

C Archive

Index

1. C Language Introduction - GeeksforGeeks
2. C Programming Language Standard - GeeksforGeeks
3. int (1 sign bit + 31 data bits) keyword in C - GeeksforGeeks
4. Is it fine to write "void main()" or "main()" in C/C++? - GeeksforGeeks
5. Difference between "int main()" and "int main(void)" in C/C++? - GeeksforGeeks
6. Interesting Facts about Macros and Preprocessors in C - GeeksforGeeks
7. Compiling a C program:- Behind the Scenes - GeeksforGeeks
8. Benefits of C language over other programming languages - GeeksforGeeks
9. Program error signals - GeeksforGeeks
10. Escape Sequences in C - GeeksforGeeks
11. Line Splicing in C/C++ - GeeksforGeeks
12. C/C++ Tokens - GeeksforGeeks
13. Variables and Keywords in C - GeeksforGeeks
14. How are variables scoped in C - Static or Dynamic? - GeeksforGeeks
15. Scope rules in C - GeeksforGeeks
16. How Linkers Resolve Global Symbols Defined at Multiple Places? - GeeksforGeeks
17. Variable Declaration and Scope in C - GeeksQuiz
18. Complicated declarations in C - GeeksforGeeks
19. Redeclaration of global variable in C - GeeksforGeeks
20. Internal Linkage and External Linkage in C - GeeksforGeeks
21. Different ways to declare variable as constant in C and C++ - GeeksforGeeks
22. Why variable name does not start with numbers in C ? - GeeksforGeeks
23. Redeclaration of global variable in C - GeeksforGeeks
24. Initialization of global and static variables in C - GeeksforGeeks
25. Data Types in C - GeeksforGeeks
26. Use of bool in C - GeeksforGeeks
27. Integer Promotions in C - GeeksforGeeks
28. Data Types in C - GeeksQuiz
29. Comparison of a float with a value in C - GeeksforGeeks
30. Is there any need of "long" data type in C and C++? - GeeksforGeeks
31. What is the size_t data type in C? - GeeksforGeeks
32. Interesting facts about data-types and modifiers in C/C++ - GeeksforGeeks
33. Difference between float and double in C/C++ - GeeksforGeeks
34. Character arithmetic in C and C++ - GeeksforGeeks
35. Type Conversion in C - GeeksforGeeks
36. Storage Classes in C - GeeksforGeeks
37. Static Variables in C - GeeksforGeeks
38. Understanding "extern" keyword in C - GeeksforGeeks
39. What are the default values of static variables in C? - GeeksforGeeks
40. Understanding "volatile" qualifier in C | Set 2 (Examples) - GeeksforGeeks
41. Const Qualifier in C - GeeksforGeeks
42. Initialization of static variables in C - GeeksforGeeks
43. Understanding "register" keyword in C - GeeksforGeeks
44. Storage Classes and Type Qualifiers in C - GeeksQuiz
45. Understanding "volatile" qualifier in C | Set 1 (Introduction) - GeeksforGeeks
46. Return values of printf() and scanf() in C/C++ - GeeksforGeeks
47. What is return type of getchar(), fgetc() and getc() ? - GeeksforGeeks
48. Scansets in C - GeeksforGeeks
49. puts() vs printf() for printing a string - GeeksforGeeks
50. What is use of %n in printf() ? - GeeksforGeeks
51. How to print % using printf()? - GeeksforGeeks
52. Input and Output in C - GeeksQuiz
53. What is the difference between printf, sprintf and fprintf? - GeeksforGeeks
54. Difference betweengetc(), getchar(), getch() and getche() - GeeksforGeeks
55. Difference between %d and %i format specifier in C language - GeeksforGeeks
56. Use of fflush(stdin) in C - GeeksforGeeks
57. Clearing The Input Buffer In C/C++ - GeeksforGeeks
58. scanf() and fscanf() in C - Simple Yet Powerful - GeeksforGeeks
59. getchar_unlocked() - faster input in C/C++ for Competitive Programming - GeeksforGeeks
60. Problem with scanf() when there is fgets()/gets()/scanf() after it - GeeksforGeeks
61. Differentiate printable and control character in C ? - GeeksforGeeks
62. rand() and srand() in C/C++ - GeeksforGeeks
63. Operators in C | Set 1 (Arithmetic Operators) - GeeksforGeeks
64. Operators in C | Set 2 (Relational and Logical Operators) - GeeksforGeeks
65. Bitwise Operators in C/C++ - GeeksforGeeks
66. Operator Precedence and Associativity in C - GeeksforGeeks
67. Evaluation order of operands - GeeksforGeeks

- 68. Comma in C and C++ - GeeksforGeeks
- 69. sizeof operator in C - GeeksforGeeks
- 70. Operands for sizeof operator - GeeksforGeeks
- 71. A comma operator question - GeeksforGeeks
- 72. Result of comma operator as l-value in C and C++ - GeeksforGeeks
- 73. Order of operands for logical operators - GeeksforGeeks
- 74. Increment (Decrement) operators require L-value Expression - GeeksforGeeks
- 75. Precedence of postfix ++ and prefix ++ in C/C++ - GeeksforGeeks
- 76. Modulus on Negative Numbers - GeeksforGeeks
- 77. C/C++ Ternary Operator - Some Interesting Observations - GeeksforGeeks
- 78. Pre-increment (or pre-decrement) in C++ - GeeksforGeeks
- 79. Difference between ++p, *p++ and *++p - GeeksforGeeks
- 80. Results of comparison operations in C and C++ - GeeksforGeeks
- 81. To find sum of two numbers without using any operator - GeeksforGeeks
- 82. Sequence Points in C | Set 1 - GeeksforGeeks
- 83. Execution of printf with ++ operators - GeeksforGeeks
- 84. Anything written in sizeof() is never executed in C - GeeksforGeeks
- 85. Difference between strlen() and sizeof() for string in C - GeeksforGeeks
- 86. # and ## Operators in C - GeeksforGeeks
- 87. Write a C macro PRINT(x) which prints x - GeeksforGeeks
- 88. Variable length arguments for Macros - GeeksforGeeks
- 89. Multiline macros in C - GeeksforGeeks
- 90. CRASH() macro - interpretation - GeeksforGeeks
- 91. The OFFSETOF() macro - GeeksforGeeks
- 92. Branch prediction macros in GCC - GeeksforGeeks
- 93. Difference between #define and const in C? - GeeksforGeeks
- 94. A C Programming Language Puzzle - GeeksforGeeks
- 95. What's difference between header files "stdio.h" and "stdlib.h" ? - GeeksforGeeks
- 96. How to print a variable name in C? - GeeksforGeeks
- 97. Constants in C/C++ - GeeksforGeeks
- 98. How a Preprocessor works in C? - GeeksforGeeks
- 99. C/C++ Preprocessors - GeeksforGeeks
- 100. C/C++ Preprocessor directives | Set 2 - GeeksforGeeks
- 101. isgraph() C library function - GeeksforGeeks
- 102. How to write your own header file in C? - GeeksforGeeks
- 103. difftime() C library function - GeeksforGeeks
- 104. tmpnam() function in C - GeeksforGeeks
- 105. _Generic keyword in C - GeeksforGeeks
- 106. C Library math.h functions - GeeksforGeeks
- 107. typedef versus #define in C - GeeksforGeeks
- 108. strftime() function in C/C++ - GeeksforGeeks
- 109. exec family of functions in C - GeeksforGeeks
- 110. Arrays in C/C++ - GeeksforGeeks
- 111. Strings in C - GeeksforGeeks
- 112. Arrays in C Language | Set 2 (Properties) - GeeksforGeeks
- 113. Do not use sizeof for array parameters - GeeksforGeeks
- 114. Initialization of variables sized arrays in C - GeeksforGeeks
- 115. Are array members deeply copied? - GeeksforGeeks
- 116. What is the difference between single quoted and double quoted declaration of char array? - GeeksforGeeks
- 117. Initialization of a multidimensional arrays in C/C++ - GeeksforGeeks
- 118. Write one line functions for strcat() and strcmp() - GeeksforGeeks
- 119. What's difference between char s[] and char *s in C? - GeeksforGeeks
- 120. gets() is risky to use! - GeeksforGeeks
- 121. C function to Swap strings - GeeksforGeeks
- 122. Storage for Strings in C - GeeksforGeeks
- 123. Difference between pointer and array in C? - GeeksforGeeks
- 124. How to dynamically allocate a 2D array in C? - GeeksforGeeks
- 125. How to pass a 2D array as a parameter in C? - GeeksforGeeks
- 126. How to write long strings in Multi-lines C/C++? - GeeksforGeeks
- 127. What are the data types for which it is not possible to create an array? - GeeksforGeeks
- 128. Variable Length Arrays in C and C++ - GeeksforGeeks
- 129. A shorthand array notation in C for repeated values - GeeksforGeeks
- 130. Accessing array out of bounds in C/C++ - GeeksforGeeks
- 131. strcpy in C/C++ - GeeksforGeeks
- 132. strcmp() in C/C++ - GeeksforGeeks
- 133. strdup() and strndup() functions in C/C++ - GeeksforGeeks
- 134. How to pass an array by value in C ? - GeeksforGeeks
- 135. Different methods to reverse a string in C/C++ - GeeksforGeeks
- 136. strpbrk() in C - GeeksforGeeks
- 137. strcoll() in C/C++ - GeeksforGeeks

- 138. `ispunct()` function in C - GeeksforGeeks
- 139. `strspn()` function in C - GeeksforGeeks
- 140. `isalpha()` and `isdigit()` functions in C with `cstring` examples. - GeeksforGeeks
- 141. Data type of case labels of switch statement in C++? - GeeksforGeeks
- 142. For Versus While - GeeksforGeeks
- 143. A nested loop puzzle - GeeksforGeeks
- 144. Interesting facts about switch statement in C - GeeksforGeeks
- 145. Difference between `while(1)` and `while(0)` in C language - GeeksforGeeks
- 146. `goto` statement in C/C++ - GeeksforGeeks
- 147. Continue Statement in C/C++ - GeeksforGeeks
- 148. Break Statement in C/C++ - GeeksforGeeks
- 149. Using range in switch case in C/C++ - GeeksforGeeks
- 150. Functions in C/C++ - GeeksforGeeks
- 151. Importance of function prototype in C - GeeksforGeeks
- 152. Functions that are executed before and after `main()` in C - GeeksforGeeks
- 153. `return` statement vs `exit()` in `main()` - GeeksforGeeks
- 154. How to Count Variable Numbers of Arguments in C? - GeeksforGeeks
- 155. What is evaluation order of function parameters in C? - GeeksforGeeks
- 156. Does C support function overloading? - GeeksforGeeks
- 157. How can I return multiple values from a function? - GeeksforGeeks
- 158. What is the purpose of a function prototype? - GeeksforGeeks
- 159. Static functions in C - GeeksforGeeks
- 160. `exit()`, `abort()` and `assert()` - GeeksforGeeks
- 161. Implicit return type `int` in C - GeeksforGeeks
- 162. What happens when a function is called before its declaration in C? - GeeksforGeeks
- 163. `_Noreturn` function specifier in C - GeeksforGeeks
- 164. `exit()` vs `_Exit()` in C and C++ - GeeksforGeeks
- 165. Predefined Identifier `_func_` in C - GeeksforGeeks
- 166. Callbacks in C - GeeksforGeeks
- 167. Nested functions in C - GeeksforGeeks
- 168. Parameter Passing Techniques in C/C++ - GeeksforGeeks
- 169. Power Function in C/C++ - GeeksforGeeks
- 170. `tolower()` function in C - GeeksforGeeks
- 171. `time()` function in C - GeeksforGeeks
- 172. Pointers in C and C++ | Set 1 (Introduction, Arithmetic and Array) - GeeksforGeeks
- 173. Double Pointer (Pointer to Pointer) in C - GeeksforGeeks
- 174. Why C treats array parameters as pointers? - GeeksforGeeks
- 175. Output of the program | Dereference, Reference, Dereference, Reference.... - GeeksforGeeks
- 176. Dangling, Void , Null and Wild Pointers - GeeksforGeeks
- 177. An Uncommon representation of array elements - GeeksforGeeks
- 178. How to declare a pointer to a function? - GeeksforGeeks
- 179. Pointer vs Array in C - GeeksforGeeks
- 180. void pointer in C / C++ - GeeksforGeeks
- 181. NULL pointer in C - GeeksforGeeks
- 182. Function Pointer in C - GeeksforGeeks
- 183. What are near, far and huge pointers? - GeeksforGeeks
- 184. Generic Linked List in C - GeeksforGeeks
- 185. `restrict` keyword in C - GeeksforGeeks
- 186. Difference between `const char *p`, `char * const p` and `const char * const p` - GeeksforGeeks
- 187. Pointer to an Array | Array Pointer - GeeksforGeeks
- 188. Enumeration (or enum) in C - GeeksforGeeks
- 189. Structures in C - GeeksforGeeks
- 190. Union in C - GeeksforGeeks
- 191. Struct Hack - GeeksforGeeks
- 192. Structure Member Alignment, Padding and Data Packing - GeeksforGeeks
- 193. Operations on struct variables in C - GeeksforGeeks
- 194. Bit Fields in C - GeeksforGeeks
- 195. Structure Sorting (By Multiple Rules) in C++ - GeeksforGeeks
- 196. Flexible Array Members in a structure in C - GeeksforGeeks
- 197. Difference between Structure and Union in C - GeeksforGeeks
- 198. Difference between C structures and C++ structures - GeeksforGeeks
- 199. Anonymous Union and Structure in C - GeeksforGeeks
- 200. Compound Literals in C - GeeksforGeeks
- 201. Memory Layout of C Programs - GeeksforGeeks
- 202. How to deallocate memory without using `free()` in C? - GeeksforGeeks
- 203. Difference Between `malloc()` and `calloc()` with Examples - GeeksforGeeks
- 204. How does `free()` know the size of memory to be deallocated? - GeeksforGeeks
- 205. Use of `realloc()` - GeeksforGeeks
- 206. What is Memory Leak? How can we avoid? - GeeksforGeeks
- 207. `fseek()` vs `rewind()` in C - GeeksforGeeks

- 208. EOF, getc() and feof() in C - GeeksforGeeks
- 209. fopen() for an existing file in write mode - GeeksforGeeks
- 210. Read/Write structure to a file in C - GeeksforGeeks
- 211. fgets() and gets() in C language - GeeksforGeeks
- 212. Basics of File Handling in C - GeeksforGeeks
- 213. fsetpos() (Set File Position) in C - GeeksforGeeks
- 214. rename function in C/C++ - GeeksforGeeks
- 215. tmpfile() function in C - GeeksforGeeks
- 216. fgetc() and fputc() in C - GeeksforGeeks
- 217. fseek() in C/C++ with example - GeeksforGeeks
- 218. ftell() in C with example - GeeksforGeeks
- 219. lseek() in C/C++ to read the alternate nth byte and write it in another file - GeeksforGeeks
- 220. C program to delete a file - GeeksforGeeks
- 221. C Program to merge contents of two files into a third file - GeeksforGeeks
- 222. C Program to print contents of file - GeeksforGeeks
- 223. C Program to print numbers from 1 to N without using semicolon? - GeeksforGeeks
- 224. To find sum of two numbers without using any operator - GeeksforGeeks
- 225. How will you show memory representation of C variables? - GeeksforGeeks
- 226. Condition To Print "HelloWord" - GeeksforGeeks
- 227. Change/add only one character and print 'x' exactly 20 times - GeeksforGeeks
- 228. Program for Sum of the digits of a given number - GeeksforGeeks
- 229. What is the best way in C to convert a number to a string? - GeeksforGeeks
- 230. Program to compute Log n - GeeksforGeeks
- 231. Print "Even" or "Odd" without using conditional statement - GeeksforGeeks
- 232. How will you print numbers from 1 to 100 without using loop? - GeeksforGeeks
- 233. Program for Sum of the digits of a given number - GeeksforGeeks
- 234. Write a C program to print "Geeks for Geeks" without using a semicolon - GeeksforGeeks
- 235. Write a one line C function to round floating point numbers - GeeksforGeeks
- 236. Implement Your Own sizeof - GeeksforGeeks
- 237. How to count set bits in a floating point number in C? - GeeksforGeeks
- 238. How to change the output of printf() in main() ? - GeeksforGeeks
- 239. How to find length of a string without string.h and loop in C? - GeeksforGeeks
- 240. Implement your own itoa() - GeeksforGeeks
- 241. Write a C program that does not terminate when Ctrl+C is pressed - GeeksforGeeks
- 242. How to measure time taken by a function in C? - GeeksforGeeks
- 243. Print a long int in C using putchar() only - GeeksforGeeks
- 244. Convert a floating point number to string in C - GeeksforGeeks
- 245. How to write a running C code without main()? - GeeksforGeeks
- 246. Write your own memcpy() and memmove() - GeeksforGeeks
- 247. C program to print characters without using format specifiers - GeeksforGeeks
- 248. C program to print a string without any quote (single or double) in the program - GeeksforGeeks
- 249. Execute both if and else statements in C/C++ simultaneously - GeeksforGeeks
- 250. Print "Hello World" in C/C++ without using any header file - GeeksforGeeks
- 251. Quine - A self-reproducing program - GeeksforGeeks
- 252. Complicated declarations in C - GeeksforGeeks
- 253. Use of bool in C - GeeksforGeeks
- 254. Sequence Points in C | Set 1 - GeeksforGeeks
- 255. Optimization Techniques | Set 2 (swapping) - GeeksforGeeks
- 256. ASCII NUL, ASCII 0 ('0') and Numeric literal 0 - GeeksforGeeks
- 257. Little and Big Endian Mystery - GeeksforGeeks
- 258. Comparator function of qsort() in C - GeeksforGeeks
- 259. Program to validate an IP address - GeeksforGeeks
- 260. Multithreading in C - GeeksforGeeks
- 261. Assertions in C/C++ - GeeksforGeeks
- 262. fork() in C - GeeksforGeeks
- 263. Interesting Facts in C Programming - GeeksforGeeks
- 264. Precision of floating point numbers in C++ (floor(), ceil(), trunc(), round() and setprecision()) - GeeksforGeeks
- 265. Concept of setjump and longjump in C - GeeksforGeeks
- 266. nextafter() and nexttoward() in C/C++ - GeeksforGeeks
- 267. pthread_cancel() in C with example - GeeksforGeeks
- 268. pthread_equal() in C with example - GeeksforGeeks
- 269. pthread_self() in C with Example - GeeksforGeeks
- 270. Local Labels in C - GeeksforGeeks
- 271. Ivalue and rvalue in C language - GeeksforGeeks
- 272. Get the stack size and set the stack size of thread attribute in C - GeeksforGeeks
- 273. Difference between fork() and exec() - GeeksforGeeks
- 274. Errors in C/C++ - GeeksforGeeks
- 275. Why is C considered faster than other languages ? - GeeksforGeeks
- 276. Incompatibilities between C and C++ codes - GeeksforGeeks
- 277. Convert C/C++ code to assembly language - GeeksforGeeks

- [278. Error Handling in C programs - GeeksforGeeks](#)
[279. Executing main\(\) in C/C++ - behind the scene - GeeksforGeeks](#)
[280. Hygienic Macros : An Introduction - GeeksforGeeks](#)
[281. Command line arguments in C/C++ - GeeksforGeeks](#)
[282. Inbuilt library functions for user Input | scanf, fscanf, sscanf, scanf_s, fscanf_s, sscanf_s - GeeksforGeeks](#)
[283. Interesting Facts in C Programming - GeeksforGeeks](#)
[284. Database Connectivity using C/C++ - GeeksforGeeks](#)
[285. Function Interposition in C with an example of user defined malloc\(\) - GeeksforGeeks](#)
[286. Macros vs Functions - GeeksforGeeks](#)
[287. Write your own memcpy\(\) and memmove\(\) - GeeksforGeeks](#)
[288. Commonly Asked C Programming Interview Questions](#)
[289. Commonly Asked C Programming Interview Questions | Set 2 - GeeksforGeeks](#)
-

C Language Introduction

C is a procedural programming language. It was initially developed by Dennis Ritchie in the year 1972. It was mainly developed as a system programming language to write an operating system. The main features of C language include low-level access to memory, a simple set of keywords, and clean style, these features make C language suitable for system programms like an operating system or compiler development.

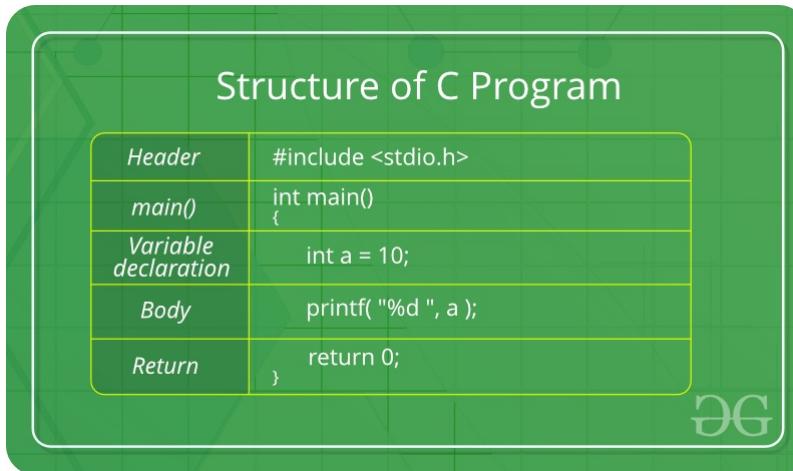
Many later languages have borrowed syntax/features directly or indirectly from C language. Like syntax of Java, PHP, JavaScript, and many other languages are mainly based on C language. C++ is nearly a superset of C language (There are few programs that may compile in C, but not in C++)�

Beginning with C programming:

1. Structure of a C program

After the above discussion, we can formally assess the structure of a C program. By structure, it is meant that any program can be written in this structure only. Writing a C program in any other structure will hence lead to a Compilation Error.

The structure of a C program is as follows:



The components of the above structure are:

1. Header Files Inclusion:

The first and foremost component is the inclusion of the Header files in a C program. A header file is a file with extension .h which contains C function declarations and macro definitions to be shared between several source files.

Some of C Header files:

- stddef.h - Defines several useful types and macros.
- stdint.h - Defines exact width integer types.
- stdio.h - Defines core input and output functions
- stdlib.h - Defines numeric conversion functions, pseudo-random number generator, memory allocation
- string.h - Defines string handling functions
- math.h - Defines common mathematical functions

Syntax to include a header file in C:

```
#include
```

2. Main Method Declaration:

The next part of a C program is to declare the main() function. The syntax to declare the main function is:

Syntax to Declare main method:

```
int main()
{
```

3. Variable Declaration:

The next part of any C program is the variable declaration. It refers to the variables that are to be used in the function. Please note that in the C program, no variable can be used without being declared. Also in a C program, the variables are to be declared before any operation in the function.

Example:

```
int main()
{
    int a;
    .
}
```

4. Body:

Body of a function in C program, refers to the operations that are performed in the functions. It can be anything like manipulations, searching, sorting, printing, etc.

Example:

```
int main()
{
    int a;
    printf("%d", a);
    .
}
```

5. Return Statement:

The last part in any C program is the return statement. The return statement refers to the returning of the values from a function. This return statement and return value depend upon the return type of the function. For example, if the return type is void, then there will be no return statement. In any other case, there will be a return statement and the return value will be of the type of the specified return type.

Example:

```
int main()
{
```

```
int a;  
printf("%d", a);  
return 0;  
}
```

2. Writing first program:

Following is first program in C

filter_none

edit

close

play_arrow

link

brightness_4

code

```
#include <stdio.h>
```

```
int main(void)
```

```
{
```

```
    printf("GeeksQuiz");  
    return 0;
```

```
}
```

chevron_right

filter_none

Let us analyze the program line by line.

Line 1: [#include <stdio.h>] In a C program, all lines that start with # are processed by **preprocessor** which is a program invoked by the compiler. In a very basic term, **preprocessor** takes a C program and produces another C program. The produced program has no lines starting with #, all such lines are processed by the preprocessor. In the above example, preprocessor copies the preprocessed code of stdio.h to our file. The .h files are called header files in C. These header files generally contain declaration of functions. We need stdio.h for the function printf() used in the program.

Line 2 [int main(void)] There must be starting point from where execution of compiled C program begins. In C, the execution typically begins with first line of main(). The void written in brackets indicates that the main doesn't take any parameter (See [this](#) for more details). main() can be written to take parameters also. We will be covering that in future posts.

The int written before main indicates return type of main(). The value returned by main indicates status of program termination. See [this](#) post for more details on return type.

Line 3 and 6: [{ and }] In C language, a pair of curly brackets define a scope and mainly used in functions and control statements like if, else, loops. All functions must start and end with curly brackets.

Line 4 [printf("GeeksQuiz");] printf() is a standard library function to print something on standard output. The semicolon at the end of printf indicates line termination. In C, semicolon is always used to indicate end of statement.

Line 5 [return 0;] The return statement returns the value from main(). The returned value may be used by operating system to know termination status of your program. The value 0 typically means successful termination.

3. How to execute the above program:

Inorder to execute the above program, we need to have a compiler to compile and run our programs. There are certain online compilers like <https://ide.geeksforgeeks.org/>, <http://ideone.com/> or <http://codepad.org/> that can be used to start C without installing a compiler.

Windows: There are many compilers available freely for compilation of C programs like **Code Blocks** and **Dev-CPP**. We strongly recommend Code Blocks.

Linux: For Linux, **gcc** comes bundled with the linux, Code Blocks can also be used with Linux.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes *arrow_drop_up*

Add your personal notes
here! (max 5000 chars)

Save

Recommended Posts:

- [Introduction to the C99 Programming Language : Part III](#)
- [Introduction to the C99 Programming Language : Part I](#)
- [Introduction to the C99 Programming Language : Part II](#)
- [kbhit in C language](#)
- [Stopwatch using C language](#)
- [Difference between while\(1\) and while\(0\) in C language](#)
- [fgets\(\) and gets\(\) in C language](#)
- [Signals in C language](#)
- [isalnum\(\) function in C Language](#)
- [isxdigit\(\) function in C Language](#)
- [isupper\(\) function in C Language](#)
- [How to use POSIX semaphores in C language](#)
- [How to clear console in C language?](#)
- [Difference between Java and C language](#)
- [chdir\(\) in C language with Examples](#)

Improved By : [RishabhPrabhu](#), [anshu8tu](#)

Article Tags :

[C](#)

[C Basics](#)

Practice Tags :

[C](#)

thumb_up

398

To-do Done
1.4

Based on 463 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) Difference between pointer and array in C?

Next

[last_page](#) Interesting Facts about Macros and Preprocessors in C

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Weekend Geeks Classes Weekdays

JAVA APP DEVELOPMENT SUMMER TRAINING COMPETITIVE PROGRAMMING LIVE Java Backend Development

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
Predefined Macros in C with Examples
C program to print odd line contents of a File followed by even line content
Introduction to the C99 Programming Language : Part II
C program to find square root of a given number

More related articles in C
Format specifiers in different Programming Languages
Problem in comparing Floating point numbers and how to compare them correctly?
Features of C Programming Language
Pointer Expressions in C with Examples
Introduction to the C99 Programming Language : Part III

C Programming Language Standard

The idea of this article is to introduce C standard.

What to do when a C program produces different results in two different compilers?

For example, consider the following simple C program.

```
filter_none
edit
close
play_arrow
link
brightness_4
code
void main() { }
```

The above program fails in gcc as the return type of main is void, but it compiles in Turbo C. How do we decide whether it is a legitimate C program or not?

Consider the following program as another example. It produces different results in different compilers.

```
filter_none
edit
close
play_arrow
link
brightness_4
code
#include<stdio.h>
int main()
{
    int i = 1;
    printf("%d %d\n", i++, i++, i);
    return 0;
}
```

```
2 1 3 - using g++ 4.2.1 on Linux.i686
1 2 3 - using SunStudio C++ 5.9 on Linux.i686
2 1 3 - using g++ 4.2.1 on SunOS.x86pc
1 2 3 - using SunStudio C++ 5.9 on SunOS.x86pc
1 2 3 - using g++ 4.2.1 on SunOS.sun4u
1 2 3 - using SunStudio C++ 5.9 on SunOS.sun4u
```

Source: [Stackoverflow](#)

Which compiler is right?

The answer to all such questions is C standard. In all such cases we need to see what C standard says about such programs.

What is C standard?

The latest C standard is ISO/IEC 9899:2011, also known as C11 as the final draft was published in 2011. Before C11, there was C99. The C11 final draft is available [here](#). See [this](#) for complete history of C standards.

Can we know behavior of all programs from C standard?

C standard leaves some behavior of many C constructs as `undefined` and some as `unspecified` to simplify the specification and allow some flexibility in implementation. For example, in C the use of any automatic variable before it has been initialized yields undefined behavior and order of evaluations of subexpressions is unspecified. This specifically frees the compiler to do whatever is easiest or most efficient,

should such a program be submitted.

So what is the conclusion about above two examples?

Let us consider the first example which is "void main() {}", the standard says following about prototype of main().

```
The function called at program startup is named main. The implementation  
declares no prototype for this function. It shall be defined with a return  
type of int and with no parameters:  
    int main(void) { /* ... */ }  
or with two parameters (referred to here as argc and argv, though any names  
may be used, as they are local to the function in which they are declared):  
    int main(int argc, char *argv[]) { /* ... */ }  
or equivalent;10) or in some other implementation-defined manner.
```

So the return type void doesn't follow the standard and it's something allowed by certain compilers.

Let us talk about second example. Note the following statement in C standard is listed under unspecified behavior.

```
The order in which the function designator, arguments, and  
subexpressions within the arguments are evaluated in a function  
call (6.5.2.2).
```

What to do with programs whose behavior is undefined or unspecified in standard?

As a programmer, it is never a good idea to use programming constructs whose behaviour is undefined or unspecified, such programs should always be discouraged. The output of such programs may change with compiler and/or machine.

This article is contributed by **Abhay Rathi**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes 

Add your personal notes
here! (max 5000 chars)

Recommended Posts:

- Features of C Programming Language
- A C Programming Language Puzzle
- Introduction to C++ Programming Language
- Introduction to the C99 Programming Language : Part I
- Introduction to the C99 Programming Language : Part II
- Benefits of C language over other programming languages
- Introduction to the C99 Programming Language : Part III
- Unordered Sets in C++ Standard Template Library
- Stopwatch using C language
- Difference between while(1) and while(0) in C language
- C Language Introduction
- Signals in C language
- kbhit in C language
- fgets() and gets() in C language
- Interesting facts about C Language

Article Tags :

C
C Basics
CPP-Basics

Practice Tags :

C



115

To-do Done
1.7

Based on 236 vote(s)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) Difference between "int main()" and "int main(void)" in C/C++?

Next

[last_page](#) C | Operators | Question 21

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
Predefined Macros in C with Examples

C program to print odd line contents of a File followed by even line content

Introduction to the C99 Programming Language : Part II

C program to find square root of a given number

Format Specifiers in different Programming Languages

Problem in Storing Floating Point numbers and how to compare them correctly?

Features of C Programming Language

Pointer Expressions in C with Examples

Introduction to the C99 Programming Language : Part III

int (1 sign bit + 31 data bits) keyword in C

In C programming language a most common keyword 'int' is used to define any positive or negative integer. But there is a difference between an integer and the numbers which can be represented with the help of the keyword 'int'. Not every integer can be represented with the keyword 'int'. According to MinGW the size of one 'int' is 4 bytes which is equal to 32 bits (1 byte=8 bits). It is still a myth somewhere that 'int' can represent an integer or 'int' is used to represent integers. Integer is a very vast category of numbers where as one 'int' has limited and exact amount of memory (size of 'int' is 4 bytes or 32 bits) to store what is being represented by it. An 'int' type variable in C language is able to store only numbers till 2147483647. Beyond this number 'int' fails to store precisely and even not correctly. 'int' is a 32 bit data type. Whenever a number is being assigned to an 'int' type variable, it is first converted to its binary representation (that is in 0's and 1's) then it is kept in memory at specific location. An 'int' is actually 1 sign bit + 31 data bits, that is 31 bits are available for storing the number being assigned to a 'int' type variable and 1 bit is reserved for maintaining the sign of the number which is either + or -. The sign is also represented by binary digits, 0 for positive sign and 1 for negative sign.

Let us understand this by an example.

Example - Consider,

```
int num=2147483647;
```

At this point first 2147483647 will be converted into its binary form which is equal to:

11111111111111111111111111111111.

11111111111111111111111111111111 is a 31 digit binary number which will be assigned to variable num's right most 31 bits and the 32nd bit will have a zero(0) as the number being assigned to variable num is a positive number. If we try to store any number greater than 2147483647 into an 'int' type variable then we will lose information.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes

Add your personal notes here! (max 5000 chars)

Recommended Posts:

- [_Generic keyword in C](#)
- [restrict keyword in C](#)
- [Use of explicit keyword in C++](#)
- [C++ mutable keyword](#)
- [C++ | Static Keyword | Question 1](#)
- [C++ | Static Keyword | Question 2](#)
- [C++ | Static Keyword | Question 3](#)
- [C++ | friend keyword | Question 1](#)
- [C++ | friend keyword | Question 2](#)
- [C++ | Static Keyword | Question 4](#)
- [C++ | const keyword | Question 1](#)
- [C++ | const keyword | Question 2](#)
- [C++ | const keyword | Question 3](#)
- [C++ | Static Keyword | Question 5](#)
- [C++ | const keyword | Question 5](#)



MohitPandey1

Check out this Author's contributed articles.

If you like GeeksforGeeks and would like to contribute, you can also write an article using [contribute.geeksforgeeks.org](#) or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please Improve this article if you find anything incorrect by clicking on the "Improve Article" button below.

Article Tags :

C
C Basics
C-Data Types

Practice Tags :

C

116

To-do Done
1.4

Based on 54 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) Measure execution time with high precision in C/C++

Next

[last_page](#) Program error signals

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Ahmed Geeks Classes Workshops

JAVA APP DEVELOPMENT SUMMER TRAINING COMPETITIVE PROGRAMMING LIVE Java Backend Development

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS

Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
Predefined Macros in C with Examples
Format specifiers in different Programming Languages
C program to print odd line contents of a file followed by even line content
C program to find square root of a given number

More related articles in C
Introduction to the C99 Programming Language : Part II
Problem in comparing Floating point numbers and how to compare them correctly?
Features of C Programming Language
Pointer Expressions in C with Examples
Introduction to the C99 Programming Language : Part III

Is it fine to write “void main()” or “main()” in C/C++?

The definition

[filter_none](#)

[edit](#)

[close](#)

[play_arrow](#)

[link](#)

[brightness_4](#)

[code](#)

`void main() { /* ... */ }`

[chevron_right](#)

[filter_none](#)

is not and never has been C++, nor has it even been C. See the ISO C++ standard 3.6.1[2] or the ISO C standard 5.1.2.2.1. A conforming implementation accepts

[filter_none](#)

[edit](#)

[close](#)

[play_arrow](#)

[link](#)

[brightness_4](#)

[code](#)

`int main() { /* ... */ }`

[chevron_right](#)

[filter_none](#)

and

```
filter_none
edit
close
play_arrow
link
brightness_4
code
int main(int argc, char* argv[]) { /* ... */ }
chevron_right
filter_none
```

A conforming implementation may provide more versions of main(), but they must all have return type int. The int returned by main() is a way for a program to return a value to "the system" that invokes it. On systems that doesn't provide such a facility the return value is ignored, but that doesn't make "void main()" legal C++ or legal C. **Even if your compiler accepts "void main()" avoid it, or risk being considered ignorant by C and C++ programmers.**

In C++, main() need not contain an explicit return statement. In that case, the value returned is 0, meaning successful execution. For example:

```
filter_none
edit
close
play_arrow
link
brightness_4
code
#include <iostream>
int main()
{
    std::cout << "This program returns the integer value 0\n";
}
chevron_right
filter_none
```

Note also that neither ISO C++ nor C99 allows you to leave the type out of a declaration. That is, in contrast to C89 and ARM C++, "int" is not assumed where a type is missing in a declaration. Consequently:

```
filter_none
edit
close
play_arrow
link
brightness_4
code
#include <iostream>

main() { /* ... */ }
chevron_right
filter_none
```

is an error because the return type of main() is missing.

Source: http://www.stroustrup.com/bs_faq2.html#void-main

To summarize above, it is never a good idea to use "void main()" or just "main()" as it doesn't confirm standards. It may be allowed by some compilers though.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up

Save

Recommended Posts:

- Write a URL in a C++ program
- Write a C program that won't compile in C++
- Write your own memcpy() and memmove()
- How to write your own header file in C?
- How can we write main as a class in C++?
- When should we write our own copy constructor?
- When should we write our own assignment operator in C++?
- C program to write an image in PGM format
- Read/Write structure to a file in C
- Write a program that produces different results in C and C++
- How to write a running C code without main()
- Write a C macro PRINT(x) which prints x
- Write a C program that does not terminate when Ctrl+C is pressed
- fopen() for an existing file in write mode
- Read/Write Class Objects from/to File in C++

Article Tags :

C
C++
C Basics
CPP-Basics
CPP-Functions
cpp-main
Practice Tags :
C
CPP

thumb_up
152

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) Can main() be overloaded in C++?

Next

[last_page](#) C++ | Virtual Functions | Question 14

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

The image shows the GeeksforGeeks website's header and a promotional banner for live courses. The header includes links for Full Stack Development, Java App Development, React.js, Geeks Classes Recorded, Design Patterns, Java Backend Development, and Django Web Development. Below the header is a large red banner with the text 'LIVE COURSES BY GEEKSFORGEEKS' and an illustration of two people looking at a screen with a Wi-Fi signal icon.

Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
ctype.h<ctype.h> library in C/C++ with Examples
How to use recursion within function in C or C++
Format specifiers in different Programming Languages
Predefined Macros in C with Examples

Most visited in C++
Vector of Vectors in C++ STL with Examples
C++ Tutorial
Which C++ libraries are useful for competitive programming?
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

Difference between "int main()" and "int main(void)" in C/C++?

Consider the following two definitions of main().

```
filter_none
edit
close

play_arrow
link
brightness_4
code

int main()
{
    /* */
    return 0;
}
```

chevron_right

filter_none

and

filter_none

edit

close

play_arrow

link

brightness_4

code

```
int main(void)
{
    /* */
    return 0;
}
```

chevron_right

filter_none

What is the difference?

In C++, there is no difference, both are same.

Both definitions work in C also, but the second definition with void is considered technically better as it clearly specifies that main can only be called without any parameter.

In C, if a function signature doesn't specify any argument, it means that the function can be called with any number of parameters or without any parameters. For example, try to compile and run following two C programs (remember to save your files as .c). Note the difference between two signatures of fun().

```
filter_none
edit
close
```

play_arrow
link
brightness_4
code

```
// Program 1 (Compiles and runs fine in C, but not in C++)
void fun() { }
int main(void)
{
    fun(10, "GfG", "GQ");
    return 0;
}
```

chevron_right

filter_none

The above program compiles and runs fine (See [this](#)), but the following program fails in compilation (see [this](#))

filter_none
edit
close

play_arrow

link
brightness_4
code

```
// Program 2 (Fails in compilation in both C and C++)
void fun(void) { }
int main(void)
{
    fun(10, "GfG", "GQ");
    return 0;
}
```

chevron_right

filter_none

Unlike C, in C++, both of the above programs fails in compilation. In C++, both `fun()` and `fun(void)` are same.

So the difference is, in C, `int main()` can be called with any number of arguments, but `int main(void)` can only be called without any argument. Although it doesn't make any difference most of the times, using "`int main(void)`" is a recommended practice in C.

Exercise:

Predict the output of following C programs.

Question 1

filter_none
edit
close

play_arrow

link
brightness_4
code

```
#include <stdio.h>
int main()
{
    static int i = 5;
    if (--i){
        printf("%d ", i);
        main(10);
    }
}
```

chevron_right

filter_none

Question 2

filter_none
edit
close

play_arrow

link
brightness_4
code

```
#include <stdio.h>
int main(void)
{
    static int i = 5;
    if (--i){
        printf("%d ", i);
        main(10);
    }
}
```

chevron_right

filter_none

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes *arrow_drop_up*

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- Difference between C and C#
- Difference between C and C++
- Difference between `++*p`, `*p++` and `*++p`
- Difference between C and Python
- Difference between `scanf()` and `gets()` in C
- Difference between Python and C++
- Difference between while(1) and while(0) in C language
- Difference between Abstraction and Encapsulation in C++
- Difference between Private and Protected in C++ with Example
- Difference between Inheritance and Polymorphism
- Difference between Keyword and Identifier
- Difference between while and do-while loop in C, C++, Java
- Difference between `std::quick_exit` and `std::abort`
- Difference between Structure and Array in C
- Difference between Java and C language

Article Tags :

C
C++
C Basics
CPP-Functions
cpp-main
Practice Tags :
C
CPP

125

To-do Done
1.8

Based on 270 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

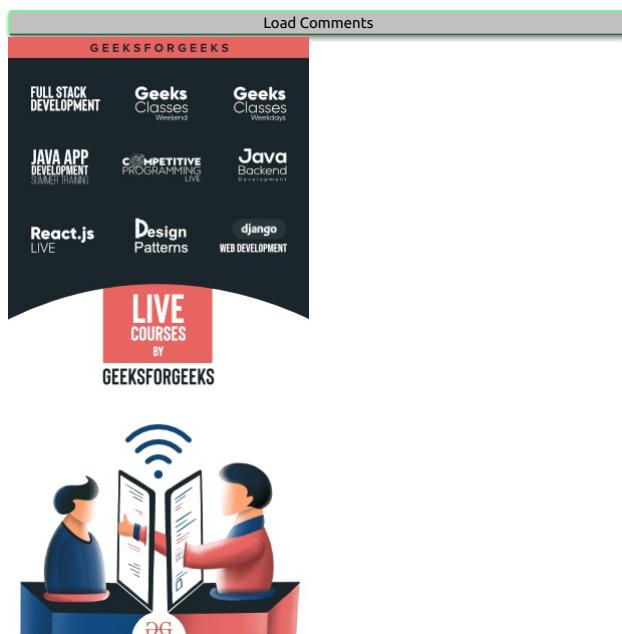
Previous

[first_page](#) C | Functions | Question 10

Next

[last_page](#) C Programming Language Standard

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
`ctype.h <cctype>` library in C/C++ with Examples
Format specifiers in different Programming Languages
C program to find square root of a given number
Predefined Macros in C with Examples

Most visited in C++
Vector of Vectors in C++ STL with Examples
C++ Tutorial
Why STL libraries are useful for competitive programming?
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

Interesting Facts about Macros and Preprocessors in C

In a C program, all lines that start with # are processed by preprocessor which is a special program invoked by the compiler. In a very basic term, preprocessor takes a C program and produces another C program without any #.

Following are some interesting facts about preprocessors in C.

1) When we use **include** directive, the contents of included header file (after preprocessing) are copied to the current file.

Angular brackets < and > instruct the preprocessor to look in the standard folder where all header files are held. Double quotes " and " instruct the preprocessor to look into the current folder (current directory).

2) When we use **define** for a constant, the preprocessor produces a C program where the defined constant is searched and matching tokens are replaced with the given expression. For example in the following program max is defined as 100.

[filter_none](#)
[edit](#)
[close](#)
[play_arrow](#)

```
link  
brightness_4  
code  
  
#include<stdio.h>  
#define max 100  
int main()  
{  
    printf("max is %d", max);  
    return 0;  
}  
chevron_right  
filter_none  
  
Output:  
max is 100
```

3) The macros can take function like arguments, the arguments are not checked for data type. For example, the following macro INCREMENT(x) can be used for x of any data type.

```
filter_none  
edit  
close  
  
play_arrow  
link  
brightness_4  
code  
  
#include <stdio.h>  
#define INCREMENT(x) ++x  
int main()  
{  
    char *ptr = "GeeksQuiz";  
    int x = 10;  
    printf("%s ", INCREMENT(ptr));  
    printf("%d", INCREMENT(x));  
    return 0;  
}  
chevron_right  
filter_none  
  
Output:  
GeeksQuiz 11
```

4) The macro arguments are not evaluated before macro expansion. For example, consider the following program

```
filter_none  
edit  
close  
  
play_arrow  
link  
brightness_4  
code  
  
#include <stdio.h>  
#define MULTIPLY(a, b) a*b  
int main()  
{  
    // The macro is expended as 2 + 3 * 3 + 5, not as 5*8  
    printf("%d", MULTIPLY(2+3, 3+5));  
    return 0;  
}  
// Output: 16  
chevron_right  
filter_none  
  
Output:  
16
```

The previous problem can be solved using following program

```
filter_none  
edit  
close  
  
play_arrow  
link  
brightness_4  
code  
  
#include <stdio.h>  
//here, instead of writing a*a we write (a)*(b)  
#define MULTIPLY(a, b) (a)*(b)  
int main()  
{  
    // The macro is expended as (2 + 3) * (3 + 5), as 5*8  
    printf("%d", MULTIPLY(2+3, 3+5));  
    return 0;  
}  
//This code is contributed by Santanu  
chevron_right  
filter_none  
  
Output:  
40
```

5) The tokens passed to macros can be concatenated using operator ## called Token-Pasting operator.

```
filter_none  
edit
```

```
close
play_arrow
link
brightness_4
code

#include <stdio.h>
#define merge(a, b) a##b
int main()
{
    printf("%d ", merge(12, 34));
}
chevron_right
```

filter_none

Output:

1234

6) A token passed to macro can be converted to a string literal by using # before it.

```
filter_none
edit
close
```

play_arrow

```
link
brightness_4
code
```

```
#include <stdio.h>
#define get(a) #a
int main()
{
    // GeeksQuiz is changed to "GeeksQuiz"
    printf("%s", get(GeeksQuiz));
}
```

chevron_right

filter_none

Output:

GeeksQuiz

7) The macros can be written in multiple lines using '\'. The last line doesn't need to have '\'.

```
filter_none
edit
close
```

play_arrow

```
link
brightness_4
code
```

```
#include <stdio.h>
#define PRINT(i, limit) while (i < limit) \
    { \
        printf("GeeksQuiz "); \
        i++; \
    }
```

int main()

{

int i = 0;
 PRINT(i, 3);
 return 0;

}

chevron_right

filter_none

Output:

GeeksQuiz GeeksQuiz GeeksQuiz

8) The macros with arguments should be avoided as they cause problems sometimes. And Inline functions should be preferred as there is type checking parameter evaluation in inline functions. From C99 onward, inline functions are supported by C language also.

For example consider the following program. From first look the output seems to be 1, but it produces 36 as output.

```
filter_none
edit
close
```

play_arrow

```
link
brightness_4
code
```

```
#include <stdio.h>
```

```
#define square(x) x*x
```

```
int main()
```

{

// Expanded as 36/6*6
int x = 36/square(6);
printf("%d", x);
return 0;

}

chevron_right

filter_none

Output:

36

If we use inline functions, we get the expected output. Also, the program given in point 4 above can be corrected using inline functions.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

#include <stdio.h>

static inline int square(int x) { return x*x; }

int main()
{
    int x = 36/square(6);
    printf("%d", x);
    return 0;
}
chevron_right
```

filter_none

Output:

(1)

9) Preprocessors also support if-else directives which are typically used for conditional compilation.

```
filter_none
edit
close
```

play_arrow

```
link
brightness_4
code
```

```
int main()
```

```
{
#define VERBOSE >= 2
    printf("Trace Message");
#endif
}
```

chevron_right

filter_none

Output:
No Output

10) A header file may be included more than one time directly or indirectly, this leads to problems of redeclaration of same variables/functions. To avoid this problem, directives like **defined**, **ifdef** and **ifndef** are used.

11) There are some standard macros which can be used to print program file (**_FILE_**), Date of compilation (**_DATE_**), Time of compilation (**_TIME_**) and Line Number in C code (**_LINE_**)

```
filter_none
edit
close
```

play_arrow

```
link
brightness_4
code
```

```
#include <stdio.h>
```

```
int main()
{
    printf("Current File :%s\n", __FILE__ );
    printf("Current Date :%s\n", __DATE__ );
    printf("Current Time :%s\n", __TIME__ );
    printf("Line Number :%d\n", __LINE__ );
    return 0;
}
```

chevron_right

filter_none

Output:
Current File :/usr/share/IDE_PROGRAMS/C/other/081c548d50135ed88cfa0296159b05ca/081c548d50135ed88cfa0296159b05ca.c
Current Date :Sep 4 2019
Current Time :10:17:43
Line Number :9

12) We can remove already defined macros using : **#undef MACRO_NAME**

```
filter_none
edit
close
```

play_arrow

```
link
brightness_4
code
```

```
#include <stdio.h>
```

```
#define LIMIT 100
int main()
{
    printf("%d",LIMIT);
    //removing defined macro LIMIT
    #undef LIMIT
    //Next line causes error as LIMIT is not defined
    printf("%d",LIMIT);
    return 0;
}
```

//This code is contributed by Santanu

chevron_right

filter_none

Following program is executed correctly as we have declared LIMIT as an integer variable after removing previously defined macro LIMIT

filter_none

edit

close

play_arrow

link

brightness_4

code

```
#include <stdio.h>
#define LIMIT 1000
int main()
{
    printf("%d",LIMIT);
    //removing defined macro LIMIT
    #undef LIMIT
    //Declare LIMIT as integer again
    int LIMIT=1001;
    printf("\n%d",LIMIT);
    return 0;
}
```

chevron_right

filter_none

Output:

```
1000
1001
```

Another interesting fact about macro using (#**undef**)

filter_none

edit

close

play_arrow

link

brightness_4

code

```
#include <stdio.h>
//div function prototype
float div(float, float);
#define div(x, y) x/y

int main()
{
    //use of macro div
    //Note: %0.2f for taking two decimal value after point
    printf("%0.2f",div(10.0,5.0));
    //removing defined macro div
    #undef div
    //function div is called as macro definition is removed
    printf("\n%0.2f",div(10.0,5.0));
    return 0;
}
```

//div function definition
float div(float x, float y){
return y/x;
}

/This code is contributed by Santanu

chevron_right

filter_none

Output:

```
2.00
0.50
```

You may like to take a [Quiz on Macros and Preprocessors in C](#)

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes *arrow_drop_up*

Add your personal notes
here! (max 5000 chars)

Recommended Posts:

- [Py-Facts - 10 interesting facts about Python](#)
- [Interesting Facts about C#](#)
- [Interesting Facts about C++](#)
- [Interesting facts about C Language](#)
- [Interesting Facts About Java](#)
- [Interesting Facts about Ubuntu](#)
- [Interesting Facts about Linux](#)
- [C++ bitset interesting facts](#)
- [Interesting Facts in C Programming](#)
- [Interesting facts about switch statement in C](#)
- [Interesting facts about null in Java](#)
- [Interesting facts about strings in Python | Set 1](#)

- Some Interesting facts about default arguments in C++
- Interesting facts about Fibonacci numbers
- Interesting facts about data-types and modifiers in C/C++

Improved By : GauravSharma16, AbhinavGarg, SantanuBasak, PearlGupta1, RahulKumawat3, [more](#)

Article Tags :

C
C Basics
cpp-macros
interesting-facts
Practice Tags :
C

143

To-do Done
2.4

Based on 306 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

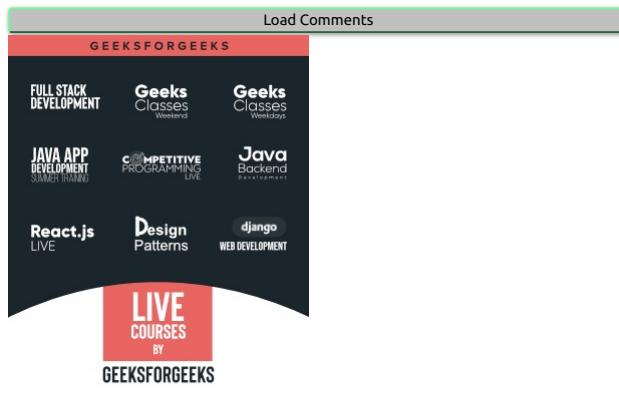
Previous

[first_page](#) C Language Introduction

Next

[last_page](#) Bitwise Operators in C/C++

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
ctype.h<ctype.h> library in C/C++ with Examples
Program to print square root with Examples
How to call function within function in C or C++
C program to print odd line contents of a File followed by even line content

More related articles In C
Introduction to the C99 Programming Language : Part II
C program to find square root of a given number
Format specifiers in different Programming Languages
Problem in comparing Floating point numbers and how to compare them correctly?
Features of C Programming Language

Compiling a C program:- Behind the Scenes

C is a high-level language and it needs a compiler to convert it into an executable code so that the program can be run on our machine.

How do we compile and run a C program?

Below are the steps we use on an Ubuntu machine with gcc compiler.

```
guest@SLICK:~/Ubuntu-/Sample
$ program for the addition of two numbers
#include<stdio.h>
#define add(a,b) (a+b) //using macros
int main()
{
    int a=5, b=6;
    printf("Addition is: %d\n", add(a,b));
    return 0;
}
```

We first create a C program using an editor and save the file as filename.c

\$ vi filename.c

The diagram on right shows a simple program to add two numbers.



Then compile it using below command.

\$ gcc -Wall filename.c -o filename

The option -Wall enables all compiler's warning messages. This option is recommended to generate better code.

The option -o is used to specify the output file name. If we do not use this option, then an output file with name a.out is generated.

```
guest@SLICK:~/Ubuntu-/Sample
guest@SLICK:~$ cd Sample
guest@SLICK:~/Ubuntu-/Samples$ ./filename
Addition is:
guest@SLICK:~/Ubuntu-/Samples$
```

After compilation executable is generated and we run the generated executable using below command.

\$./filename

What goes inside the compilation process?

Compiler converts a C program into an executable. There are four phases for a C program to become an executable:

1. Pre-processing
2. Compilation
3. Assembly
4. Linking

By executing below command, We get the all intermediate files in the current directory along with the executable.

```
$gcc -Wall -fverbose-asm filename.c -o filename
```

The following screenshot shows all generated intermediate files.



Let us one by one see what these intermediate files contain.

Pre-processing

This is the first phase through which source code is passed. This phase include:

- Removal of Comments
- Expansion of Macros
- Expansion of the included files.
- Conditional compilation

The preprocessed output is stored in the **filename.i**. Let's see what's inside filename.i using **\$vi filename.i**

A terminal window showing the content of "filename.i". The code includes various preprocessor directives like #include, #define, and #ifdef, along with standard library functions like printf and scanf. The code is heavily annotated with comments explaining the expansion of macros and inclusion of header files.

In the above output, source file is filled with lots and lots of info, but at the end our code is preserved.

Analysis:

- printf contains now a + b rather than add(a, b) that's because macros have expanded.
- Comments are stripped off.
- **#include<stdio.h>** is missing instead we see lots of code. So header files has been expanded and included in our source file.

Compiling

The next step is to compile filename.i and produce an; intermediate compiled output file **filename.s**. This file is in assembly level instructions. Let's see through this file using **\$vi filename.s**

A terminal window showing the content of "filename.s". It contains assembly language code with labels like .LC0, .LC1, .LC2, and .LC3, and various assembly instructions like movl, addl, and movw.

The snapshot shows that it is in assembly language, which assembler can understand.

Assembly

In this phase the filename.s is taken as input and turned into **filename.o** by assembler. This file contain machine level instructions. At this phase, only existing code is converted into machine language, the function calls like printf() are not resolved. Let's view this file using **\$file filename.o**

A terminal window showing the content of "filename.o". It displays assembly code with labels and instructions, similar to "filename.s" but in a more compressed form.

Linking

This is the final phase in which all the linking of function calls with their definitions are done. Linker knows where all these functions are implemented. Linker does some extra work also, it adds some extra code to our program which is required when the program starts and ends. For example, there is a code which is required for setting up the environment like passing command line arguments. This task can be easily verified by using **\$size filename.o** and **\$size filename**. Through these commands, we know that how output file increases from an object file to an executable file. This is because of the extra code that linker adds with our program.

A terminal window showing the size of "filename.o" and "filename". The output shows the size of both files and the difference between them.

Note that GCC by default does dynamic linking, so printf() is dynamically linked in above program. Refer this, [this](#) and [this](#) for more details on static and dynamic linkings.

This article is contributed by [Vikash Kumar](#). Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

My Personal Notes [arrow_drop_up](#)

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- Program error signals
- Reasons for a C++ program crash
- 8085 program to add 2-BCD numbers
- 8086 program to add two 8 bit BCD numbers
- 8085 program to add two 8 bit numbers
- 8085 program to add two 16 bit numbers
- Writing first C++ program : Hello World example
- 8086 program to add two 16 bit BCD numbers with carry
- 8085 program to find the sum of a series
- C program to demonstrate fork() and pipe()
- 8086 program for selection sort
- 8086 program to find the min value in a given array
- 8085 program for bubble sort
- 8086 program to subtract two 16 bit BCD numbers
- 8085 program to reverse 16 bit number

Improved By : akshaychaudhari2

Article Tags :

C Quiz
C Basics
system-programming

thumb_up
112

To-do Done
2.3

Based on 173 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) C Quiz - 112 | Question 5

Next

[last_page](#) C | C Quiz - 113 | Question 1

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Weekend Geeks Classes Weekdays

JAVA APP DEVELOPMENT COMPETITIVE PROGRAMMING Java Backend

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS

Most popular articles

[Must Do Coding Questions for Companies like Amazon, Microsoft, Adobe, ...](#)

[Must Do Coding Questions Company-wise](#)

[10 Projects That Every Developer Should Lay Their Hands-On](#)

[C Interview Questions](#)

[Map of Vectors in C++ STL with Examples](#)

Most Visited Articles

[Program to print all prime numbers between 1 to n](#)

[Median of sliding window in an array](#)

[Integer.MAX_VALUE and Integer.MIN_VALUE in Java with Examples](#)

[Top 10 Projects For Beginners To Practice HTML and CSS Skills](#)

[Shortest Path Faster Algorithm](#)

Benefits of C language over other programming languages

C is a middle-level programming language developed by Dennis Ritchie during the early 1970s while working at AT&T Bell Labs in the USA. The objective of its development was in the context of the re-design of the UNIX operating system to enable it to be used on multiple computers.

Earlier the language B was now used for improving the UNIX system. Being a high-level language, B allowed much faster production of code than in assembly language. Still, B suffered from drawbacks as it did not understand data-types and did not provide the use of "structures".

These drawbacks became the driving force for Ritchie for development of a new programming language called C. He kept most of language B's syntax and added data-types and many other required changes. Eventually, C was developed during 1971-73, containing both high-level functionality and the detailed features required to program an operating system. Hence, many of the UNIX components including UNIX kernel itself were eventually rewritten in C.

- As a middle-level language, C combines the features of both high-level and low-level languages. It can be used for low-level programming, such as scripting for drivers and kernels and it also supports functions of high-level programming languages, such as scripting for software applications etc.
- C is a structured programming language which allows a complex program to be broken into simpler programs called functions. It also allows free movement of data across these functions.
- Various features of C including direct access to machine level hardware APIs, the presence of C compilers, deterministic resource use and dynamic memory allocation make C language an optimum choice for scripting applications and drivers of embedded systems.
- C language is case-sensitive which means lowercase and uppercase letters are treated differently.
- C is highly portable and is used for scripting system applications which form a major part of Windows, UNIX, and Linux operating system.
- C is a general-purpose programming language and can efficiently work on enterprise applications, games, graphics, and applications requiring calculations, etc.
- C language has a rich library which provides a number of built-in functions. It also offers dynamic memory allocation.
- C implements algorithms and data structures swiftly, facilitating faster computations in programs. This has enabled the use of C in applications requiring higher degrees of calculations like MATLAB and Mathematica.

Riding on these advantages, C became dominant and spread quickly beyond Bell Labs replacing many well-known languages of that time, such as ALGOL, B, PL/I, FORTRAN, etc. C language has become available on a very wide range of platforms, from embedded microcontrollers to supercomputers.

The C language has formed the basis for many languages including C++, C-, C#, Objective-C, BitC, C-shell, csh, D, Java, JavaScript, Go, Rust, Julia, Limbo, LPC, PHP, Python, Perl, Seed7, Vala, Verilog and many more other languages are there.

References:

- Wikipedia
- Invensis

This article is contributed by Shubham Bansal. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer Placement 100 for details

My Personal Notes 

Recommended Posts:

- C Programming Language Standard
- A C Programming Language Puzzle
- getchar_unlocked() - faster input in C/C++ for Competitive Programming
- Socket Programming in C/C++: Handling multiple clients on server without multi threading
- Interesting Facts in C Programming
- Creating a Rainbow using Graphics Programming in C
- Commonly Asked C Programming Interview Questions | Set 1
- Commonly Asked C Programming Interview Questions | Set 2
- C Language Introduction
- Arrays in C Language | Set 2 (Properties)
- Convert C/C++ code to assembly language
- Signals in C language
- Constants vs Variables in C language
- Why is C considered faster than other languages ?
- Difference between %d and %i format specifier in C language

Improved By : architg98, RiturajShakti

Article Tags :

C

Practice Tags :

C



45

To-do Done
1.2

Based on 60 vote(s)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) C program to print characters without using format specifiers

Next

[last_page](#) C/C++ Tricky Programs

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
Predefined Macros in C with Examples
How to call function within function in C or C++
C program to print odd line contents of a File followed by even line content
Introduction to the C99 Programming Language : Part II

More related articles in C
C program to find square root of a given number
Format Specifiers in different Programming Languages
Problems with Floating point numbers and how to compare them correctly?
Features of C Programming Language
Pointer Expressions in C with Examples

Program error signals

Signals in computers are a way of communication between the process and the OS. When a running program undergoes some serious error then the OS sends a signal to the process and the process further may not execute. Some processes may have a signal handler which does some important tasks before the process leaves the CPU.

Signal and interrupt are basically same but a small distinction exists i.e. interrupts are generated by the processor and handled by the kernel but signals are generated by the kernel and handled by the process. Error signals generally cause termination of the program and a core dump file is created named core, which stores the state of the process at the moment of termination. This file can be investigated using debugger to know the cause of program termination.

Error signals:

■ SIGFPE -

This error signal denotes some arithmetic error that occurred like division by zero, floating point error. If a program stores integer data in a location which is then used as a floating-point operation, this causes an "invalid operation" exception as the processor cannot recognize the data as a floating-point value. But this signal does not specify the type of floating point error.

■ SIGILL -

This signal denotes illegal instruction. When a garbage instruction or instruction which a program has no privilege to execute, is executed then this signal is generated. C does not produce illegal instruction so there is no chance of facing such error signal, as the probable cause may be that the object file may be corrupted. This signal is also generated when stack overflow occurs.

■ SIGSEGV -

The signal is generated when process tries to access memory location not allocated to it, like de-referencing a wild pointer which leads to "segmentation fault". The signal is only generated when a program goes far from its memory space so that it can be detected by the memory protection mechanism.

The name is an abbreviation for "segmentation violation".

■ SIGBUS -

The name is an abbreviation for "Bus error". This signal is also produced when an invalid memory is accessed. It may seem to be same like SIGSEGV but in SIGSEGV, the memory location referenced is valid but in case of SIGBUS, memory referenced does not exist i.e. de-referencing a memory location out of memory space.

■ SIGABRT -

If an error itself is detected by the program then this signal is generated using call to abort(). This signal is also used by standard library to report an internal error. assert() function in c++ also uses abort() to generate this signal.

■ SIGSYS -

This signal is sent to process when an invalid argument is passed to a system call.

■ SIGTRAP -

This signal is sent to process when an exception is occurred. This is requested by the debugger to get informed. For example, if a variable changes its value then this will trigger it.

Refer for - [Segmentation Fault \(SIGSEGV\) vs Bus Error \(SIGBUS\)](#)

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes

Recommended Posts:

- [Signals in C language](#)
- [Communication between two process using signals in C](#)
- [Control Signals in 8155 Microprocessor](#)
- [Else without IF and L-Value Required Error in C](#)
- [How to find Segmentation Error in C & C++ ? \(Using GDB\)](#)
- [Error Handling in C programs](#)
- [Segmentation Fault \(SIGSEGV\) vs Bus Error \(SIGBUS\)](#)
- [C program to print a string without any quote \(single or double\) in the program](#)
- [Hello World Program : First program while learning Programming](#)
- [C program to detect tokens in a C program](#)
- [Output of C Program | Set 29](#)
- [How does a C program executes?](#)
- [How to compile 32-bit program on 64-bit gcc in C and C++](#)
- [Program to compute Log n](#)
- [Quine - A self-reproducing program](#)



Himanshu_Singh_Bisht

Check out this Author's contributed articles.

If you like GeeksforGeeks and would like to contribute, you can also write an article using [contribute.geeksforgeeks.org](#) or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please Improve this article if you find anything incorrect by clicking on the "Improve Article" button below.

Improved By : Neeraj Kumar 15

Article Tags :

C
C Basics
system-programming

Practice Tags :

C

thumb_up
32

To-do Done
2.6

Based on 21 vote(s)

Basic **Easy** **Medium** **Hard** **Expert**

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) [int \(1 sign bit + 31 data bits\) keyword in C](#)

Next

[last_page](#) [Different ways to initialize a variable in C/C++](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Weekend Geeks Classes Weekdays

JAVA APP DEVELOPMENT COMPETITIVE PROGRAMMING Java Backend

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS



Most popular in C

[Print all possible combinations of the string by replacing '\\$' with any other digit from the string](#)

[How to call function within function in C or C++](#)

[Format specifiers in different Programming Languages](#)

[Predefined Macros in C with Examples](#)

[C program to print odd line contents of a File followed by even line content](#)

More related articles in C

[C program to find square root of a given number](#)

[Introduction to the C99 Programming Language : Part II](#)

[Problem in comparing Floating point numbers and how to compare them correctly?](#)

[Features of C Programming Language](#)

[<cmath> float.h C/C++ with Examples](#)

Escape Sequences in C

In C programming language, there are 256 numbers of characters in character set. The entire character set is divided into 2 parts i.e. the ASCII characters set and the extended ASCII characters set. But apart from that, some other characters are also there which are not the part of any characters set, known as ESCAPE characters.

[List of Escape Sequences](#)

\a	<i>Alarm or Beep</i>
\b	<i>Backspace</i>
\f	<i>Form Feed</i>
\n	<i>New Line</i>
\r	<i>Carriage Return</i>
\t	<i>Tab (Horizontal)</i>
\v	<i>Vertical Tab</i>
\\\	<i>Backslash</i>
\'	<i>Single Quote</i>
\"	<i>Double Quote</i>
\?	<i>Question Mark</i>
\ooo	<i>octal number</i>
\xhh	<i>hexadecimal number</i>
\0	<i>Null</i>



Some coding examples of escape characters

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// C program to illustrate
// \a escape sequence
#include <stdio.h>
int main(void)
{
    printf("My mobile number "
           "is 7\08\07\03\09\02\03\04\00\08\0");
    return (0);
}
```

Chevron_right

filter_none

Output:

My mobile number is 7873923408.

filter_none

edit

close

play_arrow

link

brightness_4

code

// C program to illustrate

// \b escape sequence

#include <stdio.h>

int main(void)

{

// \b - backspace character transfers

```
// the cursor one character back with
// or without deleting on different
// compilers.
printf("Hello Geeks\b\b\b\bF");
return (0);
}
```

[chevron_right](#)

[filter_none](#)

Output:

The output is dependent upon compiler.

[filter_none](#)

[edit](#)

[close](#)

[play_arrow](#)

[link](#)

[brightness_4](#)

[code](#)

```
// C program to illustrate
// \n escape sequence
#include <stdio.h>
int main(void)
{
    // Here we are using \n, which
    // is a new line character.
    printf("Hello\n");
    printf("GeeksforGeeks");
    return (0);
}
```

[chevron_right](#)

[filter_none](#)

Output:

Hello
GeeksforGeeks

[filter_none](#)

[edit](#)

[close](#)

[play_arrow](#)

[link](#)

[brightness_4](#)

[code](#)

```
// C program to illustrate
// \t escape sequence
#include <stdio.h>
int
main(void)
{
    // Here we are using \t, which is
    // a horizontal tab character.
    // It will provide a tab space
    // between two words.
    printf("Hello \t GFG");
    return (0);
}
```

[chevron_right](#)

[filter_none](#)

Output:

Hello GFG

The escape sequence "\t" is very frequently used in loop based pattern printing programs.

[filter_none](#)

[edit](#)

[close](#)

[play_arrow](#)

[link](#)

[brightness_4](#)

[code](#)

```
// C program to illustrate
// \v escape sequence
#include <stdio.h>
int main(void)
{
    // Here we are using \v, which
    // is vertical tab character.
    printf("Hello friends");

    printf("\v Welcome to GFG");

    return (0);
}
```

[chevron_right](#)

[filter_none](#)

Output:

Hello Friends
Welcome to GFG

filter_none
edit
close
play_arrow
link
brightness_4
code

```
// C program to illustrate \r escape
// sequence
#include <stdio.h>
int main(void)
{
    // Here we are using \r, which
    // is carriage return character.
    printf("Hello fri \r ends");
    return (0);
}
```

chevron_right

filter_none

Output: (Depend upon compiler)

ends

filter_none

edit

close

play_arrow

link

brightness_4

code

```
// C program to illustrate \\(Backslash)
// escape sequence to print backslash.
#include <stdio.h>
int main(void)
{
    // Here we are using \,
    // It contains two escape sequence
    // means \ and \n.
    printf("Hello\\GFG");
    return (0);
}
```

chevron_right

filter_none

Output: (Depend upon compiler)

Hello\GFG

Explanation : It contains two escape sequence means it after printing the \ the compiler read the next \ as as new line character i.e. \n, which print the GFG in the next line

filter_none
edit
close
play_arrow
link
brightness_4
code

```
// C program to illustrate \' escape
// sequence/ and \" escape sequence to
// print single quote and double quote.
#include <stdio.h>
int main(void)
{
    printf("\' Hello Geeks\n");
    printf("\" Hello Geeks");
    return 0;
}
```

chevron_right

filter_none

Output:

' Hello Geeks
" Hello Geeks

filter_none

edit

close

play_arrow

link

brightness_4

code

```
// C program to illustrate
// \? escape sequence
#include <stdio.h>
int main(void)
{
    // Here we are using \?, which is
    // used for the presentation of trigraph
    // in the early of C programming. But
    // now we don't have any use of it.
```

```

printf("\?!\n");
return 0;
}
chevron_right
filter_none
Output:
??!
filter_none
edit
close
play_arrow
link
brightness_4
code

// C program to illustrate \000 escape sequence
#include <stdio.h>
int main(void)
{
    // we are using \000 escape sequence, here
    // each 0 in "000" is one to three octal
    // digits(0....7).
    char* s = "A\0725";
    printf("%s", s);
    return 0;
}
chevron_right
filter_none

```

Explanation : Here 000 is one to three octal digits(0....7) means there must be atleast one octal digit after \ and maximum three. Here 072 is the octal notation, first it is converted to decimal notation that is the ASCII value of char ':'. At the place of \072 there is : and the output is A:5.

```

filter_none
edit
close
play_arrow
link
brightness_4
code

// C program to illustrate \XHH escape
// sequence
#include <stdio.h>
int main(void)
{
    // We are using \xhh escape sequence.
    // Here hh is one or more hexadecimal
    // digits(0....9, a...f, A...F).
    char* s = "B\x4a";
    printf("%s", s);
    return 0;
}
chevron_right
filter_none

```

Explanation : Here hh is one or more hexadecimal digits(0....9, a...f, A...F). There can be more than one hexadecimal number after \x. Here, '\x4a' is a hexadecimal number and it is a single char. Firstly it will get converted into decimal notation and it is the ASCII value of char 'J'. Therefore at the place of \x4a, we can write J. So the output is Bj.

This article is contributed by **Bishal Kumar Dubey**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes

Add your personal notes
here! (max 5000 chars)

Recommended Posts:

- [Check if two same sub-sequences exist in a string or not](#)
- [Check if the number is a Prime power number](#)
- [Check if there exists a permutation of given string which doesn't contain any monotonous substring](#)
- [Count of array elements which is smaller than both its adjacent elements](#)
- [Check if a number is Full Fibonacci or not](#)
- [Check whether two numbers are in silver ratio](#)
- [Find N fractions that sum upto a given fraction N/D](#)
- [Count of decreasing pairs formed from numbers 1 to N](#)
- [Count the minimum steps to reach 0 from the given integer N](#)
- [Format specifiers in different Programming Languages](#)
- [Construct an Array of size N whose sum of cube of all elements is a perfect square](#)
- [Find N distinct integers with sum N](#)
- [Find the middle digit of a given Number](#)
- [Program to find absolute value of a given number](#)

Article Tags :

C
 School Programming
 C Basics
 Practice Tags :
 C

thumb_up
46

To-do Done
1.5

Based on 45 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

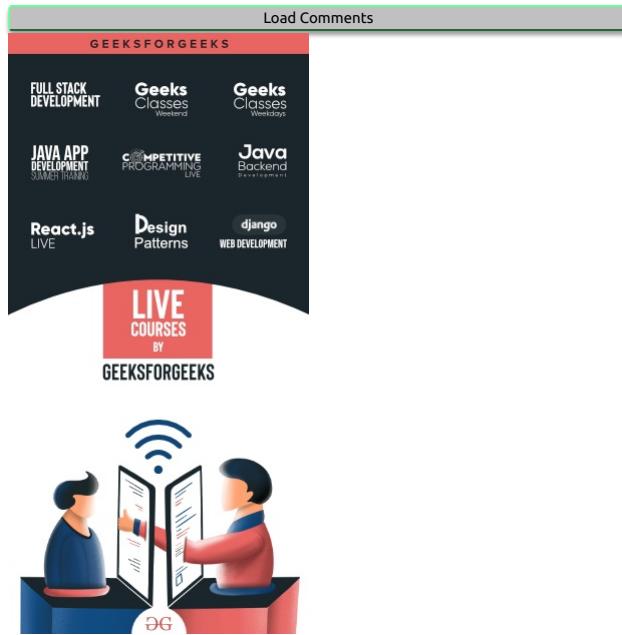
Previous

[first_page](#) Rounding Floating Point Number To two Decimal Places in C and C++

Next

[last_page](#) Octal numbers in c

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Line Splicing in C/C++

While writing a program, sometimes we give comment about the working of the code in the comment section with the help of single/double comment line. But we had never thought that if at the end of this comment line if we use \ (backslash) character then what will happen?

The answer of the above question is line Splicing. Lines terminated by a \ are spliced together with the next line very early in the process of translation. [\\$2.2 Phases of translation](#).

Actually whenever at the end of the comment line if we use \ (backslash) character then it deletes the backslash character and the preceding next line of code only from the entire program or we can say that the ending \ (backslash) **makes the new line also as a comment for the compiler**.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// C program to illustrate the concept of Line splicing.
#include <stdio.h>
int main()
{
    // Line Splicing\
    printf("Hello GFG\n");
    printf("welcome");
    return (0);
}
chevron_right
filter_none
Output:
```

Welcome

Explanation: In the above program as we can see when we use the \ (backslash) character at the end of comment line. Then the next line of code is treated as comment in the program and the output is

welcome.

This article is contributed by **Bishal Kumar Dubey**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes

Add your personal notes here! (max 5000 chars)

Recommended Posts:

- Equation of straight line passing through a given point which bisects it into two equal line segments
- C program to print odd line contents of a File followed by even line content
- Program to delete a line given the line number from a file
- Draw a line in C++ graphics
- Command line arguments in C/C++
- Command line arguments example in C
- LEX program to add line numbers to a given file
- Swap two variables in one line in C/C++, Python, PHP and Java
- Write one line functions for strcat() and strcmp()
- getopt() function in C to parse command line arguments
- Write a one line C function to round floating point numbers
- Rust vs C++: Will Rust Replace C++ in Future ?
- Priority queue of pairs in C++ with ordering by first and second element
- Implementation of lower_bound() and upper_bound() in Vector of Pairs in C++

Article Tags :

C

C++

Practice Tags :

C

CPP



52

To-do Done
1.4

Based on 60 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) [getppid\(\)](#) and [getpid\(\)](#) in Linux

Next

[last_page](#) Core Dump (Segmentation fault) in C/C++

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Weekend Geeks Classes Weekdays

JAVA APP DEVELOPMENT COMPETITIVE PROGRAMMING DAY Java Backend

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
Format specifiers in different Programming Languages
C program to find square root of a given number
Predefined Macros in C with Examples
How to call function within function in C or C++

Most visited in C++
Vector of Vectors in C++ STL with Examples
C++ Circular Queue
Which C++ libraries are useful for competitive programming?
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

C/C++ Tokens

A token is the smallest element of a program that is meaningful to the compiler. Tokens can be classified as follows:

1. Keywords
2. Identifiers
3. Constants
4. Strings

- Special Symbols
- Operators

1. Keyword: Keywords are pre-defined or reserved words in a programming language. Each keyword is meant to perform a specific function in a program. Since keywords are referred names for a compiler, they can't be used as variable names because by doing so, we are trying to assign a new meaning to the keyword which is not allowed. You cannot redefine keywords. However, you can specify text to be substituted for keywords before compilation by using C/C++ preprocessor directives. C language supports **32** keywords which are given below:

```
auto      double    int       struct
break     else      long      switch
case      enum      register  typedef
char      extern   return   union
const     float     short    unsigned
continue  for      signed   void
default   goto    sizeof   volatile
do       if       static   while
```

While in C++ there are **31** additional keywords other than C Keywords they are:

```
asm      bool      catch    class
const_cast delete   dynamic_cast explicit
export   false     friend   inline
mutable  namespace new      operator
private  protected public   reinterpret_cast
static_cast template this    throw
true     try      typeid  typename
using    virtual   wchar_t
```

- Identifiers:** Identifiers are used as the general terminology for naming of variables, functions and arrays. These are user defined names consisting of arbitrarily long sequence of letters and digits with either a letter or the underscore(_) as a first character. Identifier names must differ in spelling and case from any keywords. You cannot use keywords as identifiers; they are reserved for special use. Once declared, you can use the identifier in later program statements to refer to the associated value. A special kind of identifier, called a statement label, can be used in goto statements.

There are certain rules that should be followed while naming c identifiers:

- They must begin with a letter or underscore(_).
- They must consist of only letters, digits, or underscore. No other special character is allowed.
- It should not be a keyword.
- It must not contain white space.
- It should be up to 31 characters long as only first 31 characters are significant.

Some examples of c identifiers:

NAME	REMARK
_A9	Valid
Temp.var	Invalid as it contains special character other than the underscore
void	Invalid as it is a keyword

C program:

```
void main()
{
    int a = 10;
}
```

In the above program there are **2** identifiers:

- main:** method name.
- a:** variable name.

- Constants:** Constants are also like normal variables. But, only difference is, their values can not be modified by the program once they are defined. Constants refer to fixed values. They are also called as literals.

Constants may belong to any of the data type. **Syntax:**

```
const data_type variable_name; (or) const data_type *variable_name;
```

Types of Constants:

- Integer constants - Example: 0, 1, 1218, 12482
- Real or Floating point constants - Example: 0.0, 1203.03, 30486.184
- Octal & Hexadecimal constants - Example: octal: (013)₈ = (11)₁₀, Hexadecimal: (013)₁₆ = (19)₁₀
- Character constants -Example: 'a', 'A', 'z'
- String constants -Example: "GeeksforGeeks"

- Strings:** Strings are nothing but an array of characters ended with a null character ('\0'). This null character indicates the end of the string. Strings are always enclosed in double quotes. Whereas, a character is enclosed in single quotes in C and C++. **Declarations for String:**

- char string[20] = {'g', 'e', 'e', 'k', 's', ' ', 'f', 'o', 'r', 'g', 'e', 'e', 'k', '\0';}
- char string[20] = "geeksforgeeks";
- char string [] = "geeksforgeeks";

Difference between above declarations are:

- when we declare char as "string[20]", 20 bytes of memory space is allocated for holding the string value.
- When we declare char as "string()", memory space will be allocated as per the requirement during execution of the program.

- Special Symbols:** The following special symbols are used in C having some special meaning and thus, cannot be used for some other purpose. [] () {} ; *= #

- Brackets[]:** Opening and closing brackets are used as array element reference. These indicate single and multidimensional subscripts.
- Parentheses():** These special symbols are used to indicate function calls and function parameters.
- Braces{}:** These opening and ending curly braces marks the start and end of a block of code containing more than one executable statement.
- comma (,):** It is used to separate more than one statements like for separating parameters in function calls.
- semi colon :** It is an operator that essentially invokes something called an initialization list.
- asterick (*):** It is used to create pointer variable.
- assignment operator:** It is used to assign values.
- pre processor(#):** The preprocessor is a macro processor that is used automatically by the compiler to transform your program before actual compilation.

- Operators:** Operators are symbols that triggers an action when applied to C variables and other objects. The data items on which operators act upon are called operands.

Depending on the number of operands that an operator can act upon, operators can be classified as follows:

- Unary Operators:** Those operators that require only single operand to act upon are known as unary operators. For Example increment and decrement operators

- Binary Operators:** Those operators that require two operands to act upon are called binary operators. **Binary operators are classified into :**

- Arithmetic operators
- Relational Operators
- Logical Operators
- Assignment Operators
- Conditional Operators
- Bitwise Operators

Ternary Operators: These operators requires three operands to act upon. For Example Conditional operator(?:).

For more information about operators [click](#)

This article is contributed by **I.HARISH KUMAR**. If you like GeeksforGeeks and would like to contribute, you can also write an article using [contribute.geeksforgeeks.org](#) or mail your article to [contribute@geeksforgeeks.org](#). See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes [arrow_drop_up](#)

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- C program to detect tokens in a C program
- std::basic_istream::ignore in C++ with Examples
- std::basic_istream::getline in C++ with Examples
- Format specifiers in different Programming Languages
- C program to find square root of a given number
- C program to print odd line contents of a File followed by even line content
- std::is_heap() in C++ with Examples
- std::basic_istream::gcount() in C++ with Examples
- new vs malloc() and free() vs delete in C++
- std::string::rfind in C++ with Examples
- Sum of factorials of Prime numbers in a Linked list
- Sum of all Palindrome Numbers present in a Linked list
- Generate an array of given size with equal count and sum of odd and even numbers
- Namespaces in C++ | Set 4 (Overloading, and Exchange of Data in different Namespaces)

Improved By : ashutoshjoshi1995

Article Tags :

C
C++
CBSE - Class 11
school-programming
Practice Tags :
C
CPP

50

To-do Done
1.3

Based on 56 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) Difference between strlen() and sizeof() for string in C

Next

[last_page](#) C Library math.h functions

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Weekend Geeks Classes Weekdays

Java APP DEVELOPMENT COMPETITIVE PROGRAMMING LIVE Java Backend Development

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS

Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
Predefined Macros in C with Examples
C programs to print odd line contents of a File followed by even line content
Introduction to the C99 Programming Language : Part II
C program to find square root of a given number

Most visited in C++
Vector of Vectors in C++ STL with Examples
C++ Tutorial
Which C++ libraries are useful for competitive programming?
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

Variables and Keywords in C

A **variable** in simple terms is a storage place which has some memory allocated to it. Basically, a variable used to store some form of data. Different types of variables require different amounts of memory, and have some specific set of operations which can be applied on them.

Variable Declaration:

A typical variable declaration is of the form:

```
type variable_name;  
or for multiple variables:  
type variable1_name, variable2_name, variable3_name;
```

A variable name can consist of alphabets (both upper and lower case), numbers and the underscore '_' character. However, the name must not start with a number.

Difference b/w variable declaration and definition

Variable declaration refers to the part where a variable is first declared or introduced before its first use. Variable definition is the part where the variable is assigned a memory location and a value. Most of the times, variable declaration and definition are done together.

See the following C program for better clarification:

```
filter_none
edit
close

play_arrow
link
brightness_4
code

#include <stdio.h>
int main()
{
    // declaration and definition of variable 'a123'
    char a123 = 'a';

    // This is also both declaration and definition as 'b' is allocated
    // memory and assigned some garbage value.
    float b;

    // multiple declarations and definitions
    int _c, _d45, e;

    // Let us print a variable
    printf("%c \n", a123);

    return 0;
}
```

chevron_right

filter_none

Output:

```
a
```

Is it possible to have separate declaration and definition?

It is possible in case of [extern variables](#) and [functions](#). See question 1 of [this](#) for more details.

Keywords are specific reserved words in C each of which has a specific feature associated with it. Almost all of the words which help us use the functionality of the C language are included in the list of keywords. So you can imagine that the list of keywords is not going to be a small one!

There are a total of 32 keywords in C:

```
auto      break     case      char      const      continue
default   do        double    else      enum       extern
float     for       goto     if        int        long
register return   short    signed   sizeof    static
struct    switch   typedef  union    unsigned  void
volatile while
```

Most of these keywords have already been discussed in the various sub-sections of the [C language](#), like Data Types, Storage Classes, Control Statements, Functions etc.

Let us discuss some of the other keywords which allow us to use the basic functionality of C:

const: const can be used to declare constant variables. Constant variables are variables which, when initialized, can't change their value. Or in other words, the value assigned to them cannot be modified further down in the program.

Syntax:

```
const data_type var_name = var_value;
```

Note: Constant variables must be initialized during their declaration. const keyword is also used with pointers. Please refer the [const qualifier in C](#) for understanding the same.

extern: extern simply tells us that the variable is defined elsewhere and not within the same block where it is used. Basically, the value is assigned to it in a different block and this can be overwritten/changed in a different block as well. So an extern variable is nothing but a global variable initialized with a legal value where it is declared in order to be used elsewhere. It can be accessed within any function/block. Also, a normal global variable can be made extern as well by placing the 'extern' keyword before its declaration/definition in any function/block. This basically signifies that we are not initializing a new variable but instead we are using/accessing the global variable only. The main purpose of using extern variables is that they can be accessed between two different files which are part of a large program.

Syntax:

```
extern data_type var_name = var_value;
```

static: static keyword is used to declare static variables, which are popularly used while writing programs in C language. Static variables have a property of preserving their value even after they are out of their scope! Hence, static variables preserve the value of their last use in their scope. So we can say that they are initialized only once and exist till the termination of the program. Thus, no new memory is allocated because they are not re-declared. Their scope is local to the function to which they were defined. Global static variables can be accessed anywhere within that file as their scope is local to the file. By default, they are assigned the value 0 by the compiler.

Syntax:

```
static data_type var_name = var_value;
```

void: void is a special data type. But what makes it so special? void, as it literally means, is an empty data type. It means it has nothing or it holds no value. For example, when it is used as the return data type for a function it simply represents that the function returns no value. Similarly, when it's added to a function heading, it represents that the function takes no arguments.

Note: void also has a significant use with pointers. Please refer to the [void pointer in C](#) for understanding the same.

typedef: typedef is used to give a new name to an already existing or even a custom data type (like a structure). It comes in very handy at times, for example in a case when the name of the structure defined by you is very long or you just need a short-hand notation of a pre-existing data type.

Let's implement the keywords which we have discussed above. Take a look at the following code which is a working example to demonstrate these keywords:

```
filter_none
edit
close

play_arrow
link
brightness_4
code

#include <stdio.h>
```

```

// declaring and initializing an extern variable
extern int x = 9;

// declaring and initializing a global variable
// simply int z; would have initialized z with
// the default value of a global variable which is 0
int z=10;

// using typedef to give a short name to long long int
// very convenient to use now due to the short name
typedef long long int LL;

// function which prints square of a no. and which has void as its
// return data type
void calSquare(int arg)
{
    printf("The square of %d is %d\n",arg,arg*arg);
}

// Here void means function main takes no parameters
int main(void)
{
    // declaring a constant variable, its value cannot be modified
    const int a = 32;

    // declaring a char variable
    char b = 'G';

    // telling the compiler that the variable z is an extern variable
    // and has been defined elsewhere (above the main function)
    extern int z;

    LL c = 1000000;

    printf("Hello World!\n");

    // printing the above variables
    printf("This is the value of the constant variable 'a': %d\n",a);
    printf("'b' is a char variable. Its value is %c\n",b);
    printf("'c' is a long long int variable. Its value is %lld\n",c);
    printf("These are the values of the extern variables 'x' and 'z'"
    " respectively: %d and %d\n",x,z);

    // value of extern variable x modified
    x=2;

    // value of extern variable z modified
    z=5;

    // printing the modified values of extern variables 'x' and 'z'
    printf("These are the modified values of the extern variables"
    "'x' and 'z' respectively: %d and %d\n",x,z);

    // using a static variable
    printf("The value of static variable 'y' is NOT initialized to 5 after the "
    "first iteration! See for yourself :)\n");

    while (x > 0)
    {
        static int y = 5;
        y++;
        // printing value at each iteration
        printf("The value of y is %d\n",y);
        x--;
    }

    // print square of 5
    calSquare(5);

    printf("Bye! See you soon. :)\n");

    return 0;
}

```

chevron_right

filter_none

Output:

```

Hello World
This is the value of the constant variable 'a': 32
'b' is a char variable. Its value is G
'c' is a long long int variable. Its value is 1000000
These are the values of the extern variables 'x' and 'z' respectively: 9 and 10
These are the modified values of the extern variables 'x' and 'z' respectively: 2 and 5
The value of static variable 'y' is NOT initialized to 5 after the first iteration! See for yourself :)
The value of y is 6
The value of y is 7
The square of 5 is 25
Bye! See you soon. :)
```

This article is contributed by Ayush Jaggi. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes

Add your personal notes here! (max 5000 chars)

Recommended Posts:

- Difference between Static variables and Register variables in C
- Python | Set 4 (Dictionary, Keywords in Python)
- Variables in C++
- Variables in Java
- Static Variables in C
- Scope of Variables In Java
- Initialization of static variables in C
- Can Global Variables be dangerous ?
- Operations on struct variables in C
- Implicit initialization of variables with 0 or 1 in C
- Constants vs Variables in C language
- How will you show memory representation of C variables?
- Initialization of variables sized arrays in C
- Global and Local Variables in Python
- C Program to print environment variables

Improved By : [InathiSirayi, Srinivas Nekkanti](#)

Article Tags :

C
School Programming

C Basics
C-Variable Declaration and Scope

Practice Tags :

C

 52

To-do Done
1.5

Based on 89 vote(s)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

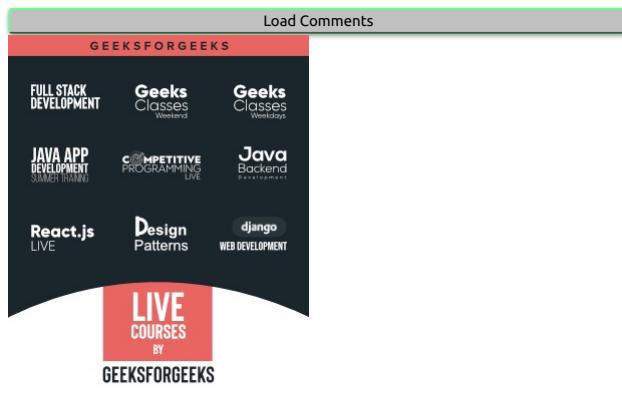
Previous

[first_page](#) sizeof operator in C

Next

[last_page](#) Data Types in C

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
Format specifiers in different Programming Languages
Predefined Macros in C with Examples
C program to find square root of a given number
C program print odd line contents of a file followed by even line content

Most visited in School Programming
Make A, B and C equal by adding total value N to them
Check if a Number is Odd or Even using Bitwise Operators
C program to sort an array in ascending order
Student Information Management System
Calculate the frequency of each word in the given string

How are variables scoped in C – Static or Dynamic?

In C, variables are always **statically** (or **lexically**) **scoped** i.e., binding of a variable can be determined by program text and is independent of the run-time function call stack.

For example, output for the below program is 0, i.e., the value returned by f() is not dependent on who is calling it. f() always returns the value of global variable x.

```
link  
brightness_4  
code  
  
# include <stdio.h>  
  
int x = 0;  
int f()  
{  
    return x;  
}  
int g()  
{  
    int x = 1;  
    return f();  
}  
int main()  
{  
    printf("%d", g());  
    printf("\n");  
    getchar();  
}
```

[chevron_right](#)

[filter_none](#)

References:

http://en.wikipedia.org/wiki/Scope_%28programming%29

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes [arrow_drop_up](#)

Add your personal notes here! (max 5000 chars)

[Save](#)

Recommended Posts:

- Difference between Static variables and Register variables in C
- Static Variables in C
- Initialization of static variables in C
- What are the default values of static variables in C?
- Initialization of global and static variables in C
- Internal static variable vs. External static variable with Examples in C
- Static functions in C
- When are static objects destroyed?
- C++ | Static Keyword | Question 6
- Can static functions be virtual in C++?
- C++ | Static Keyword | Question 5
- C++ | Static Keyword | Question 4
- C++ | Static Keyword | Question 3
- C++ | Static Keyword | Question 2
- C++ | Static Keyword | Question 1

Improved By : [InathiSirayi](#)

Article Tags :

C

GFACTS

C-Variable Declaration and Scope

Practice Tags :

C

[thumb_up](#)
39

To-do Done
1.4

Based on 122 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) G-Fact 7

Next

[last_page](#) G-Fact 8

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

[Load Comments](#)



Most popular in C
 Program to calculate Electricity Bill
 Program to print half Diamond Star pattern
 C/C++ program for calling main() in main()
 C/C++ #include directive with Examples
 Output of C programs | Set 66 (Accessing Memory Locations)

Most visited in GFacts
 Interesting and Cool Tricks in Java
 Interesting Facts About Perl
 How to Install and Configure MongoDB in Ubuntu?

Scope rules in C

Scope of an identifier is the part of the program where the identifier may directly be accessible. In C, all identifiers are **lexically**(or **statically**) **scoped**. C scope rules can be covered under the following two categories.

There are basically 4 scope rules:

SCOPE

File Scope

Block Scope

Function Prototype Scope
 Function scope

MEANING

Scope of a Identifier starts at the beginning of the file and ends at the end of the file. It refers to only those Identifiers that are declared outside of all functions. The Identifiers of File scope are visible all over the file Identifiers having file scope are global

Scope of a Identifier begins at opening of the block / '{' and ends at the end of the block / '}'. Identifiers with block scope are local to their block

Identifiers declared in function prototype are visible within the prototype

Function scope begins at the opening of the function and ends with the closing of it. Function scope is applicable to labels only. A label declared is used as a target to goto statement and both goto and label statement must be in same function

Let's discuss each scope rules with examples.

1. File Scope: These variables are usually declared outside of all of the functions and blocks, at the top of the program and can be accessed from any portion of the program. These are also called the global scope variables as they can be globally accessed.

Example 1:

```
filter_none
edit
close

play_arrow
link
brightness_4
code

#include<stdio.h>

int main()
{
    int x = 10, y = 20;
    {
        // The outer block contains declaration of x and y, so
        // following statement is valid and prints 10 and 20
        printf("x = %d, y = %d\n", x, y);
        {
            // y is declared again, so outer block y is not accessible
            // in this block
            int y = 40;

            // Changes the outer block variable x to 11
            x++;

            // Changes this block's variable y to 41
            y++;

            printf("x = %d, y = %d\n", x, y);
        }

        // This statement accesses only outer block's variables
        printf("x = %d, y = %d\n", x, y);
    }
    return 0;
}
chevron_right
```

filter_none

Output:

```
Before change within main: 5
After change within main: 10
```

Example 2:

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// C program to illustrate
// function prototype scope

#include <stdio.h>

// function prototype scope
//(not part of a function definition)
int Sub(int num1, int num2);

// file scope
int num1;

// Function to subtract
int Sub(int num1, int num2)
{
    return (num1-num2);
}

// Driver method
int main(void)
{
    printf("%d\n", Sub(10,5));
    return 0;
}
```

chevron_right

filter_none

```
filter_none
edit
close
play_arrow
link
brightness_4
code

void func1()
{
    {
        // label in scope even
        // though declared later
        goto label_exec;

label_exec:;
    }
}

// label ignores block scope
goto label_exec;
}
```

chevron_right

filter_none

Note: To restrict access to the current file only, global variables can be marked as static.

2. **Block Scope:** A Block is a set of statements enclosed within left and right braces i.e. '{' and '}' respectively. Blocks may be nested in C(a block may contain other blocks inside it). A variable declared inside a block is accessible in the block and all inner blocks of that block, but not accessible outside the block. Basically these are local to the blocks in which the variables are defined and are not accessible outside.

filter_none

edit

close

play_arrow

link

brightness_4

code

```
int main()
{
    {
        int x = 10;
    }
}
```

```

    // Error: x is not accessible here
    printf("%d", x);
}
return 0;
}
chevron_right
filter_none

```

Output:

```
x = 10, y = 20
x = 11, y = 41
x = 11, y = 20
```

3. **Function Prototype Scope:** These variables range includes within the function parameter list. The scope of the these variables begins right after the declaration in the function prototype and runs to the end of the declarations list. These scope doesnt include the function definition, but just the function prototype.

Example:

```

filter_none
edit
close
play_arrow
link
brightness_4
code

// C program to illustrate scope of variables

#include<stdio.h>

int main()
{
    // Initialization of local variables
    int x = 1, y = 2, z = 3;

    printf("x = %d, y = %d, z = %d\n",
    x, y, z);
    {

        // changing the variables x & y
        int x = 10;
        float y = 20;

        printf("x = %d, y = %f, z = %d\n",
        x, y, z);
        {

            // changing z
            int z = 100;
            printf("x = %d, y = %f, z = %d\n",
            x, y, z);
        }
    }
    return 0;
}
chevron_right
filter_none

```

Output:

```
5
```

4. **Function Scope:** A Function scope begins at the opening of the function and ends with the closing of it. Function scope is applicable to labels only. A label declared is used as a target to go to the statement and both goto and label statement must be in the same function.

Example:

```

filter_none
edit
close
play_arrow
link
brightness_4
code

chevron_right
filter_none

```

Now various questions may arise with respect to the scope of access of variables:

What if the inner block itself has one variable with the same name?

If an inner block declares a variable with the same name as the variable declared by the outer block, then the visibility of the outer block variable ends at the point of the declaration by inner block.

What about functions and parameters passed to functions?

A function itself is a block. Parameters and other local variables of a function follow the same block scope rules.

Can variables of the block be accessed in another subsequent block?

No, a variable declared in a block can only be accessed inside the block and all inner blocks of this block.

For example: the following program produces a compiler error.

```

filter_none
edit
close
play_arrow
link
brightness_4
code

int main()
```

```

{
    int x = 10;
}
{
    // Error: x is not accessible here
    printf("%d", x);
}
return 0;
}
chevron_right
filter_none

```

Error:

```

prog.c: In function 'main':
prog.c:8:15: error: 'x' undeclared (first use in this function)
    printf("%d", x); // Error: x is not accessible here
               ^
prog.c:8:15: note: each undeclared identifier is
reported only once for each function it appears in

```

Example:

```

filter_none
edit
close

play_arrow
link
brightness_4
code

// C program to illustrate scope of variables

#include<stdio.h>

int main()
{
    // Initialization of local variables
    int x = 1, y = 2, z = 3;

    printf("x = %d, y = %d, z = %d\n",
           x, y, z);
    {

        // changing the variables x & y
        int x = 10;
        float y = 20;

        printf("x = %d, y = %f, z = %d\n",
               x, y, z);
        {

            // changing z
            int z = 100;
            printf("x = %d, y = %f, z = %d\n",
                   x, y, z);
        }
    }
    return 0;
}
chevron_right
filter_none

```

Output:

```

x = 1, y = 2, z = 3
x = 10, y = 20.000000, z = 3
x = 10, y = 20.000000, z = 100

```

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up

Add your personal notes
here! (max 5000 chars)

Save

Recommended Posts:

- [Structure Sorting \(By Multiple Rules\) in C++](#)
- [C | Variable Declaration and Scope | Question 6](#)
- [C | Variable Declaration and Scope | Question 1](#)
- [C | Variable Declaration and Scope | Question 8](#)
- [C | Variable Declaration and Scope | Question 7](#)
- [C | Variable Declaration and Scope | Question 5](#)
- [C | Variable Declaration and Scope | Question 2](#)
- [C | Variable Declaration and Scope | Question 3](#)
- [C | Variable Declaration and Scope | Question 4](#)
- [Scope Resolution Operator Versus this pointer in C++?](#)
- [Format specifiers in different Programming Languages](#)
- [C program to find square root of a given number](#)
- [C program to print odd line contents of a File followed by even line content](#)
- [Getting System and Process Information Using C Programming and Shell in Linux](#)

Article Tags :

C
C-Variable Declaration and Scope

Practice Tags :

C



47

To-do Done
1.5

Based on 140 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) A comma operator question

Next

[last_page](#) Complicated declarations in C

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

The screenshot shows the GeeksforGeeks homepage. At the top, there's a navigation bar with 'Load Comments' and the 'GEEKSFORGEEKS' logo. Below the logo are several course categories: 'FULL STACK DEVELOPMENT', 'Geeks Classes' (Weekend), 'Geeks Classes' (Weekdays), 'JAVA APP DEVELOPMENT SUMMER TRAINING', 'COMPETITIVE PROGRAMMING LIVE', 'Java Backend Environment', 'React.js LIVE', 'Design Patterns', and 'django WEB DEVELOPMENT'. A prominent red banner in the center says 'LIVE COURSES BY GEEKSFORGEEKS'. Below the banner, there's an illustration of two people looking at a screen with a Wi-Fi signal. A small 'GG' logo is at the bottom left. A sidebar on the right lists 'Most popular in C' with links to programs like 'Program to calculate Electricity Bill', 'Program to print half Diamond star pattern', 'C program to calling main() in main()', 'C/C++ #include directive with Examples', 'Output of C programs | Set 66 (Accessing Memory Locations)', and more related articles.

How Linkers Resolve Global Symbols Defined at Multiple Places?

At compile time, the compiler exports each global symbol to the assembler as either strong or weak, and the assembler encodes this information implicitly in the symbol table of the relocatable object file.

Functions and initialized global variables get strong symbols. Uninitialized global variables get weak symbols.

For the following example programs, buf, bufp0, main, and swap are strong symbols; bufp1 is a weak symbol.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

/* main.c */
void swap();
int buf[2] = {1, 2};
int main()
{
    swap();
    return 0;
}

/* swap.c */
extern int buf[];

int *bufp0 = &buf[0];
int *bufp1;

void swap()
{
    int temp;

    bufp1 = &buf[1];
```

```
temp = *bufp0;
*bufp0 = *bufp1;
*bufp1 = temp;
}
chevron_right
```

```
filter_none
```

Given this notion of strong and weak symbols, Unix linkers use the following rules for dealing with multiple defined symbols:

Rule 1: Multiple strong symbols (with same variable name) are not allowed.

Rule 2: Given a strong symbol and multiple weak symbols, choose the strong symbol.

Rule 3: Given multiple weak symbols, choose any of the weak symbols.

For example, suppose we attempt to compile and link the following two C modules:

```
filter_none
edit
close

play_arrow

link
brightness_4
code
```

```
/* foo1.c */
int main()
{
    return 0;
}
```

```
/* bar1.c */
int main()
{
    return 0;
}
```

```
chevron_right
```

```
filter_none
```

In this case, the linker will generate an error message because the strong symbol main is defined multiple times (**rule 1**):

```
$ gcc foo1.c bar1.c
/tmp/cca015022.o: In function `main':
/tmp/cca015022.o(.text+0x0): multiple definition of `main'
/tmp/cca015021.o(.text+0x0): first defined here
```

Similarly, the linker will generate an error message for the following modules because the strong symbol x is defined twice (**rule 1**):

```
filter_none
edit
close

play_arrow

link
brightness_4
code
```

```
/* foo2.c */
int x = 15213;
int main()
{
    return 0;
}
```

```
/* bar2.c */
int x = 15213;
void f()
{
}
chevron_right
```

```
filter_none
```

However, if x is uninitialized in one module, then the linker will quietly choose the strong symbol defined in the other (**rule 2**) as is the case in following program:

```
filter_none
edit
close

play_arrow

link
brightness_4
code
```

```
/* foo3.c */
#include <stdio.h>
void f(void);
int x = 15213;
int main()
{
    f();
    printf("x = %d\n", x);
    return 0;
}
```

```
/* bar3.c */
int x;
void f()
{
    x = 15212;
}
```

```
chevron_right
```

filter_none

At run time, function f() changes the value of x from 15213 to 15212, which might come as a unwelcome surprise to the author of function main! Notice that the linker normally gives no indication that it has detected multiple definitions of x.

```
$ gcc -o gfg foo3.c bar3.c
$ ./gfg
x = 15212
```

The same thing can happen if there are two weak definitions of x (**rule 3**):

```
filter_none
edit
close

play_arrow

link
brightness_4
code

/*a.c*/
#include <stdio.h>
void b(void);

int x;
int main()
{
    x = 2016;
    b();
    printf("x = %d ",x);
    return 0;
}
/*b.c*/
#include <stdio.h>

int x;

void b()
{
    x = 2017;
}
```

chevron_right
filter_none

The **application of rules 2 and 3** can introduce some insidious run-time bugs that are incomprehensible to the unwary programmer, especially if the duplicate symbol definitions have different types.
Example : "x" is defined as an int in one module and a double in another.

```
filter_none
edit
close

play_arrow

link
brightness_4
code

/*a.c*/
#include <stdio.h>
void b(void);

int x = 2016;
int y = 2017;
int main()
{
    b();
    printf("x = 0x%x y = 0x%x \n", x, y);
    return 0;
}
/*b.c*/
double x;

void b()
{
    x = -0.0;
}
```

chevron_right
filter_none

Execution:

```
$ gcc a.c b.c -o geeksforgeeks
$ ./geeksforgeeks
x = 0x0 y = 0x80000000
```

This is a subtle and nasty bug, especially because it occurs silently, with no warning from the compilation system, and because it typically manifests itself much later in the execution of the program, far away from where the error occurred. In a large system with hundreds of modules, a bug of this kind is extremely hard to fix, especially because many programmers are not aware of how linkers work. When in doubt, invoke the linker with a flag such as the `gcc -fno-common` flag, which triggers an error if it encounters multiple defined global symbols.

Source : <http://csapp.cs.cmu.edu/public/ch7-preview.pdf>

This article is contributed by **Sahil Rajput**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes *arrow_drop_up*

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- Can Global Variables be dangerous ?
- Redeclaration of global variable in C
- Initialization of global and static variables in C
- Rounding Floating Point Number To two Decimal Places in C and C++
- Function Interposition in C with an example of user defined malloc()
- Can we access global variable if there is a local variable with same name?
- Multiple Inheritance in C++
- How to return multiple values from a function in C or C++?
- Assigning multiple characters in an int in C language
- Structure Sorting (By Multiple Rules) in C++
- How can I return multiple values from a function?
- Lex program to take input from file and remove multiple spaces, lines and tabs
- Socket Programming in C/C++: Handling multiple clients on server without multi threading
- std::tuple, std::pair | Returning multiple values from a function using Tuple and Pair in C++

Improved By : Pankaj_KS

Article Tags :

C
C-Variable Declaration and Scope

Practice Tags :

C

thumb_up
34

To-do Done
3.1

Based on 122 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

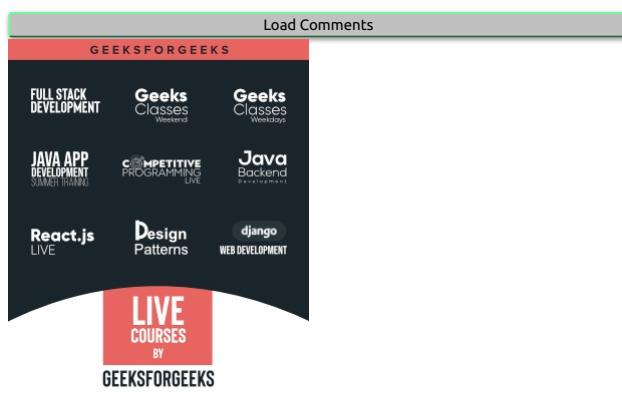
Previous

first_page Initialization of static variables in C

Next

last_page puts() vs printf() for printing a string

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.



Most popular in C
Program to calculate Electricity Bill
Program to print half Diamond star pattern
C/C++ program for calling main() in main()
C/C++ #include directive with Examples
Output of C programs | Set 66 (Accessing Memory Locations)

More related articles in C
Chain of Pointers in C with Examples
Difference between Type Casting and Type Conversion
C program to display month by month calendar for a given year
Print all possible combinations of the string by replacing 's' with any other digit from the string
getch() function in C with Examples

C Variable Declaration and Scope

12

Question 1

Consider the following two C lines

```
int var1;  
extern int var2;
```

Which of the following statements is correct

A Both statements only declare variables, don't define them.

B First statement declares and defines var1, but second statement only declares var2

C Both statements declare define variables var1 and var2

C Variable Declaration and Scope

Discuss it

Question 1 Explanation:

See Understanding "extern" keyword in C

Question 2

Predict the output

```
#include <stdio.h>
int var = 20;
int main()
{
    int var = var;
    printf("%d ", var);
    return 0;
}
```

AGarbage Value

B20

CCompiler Error

C Variable Declaration and Scope

Discuss it

Question 2 Explanation:

First var is declared, then value is assigned to it. As soon as var is declared as a local variable, it hides the global variable var.

Question 3

```
#include <stdio.h>
extern int var;
int main()
{
    var = 10;
    printf("%d ", var);
    return 0;
}
```

ACompiler Error: var is not defined

B20

C0

C Variable Declaration and Scope

Discuss it

Question 3 Explanation:

var is only declared and not defined (no memory allocated for it) Refer: Understanding "extern" keyword in C

Question 4

```
#include <stdio.h>
extern int var = 0;
int main()
{
    var = 10;
    printf("%d ", var);
    return 0;
}
```

A10

BCompiler Error: var is not defined

C0

C Variable Declaration and Scope

Discuss it

Question 4 Explanation:

If a variable is only declared and an initializer is also provided with that declaration, then the memory for that variable will be allocated i.e. that variable will be considered as defined. Refer: Understanding "extern" keyword in C

Question 5

Output?

```
int main()
{
{
    int var = 10;
}
{
    printf("%d", var);
}
return 0;
}
```

A10

BCompiler Error

CGarbage Value

C Variable Declaration and Scope

Discuss it

Question 5 Explanation:

x is not accessible. The curly brackets define a block of scope. Anything declared between curly brackets goes out of scope after the closing bracket.

Question 6

Output?

```
#include <stdio.h>
int main()
{
    int x = 1, y = 2, z = 3;
    printf(" x = %d, y = %d, z = %d n", x, y, z);
    {
        int x = 10;
        float y = 20;
        printf(" x = %d, y = %f, z = %d n", x, y, z);
        {
            int z = 100;
            printf(" x = %d, y = %f, z = %d n", x, y, z);
        }
    }
    return 0;
}
```

A x = 1, y = 2, z = 3
x = 10, y = 20.000000, z = 3
x = 1, y = 2, z = 100

B Compiler Error

C x = 1, y = 2, z = 3
x = 10, y = 20.000000, z = 3
x = 10, y = 20.000000, z = 100

D x = 1, y = 2, z = 3
x = 1, y = 2, z = 3
x = 1, y = 2, z = 3

C Variable Declaration and Scope

Discuss it

Question 6 Explanation:

See Scope rules in C

Question 7

```
int main()
{
    int x = 032;
    printf("%d", x);
    return 0;
}
```

A32

B0

C26

D50

C Variable Declaration and Scope

Discuss it

Question 7 Explanation:

When a constant value starts with 0, it is considered as octal number. Therefore the value of x is $3*8 + 2 = 26$

Question 8

Consider the following C program, which variable has the longest scope?

```
int a;
int main()
{
    int b;
    // ...
    // ...
}
int c;
```

Aa

Bb

Cc

DAll have same scope

C Variable Declaration and Scope

Discuss it

Question 8 Explanation:

a is accessible everywhere. b is limited to main() c is accessible only after its declaration.

Question 9

Consider the following variable declarations and definitions in C

```
i) int var_9 = 1;
ii) int 9_var = 2;
iii) int _ = 3;
```

Choose the correct statement w.r.t. above variables.

ABoth i) and iii) are valid.

BOnly i) is valid.

CBoth i) and ii) are valid.

DAll are valid.

C Variable Declaration and Scope C Quiz - 101

Discuss it

Question 9 Explanation:

In C language, a variable name can consists of letters, digits and underscore i.e. `_`. But a variable name has to start with either letter or underscore. It can't start with a digit. So valid variables are `var_9` and `_` from the above question. Even two back to back underscore i.e. `__` is also a valid variable name. Even `_9` is a valid variable. But `9var` and `9_` are invalid variables in C. This will be caught at the time of compilation itself. That's why the correct answer is A).

Question 10

Find out the correct statement for the following program.

```
#include "stdio.h"

int * gPtr;

int main()
{
    int * lPtr = NULL;

    if(gPtr == lPtr)
    {
        printf("Equal!");
    }
    else
    {
        printf("Not Equal");
    }

    return 0;
}
```

Alt'll always print Equal.

Bit'll always print Not Equal.

CSince gPtr isn't initialized in the program, it'll print sometimes Equal and at other times Not Equal.

C Variable Declaration and Scope C Quiz - 109

Discuss it

Question 10 Explanation:

It should be noted that global variables such `gPtr` (which is a global pointer to `int`) are initialized to ZERO. That's why `gPtr` (which is a global pointer and initialized implicitly) and `lPtr` (which `a` is local pointer and initialized explicitly) would have same value i.e. correct answer is a.

There are 17 questions to complete.

12

My Personal Notes 

Add your personal notes here! (max 5000 chars)

Complicated declarations in C

Most of the times declarations are simple to read, but it is hard to read some declarations which involve pointer to functions. For example, consider the following declaration from "signal.h".

```
filter_none
edit
close

play_arrow

link
brightness_4
code

void (*bsd_signal(int, void (*)(int)))(int);
chevron_right
```

Let us see the steps to read complicated declarations.

1) Convert C declaration to postfix format and read from right to left.

2) To convert expression to postfix, start from innermost parenthesis, If innermost parenthesis is not present then start from declarations name and go right first. When first ending parenthesis encounters then go left. Once whole parenthesis is parsed then come out from parenthesis.

3) Continue until complete declaration has been parsed.

Let us start with simple example. Below examples are from "K & R" book.

```
filter_none
edit
close

play_arrow

link
brightness_4
code
```

```
1) int (*fp) ();
```

```
chevron_right
```

```
filter_none
```

Let us convert above expression to postfix format. For the above example, there is no innermost parenthesis, that's why, we will print declaration name i.e. "fp". Next step is, go to right side of expression, but there is nothing on right side of "fp" to parse, that's why go to left side. On left side we found "", now print "" and come out of parenthesis. We will get postfix expression as below.

```
fp * () int
```

Now read postfix expression from left to right. e.g. fp is pointer to function returning int

Let us see some more examples.

```
filter_none
```

```
edit
```

```
close
```

```
play_arrow
```

```
link
```

```
brightness_4
```

```
code
```

```
2) int (*daytab)[13]
```

```
chevron_right
```

```
filter_none
```

Postfix : daytab * [13] int

Meaning : daytab is pointer to array of 13 integers.

```
filter_none
```

```
edit
```

```
close
```

```
play_arrow
```

```
link
```

```
brightness_4
```

```
code
```

```
3) void (*f[10]) (int, int)
```

```
chevron_right
```

```
filter_none
```

Postfix : f[10] * (int, int) void

Meaning : f is an array of 10 pointer to function(which takes 2 arguments of type int) returning void

```
filter_none
```

```
edit
```

```
close
```

```
play_arrow
```

```
link
```

```
brightness_4
```

```
code
```

```
4) char (*(*x())[1]) ()
```

```
chevron_right
```

```
filter_none
```

Postfix : x () * [] * () char

Meaning : x is a function returning pointer to array of pointers to function returning char

```
filter_none
```

```
edit
```

```
close
```

```
play_arrow
```

```
link
```

```
brightness_4
```

```
code
```

```
5) char (*(*x[3])())[5]
```

```
chevron_right
```

```
filter_none
```

Postfix : x[3] * () * [5] char

Meaning : x is an array of 3 pointers to function returning pointer to array of 5 char's

```
filter_none
```

```
edit
```

```
close
```

```
play_arrow
```

```
link
```

```
brightness_4
```

```
code
```

```
6) int *(*(*arr[5])()) ()
```

```
chevron_right
```

```
filter_none
```

Postfix : arr[5] * () * () * int

Meaning : arr is an array of 5 pointers to functions returning pointer to function returning pointer to integer

```
filter_none
```

```
edit
```

```
close
```

```
play_arrow
```

```
link
```

```
brightness_4
```

```
code
```

```
7) void (*bsd_signal(int sig, void (*func)(int)))(int);
```

```
chevron_right
```

```
filter_none
```

Postfix : bsd_signal(int sig, void(*func)(int)) * (int) void

Meaning : bsd_signal is a function that takes integer & a pointer to a function(that takes integer as argument and returns void) and returns pointer to a function(that take integer as argument and returns void)
This article is compiled by "Narendra Kangalkar" and reviewed by GeeksforGeeks team. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes [arrow_drop_up](#)

Add your personal notes here! (max 5000 chars)

[Save](#)

Recommended Posts:

- Format specifiers in different Programming Languages
- C program to find square root of a given number
- C program to print odd line contents of a File followed by even line content
- Getting System and Process Information Using C Programming and Shell in Linux
- Program to calculate Electricity Bill
- Program to print half Diamond star pattern
- C/C++ program for calling main() in main()
- Print all possible combinations of the string by replacing '\$' with any other digit from the string
- How to use make utility to build C projects?
- How to call function within function in C or C++
- Role of SemiColon in various Programming Languages
- C program to display month by month calendar for a given year
- Difference between Type Casting and Type Conversion
- Pi(pi) in C++ with Examples

Improved By : Karthik, aakash nandi

Article Tags :

C
C-Variable Declaration and Scope
pointer

Practice Tags :

C

61

To-do Done
3.7

Based on 214 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) Scope rules in C

Next

[last_page](#) Initialization of variables sized arrays in C

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Weekend Geeks Classes Weekdays

JAVA APP DEVELOPMENT COMPETITIVE PROGRAMMING DN Java Backend

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS

Most popular in C
Program to calculate Electricity Bill
Program to print half Diamond star pattern
C/C++ program for calling main() in main()
C/C++ #include directive with Examples
Output of C programs | Set 68 (Accessing Memory Locations)

More related articles in C
C and Pointers in C with Examples
Difference between Type Casting and Type Conversion
C program to display month by month calendar for a given year
Print all possible combinations of the string by replacing '\$' with any other digit from the string
getch() function in C with Examples

Redeclaration of global variable in C

Consider the below two programs:

[filter_none](#)

[edit](#)

[close](#)

[play_arrow](#)

[link](#)

[brightness_4](#)

[code](#)

// Program 1

int main()

{

 int x;

 int x = 5;

 printf("%d", x);

 return 0;

}

[chevron_right](#)

[filter_none](#)

Output in C:

`redeclaration of 'x' with no linkage`

[filter_none](#)

[edit](#)

[close](#)

[play_arrow](#)

[link](#)

[brightness_4](#)

[code](#)

// Program 2

int x;

int x = 5;

int main()

{

 printf("%d", x);

 return 0;

}

[chevron_right](#)

[filter_none](#)

Output in C:

5

In C, the first program fails in compilation, but second program works fine. In C++, both programs fail in compilation.

C allows a global variable to be declared again when first declaration doesn't initialize the variable.

The below program fails in both C also as the global variable is initialized in first declaration itself.

[filter_none](#)

[edit](#)

[close](#)

[play_arrow](#)

[link](#)

[brightness_4](#)

[code](#)

int x = 5;

int x = 10;

int main()

{

 printf("%d", x);

 return 0;

}

[chevron_right](#)

[filter_none](#)

Output:

`error: redefinition of 'x'`

This article is contributed by **Abhay Rathi**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes [arrow_drop_up](#)

Add your personal notes
here! (max 5000 chars)

[Save](#)

Recommended Posts:

- [Can we access global variable if there is a local variable with same name?](#)
- [Internal static variable vs. External static variable with Examples in C](#)
- [Can Global Variables be dangerous ?](#)
- [Initialization of global and static variables in C](#)
- [How Linkers Resolve Global Symbols Defined at Multiple Places?](#)
- [How to print a variable name in C?](#)
- [Why variable name does not start with numbers in C ?](#)
- [Variable Length Arrays in C and C++](#)

- Different ways to initialize a variable in C/C++
- How to modify a const variable in C?
- Variable Length Argument in C
- C | Variable Declaration and Scope | Question 4
- C | Variable Declaration and Scope | Question 5
- C | Variable Declaration and Scope | Question 3
- C | Variable Declaration and Scope | Question 1

Article Tags :

C
C-Variable Declaration and Scope
cpp-storage-classes
Practice Tags :
C

thumb_up
38

To-do Done

2

Based on 114 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

first_page How to sort an array of dates in C/C++?

Next

last_page sprintf() in C

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

[Load Comments](#)



FULL STACK DEVELOPMENT

Geeks Classes Weekend

Geeks Classes Weekdays



JAVA APP DEVELOPMENT SUMMER TRAINING



COMPETITIVE PROGRAMMING LIVE



Java Backend Development



React.js LIVE



Design Patterns



django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEEKS



Most popular in C
[Program to calculate Electricity Bill](#)
[Program to print Diamond star pattern](#)
[C/C++ program for calling main\(\) in main\(\)](#)
[C/C++ #include directive with Examples](#)
[Output of C programs | Set 66 \(Accessing Memory Locations\)](#)

More related articles in C
[Difference between Type Casting and Type Conversion](#)
[getch\(\) function in C with Examples](#)
[Chain of Pointers in C with Examples](#)
[Jagged Array or Array of Arrays in C with Examples](#)
[Print all possible combinations of the string by replacing '\\$' with any other digit from the string](#)

Internal Linkage and External Linkage in C

It is often quite hard to distinguish between scope and linkage, and the roles they play. This article focuses on scope and linkage, and how they are used in C language.

Note: All C programs have been compiled on 64 bit GCC 4.9.2. Also, the terms "identifier" and "name" have been used interchangeably in this article.

Definitions

- Scope :** Scope of an identifier is the part of the program where the identifier may directly be accessible. In C, all identifiers are lexically (or statically) scoped.
- Linkage :** Linkage describes how names can or can not refer to the same entity throughout the whole program or one single translation unit.
The above sounds similar to Scope, but it is not so. To understand what the above means, let us dig deeper into the compilation process.
- Translation Unit :** A translation unit is a file containing source code, header files and other dependencies. All of these sources are grouped together in a file for they are used to produce one single executable object. It is important to link the sources together in a meaningful way. For example, the compiler should know that printf definition lies in stdio header file.

In C and C++, a program that consists of multiple source code files is compiled *one at a time*. Until the compilation process, a variable can be described by its scope. It is only when the linking process starts, that linkage property comes into play. Thus, **scope is a property handled by compiler, whereas linkage is a property handled by linker**.

The Linker links the resources together in the *linking* stage of compilation process. The Linker is a program that takes multiple machine code files as input, and produces an executable object code. It resolves symbols (i.e., fetches definition of symbols such as "+" etc..) and arranges objects in address space.

Linkage is a property that describes how variables should be linked by the linker. Should a variable be available for another file to use? Should a variable be used only in the file declared? Both are decided by linkage.

Linkage thus allows you to couple names together on a per file basis, scope determines visibility of those names.

There are 2 types of linkage:

- Internal Linkage:** An identifier implementing internal linkage is not accessible outside the translation unit it is declared in. Any identifier within the unit can access an identifier having internal linkage. It is implemented by the keyword `static`. An internally linked identifier is stored in initialized or uninitialized segment of RAM. (**Note:** `static` also has a meaning in reference to scope, but that is not discussed here).

Some Examples:

filter_none
 edit
 close
 play_arrow
 link
 brightness_4
 code

```
// C code to illustrate Internal Linkage
#include <stdio.h>

static int animals = 8;
const int i = 5;
```

int call_me(void)
 {
 printf("%d %d", i, animals);
 }

chevron_right
 filter_none

The above code implements static linkage on identifier animals. Consider Feed.cpp is located in the same translation unit.

Feed.cpp

filter_none
 edit
 close
 play_arrow
 link
 brightness_4
 code

```
// C code to illustrate Internal Linkage
#include <stdio.h>
```

```
int main()
{
  call_me();
  animals = 2;
  printf("%d", animals);
  return 0;
}
```

chevron_right
 filter_none

On compiling Animals.cpp first and then Feed.cpp, we get

Output : 5 8 2

Now, consider that Feed.cpp is located in a different translation unit. It will compile and run as above only if we use #include "Animals.cpp". Consider Wash.cpp located in a 3rd translation unit.

Wash.cpp

filter_none
 edit
 close
 play_arrow
 link
 brightness_4
 code

```
// C code to illustrate Internal Linkage
#include <stdio.h>
#include "animal.cpp" // note that animal is included.
```

```
int main()
{
  call_me();
  printf("\n having fun washing!");
  animals = 10;
  printf("%d\n", animals);
  return 0;
}
```

chevron_right
 filter_none

On compiling, we get:

**Output : 5 8
having fun washing!
10**

There are 3 translation units (Animals, Feed, Wash) which are using animals code.

This leads us to conclude that each translation unit accesses its own copy of animals. That is why we have animals = 8 for Animals.cpp, animals = 2 for Feed.cpp and animals = 10 for Wash.cpp. A file. This behavior eats up memory and decreases performance.

Another property of internal linkage is that it is **only implemented when the variable has global scope**, and all constants are by default internally linked.

Usage : As we know, an internally linked variable is passed by copy. Thus, if a header file has a function fun1() and the source code in which it is included also has fun1() but with a different definition, then the 2 functions will not clash with each other. Thus, we commonly use internal linkage to hide translation-unit-local helper functions from the global scope. For example, we might include a header file that contains a method to read input from the user, in a file that may describe another method to read input from the user. Both of these functions are independent of each other when linked.

2. **External Linkage:** An identifier implementing external linkage is visible to **every translation unit**. Externally linked identifiers are *shared* between translation units and are considered to be located at the outermost level of the program. In practice, this means that you must define an identifier in a place which is visible to all, such that it has only one visible definition. It is the default linkage for globally scoped variables and functions. Thus, all instances of a particular identifier with external linkage refer to the same identifier in the program. The keyword `extern` implements external linkage.

When we use the keyword `extern`, we tell the linker to look for the definition elsewhere. Thus, the declaration of an externally linked identifier does not take up any space. `Extern` identifiers are generally stored in initialized/uninitialized or text segment of RAM.

Please do go through [Understanding extern keyword in C](#) before proceeding to the following examples.

It is possible to use an `extern` variable in a local scope. This shall further outline the differences between linkage and scope. Consider the following code:

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// C code to illustrate External Linkage
#include <stdio.h>

void foo()
{
    int a;
    extern int b; // line 1
}

void bar()
{
    int c;
    c = b; // error
}

int main()
{
    foo();
    bar();
}
chevron_right
```

filter_none

Error: 'b' was not declared in this scope

Explanation : The variable `b` has local scope in the function `foo()`, even though it is an `extern` variable. Note that compilation takes place before linking; i.e scope is a concept that can be used only during compile phase. After the program is compiled there is no such concept as "scope of variable".

During compilation, scope of `b` is considered. It has local scope in `foo()`. When the compiler sees the `extern` declaration, it trusts that there is a definition of `b` somewhere and lets the linker handle the rest.

However, the same compiler will go through the `bar()` function and try to find variable `b`. Since `b` has been declared `extern`, it has not been given memory yet by the compiler; it does not exist yet. The compiler will let the linker find the definition of `b` in the translation unit, and then the linker will assign `b` the value specified in definition. It is only then that `b` will exist and be assigned memory. However, since there is no declaration given at compile time within the scope of `bar()`, or even in global scope, the compiler complains with the error above.

Given that it is the compiler's job to make sure that all variables are used within their scopes, it complains when it sees `b` in `bar()`, when `b` has been declared in `foo()`'s scope. The compiler will stop compiling and the program will not be passed to the linker.

We can fix the program by declaring `b` as a global variable, by moving line 1 to before `foo`'s definition.

Let us look at another example

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// C code to illustrate External Linkage
#include <stdio.h>

int x = 10;
int z = 5;

int main()
{
    extern int y; // line 2
    extern int z;
    printf("%d %d %d", x, y, z);
}
```

int y = 2;
chevron_right

filter_none

Output: 10 2 5

We can explain the output by observing behaviour of external linkage. We define 2 variables `x` and `z` in *global* scope. By default, both of them have external linkage. Now, when we declare `y` as `extern`, we tell the compiler that there exists a `y` with some definition within the same translation unit. Note that this is during the compile time phase, where the compiler trusts the `extern` keyword and compiles the rest of the program. The next line, `extern int z` has no effect on `z`, as `z` is externally linked by default when we declared it as a global variable outside the program. When we encounter `printf` line, the compiler sees 3 variables, all 3 having been declared before, and all 3 being used within their scopes (in the `printf` function). The program thus compiles successfully, even though the compiler does not know the definition of `y`.

The next phase is linking. The linker goes through the compiled code and finds `x` and `z` first. As they are global variables, they are externally linked by default. The linker then updates value of `x` and `z` throughout the entire translation unit as 10 and 5. If there are any references to `x` and `z` in any other file in the translation unit, they are set to 10 and 5.

Now, the linker comes to `extern int y` and tries to find any definition of `y` within the translation unit. It looks through every file in the translation unit to find definition of `y`. If it does not find any definition, a linker error will be thrown. In our program, we have given the definition outside `main()`, which has already been compiled for us. Thus, the linker finds that definition and updates `y`.

This article is contributed by [simran dhamija](#). If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

My Personal Notes [arrow_drop_up](#)

[Save](#)

Recommended Posts:

- Internal static variable vs. External static variable with Examples in C
- Format specifiers in different Programming Languages
- C program to find square root of a given number
- C program to print odd line contents of a File followed by even line content
- Getting System and Process Information Using C Programming and Shell in Linux
- Program to calculate Electricity Bill
- Program to print half Diamond star pattern
- C/C++ program for calling main() in main()
- Print all possible combinations of the string by replacing '\$' with any other digit from the string
- How to use make utility to build C projects?
- How to call function within function in C or C++
- Role of SemiColon in various Programming Languages
- C program to display month by month calendar for a given year
- Difference between Type Casting and Type Conversion

Improved By : BitsPlease

Article Tags :

C

C-Variable Declaration and Scope

Practice Tags :

C

[thumb_up](#)

25

To-do Done

3.7

Based on 34 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) [isgraph\(\)](#) C library function

Next

[last_page](#) How to use POSIX semaphores in C language

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

[Load Comments](#)



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
Predefined Macros in C with Examples
C program to print odd line contents of a File followed by even line content
Introduction to the C99 Programming Language : Part II
C program to find square root of a given number

More related articles in C
Format specifiers in different Programming Languages
Problem in comparing Floating point numbers and how to compare them correctly?
Features of C Programming Language
Pointer Expressions in C with Examples
Introduction to the C99 Programming Language : Part III

Different ways to declare variable as constant in C and C++

There are many different ways to make the variable as constant

1. **Using const keyword:** The const keyword specifies that a variable or object value is constant and can't be modified at the compilation time.

```
filter_none
edit
close

play_arrow
link
brightness_4
code

// C program to demonstrate const specifier
#include <stdio.h>
int main()
{
    const int num = 1;

    num = 5; // Modifying the value
    return 0;
}
chevron_right
filter_none

It will throw as error like:
error: assignment of read-only variable 'num'
```

2. **Using enum keyword:** Enumeration (or enum) is a user defined data type in C and C++. It is mainly used to assign names to integral constants, that make a program easy to read and maintain.

```
filter_none
edit
close

play_arrow
link
brightness_4
code

// In C and C++ internally the default
// type of 'var' is int
enum VARS { var = 42 };

// In C++ 11 (can have any integral type):
enum : type { var = 42; }

// where mytype = int, char, long etc.
// but it can't be float, double or
// user defined data type.
chevron_right
filter_none
```

Note: The data types of **enum** are of course limited as we can see in above example.

3. **Using constexpr keyword:** Using constexpr in C++(not in C) can be used to declare variable as a guaranteed constant. But it would fail to compile if its initializer isn't a constant expression.

```
filter_none
edit
close

play_arrow
link
brightness_4
code

#include <iostream>
```

```
int main()
{
    int var = 5;
    constexpr int k = var;
    std::cout << k;
    return 0;
}
```

chevron_right

filter_none

Above program will throw an error i.e.,

```
error: the value of 'var' is not usable in a constant expression
```

because the variable 'var' is not constant expression. Hence in order to make it as constant, we have to declare the variable 'var' with **const** keyword.

4. **Using Macros:** We can also use Macros to define constant, but there is a catch,

```
#define var 5
```

Since Macros are handled by the pre-processor(the pre-processor does text replacement in our source file, replacing all occurrences of 'var' with the literal 5) not by the compiler. Hence it wouldn't be recommended because Macros doesn't carry type checking information and also prone to error. In fact not quite constant as 'var' can be redefined like this.

```
C++
filter_none
edit
close
play_arrow
link
brightness_4
code

// C program to demonstrate the problems
// in Macros
#include <iostream>
using namespace std;

#define var 5
int main()
{
    printf("%d ", var);

    #ifdef var
    #undef var
    // redefine var as 10
    #define var 10
    #endif

    printf("%d", var);
    return 0;
}
chevron_right
filter_none
```

```

filter_none
edit
close
play_arrow
link
brightness_4
code
// C program to demonstrate the problems
// in 'Macros'
#include <stdio.h>

#define var 5
int main()
{
    printf("%d ", var);

#ifndef var
#define var 10
#endif

    printf("%d", var);
    return 0;
}
chevron_right
filter_none
Note: preprocessor and enum only works as a literal constant and integers constant respectively. Hence they only define the symbolic name of constant. Therefore if you need a constant variable with a specific memory address use either 'const' or 'constexpr' according to the requirement.
This article is contributed by Shubham Bansal. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.
GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer Placement 100 for details

```

My Personal Notes arrow_drop_up
Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

Different ways to initialize a variable in C/C++
How to declare a pointer to a function?
Can we access global variable if there is a local variable with same name?
Declaring a C/C++ function returning pointer to array of integer pointers
Internal static variable vs. External static variable with Examples in C
errno constant in C++
sizeoff() for Floating Constant in C
How to print range of basic data types without any library function and constant in C?
How to print a value of character in C
Using %c as format specifier in C
How to modify a const variable in C?
Redeclaration of global variable in C
Variable Length Arrays in C and C++
Variable Length Argument in C
Why variable name does not start with numbers in C ?

Article Tags :

C
C++
C/C++ data-types
Practice Tags :
C
CPP

thumb_up

17

To-do Done
2.2

Based on 20 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.
Post navigation
Previous
[first_page](#) Differentiate printable and control character in C ?
Next
[last_page](#) tmpnam() function in C

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

Why variable name does not start with numbers in C ?

In C, apart from keywords everything in the C program is treated as Identifier. Identifier can be the names given to variables, constants, functions and user-defined data. A variable name can consist of alphabets (upper case, lower case), numbers (0-9) and _ (underscore) character. But the name of any variable must not start with a number. Now we must have the answer that why can't we name a variable starting with number. Following might be the reason for it. The compiler has 7 phase as follows:

Lexical Analysis
Syntax Analysis
Semantic Analysis
Intermediate Code Generation
Code Optimization
Code Generation
Symbol Table

Backtracking is avoided in lexical analysis phase while compiling the piece of code. The variable like Apple;, the compiler will know its a identifier right away when it meets letter 'A' character in the lexical analysis phase. However, a variable like 123apple;, compiler won't be able to decide if its a number or identifier until it hits 'a' and it needs backtracking to go in the lexical analysis phase to identify that it is a variable. But it is not supported in compiler.

When you're parsing the token you only have to look at the first character to determine if it's an identifier or literal and then send it to the correct function for processing. So that's a performance optimization.

This article is contributed by [Bishal Kumar Dubey](#). If you like GeeksforGeeks and would like to contribute, you can also write an article using [contribute.geeksforgeeks.org](#) or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up
Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- How will you print numbers from 1 to 100 without using loop?
- Write a one line C function to round floating point numbers
- Can we access global variable if there is a local variable with same name?
- Modulus on Negative Numbers
- How to Count Variable Numbers of Arguments in C?
- Variable length arguments for Macros
- To find sum of two numbers without using any operator
- Redeclaration of global variable in C
- Variable Length Arrays in C and C++
- Converting Strings to Numbers in C/C++
- Set a variable without using Arithmetic, Relational or Conditional Operator
- Precision of floating point numbers in C++ (floor(), ceil(), trunc(), round() and setprecision())
- How to print a variable name in C?
- C Program to print numbers from 1 to N without using semicolon?
- Program to print a pattern of numbers

Article Tags :

C
Practice Tags :
C

thumb_up
30

To-do Done
2.2

Based on 31 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

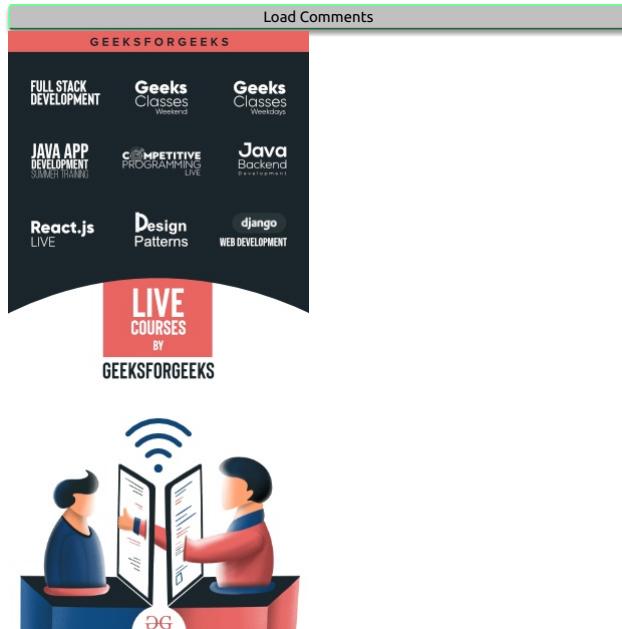
Previous

[first_page](#) Generic keyword in C

Next

[last_page](#) `pthread_self()` in C with Example

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
Predefined Macros in C with Examples
C program to print the line contents of a File followed by even line content
Introduction to the C99 Programming Language : Part II
C program to find square root of a given number

More related articles in C
Features of C Programming Languages
Problem in comparing Floating point numbers and how to compare them correctly?
Features of C Programming Language
Pointer Expressions in C with Examples
Introduction to the C99 Programming Language : Part III

Redeclaration of global variable in C

Consider the below two programs:

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// Program 1
int main()
{
    int x;
    int x = 5;
    printf("%d", x);
    return 0;
}
chevron_right
```

Output in C:

```
redeclaration of 'x' with no linkage
```

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// Program 2
int x;
int x = 5;
```

```
int main()
{
    printf("%d", x);
    return 0;
}
chevron_right
filter_none
```

Output in C:

5

In C, the first program fails in compilation, but second program works fine. In C++, both programs fail in compilation.

C allows a global variable to be declared again when first declaration doesn't initialize the variable.

The below program fails in both C also as the global variable is initialized in first declaration itself.

```
filter_none
edit
close
play_arrow
link
brightness_4
code
```

```
int x = 5;
int x = 10;
```

```
int main()
{
```

```
    printf("%d", x);
    return 0;
}
```

```
chevron_right
filter_none
```

Output:

error: redefinition of 'x'

This article is contributed by **Abhay Rathi**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up

Add your personal notes
here! (max 5000 chars)

Save

Recommended Posts:

- [Can we access global variable if there is a local variable with same name?](#)
- [Internal static variable vs. External static variable with Examples in C](#)
- [Can Global Variables be dangerous ?](#)
- [Initialization of global and static variables in C](#)
- [How Linkers Resolve Global Symbols Defined at Multiple Places?](#)
- [How to print a variable name in C?](#)
- [Why variable name does not start with numbers in C ?](#)
- [Variable Length Arrays in C and C++](#)
- [Different ways to initialize a variable in C/C++](#)
- [How to modify a const variable in C?](#)
- [Variable Length Argument in C](#)
- [C | Variable Declaration and Scope | Question 4](#)
- [C | Variable Declaration and Scope | Question 5](#)
- [C | Variable Declaration and Scope | Question 3](#)
- [C | Variable Declaration and Scope | Question 1](#)

Article Tags :

C
C-Variable Declaration and Scope
cpp-storage-classes
Practice Tags :
C

thumb_up
38

To-do Done

2

Based on 114 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) How to sort an array of dates in C/C++?

Next

[last_page](#) sprintf() in C

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments



Most popular in C
Program to calculate Electricity Bill
Program to print half Diamond star pattern
C/C++ program for calling main() in main()
C/C++ #include directive with Examples
Output of C programs | Set 66 (Accessing Memory Locations)

More related articles in C
Difference between Type Casting and Type Conversion
getchar() Function Examples
Chain of Pointers in C with Examples
Jagged Array or Array of Arrays in C with Examples
Print all possible combinations of the string by replacing '\$' with any other digit from the string

Initialization of global and static variables in C

Predict the output of following C programs.

```
filter_none
edit
close

play_arrow
link
brightness_4
code

// PROGRAM 1
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
    static int *p = (int*)malloc(sizeof(p));
    *p = 10;
    printf("%d", *p);
}
chevron_right
```

```
filter_none
filter_none
edit
close

play_arrow
link
brightness_4
code

// PROGRAM 2
#include <stdio.h>
#include <stdlib.h>
int *p = (int*)malloc(sizeof(p));

int main(void)
{
    *p = 10;
    printf("%d", *p);
}
chevron_right
```

Both of the above programs don't compile in C. We get the following compiler error in C.

```
error: initializer element is not constant
```

In C, static and global variables are initialized by the compiler itself. Therefore, they must be initialized with a constant value.

Note that the above programs compile and run fine in C++, and produce the output as 10.

As an exercise, predict the output of following program in both C and C++.

```
filter_none
edit
close

play_arrow
link
```

```
brightness_4
code

#include <stdio.h>
int fun(int x)
{
    return (x+5);
}

int y = fun(20);
```

int main()

```
{
    printf("%d ", y);
}
```

chevron_right

filter_none

This article is contributed by [Shankar Shastri](#). Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes *arrow_drop_up*

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- [Initialization of static variables in C](#)
- [Difference between Static variables and Register variables in C](#)
- [Implicit initialization of variables with 0 or 1 in C](#)
- [Initialization of variables sized arrays in C](#)
- [Can Global Variables be dangerous ?](#)
- [Static Variables in C](#)
- [How are variables scoped in C - Static or Dynamic?](#)
- [What are the default values of static variables in C?](#)
- [Internal static variable vs. External static variable with Examples in C](#)
- [Initialization of a multidimensional arrays in C/C++](#)
- [Initialization of data members](#)
- [Redeclaration of global variable in C](#)
- [How Linkers Resolve Global Symbols Defined at Multiple Places?](#)
- [Static functions in C](#)
- [C++ | Static Keyword | Question 4](#)

Article Tags :

C

C Basics

Practice Tags :

C

thumb_up

21

To-do Done
3.1

Based on 26 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

first_page [What is the difference between "char a" and "char a\[1\]"?](#)

Next

last_page [Write a C program that displays contents of a given file like 'more' utility in Linux](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
Predefined Macros in C with Examples
Format specifiers in different Programming Languages
C program to print odd line contents of a file followed by even line content
Introduction to the C99 Programming Language : Part II

More related articles in C
C program to find square root of a given number
Problems of comparing floating point numbers and how to compare them correctly?
Features of C Programming Language
Pointer Expressions in C with Examples
Introduction to the C99 Programming Language : Part III

Data Types in C

Each variable in C has an associated data type. Each data type requires different amounts of memory and has some specific operations which can be performed over it. Let us briefly describe them one by one:

Following are the examples of some very common data types used in C:

- **char:** The most basic data type in C. It stores a single character and requires a single byte of memory in almost all compilers.
- **int:** As the name suggests, an int variable is used to store an integer.
- **float:** It is used to store decimal numbers (numbers with floating point value) with single precision.
- **double:** It is used to store decimal numbers (numbers with floating point value) with double precision.

Different data types also have different ranges upto which they can store numbers. These ranges may vary from compiler to compiler. Below is list of ranges along with the memory requirement and format specifiers on 32 bit gcc compiler.

DATA TYPE	MEMORY (BYTES)	RANGE	FORMAT SPECIFIER
short int	2	-32,768 to 32,767	%hd
unsigned short int	2	0 to 65,535	%hu
unsigned int	4	0 to 4,294,967,295	%u
int	4	-2,147,483,648 to 2,147,483,647	%d
long int	8	-2,147,483,648 to 2,147,483,647	%ld
unsigned long int	8	0 to 4,294,967,295	%lu
long long int	8	-(2^63) to (2^63)-1	%lld
unsigned long long int	8	0 to 18,446,744,073,709,551,615	%llu
signed char	1	-128 to 127	%c
unsigned char	1	0 to 255	%c
float	4		%f
double	8		%lf
long double	16		%Lf

We can use the `sizeof()` operator to check the size of a variable. See the following C program for the usage of the various data types:

```

filter_none
edit
close
play_arrow
link
brightness_4
code

#include <stdio.h>
int main()
{
    int a = 1;
    char b ='G';
    double c = 3.14;
    printf("Hello World!\n");

    //printing the variables defined above along with their sizes
    printf("Hello! I am a character. My value is %c and "
           "my size is %lu byte.\n", b,sizeof(char));
    //can use sizeof(b) above as well

    printf("Hello! I am an integer. My value is %d and "
           "my size is %lu bytes.\n", a,sizeof(int));
    //can use sizeof(a) above as well

    printf("Hello! I am a double floating point variable."
           " My value is %lf and my size is %lu bytes.\n",c,sizeof(double));
    //can use sizeof(c) above as well

    printf("Bye! See you soon. :)\\n");

    return 0;
}

```

}

chevron_right

filter_none

Output:

```
Hello World!
Hello! I am a character. My value is G and my size is 1 byte.
Hello! I am an integer. My value is 1 and my size is 4 bytes.
Hello! I am a double floating point variable. My value is 3.140000 and my size is 8 bytes.
Bye! See you soon. :)
```

Quiz on Data Types in C

This article is contributed by Ayush Jaggi. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes *arrow_drop_up*

Save

Recommended Posts:

- Difference between fundamental data types and derived data types
- C++ Data Types
- Derived Data Types in C++
- C | Data Types | Question 4
- C | Data Types | Question 5
- C | Data Types | Question 6
- C | Data Types | Question 7
- C | Data Types | Question 8
- C | Data Types | Question 9
- C | Data Types | Question 2
- C | Data Types | Question 1
- What are the data types for which it is not possible to create an array?
- Uninitialized primitive data types in C/C++
- Linking Files having same variables with different data types in C
- Interesting facts about data-types and modifiers in C/C++

Improved By : Srichandrahaas

Article Tags :

C

C Basics

C-Data Types

Practice Tags :

C

thumb_up

45

To-do Done
1.5

Based on 58 vote(s)

Basic **Easy** **Medium** **Hard** **Expert**

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) Variables and Keywords in C

Next

[last_page](#) Static Variables in C

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Weekend Geeks Classes Weekdays

JAVA APP DEVELOPMENT COMPETITIVE PROGRAMMING LIVE Java Backend Development

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS



Most popular in C
Program to calculate Electricity Bill

Use of bool in C

Prerequisite: [Bool Data Type in C++](#)

The C99 standard for C language supports bool variables. Unlike C++, where no header file is needed to use bool, a header file "stdbool.h" must be included to use bool in C. If we save the below program as .c, it will not compile, but if we save it as .cpp, it will work fine.

```
filter_none
edit
close

play_arrow

link
brightness_4
code

int main()
{
    bool arr[2] = {true, false};
    return 0;
}
chevron_right
filter_none
```

If we include the header file "stdbool.h" in the above program, it will work fine as a C program.

```
filter_none
edit
close

play_arrow

link
brightness_4
code

#include <stdbool.h>
int main()
{
    bool arr[2] = {true, false};
    return 0;
}
chevron_right
filter_none
```

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes [arrow_drop_up](#)

Add your personal notes here! (max 5000 chars)

[Save](#)

Recommended Posts:

- Format specifiers in different Programming Languages
- C program to find square root of a given number
- C program to print odd line contents of a File followed by even line content
- Getting System and Process Information Using C Programming and Shell in Linux
- Program to calculate Electricity Bill
- Program to print half Diamond star pattern
- C/C++ program for calling main() in main()
- Print all possible combinations of the string by replacing '\$' with any other digit from the string
- How to use make utility to build C projects?
- How to call function within function in C or C++
- Role of SemiColon in various Programming Languages
- C program to display month by month calendar for a given year
- Difference between Type Casting and Type Conversion
- Pi(π) in C++ with Examples

Article Tags :

C
C-Data Types
Practice Tags :

C

[thumb_up](#)
28

To-do Done
1.3

Based on 105 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) Static data members in C++

Next

[last_page](#) Private Destructor

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Weekend Geeks Classes Weekdays

JAVA APP DEVELOPMENT SUMMER TRAINING COMPETITIVE PROGRAMMING LN Java Backend

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS



Most popular in C
Program to calculate Electricity Bill
Program to print half Diamond star pattern
C/C++ program for calling main() in main()
C/C++ #include directive with Examples
Output of C programs | Set 66 (Accessing Memory Locations)

More related articles in C
Character Pointers with Examples
Difference between Type Casting and Type Conversion
C program to display month by month calendar for a given year
Print all possible combinations of the string by replacing '\$' with any other digit from the string
getch() function in C with Examples

Integer Promotions in C

Some data types like `char`, `short int` take less number of bytes than `int`, these data types are automatically promoted to `int` or `unsigned int` when an operation is performed on them. This is called integer promotion. For example no arithmetic calculation happens on smaller types like `char`, `short` and `enum`. They are first converted to `int` or `unsigned int`, and then arithmetic is done on them. If an `int` can represent all values of the original type, the value is converted to an `int`. Otherwise, it is converted to an `unsigned int`.

For example see the following program.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

#include <stdio.h>
int main()
{
    char a = 30, b = 40, c = 10;
    char d = (a * b) / c;
    printf ("%d ", d);
    return 0;
}
```

chevron_right

filter_none

Output:

120

At first look, the expression `(a*b)/c` seems to cause arithmetic overflow because signed characters can have values only from -128 to 127 (in most of the C compilers), and the value of subexpression '`(a*b)`' is 1200 which is greater than 128. But integer promotion happens here in arithmetic done on `char` types and we get the appropriate result without any overflow.

Consider the following program as **another example**.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

#include <stdio.h>

int main()
{
    char a = 0xfb;
    unsigned char b = 0xfb;

    printf("a = %c", a);
    printf("\nb = %c", b);

    if (a == b)
        printf("\nSame");
    else
        printf("\nNot Same");
```

```
return 0;
```

```
}
```

```
chevron_right
```

```
filter_none
```

Output:

```
a = ?  
b = ?  
Not Same
```

When we print 'a' and 'b', same character is printed, but when we compare them, we get the output as "Not Same".

'a' and 'b' have same binary representation as *char*. But when comparison operation is performed on 'a' and 'b', they are first converted to *int*. 'a' is a signed *char*, when it is converted to *int*, its value becomes -5 (signed value of 0xfb). 'b' is *unsigned char*, when it is converted to *int*, its value becomes 251. The values -5 and 251 have different representations as *int*, so we get the output as "Not Same".

We will soon be discussing integer conversion rules between signed and unsigned, int and long int, etc.

References:

http://www.tru64unix.compaq.com/docs/base_doc/DOCUMENTATION/V40F_HTML/AQTLBTE/DOCU_067.HTM

This article is contributed by **Abhay Rathi**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes *arrow_drop_up*

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- [How to Read and Print an Integer value in C](#)
- [Input an integer array without spaces in C](#)
- [Integer literal in C/C++ \(Prefixes and Suffixes\)](#)
- [Assigning an integer to float and comparison in C/C++](#)
- [Check for integer overflow on multiplication](#)
- [Storage of integer and character values in C](#)
- [How to concatenate two integer arrays without using loop in C ?](#)
- [Extended Integral Types \(Choosing the correct integer size in C/C++\)](#)
- [Format specifiers in different Programming Languages](#)
- [C program to find square root of a given number](#)
- [C program to print odd line contents of a File followed by even line content](#)
- [Getting System and Process Information Using C Programming and Shell in Linux](#)
- [Program to calculate Electricity Bill](#)
- [Program to print half Diamond star pattern](#)

Article Tags :

C

C-Data Types

cpp-data-types

Practice Tags :

C

thumb_up

31

To-do Done
2.5

Based on 90 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) Print a long int in C using putchar() only

Next

[last_page](#) C | File Handling | Question 3

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Geeks Classes
JAVA APP DEVELOPMENT COMPETITIVE PROGRAMMING LIVE Java Backend
React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS



Most popular in C
Program to calculate Electricity Bill
Program to print half Diamond star pattern
C/C++ program for calling main() in main()
C/C++ #include directive with Examples
Output of C programs | Set 66 (Accessing Memory Locations)

More related articles in C
Difference between Type Casting and Type Conversion
getch() function in C with Examples
Chain of Pointers in C with Examples
Jagged Array or Array of Arrays in C with Examples
Print all possible combinations of the string by replacing '\$' with any other digit from the string

C Data Types

12

Question 1

Predict the output of following program. Assume that the numbers are stored in 2's complement form.

```
#include <stdio.h>
int main()
{
    unsigned int x = -1;
    int y = ~0;
    if (x == y)
        printf("same");
    else
        printf("not same");
    return 0;
}
```

A same

B not same

C Data Types

Discuss it

Question 1 Explanation:

-1 and ~0 essentially have same bit pattern, hence x and y must be same. In the comparison, y is promoted to unsigned and compared against x (See [this](#) for promotion rules). The result is "same". However, when interpreted as signed and unsigned their numerical values will differ. x is MAXUNIT and y is -1. Since we have %u for y also, the output will be MAXUNIT and MAXUNIT.

Question 2

Which of the following is not a valid declaration in C?

1. `short int x;`
2. `signed short x;`
3. `short x;`
4. `unsigned short x;`

A3 and 4

B2

C1

D All are valid

C Data Types

Discuss it

Question 2 Explanation:

All are valid. First 3 mean the same thing. 4th means unsigned.

Question 3

Predict the output

```
#include <stdio.h>
int main()
{
    float c = 5.0;
    printf ("Temperature in Fahrenheit is %.2f", (9/5)*c + 32);
    return 0;
}
```

A Temperature in Fahrenheit is 41.00

B Temperature in Fahrenheit is 37.00

C Temperature in Fahrenheit is 0.00

D Compiler Error

C Data Types

Discuss it

Question 3 Explanation:

Since 9 and 5 are integers, integer arithmetic happens in subexpression (9/5) and we get 1 as its value. To fix the above program, we can use 9.0 instead of 9 or 5.0 instead of 5 so that floating point arithmetic happens.

Question 4

Predict the output of following C program

```
#include <stdio.h>
int main()
{
    char a = ' 12';
    printf("%d", a);
    return 0;
}
```

A Compiler Error

B 12

C 10

D Empty

C Data Types

Discuss it

Question 4 Explanation:

The value '012' means the character with value 12 in octal, which is decimal 10. Note: It is equivalent to char a = 012 and int a = '\012' and int a = 012.

Question 5

In C, sizes of an integer and a pointer must be same.

A True

B False

C Data Types

Discuss it

Question 5 Explanation:

Sizes of integer and pointer are compiler dependent. The both sizes need not be same.

Question 6

Output?

```
int main()
{
    void *vptr, v;
    v = 0;
    vptr = &v;
    printf("%v", *vptr);
    getchar();
    return 0;
}
```

A 0

B Compiler Error

C Garbage Value

C Data Types

Discuss it

Question 6 Explanation:
void is not a valid type for declaring variables. void * is valid though.

Question 7

Assume that the size of char is 1 byte and negatives are stored in 2's complement form

```
#include<stdio.h>
int main()
{
    char c = 125;
    c = c+10;
    printf("%d", c);
    return 0;
}
```

A135

B+INF

C-121

D-8

C Data Types

Discuss it

Question 7 Explanation:

125 is represented as 0111101 in binary and when we add 10 i.e 1010 in binary it becomes : 10000111. Now what does this number represent? Firstly, you should know that char can store numbers only -128 to 127 since the most significant bit is kept for sign bit. Therefore 10000111 represents a negative number. To check which number it represents we find the 2's complement of it and get 01111001 which is = 121 in decimal system. Hence, the number 10000111 represents -121.

Question 8

```
#include <stdio.h>
int main()
{
    if (sizeof(int) > -1)
        printf("Yes");
    else
        printf("No");
    return 0;
}
```

AYes

BNo

CCompiler Error

DRuntime Error

C Data Types

Discuss it

Question 8 Explanation:

In C, when an integer value is compared with an unsigned int, the int is promoted to unsigned. Negative numbers are stored in 2's complement form and unsigned value of the 2's complement form is much higher than the sizeof int.

Question 9

Suppose n and p are unsigned int variables in a C program. We wish to set p to $\lceil \frac{n}{3} \rceil$. If n is large, which of the following statements is most likely to set p correctly?

Ap = n * (n-1) * (n-2) / 6;
Bp = n * (n-1) / 2 * (n-2) / 3;
Cp = n * (n-1) / 3 * (n-2) / 2;
Dp = n * (n-1) * (n-2) / 6.0;

C Data Types GATE-CS-2014-(Set-2)

Discuss it

Question 9 Explanation:

As n is large, the product $n*(n-1)*(n-2)$ will go out of the range(overflow) and it will return a value different from what is expected. Therefore, option (A) and (D) are eliminated. So we consider a shorter product $n*(n-1)$. $n*(n-1)$ is always an even number. So the subexpression " $n * (n-1) / 2$ " in option B would always produce an integer, which means no precision loss in this subexpression. And when we consider " $n*(n-1)/2*(n-2)$ ", it will always give a number which is a multiple of 3. So dividing it with 3 won't have any loss.

Question 10

Output of following program?

```
#include<stdio.h>
int main()
{
    float x = 0.1;
    if ( x == 0.1 )
        printf("IF");
    else if (x == 0.1f)
        printf("ELSE IF");
    else
        printf("ELSE");
}
```

AELSE IF

BIF

CELSE

C Data Types

Discuss it

There are 14 questions to complete.

12

My Personal Notes 

Add your personal notes here! (max 5000 chars)

Comparison of a float with a value in C

Predict the output of following C program.

filter_none
edit
close

play_arrow

link
brightness_4
code


```
#include<stdio.h>
int main()
{
    float x = 0.1;
    if (x == 0.1)
        printf("IF");
    else if (x == 0.1f)
        printf("ELSE IF");
    else
        printf("ELSE");
}
```



filter_none

The output of above program is "**ELSE IF**" which means the expression "x == 0.1" returns false and expression "x == 0.1f" returns true.

Let consider the of following program to understand the reason behind the above output.

```

filter_none
edit
close
play_arrow
link
brightness_4
code

#include<stdio.h>
int main()
{
    float x = 0.1;
    printf("%d %d %d", sizeof(x), sizeof(0.1), sizeof(0.1f));
    return 0;
}
chevron_right

```

filter_none

The output of above program is **4 8 4** on a typical C compiler.
It actually prints size of float, size of double and size of float.

The values used in an expression are considered as double (**double precision floating point format**) unless a 'f' is specified at the end. So the expression "x==0.1" has a double on right side and float which are stored in a **single precision floating point format** on left side. In such situations float is promoted to double (see [this](#)). The double precision format uses more bits for precision than single precision format. The binary equivalent of 0.1_{10} can be written as $(0.00011001100110011001)_{2}$ which will goes up to infinity (See [this article](#) to know more about conversion). Since the precision of float is less than the double therefore after certain point (23 in float and 52 in double) it would truncate the result. Hence after promotion of float into double(at the time of comparison) compiler will pad the remaining bits with zeroes. Hence we get the different result in which decimal equivalent of both would be different. For instance,

```

In float
=> (0.1)10 = (0.00011001100110011001)2
In double after promotion of float ... (1)
=> (0.1)10 = (0.000110011001100110011000000000000000000...)2
                                         ^ padding zeroes here
In double without promotion ... (2)
=> (0.1)10 = (0.00011001100110011001100110011001100110011001)2

Hence we can see the result of both equations are different.
Therefore 'if' statement can never be executed.

```

Note that the promotion of float to double can only cause mismatch when a value (like 0.1) uses more precision bits than the bits of single precision. For example, the following C program prints "IF".

```

filter_none
edit
close
play_arrow
link
brightness_4
code

#include<stdio.h>
int main()
{
    float x = 0.5;
    if (x == 0.5)
        printf("IF");
    else if (x == 0.5f)
        printf("ELSE IF");
    else
        printf("ELSE");
}
chevron_right

```

filter_none

Output:

IF

Here binary equivalent of 0.5_{10} is $(0.100000...)_{2}$

(No precision will be lost in both float and double type). Therefore if compiler pad the extra zeroes at the time of promotion then we would get the same result in decimal equivalent of both left and right side in comparison($x == 0.5$).

You can refer [Floating Point Representation - Basics](#) for representation of floating point numbers.

This article is contributed by **Abhay Rathi** and improved by **SHUBHAM BANSAL**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes 

Add your personal notes
here! (max 5000 chars)

Save

Recommended Posts:

- [Assigning an integer to float and comparison in C/C++](#)
- [Results of comparison operations in C and C++](#)
- [Ratio Manipulations in C++ | Set 2 \(Comparison\)](#)
- [Comparison of boolean data type in C++ and Java](#)
- [How to check whether a number is in the range\[low, high\] using one comparison ?](#)
- [Difference between float and double in C/C++](#)
- [Function Overloading and float in C++](#)
- [<cfloat> float.h in C/C++ with Examples](#)
- [Modulus of two float or double numbers](#)
- [gcvt\(\) | Convert float value to string in C](#)
- [C/C++ program to find the size of int, float, double and char](#)
- [Format specifiers in different Programming Languages](#)
- [C program to find square root of a given number](#)
- [C program to print odd line contents of a File followed by even line content](#)

Article Tags :

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page C | Data Types | Question 9](#)

Next

[last_page Difference between #define and const in C?](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Weekday Geeks Classes Weekdays

JAVA APP DEVELOPMENT COMPETITIVE PROGRAMMING LIVE Java Backend SUMMER TRAINING

React.js Design Patterns django

LIVE COURSES BY GEEKSFORGEEKS



Most popular in C
Program to calculate Electricity Bill
Program to print half Diamond star pattern
C/C++ program for calling main() in main()
C/C++ #include directive with Examples
Output of C programs | Set 66 (Accessing Memory Locations)

More related articles in C
Difference between Type Casting and Type Conversion
getch() function in C with Examples
Chain of Pointers in C with Examples
Jagged Array or Array of Arrays in C with Examples
Print all possible combinations of the string by replacing '\$' with any other digit from the string

Is there any need of “long” data type in C and C++?

In C and C++, there are four different data type available for holding the integers i.e., **short**, **int**, **long** and **long long**. Each of these data type requires different amounts of memory.

But there is a catch, the size of “**long**” data type is not fixed unlike other data types. It varies from architectures, operating system and even with compiler that we are using. In some of the systems it behaves like an **int** data type or a **long long** data type as follows:

OS	Architecture	Size
Windows	IA-32	4 bytes
Windows	Intel® 64 or IA-64	4 bytes
Linux	IA-32	4 bytes
Linux	Intel® 64 or IA-64	8 bytes
Mac OS X	IA-32	4 bytes
Mac OS X	Intel® 64 or IA-64	8 bytes

Well it also varies from compiler. But before this, let's understand about the concept of **cross compiler**.

A **cross compiler** is a compiler capable of creating executable code for a platform other than the one on which the compiler is running. For instance, if I compile the following programs in 64 bit architecture running a 64 bit Ubuntu, I will get the result like this:

C++

filter_none
edit
close

play_arrow

link

brightness_4
code

```
// C++ program to check the size of 'long'  
// data type
```

```
#include <bits/stdc++.h>  
using namespace std;
```

```
int main()  
{  
    cout << "Size of int = "<< sizeof(int) << endl;  
    cout << "Size of long = " << sizeof(long) << endl;  
    cout << "Size of long long = " << sizeof(long long);
```

}

```
// This code is contributed by shubhamsingh10
chevron_right
```

```
filter_none
```

C

```
filter_none
```

```
edit
```

```
close
```

```
play_arrow
```

```
link
```

```
brightness_4
```

```
code
```

```
// C program to check the size of 'long'
```

```
// data type
```

```
#include<stdio.h>
```

```
int main()
```

```
{
```

```
    printf("Size of int = %ld\n", sizeof(int));
```

```
    printf("Size of long = %ld\n", sizeof(long));
```

```
    printf("Size of long long = %ld", sizeof(long long));
```

```
}
```

```
chevron_right
```

```
filter_none
```

```
Output in 32 bit gcc compiler:-
```

```
Size of int = 4
```

```
Size of long = 4
```

```
Size of long long = 8
```

```
Output in 64 bit gcc compiler:-
```

```
Size of int = 4
```

```
Size of long = 8
```

```
Size of long long = 8
```

See [this](#) article to know more about how to compile a program with 32-bit or 64-bit gcc compiler.

From above we conclude that size of only "long" data type varies from compiler. Now the question is what exactly is happening here? Let's discuss it in the way of how compiler allocates memory internally.

CPU calls data from RAM by giving the address of the location to MAR (Memory Address Register). The location is found and the data is transferred to MDR (Memory Data Register). This data is recorded in one of the Registers in the Processor for further processing. That's why size of Data Bus determines the size of Registers in Processor. Now, a 32 bit register can call data of 4 bytes size only, at a time. And if the data size exceeds 32 bits, then it would require two cycles of fetching to have the data in it. This slows down the speed of 32 bit Machine compared to 64 bit, which would complete the operation in ONE fetch cycle only. So, obviously for the smaller data, it makes no difference if my processors are clocked at the same speed. Compilers are designed to generate the most efficient code for the target machine architecture.

So, in short the size of a variable is compiler dependent as it generates the instructions based on the target architecture and system architecture that only deals with the size of data bus and its transfer.

Note: Interestingly we don't have any need of "long" data type as their replacement(int, long long) is already available from [C99](#) standard.

Suggestion: If it is important to you for integer types to have the same size on all Intel platforms, then consider replacing "**long**" by either "**int**" or "**long long**". The size of the "int" integer type is 4 bytes and the size of the "long long" integer type is 8 bytes for all the above combinations of operating system, architecture and compiler.

References:

<https://software.intel.com/en-us/articles/size-of-long-integer-type-on-different-architecture-and-os>

This article is contributed by Shubham Bansal. If you like GeeksforGeeks and would like to contribute, you can also write an article using [contribute.geeksforgeeks.org](#) or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up

Add your personal notes
here! (max 5000 chars)

Save

Recommended Posts:

- [What is the size_t data type in C?](#)
- [What is data type of FILE in C ?](#)
- [Bool data type in C++](#)
- [C++ map having key as a user define data type](#)
- [C++ set for user define data type](#)
- [Data type of character constants in C and C++](#)
- [Data Type Ranges and their macros in C++](#)
- [Multi-set for user defined data type](#)
- [Comparison of boolean data type in C++ and Java](#)
- [Data type of case labels of switch statement in C++?](#)
- [Conversion of Struct data type to Hex String and vice versa](#)
- [Difference between Type Casting and Type Conversion](#)
- [Difference between fundamental data types and derived data types](#)
- [Type of 'this' pointer in C++](#)
- [Type Conversion in C](#)

Improved By : SHUBHAMSINGH10

Article Tags :

C

C++

cpp-data-types

Practice Tags :

C

CPP

thumb_up

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.
Post navigation
Previous
[first_page strcat\(\) vs strncat\(\) in C++](#)
Next
[last_page Segmentation Fault \(SIGSEGV\) vs Bus Error \(SIGBUS\)](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

[Load Comments](#)

GEEKSFORGEEKS

[FULL STACK DEVELOPMENT](#) [Geeks Classes Weekend](#) [Geeks Classes Weekdays](#)
[JAVA APP DEVELOPMENT](#) [COMPETITIVE PROGRAMMING LVL](#) [Java Backend](#)
[React.js LIVE](#) [Design Patterns](#) [django WEB DEVELOPMENT](#)

LIVE COURSES BY GEEKSFORGEEKS



Most popular in C
[Print all possible combinations of the string by replacing '\\$' with any other digit from the string](#)
[Predefined Macros in C with Examples](#)
[Format specifiers in different Programming Languages](#)
[C program to print odd line contents of a File followed by even line content](#)
[C program to find square root of a given number](#)

Most visited in C++
[Vector of Vectors in C++ STL with Examples](#)
[C++ Operator Overloading](#)
[Which C++ libraries are useful for competitive programming?](#)
[Array of Vectors in C++ STL](#)
[Map of Vectors in C++ STL with Examples](#)

What is the `size_t` data type in C?

`size_t` is an unsigned integral data type which is defined in various header files such as:

```
filter_none
edit
close
play_arrow
link
brightness_4
code

<stddef.h>, <stdio.h>, <stdlib.h>, <string.h>, <time.h>, <wchar.h>
chevron_right
filter_none
```

It's a type which is used to represent the size of objects in bytes and is therefore used as the return type by the `sizeof` operator. It is guaranteed to be big enough to contain the size of the biggest object the host system can handle. Basically the maximum permissible size is dependent on the compiler; if the compiler is 32 bit then it is simply a typedef(i.e., alias) for `unsigned int` but if the compiler is 64 bit then it would be a typedef for `unsigned long long`. The `size_t` data type is never negative.

Therefore many C library functions like `malloc`, `memcpy` and `strlen` declare their arguments and return type as `size_t`. For instance,

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// Declaration of various standard library functions.
```

```
// Here argument of 'n' refers to maximum blocks that can be
// allocated which is guaranteed to be non-negative.
void *malloc(size_t n);
```

```
// While copying 'n' bytes from 's2' to 's1'
// n must be non-negative integer.
void *memcpy(void *s1, void const *s2, size_t n);
```

```
// strlen() uses size_t because the length of any string
// will always be at least 0.
```

```
size_t strlen(char const *s);
chevron_right
filter_none
```

size_t or any unsigned type might be seen used as loop variable as loop variables are typically greater than or equal to 0.

Note: When we use a **size_t** object, we have to make sure that in all the contexts it is used, including arithmetic, we want only non-negative values. For instance, the following program would definitely give the unexpected result:

```
filter_none
edit
close

play_arrow

link
brightness_4
code

// C program to demonstrate that size_t or
// any unsigned int type should be used
// carefully when used in a loop.
#include<stdio.h>

#define N 10

int main()
{
    int a[N];

    // This is fine.
    for (size_t n = 0; n < N; ++n) {
        a[n] = n;
    }

    // But reverse cycles are tricky for unsigned
    // types as they can lead to infinite loops.
    for (size_t n = N-1; n >= 0; --n)
        printf("%d ", a[n]);
}
```

```
chevron_right
```

```
filter_none
```

Output
Infinite loop and then segmentation fault

This article is contributed by Shubham Bansal. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer Placement 100 for details

My Personal Notes arrow_drop_up

Add your personal notes
here! (max 5000 chars)

Save

Recommended Posts:

- Bool data type in C++
- What is data type of FILE in C ?
- Is there any need of "long" data type in C and C++?
- C++ set for user define data type
- Data Type Ranges and their macros in C++
- C++ map having key as a user define data type
- Data type of character constants in C and C++
- Multi-set for user defined data type
- Comparison of boolean data type in C++ and Java
- Data type of case labels of switch statement in C++?
- Conversion of Struct data type to Hex String and vice versa
- Difference between Type Casting and Type Conversion
- Difference between fundamental data types and derived data types
- Type of 'this' pointer in C++
- Type Conversion in C++

Improved By : [jflopezfernandez](#)

Article Tags :

C
C++
cpp-data-types

Practice Tags :

C
CPP

thumb_up
32

To-do Done
2.6

Based on 33 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) [iscntrl\(\)](#) in C++ and its application to find control characters

Next

[last_page](#) Operator overloading in C++ to print contents of vector, map, pair, ..

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Weekdays Geeks Classes Weekends

Java APP DEVELOPMENT Summer Training COMPETITIVE PROGRAMMING LIVE Java Backend Part-time

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS



Most popular in C
Print all possible combinations of the string by replacing 'S' with any other digit from the string
Predefined Macros in C with Examples
C Program to print content of a File followed by even line content
Introduction to the C99 Programming Language : Part II
C program to find square root of a given number

Most visited in C++
Vector of Vectors in C++ STL with Examples
C++ Tutorial
Which C++ libraries are useful for competitive programming?
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

Interesting facts about data-types and modifiers in C/C++

Here are some logical and interesting facts about data-types and the modifiers associated with data-types:-

1. If no data type is given to a variable, the compiler automatically converts it to int data type.

C++

```
filter_none
edit
close
play_arrow
link
brightness_4
code

#include <iostream>
using namespace std;

int main()
{
    signed a;
    signed b;

    // size of a and b is equal to the size of int
    cout << "The size of a is " << sizeof(a) << endl;
    cout << "The size of b is " << sizeof(b);
    return (0);
}

// This code is contributed by shubhamsingh10
chevron_right
```

C

```
filter_none
edit
close
play_arrow
link
brightness_4
code

#include <stdio.h>
int main()
{
    signed a;
    signed b;

    // size of a and b is equal to the size of int
    printf("The size of a is %d\n", sizeof(a));
    printf("The size of b is %d", sizeof(b));
    return (0);
}
```

```
}
```

```
chevron_right
```

```
filter_none
```

Output:

```
The size of a is 4  
The size of b is 4
```

2. Signed is the default modifier for char and int data types.

C++

```
filter_none  
edit  
close  
  
play_arrow  
  
link  
brightness_4  
code  
  
#include <iostream>  
using namespace std;
```

```
int main()  
{  
    int x;  
    char y;  
    x = -1;  
    y = -2;  
    cout << "x is " << x << " and y is " << y << endl;  
}
```

```
// This code is contributed by shubhamsingh10  
chevron_right
```

```
filter_none
```

C

```
filter_none  
edit  
close  
  
play_arrow  
  
link  
brightness_4  
code  
  
#include <stdio.h>  
int main()  
{  
    int x;  
    char y;  
    x = -1;  
    y = -2;  
    printf("x is %d and y is %d", x, y);  
}
```

```
chevron_right
```

```
filter_none
```

Output:

```
x is -1 and y is -2.
```

3. We can't use any modifiers in float data type. If programmer tries to use it ,the compiler automatically gives compile time error.

C++

```
filter_none  
edit  
close  
  
play_arrow  
  
link  
brightness_4  
code  
  
#include <iostream>  
using namespace std;  
int main()  
{  
    signed float a;  
    short float b;  
    return (0);  
}

```
//This article is contributed by shivanisinghss2110
chevron_right
```


```

```
filter_none
```

```
edit  
close
```

C

```
filter_none  
edit  
close
```

```
play_arrow  
link  
brightness_4  
code  
  
#include <stdio.h>  
int main()  
{  
    signed float a;  
    short float b;  
    return (0);  
}  
chevron_right
```

filter_none

Output:
[Error] both 'signed' and 'float' in declaration specifiers
[Error] both 'short' and 'float' in declaration specifiers

4. Only long modifier is allowed in double data types. We cant use any other specifier with double data type. If we try any other specifier, compiler will give compile time error.

C++

```
filter_none  
edit  
close  
  
play_arrow  
link  
brightness_4  
code  
  
#include <iostream>  
using namespace std;  
int main()  
{  
    long double a;  
    return (0);  
}
```

// This code is contributed by shubhamsingh10
chevron_right

filter_none

```
filter_none  
edit  
close  
  
play_arrow  
link  
brightness_4  
code  
  
#include <stdio.h>  
int main()  
{  
    long double a;  
    return (0);  
}  
chevron_right
```

filter_none

```
filter_none  
edit  
close  
  
play_arrow  
link  
brightness_4  
code  
  
#include <iostream>  
using namespace std;  
int main()  
{  
    short double a;  
    signed double b;  
    return (0);  
}
```

// This code is contributed by shubhamsingh10
chevron_right

filter_none

```
filter_none  
edit  
close  
  
play_arrow
```

C

```
link  
brightness_4  
code  
  
#include <stdio.h>  
int main()  
{  
    short double a;  
    signed double b;  
    return (0);  
}  
chevron_right
```

filter_none

Output:

```
[Error] both 'short' and 'double' in declaration specifiers  
[Error] both 'signed' and 'double' in declaration specifiers
```

This article is contributed by **Bishal Kumar Dubey**. If you like GeeksforGeeks and would like to contribute, you can also write an article using [contribute.geeksforgeeks.org](#) or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- Py-Facts - 10 interesting facts about Python
- Interesting Facts about C#
- Interesting Facts about C++
- Interesting facts about C Language
- Interesting Facts about Linux
- Interesting Facts About Java
- Interesting Facts about Ubuntu
- Interesting Facts in C Programming
- C++ bitset interesting facts
- Interesting facts about strings in Python | Set 1
- Interesting facts about null in Java
- Interesting Facts about Macros and Preprocessors in C
- Interesting facts about Fibonacci numbers
- Some Interesting facts about default arguments in C++
- Interesting facts about switch statement in C

Improved By : [Rhythm1](#), [SHUBHAMSINGH10](#), [shivanisinghs2110](#)

Article Tags :

C
C++
cpp-data-types
interesting-facts

Practice Tags :

C
CPP

thumb_up
19

To-do Done
1.5

Based on 28 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) [std::none_of in C++](#)

Next

[last_page](#) [Nested functions in C](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
Predefined Macros in C with Examples
Format specifiers in different Programming Languages
C program to print odd line contents of a File followed by even line content
Introduction to the C99 Programming Language : Part II

Most visited in C++
Vector of Vectors in C++ STL with Examples
C++ STL libraries are useful for competitive programming?
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

Difference between float and double in C/C++

For representing floating point numbers, we use **float**, **double** and **long double**.

What's the difference ?

double has 2x more precision than **float**.

float is a 32 bit IEEE 754 single precision Floating Point Number 1 bit for the sign, (8 bits for the exponent, and 23* for the value), i.e. float has 7 decimal digits of precision.

double is a 64 bit IEEE 754 double precision Floating Point Number (1 bit for the sign, 11 bits for the exponent, and 52* bits for the value), i.e. double has 15 decimal digits of precision.

Let's take a example(example taken from [here](#)) :
For a quadratic equation $x^2 - 4.0000000 x + 3.9999999 = 0$, the exact roots to 10 significant digits are, $r1 = 2.000316228$ and $r2 = 1.999683772$

`filter_none`
`edit`
`close`

`play_arrow`

`link`

`brightness_4`

`code`

// C program to demonstrate
// double and float precision values

```
#include <stdio.h>  
#include <math.h>
```

```
// utility function which calculate roots of  
// quadratic equation using double values
```

```
void double_solve(double a, double b, double c){  
    double d = b*b - 4.0*a*c;  
    double sd = sqrt(d);  
    double r1 = (-b + sd) / (2.0*a);  
    double r2 = (-b - sd) / (2.0*a);  
    printf("%.5f\t%.5f\n", r1, r2);  
}
```

```
// utility function which calculate roots of  
// quadratic equation using float values
```

```
void float_solve(float a, float b, float c){  
    float d = b*b - 4.0f*a*c;  
    float sd = sqrtf(d);  
    float r1 = (-b + sd) / (2.0f*a);  
    float r2 = (-b - sd) / (2.0f*a);  
    printf("%.5f\t%.5f\n", r1, r2);  
}
```

```
// driver program
```

```
int main(){
```

```
    float fa = 1.0f;  
    float fb = -4.000000f;  
    float fc = 3.999999f;  
    double da = 1.0;  
    double db = -4.000000;  
    double dc = 3.999999;  
  
    printf("roots of equation x^2 - 4.000000 x + 3.999999 = 0 are : \n");  
    printf("for float values: \n");  
    float_solve(fa, fb, fc);
```

```
printf("for double values: \n");
double_solve(da, db, dc);
return 0;
}
```

Chevron_right

filter_none

Output:

```
roots of equation x2 - 4.000000 x + 3.999999 = 0 are :
for float values:
2.00000  2.00000
for double values:
2.00032  1.99968
```

This article is contributed by **Mandeep Singh**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer Placement 100 for details

My Personal Notes *arrow_drop_up*

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- Modulus of two float or double numbers
- C/C++ program to find the size of int, float, double and char
- What is the difference between single quoted and double quoted declaration of char array?
- Difference between Single Bus Structure and Double Bus Structure
- Difference between Single Precision and Double Precision
- Comparison of a float with a value in C
- <cfloat> float.h in C/C++ with Examples
- Function Overloading and float in C++
- gconv() | Convert float value to string in C
- Assigning an integer to float and comparison in C/C++
- Double forking to prevent Zombie process
- C program to print a string without any quote (single or double) in the program
- Difference between DFA and NFA
- Difference between H.323 and SIP
- Difference Between SMO and SEO

Improved By : msdeep14

Article Tags :

C
Difference Between
cpp-data-types

Practice Tags :

C

thumb_up

18

To-do Done
2.3

Based on 19 vote(s)

Basic **Easy** **Medium** **Hard** **Expert**

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

first_page Differences between JDK, JRE and JVM

Next

last_page std::next vs std::advance in C++

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
ctype.h<ctype> library in C/C++ with Examples

How to call function within function in C or C++
Format specifiers in different Programming Languages
Predefined Macros in C with Examples

Most visited in Difference Between
Difference between PostgreSQL and MongoDB
Difference between DELETE and TRUNCATE
Difference Between SMO and SEO
Difference between Prim's and Kruskal's algorithm for MST
Monolithic vs Microservices architecture

Character arithmetic in C and C++

As already known character known character range is between -128 to 127 or 0 to 255. This point has to be kept in mind while doing character arithmetic. To understand better let's take an example.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// C program to demonstrate character arithmetic.

#include <stdio.h>
```

```
int main()
{
    char ch1 = 125, ch2 = 10;
    ch1 = ch1 + ch2;
    printf("%d\n", ch1);
    printf("%c\n", ch1 - ch2 - 4);
    return 0;
}
```

chevron_right

filter_none

Output:

```
-121
y
```

So %d specifier causes an integer value to be printed and %c specifier causes a character value to printed. But care has to be taken that while using %c specifier the integer value should not exceed 127. So far so good.

But for C++ it plays out a little different.

Look at this example to understand better.

```
filter_none
edit
close
play_arrow
link
brightness_4
code
```

```
// A C++ program to demonstrate character
// arithmetic in C++.

#include <bits/stdc++.h>
using namespace std;
```

```
int main()
{
    char ch = 65;
    cout << ch << endl;
    cout << ch + 0 << endl;
    cout << char(ch + 32) << endl;
    return 0;
}
```

chevron_right

filter_none

Output:

```
A  
65  
a
```

Without a '+' operator character value is printed. But when used along with '+' operator behaved differently. Use of '+' operator implicitly typecasts it to an 'int'. So to conclude, in character arithmetic, typecasting of char variable to 'char' is explicit and to 'int' it is implicit.

This article is contributed by [Parveen Kumar](#). If you like GeeksforGeeks and would like to contribute, you can also write an article using [contribute.geeksforgeeks.org](#) or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes

Add your personal notes here! (max 5000 chars)

Recommended Posts:

- [Check input character is alphabet, digit or special character](#)
- [Sum of array using pointer arithmetic](#)
- [Ratio Manipulations in C++ | Set 1 \(Arithmetical\)](#)
- [How to sum two integers without using arithmetic operators in C/C++?](#)
- [Multidimensional Pointer Arithmetic in C/C++](#)
- [Pointers in C and C++ | Set 1 \(Introduction, Arithmetic and Array\)](#)
- [Conditionally assign a value without using conditional and arithmetic operators](#)
- [Set a variable without using Arithmetic, Relational or Conditional Operator](#)
- [Character Classification in C++ : cctype](#)
- [Type difference of character literals in C and C++](#)
- [Change/add only one character and print '*' exactly 20 times](#)
- [Data type of character constants in C and C++](#)
- [getline\(\) function and character array](#)
- [Frequency of each character in a String using unordered_map in C++](#)
- [Differentiate printable and control character in C ?](#)

Article Tags :

[C](#)

[C++](#)

[cpp-data-types](#)

Practice Tags :

[C](#)

[CPP](#)

 [thumb_up](#)

19

To-do Done
1.6

Based on 18 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) [C Program to find IP Address, Subnet Mask & Default Gateway](#)

Next

[last_page](#) [std::string::find_first_not_of in C++](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT **Geeks Classes** Weekend **Geeks Classes** Weekdays

JAVA APP DEVELOPMENT COMPETITIVE PROGRAMMING DN Java Backend

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
Format specifiers in different Programming Languages
C program to find square root of a given number
Predefined Macros in C with Examples
C program to print odd line contents of a File followed by even line content

Most visited in C++

Vectors in C++ STL with Examples

C++ Tutorials

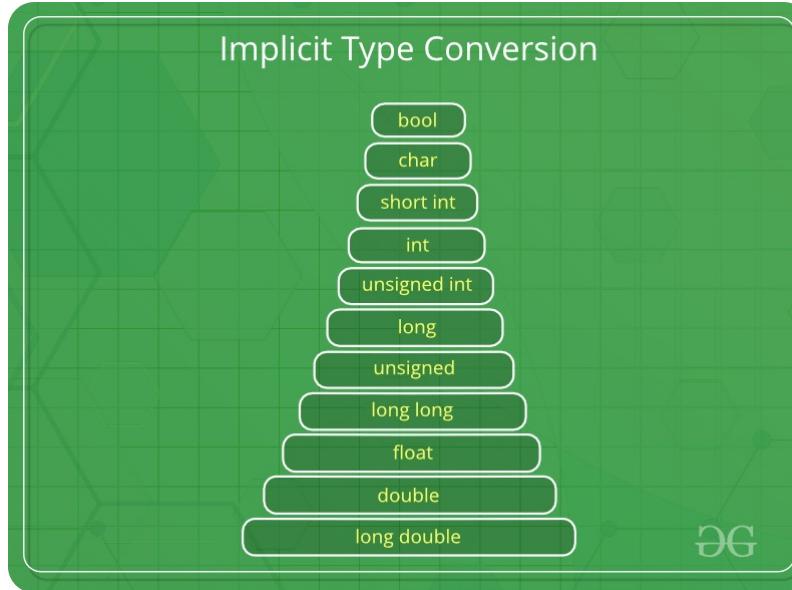
Which C++ libraries are useful for competitive programming?

Array of Vectors in C++ STL

Type Conversion in C

A type cast is basically a conversion from one type to another. There are two types of type conversion:

1. Implicit Type Conversion



Also known as 'automatic type conversion'.

- Done by the compiler on its own, without any external trigger from the user.
- Generally takes place when in an expression more than one data type is present. In such condition type conversion (type promotion) takes place to avoid lose of data.
- All the data types of the variables are upgraded to the data type of the variable with largest data type.

```
bool -> char -> short int -> int ->
unsigned int -> long -> unsigned ->
long long -> float -> double -> long double
```

- It is possible for implicit conversions to lose information, signs can be lost (when signed is implicitly converted to unsigned), and overflow can occur (when long long is implicitly converted to float).

Example of Type Implicit Conversion:

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// An example of implicit conversion
#include<stdio.h>
int main()
{
    int x = 10;    // integer x
    char y = 'a'; // character c

    // y implicitly converted to int. ASCII
    // value of 'a' is 97
    x = x + y;

    // x is implicitly converted to float
    float z = x + 1.0;

    printf("x = %d, z = %f", x, z);
    return 0;
}
```

chevron_right

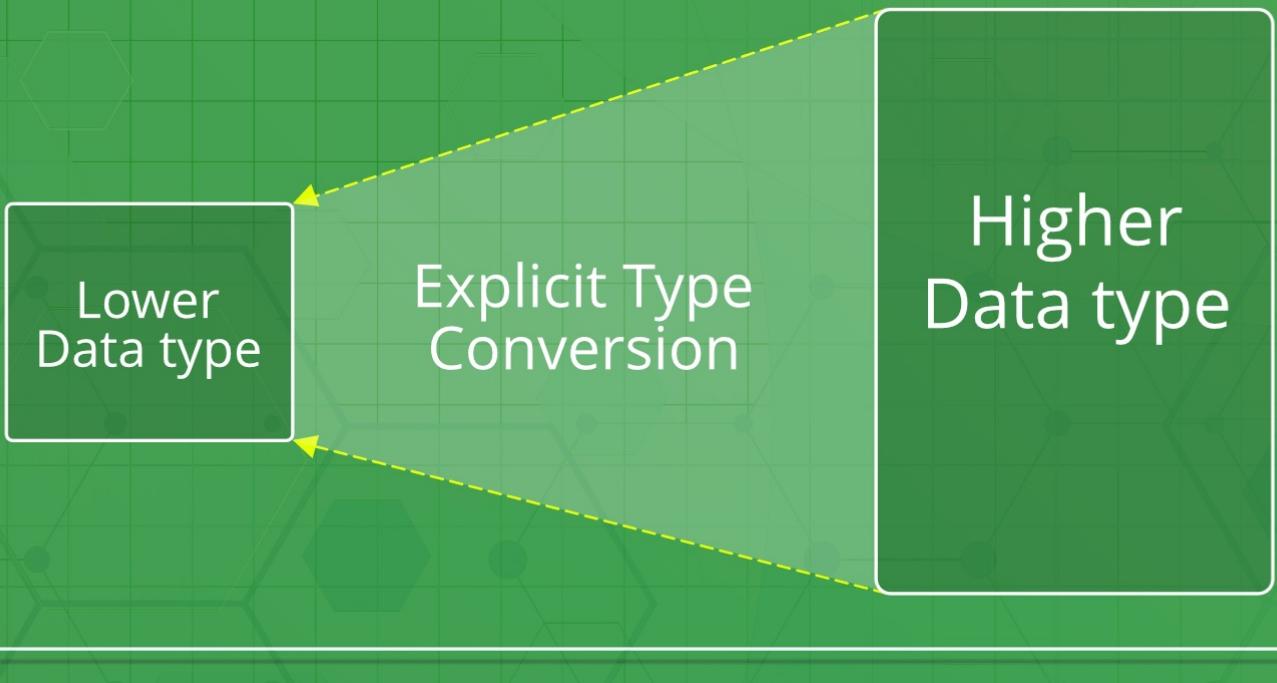
filter_none

Output:

```
x = 107, z = 108.000000
```

2. Explicit Type Conversion-

Explicit Type Conversion



This process is also called type casting and it is user defined. Here the user can type cast the result to make it of a particular data type.

The syntax in C:

```
(type) expression
```

Type indicated the data type to which the final result is converted.

```
filter_none  
edit  
close  
  
play_arrow  
link  
brightness_4  
code  
  
// C program to demonstrate explicit type casting  
#include<stdio.h>  
  
int main()  
{  
    double x = 1.2;  
  
    // Explicit conversion from double to int  
    int sum = (int)x + 1;  
  
    printf("sum = %d", sum);  
  
    return 0;  
}
```

chevron_right

```
filter_none
```

Output:

```
sum = 2
```

Advantages of Type Conversion

- This is done to take advantage of certain features of type hierarchies or type representations.
- It helps us to compute expressions containing variables of different data types.

Related articles:

- [Catch Block and Type Casting in C](#)
- [Integer Promotion](#)

This article is contributed by **Ankita Dutta**. If you like GeeksforGeeks and would like to contribute, you can also write an article and mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- Difference between Type Casting and Type Conversion
- Implicit Type Conversion in C with Examples
- Catch block and type conversion in C++
- Conversion of Struct data type to Hex String and vice versa
- Advanced C++ | Conversion Operators
- Is there any need of "long" data type in C and C++?
- What is data type of FILE in C ?
- What is the size_t data type in C?
- Implicit return type int in C
- Array Type Manipulation in C++
- Type Inference in C++ (auto and decltype)
- Data Type Ranges and their macros in C++
- const_cast in C++ | Type Casting operators
- Data type of character constants in C and C++
- C | Storage Classes and Type Qualifiers | Question 12

Article Tags :

C
C-Data Types
Practice Tags :

C



31

To-do Done
1.7

Based on 28 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

first_page Understanding nullptr in C++

Next

last_page Type Inference in C++ (auto and decltype)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Weekend Geeks Classes Weekdays

JAVA APP DEVELOPMENT COMPETITIVE PROGRAMMING LIVE Java Backend Development

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS

Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
Predefined Macros in C with Examples
Format specifiers in different Programming Languages
C program to print end line contents of a File followed by even line content
Introduction to the C99 Programming Language : Part II

More related articles in C
C program to calculate the root of a given number
Program in comparing Floating point numbers and how to compare them correctly?
Features of C Programming Language
Pointer Expressions in C with Examples
Introduction to the C99 Programming Language : Part III

Storage Classes in C

Storage Classes are used to describe the features of a variable/function. These features basically include the scope, visibility and life-time which help us to trace the existence of a particular variable during the runtime of a program.

C language uses 4 storage classes, namely:

Storage classes in C

Storage Specifier	Storage	Initial value	Scope	Life
auto	stack	Garbage	Within block	End of block
extern	Data segment	Zero	global Multiple files	Till end of program
static	Data segment	Zero	Within block	Till end of program
register	CPU Register	Garbage	Within block	End of block



1. **auto:** This is the default storage class for all the variables declared inside a function or a block. Hence, the keyword auto is rarely used while writing programs in C language. Auto variables can be only accessed within the block/function they have been declared and not outside them (which defines their scope). Of course, these can be accessed within nested blocks within the parent block/function in which the auto variable was declared. However, they can be accessed outside their scope as well using the concept of pointers given here by pointing to the very exact memory location where the variables resides. They are assigned a garbage value by default whenever they are declared.

2. **extern:** Extern storage class simply tells us that the variable is defined elsewhere and not within the same block where it is used. Basically, the value is assigned to it in a different block and this can be overwritten/changed in a different block as well. So an extern variable is nothing but a global variable initialized with a legal value where it is declared in order to be used elsewhere. It can be accessed within any function/block. Also, a normal global variable can be made extern as well by placing the 'extern' keyword before its declaration/definition in any function/block. This basically signifies that we are not initializing a new variable but instead we are using/accessing the global variable only. The main purpose of using extern variables is that they can be accessed between two different files which are part of a large program. For more information on how extern variables work, have a look at this [link](#).

3. **static:** This storage class is used to declare static variables which are popularly used while writing programs in C language. Static variables have a property of preserving their value even after they are out of their scope! Hence, static variables preserve the value of their last use in their scope. So we can say that they are initialized only once and exist till the termination of the program. Thus, no new memory is allocated because they are not re-declared. Their scope is local to the function to which they were defined. Global static variables can be accessed anywhere in the program. By default, they are assigned the value 0 by the compiler.

4. **register:** This storage class declares register variables which have the same functionality as that of the auto variables. The only difference is that the compiler tries to store these variables in the register of the microprocessor if a free register is available. This makes the use of register variables to be much faster than that of the variables stored in the memory during the runtime of the program. If a free register is not available, these are then stored in the memory only. Usually few variables which are to be accessed very frequently in a program are declared with the register keyword which improves the running time of the program. An important and interesting point to be noted here is that we cannot obtain the address of a register variable using pointers.

To specify the storage class for a variable, the following syntax is to be followed:

Syntax:

```
storage_class var_data_type var_name;
```

Functions follow the same syntax as given above for variables. Have a look at the following C example for further clarification:

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// A C program to demonstrate different storage
// classes
#include <stdio.h>

// declaring the variable which is to be made extern
// an intial value can also be initialized to x
int x;

void autoStorageClass()
{

    printf("\nDemonstrating auto class\n\n");

    // declaring an auto variable (simply
    // writing "int a=32;" works as well)
    auto int a = 32;

    // printing the auto variable 'a'
    printf("Value of the variable 'a'"
           " declared as auto: %d\n",
           a);

    printf("-----");
}

void registerStorageClass()
{

    printf("\nDemonstrating register class\n\n");

    // declaring a register variable
    register char b = 'G';

    // printing the register variable 'b'
    printf("Value of the variable 'b'"
           " declared as register: %d\n",
           b);

    printf("-----");
}
```

```

void externStorageClass()
{
    printf("\nDemonstrating extern class\n\n");

    // telling the compiler that the variable
    // z is an extern variable and has been
    // defined elsewhere (above the main
    // function)
    extern int x;

    // printing the extern variables 'x'
    printf("Value of the variable 'x'"
        " declared as extern: %d\n",
        x);

    // value of extern variable x modified
    x = 2;

    // printing the modified values of
    // extern variables 'x'
    printf("Modified value of the variable 'x'"
        " declared as extern: %d\n",
        x);

    printf("-----");
}

void staticStorageClass()
{
    int i = 0;

    printf("\nDemonstrating static class\n\n");

    // using a static variable 'y'
    printf("Declaring 'y' as static inside the loop.\n"
        "But this declaration will occur only"
        " once as 'y' is static.\n"
        "If not, then every time the value of 'y' "
        "will be the declared value 5"
        " as in the case of variable 'p'\n");

    printf("\nLoop started:\n");

    for (i = 1; i < 5; i++) {

        // Declaring the static variable 'y'
        static int y = 5;

        // Declare a non-static variable 'p'
        int p = 10;

        // Incrementing the value of y and p by 1
        y++;
        p++;

        // printing value of y at each iteration
        printf("\nThe value of 'y', "
            "declared as static, in %d "
            "iteration is %d\n",
            i, y);

        // printing value of p at each iteration
        printf("The value of non-static variable 'p', "
            "in %d iteration is %d\n",
            i, p);
    }

    printf("\nLoop ended:\n");

    printf("-----");
}

int main()
{
    printf("A program to demonstrate"
        " Storage Classes in C\n\n");

    // To demonstrate auto Storage Class
    autoStorageClass();

    // To demonstrate register Storage Class
    registerStorageClass();

    // To demonstrate extern Storage Class
    externStorageClass();

    // To demonstrate static Storage Class
    staticStorageClass();
}

```

```
// exiting
printf("\n\nStorage Classes demonstrated;

return 0;
}

// This code is improved by RishabhPrabhu
```

filter_none

Output:

A program to demonstrate Storage Classes in C

Demonstrating auto class

Value of the variable 'a' declared as auto: 32

Demonstrating register class

Value of the variable 'b' declared as register: 71

Demonstrating extern class

Value of the variable 'x' declared as extern: 0

Modified value of the variable 'x' declared as extern: 2

Demonstrating static class

Declaring 'y' as static inside the loop.

But this declaration will occur only once as 'y' is static.

If not, then every time the value of 'y' will be the declared value 5 as in the case of variable 'p'

Loop started:

The value of 'y', declared as static, in 1 iteration is 6

The value of non-static variable 'p', in 1 iteration is 11

The value of 'y', declared as static, in 2 iteration is 7

The value of non-static variable 'p', in 2 iteration is 11

The value of 'y', declared as static, in 3 iteration is 8

The value of non-static variable 'p', in 3 iteration is 11

The value of 'y', declared as static, in 4 iteration is 9

The value of non-static variable 'p', in 4 iteration is 11

Loop ended:

Storage Classes demonstrated

Quiz on Storage Classes

This article is contributed by Ayush Jaggi. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- [C | Storage Classes and Type Qualifiers | Question 17](#)
- [C | Storage Classes and Type Qualifiers | Question 18](#)
- [C | Storage Classes and Type Qualifiers | Question 19](#)
- [C | Storage Classes and Type Qualifiers | Question 19](#)
- [C | Storage Classes and Type Qualifiers | Question 14](#)
- [C | Storage Classes and Type Qualifiers | Question 19](#)
- [C | Storage Classes and Type Qualifiers | Question 19](#)
- [C | Storage Classes and Type Qualifiers | Question 6](#)
- [C | Storage Classes and Type Qualifiers | Question 12](#)
- [C | Storage Classes and Type Qualifiers | Question 19](#)
- [C | Storage Classes and Type Qualifiers | Question 19](#)
- [C | Storage Classes and Type Qualifiers | Question 11](#)
- [C | Storage Classes and Type Qualifiers | Question 8](#)
- [C | Storage Classes and Type Qualifiers | Question 19](#)
- [C | Storage Classes and Type Qualifiers | Question 3](#)

Improved By : [skbarnwal](#), [RishabhPrabhu](#), [harkiran78](#)

Article Tags :

C

C Basics

C-Storage Classes and Type Qualifiers

Veritas

Practice Tags :

Veritas

C

31

To-do Done
2.6

Based on 51 vote(s)

Basic **Easy** **Medium** **Hard** **Expert**

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Weekend Geeks Classes Weekdays

Java APP DEVELOPMENT COMPETITIVE PROGRAMMING LIVE Java Backend Development

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS

Most popular in C
Program to calculate Electricity Bill
Program to print half Diamond star pattern
C/C++ Program to calculate Factorial
C/C++ #include directive with Examples
Output of C programs | Set 66 (Accessing Memory Locations)

More related articles in C
Difference between Type Casting and Type Conversion
Chain of Pointers in C with Examples
Print all possible combinations of the string by replacing '\$' with any other digit from the string
getch() function in C with Examples
Jagged Array Array of Arrays in C with Examples

Static Variables in C

Static variables have a property of preserving their value even after they are out of their scope! Hence, static variables preserve their previous value in their previous scope and are not initialized again in the new scope.

Syntax:

```
static data_type var_name = var_value;
```

Following are some interesting facts about static variables in C.

1) A static int variable remains in memory while the program is running. A normal or auto variable is destroyed when a function call where the variable was declared is over.

For example, we can use static int to count a number of times a function is called, but an auto variable can't be used for this purpose.

For example below program prints "1 2"

```
filter_none
edit
close
play_arrow
link
brightness_4
code

#include<stdio.h>
int fun()
{
    static int count = 0;
    count++;
    return count;
}
```

```
int main()
{
    printf("%d ", fun());
    printf("%d ", fun());
    return 0;
}
chevron_right
```

Output:

```
1 2
```

But below program prints 1 1

```
filter_none
edit
close
play_arrow
```

```
link  
brightness_4  
code  
  
#include<stdio.h>  
int fun()  
{  
    int count = 0;  
    count++;  
    return count;  
}  
  
int main()  
{  
    printf("%d ", fun());  
    printf("%d ", fun());  
    return 0;  
}
```

[chevron_right](#)
[filter_none](#)

Output:

```
1 1
```

2) Static variables are allocated memory in data segment, not stack segment. See [memory layout of C programs](#) for details.

3) Static variables (like global variables) are initialized as 0 if not initialized explicitly. For example in the below program, value of x is printed as 0, while value of y is something garbage. See [this](#) for more details.

```
filter_none  
edit  
close  
  
play_arrow  
link  
brightness_4  
code  
  
#include <stdio.h>  
int main()  
{  
    static int x;  
    int y;  
    printf("%d \n %d", x, y);  
}
```

[chevron_right](#)
[filter_none](#)

Output:

```
0  
[some_garbage_value]
```

4) In C, static variables can only be initialized using constant literals. For example, following program fails in compilation. See [this](#) for more details.

```
filter_none  
edit  
close  
  
play_arrow  
link  
brightness_4  
code  
  
#include<stdio.h>  
int initializer(void)  
{  
    return 50;  
}
```

```
int main()  
{  
    static int i = initializer();  
    printf(" value of i = %d", i);  
    getchar();  
    return 0;  
}
```

[chevron_right](#)
[filter_none](#)

Output

```
In function 'main':  
9:5: error: initializer element is not constant  
    static int i = initializer();  
    ^
```

Please note that this condition doesn't hold in C++. So if you save the program as a C++ program, it would compile \and run fine.

5) Static global variables and functions are also possible in C/C++. The purpose of these is to limit scope of a variable or function to a file. Please refer [Static functions in C](#) for more details.

6) Static variables should not be declared inside structure. The reason is C compiler requires the entire structure elements to be placed together (i.e.) memory allocation for structure members should be contiguous. It is possible to declare structure inside the function (stack segment) or allocate memory dynamically(heap segment) or it can be even global (BSS or data segment). Whatever might be the case, all structure members should reside in the same memory segment because the value for the structure element is fetched by counting the offset of the element from the beginning address of the structure. Separating out one member alone to data segment defeats the purpose of static variable and it is possible to have an entire structure as static.

Related Articles:

- Static Keyword in C++
- Quiz on Static Keyword
- Static data members in C++
- When are static objects destroyed?
- Interesting facts about static member functions
- Can static functions be virtual?
- Comparison of static keyword in C++ and Java
- Static functions in C

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes

Add your personal notes here! (max 5000 chars)

Recommended Posts:

- Difference between Static variables and Register variables in C
- Initialization of static variables in C
- Initialization of global and static variables in C
- How are variables scoped in C - Static or Dynamic?
- What are the default values of static variables in C?
- Internal static variable vs. External static variable with Examples in C
- Static functions in C
- C++ | Static Keyword | Question 2
- C++ | Static Keyword | Question 1
- C++ | Static Keyword | Question 6
- C++ | Static Keyword | Question 5
- C++ | Static Keyword | Question 4
- C++ | Static Keyword | Question 3
- Can static functions be virtual in C++?
- When are static objects destroyed?

Improved By : [DibyajyotiMohapatra, kani_26](#)

Article Tags :

C
C Basics
C-Storage Classes and Type Qualifiers

Practice Tags :

C



39

To-do Done
2

Based on 52 vote(s)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) Data Types in C

Next

[last_page](#) A C Programming Language Puzzle

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Weekend Geeks Classes Weekdays

JAVA APP DEVELOPMENT SUMMER TRAINING COMPETITIVE PROGRAMMING LITE Java Backend Development

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS



Understanding “extern” keyword in C

I'm sure this post will be as interesting and informative to C virgins (i.e. beginners) as it will be to those who are well-versed in C. So let me start by saying that the `extern` keyword applies to C variables (data objects) and C functions. Basically, the `extern` keyword extends the visibility of the C variables and C functions. That's probably the reason why it was named `extern`.

Though most people probably understand the difference between the “declaration” and the “definition” of a variable or function, for the sake of completeness, I would like to clarify them.

- **Declaration** of a variable or function simply declares that the variable or function exists somewhere in the program, but the memory is not allocated for them. The declaration of a variable or function serves an important role—it tells the program what its type is going to be. In case of *function* declarations, it also tells the program the arguments, their data types, the order of those arguments, and the return type of the function. So that's all about the declaration.
- Coming to the **definition**, when we *define* a variable or function, in addition to everything that a declaration does, it also allocates memory for that variable or function. Therefore, we can think of definition as a superset of the declaration (or declaration as a subset of definition).

A variable or function can be *declared* any number of times, but it can be *defined* only once. (Remember the basic principle that you can't have two locations of the same variable or function).

Now back to the `extern` keyword. First, Let's consider the use of `extern` in functions. It turns out that when a function is declared or defined, the `extern` keyword is implicitly assumed. When we write:

```
int foo(int arg1, char arg2);
```

The compiler treats it as:

```
extern int foo(int arg1, char arg2);
```

Since the `extern` keyword extends the function's visibility to the whole program, the function can be used (called) anywhere in any of the files of the whole program, provided those files contain a declaration of the function. (With the declaration of the function in place, the compiler knows the definition of the function exists somewhere else and it goes ahead and compiles the file). So that's all about `extern` and functions.

Now let's consider the use of `extern` with variables. To begin with, how would you *declare* a variable without *defining* it? You would do something like this:

```
extern int var;
```

Here, an integer type variable called `var` has been declared (it hasn't been defined yet, so no memory allocation for `var` so far). And we can do this declaration as many times as we want. So far so good.

Now, how would you *define* `var`? You would do this:

```
int var;
```

In this line, an integer type variable called `var` has been both declared **and** defined (remember that *definition* is the superset of *declaration*). Since this is a *definition*, the memory for `var` is also allocated. Now here comes the surprise. When we declared/defined a function, we saw that the `extern` keyword was present implicitly. But this isn't the case with variables. If it were, memory would never be allocated for them. Therefore, we need to include the `extern` keyword explicitly when we want to declare variables without defining them. Also, as the `extern` keyword extends the visibility to the whole program, by using the `extern` keyword with a variable, we can use the variable anywhere in the program provided we include its declaration the variable is defined somewhere.

Now let us try to understand `extern` with examples.

Example 1:

```
filter_none
edit
close

play_arrow

link
brightness_4
code

int var;
int main(void)
{
    var = 10;
    return 0;
}
chevron_right
filter_none
```

This program compiles successfully. `var` is defined (and declared implicitly) globally.

Example 2:

```
filter_none
edit
close

play_arrow

link
brightness_4
code

extern int var;
int main(void)
{
    return 0;
}
chevron_right
filter_none
```

This program compiles successfully. Here `var` is declared only. Notice `var` is never used so no problems arise.

Example 3:

```
filter_none
edit
close

play_arrow

link
brightness_4
code

extern int var;
int main(void)
```

```
{  
var = 10;  
return 0;  
}  
chevron_right  
filter_none
```

This program throws an error in compilation because var is declared but not defined anywhere. Essentially, the var isn't allocated any memory. And the program is trying to change the value to 10 of a variable that doesn't exist at all.

Example 4:

```
filter_none  
edit  
close  
play_arrow  
link  
brightness_4  
code  
  
#include "somefile.h"  
extern int var;  
int main(void)  
{  
    var = 10;  
    return 0;  
}  
chevron_right  
filter_none
```

Assuming that somefile.h contains the definition of var, this program will compile successfully.

Example 5:

```
filter_none  
edit  
close  
play_arrow  
link  
brightness_4  
code  
  
extern int var = 0;  
int main(void)  
{  
    var = 10;  
    return 0;  
}  
chevron_right  
filter_none
```

Do you think this program will work? Well, here comes another surprise from C standards. They say that..if a variable is only declared and an initializer is also provided with that declaration, then the memory for that variable will be allocated—in other words, that variable will be considered as defined. Therefore, as per the C standard, this program will compile successfully and work.

So that was a preliminary look at the extern keyword in C.

In short, we can say:

1. A declaration can be done any number of times but definition only once.
2. the extern keyword is used to extend the visibility of variables/functions.
3. Since functions are visible throughout the program by default, the use of extern is not needed in function declarations or definitions. Its use is implicit.
4. When extern is used with a variable, it's only declared, not defined.
5. As an exception, when an extern variable is declared with initialization, it is taken as the definition of the variable as well.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes [arrow_drop_up](#)

Add your personal notes
here! (max 5000 chars)

Save

Recommended Posts:

- Understanding "register" keyword in C
- Understanding nullptr in C++
- Understanding Hypothesis Testing
- Understanding constexpr specifier in C++
- Understanding "volatile" qualifier in C | Set 1 (Introduction)
- Understanding "volatile" qualifier in C | Set 2 (Examples)
- Understanding Lvalues, PRvalues and Xvalues in C/C++ with Examples
- restrict keyword in C
- _Generic keyword in C
- Use of explicit keyword in C++
- C++ mutable keyword
- C++ | Static Keyword | Question 1
- C++ | const keyword | Question 5
- C++ | Static Keyword | Question 2
- C++ | Static Keyword | Question 5

Improved By : [anupamgodse121](#), [NikiHerl](#), [derekhullinger](#)

Article Tags :

Articles

C

C-Storage Classes and Type Qualifiers

Practice Tags :

C

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) Little and Big Endian Mystery

Next

[last_page](#) Let's experiment with Networking

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

The image shows the top navigation bar of the GeeksforGeeks website. It features a red header with the text "Load Comments". Below the header is a dark blue navigation bar with several categories: "FULL STACK DEVELOPMENT", "JAVA APP DEVELOPMENT SUMMER TRAINING", "React.js LIVE", "Geeks Classes Weekend", "Geeks Classes Weekdays", "COMPETITIVE PROGRAMMING LIVE", "Java Backend Development", "Design Patterns", and "django WEB DEVELOPMENT". A prominent red banner in the center says "LIVE COURSES BY GEEKSFORGEEKS" with an illustration of two people looking at a screen. At the bottom of the page, there is a list of most popular articles and a "Most visited in C" section.

Most popular in Articles
new vs malloc() and free() vs delete in C++
How to change the default icon of Android App
Difference between sum of degrees of odd and even degree nodes in an Undirected Graph
When to use which sorting algorithms
Group all co-prime numbers from 1 to N

Most visited in C
Print in C++ with Examples
Speed up Code executions with help of Pragma in C/C++
Program to calculate Electricity Bill
Modulo Operator (%) In C/C++ with Examples
Role of SemiColon in various Programming Languages

What are the default values of static variables in C?

In C, if an object that has static storage duration is not initialized explicitly, then:

- if it has pointer type, it is initialized to a NULL pointer;
- if it has arithmetic type, it is initialized to (positive or unsigned) zero;
- if it is an aggregate, every member is initialized (recursively) according to these rules;
- if it is a union, the first named member is initialized (recursively) according to these rules.

For example, following program prints:

```
Value of g = 0
Value of sg = 0
Value of s = 0

filter_none
edit
close
play_arrow
link
brightness_4
code

#include<stdio.h>
int g; //g = 0, global objects have static storage duration
static int gs; //gs = 0, global static objects have static storage duration
int main()
{
    static int s; //s = 0, static objects have static storage duration
    printf("Value of g = %d", g);
    printf("\nValue of gs = %d", gs);
    printf("\nValue of s = %d", s);

    getchar();
    return 0;
}
chevron_right
filter_none
```

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

My Personal Notes [arrow_drop_up](#)

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- Difference between Static variables and Register variables in C
- Static Variables in C
- Initialization of static variables in C
- Initialization of global and static variables in C
- How are variables scoped in C - Static or Dynamic?
- Internal static variable vs. External static variable with Examples in C
- C++ Internals | Default Constructors | Set 1
- Templates and Default Arguments
- Some Interesting facts about default arguments in C++
- Default Assignment Operator and References
- Default arguments and virtual function
- C++ default constructor | Built-in types
- When does compiler create default and copy constructors in C++?
- C++ | Function Overloading and Default Arguments | Question 2
- C++ | Function Overloading and Default Arguments | Question 5

Article Tags :

C
C-Storage Classes and Type Qualifiers

Practice Tags :

C

16

To-do Done
1.4

Based on 66 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

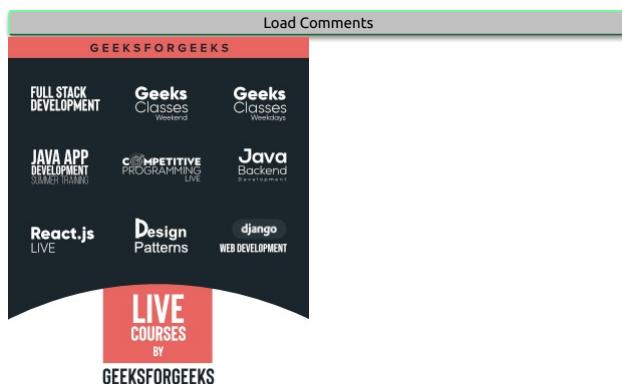
Previous

[first_page](#) Data type of case labels of switch statement in C++?

Next

[last_page](#) C/C++ Ternary Operator - Some Interesting Observations

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
ctype.h(<cctype>) library in C/C++ with Examples
C program to display month by month calendar for a given year
Predictable Macros in C with Examples
Implicit Type Conversion in C with Examples

More related articles in C
For loop specific to different Programming Languages
How to use function within function in C or C++
C program to print odd line contents of a file followed by even line content
Introduction to the C99 Programming Language : Part II
C program to find square root of a given number

Understanding “volatile” qualifier in C | Set 2 (Examples)

The volatile keyword is intended to prevent the compiler from applying any optimizations on objects that can change in ways that cannot be determined by the compiler.

Objects declared as volatile are omitted from optimization because their values can be changed by code outside the scope of current code at any time. The system always reads the current value of a volatile object from the memory location rather than keeping its value in temporary register at the point it is requested, even if a previous instruction asked for a value from the same object. So the simple question is, how can value of a variable change in such a way that compiler cannot predict. Consider the following cases for answer to this question.

1) *Global variables modified by an interrupt service routine outside the scope:* For example, a global variable can represent a data port (usually global pointer referred as memory mapped IO) which will be updated dynamically. The code reading data port must be declared as volatile in order to fetch latest data available at the port. Failing to declare variable as volatile, the compiler will optimize the code in such a way that it will read the port only once and keeps using the same value in a temporary register to speed up the program (speed optimization). In general, an ISR used to update these data port when there is an interrupt due to availability of new data

2) **Global variables within a multi-threaded application:** There are multiple ways for threads communication, viz, message passing, shared memory, mail boxes, etc. A global variable is weak form of shared memory. When two threads sharing information via global variable, they need to be qualified with volatile. Since threads run asynchronously, any update of global variable due to one thread should be fetched freshly by another consumer thread. Compiler can read the global variable and can place them in temporary variable of current thread context. To nullify the effect of compiler optimizations, such global variables need to be qualified as volatile

If we do not use volatile qualifier, the following problems may arise

- 1) Code may not work as expected when optimization is turned on.
- 2) Code may not work as expected when interrupts are enabled and used.

Let us see an example to understand how compilers interpret volatile keyword. Consider below code, we are changing value of const object using pointer and we are compiling code without optimization option. Hence compiler won't do any optimization and will change value of const object.

```
filter_none
edit
close

play_arrow
link
brightness_4
code

/* Compile code without optimization option */
#include <stdio.h>
int main(void)
{
    const int local = 10;
    int *ptr = (int*) &local;

    printf("Initial value of local : %d \n", local);

    *ptr = 100;

    printf("Modified value of local: %d \n", local);

    return 0;
}
chevron_right
```

When we compile code with “-save-temps” option of gcc it generates 3 output files

- 1) preprocessed code (having .i extention)
- 2) assembly code (having .s extention) and
- 3) object code (having .o option).

We compile code without optimization, that's why the size of assembly code will be larger (which is highlighted in red color below).

Output:

```
[narendra@ubuntu]$ gcc volatile.c -o volatile --save-temps
[narendra@ubuntu]$ ./volatile
Initial value of local : 10
Modified value of local: 100
[narendra@ubuntu]$ ls -l volatile.s
-rw-r--r-- 1 narendra narendra 731 2016-11-19 16:19 volatile.s
[narendra@ubuntu]$
```

Let us compile same code with optimization option (i.e. -O option). In the below code, “local” is declared as const (and non-volatile), GCC compiler does optimization and ignores the instructions which try to change value of const object. Hence value of const object remains same.

```
filter_none
edit
close

play_arrow
link
brightness_4
code

/* Compile code with optimization option */
#include <stdio.h>
```

```
int main(void)
{
    const int local = 10;
    int *ptr = (int*) &local;

    printf("Initial value of local : %d \n", local);

    *ptr = 100;

    printf("Modified value of local: %d \n", local);

    return 0;
}
```

For above code, compiler does optimization, that's why the size of assembly code will reduce.

Output:

```
[narendra@ubuntu]$ gcc -O3 volatile.c -o volatile --save-temps
[narendra@ubuntu]$ ./volatile
Initial value of local : 10
Modified value of local: 10
[narendra@ubuntu]$ ls -l volatile.s
-rw-r--r-- 1 narendra narendra 626 2016-11-19 16:21 volatile.s
```

Let us declare const object as volatile and compile code with optimization option. Although we compile code with optimization option, value of const object will change, because variable is declared as volatile that means don't do any optimization.

```
filter_none
edit
close
play_arrow
link
brightness_4
code
/* Compile code with optimization option */
#include <stdio.h>
```

```
int main(void)
{
    const volatile int local = 10;
    int *ptr = (int*) &local;

    printf("Initial value of local : %d \n", local);

    *ptr = 100;

    printf("Modified value of local: %d \n", local);

    return 0;
}
```

chevron_right

filter_none

Output:

```
[narendra@ubuntu]$ gcc -O3 volatile.c -o volatile --save-temp
[narendra@ubuntu]$ ./volatile
Initial value of local : 10
Modified value of local: 100
[narendra@ubuntu]$ ls -l volatile.s
-rw-r--r-- 1 narendra narendra 711 2016-11-19 16:22 volatile.s
[narendra@ubuntu]$
```

The above example may not be a good practical example, the purpose was to explain how compilers interpret volatile keyword. As a practical example, think of touch sensor on mobile phones. The driver abstracting touch sensor will read the location of touch and send it to higher level applications. The driver itself should not modify (const-ness) the read location, and make sure it reads the touch input every time fresh (volatile-ness). Such driver must read the touch sensor input in const volatile manner.

Note : The above codes are compiler specific and may not work on all compilers. The purpose of the examples is to make readers understand the concept.

Related Article :

[Understanding "volatile" qualifier in C | Set 1 \(Introduction\)](#)

Refer following links for more details on volatile keyword:

[Volatile: A programmer's best friend](#)

[Do not use volatile as a synchronization primitive](#)

This article is compiled by "Narendra Kangarkar" and reviewed by GeeksforGeeks team. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up

Add your personal notes
here! (max 5000 chars)

Save

Recommended Posts:

- [Understanding "volatile" qualifier in C | Set 1 \(Introduction\)](#)
- [Understanding Lvalues, PRvalues and Xvalues in C/C++ with Examples](#)
- [Const Qualifier in C](#)
- [Understanding nullptr in C++](#)
- [Understanding "register" keyword in C](#)
- [Understanding constexpr specifier in C++](#)
- [Understanding "extern" keyword in C](#)
- [P\(i\)n in C++ with Examples](#)
- [mbrtoc16\(\) in C/C++ with Examples](#)
- [mbrtoc32\(\) in C/C++ with Examples](#)
- [iswgraph\(\) in C/C++ with Examples](#)
- [SDL library in C/C++ with examples](#)
- [iswprint\(\) in C/C++ with Examples](#)
- [wmemset\(\) in C/C++ with Examples](#)
- [C/C++ if else statement with Examples](#)

Improved By : shohamziner

Article Tags :

C

[C-Storage Classes and Type Qualifiers](#)

Practice Tags :

C

thumb_up

30

To-do Done
3.2

Based on 86 vote(s)

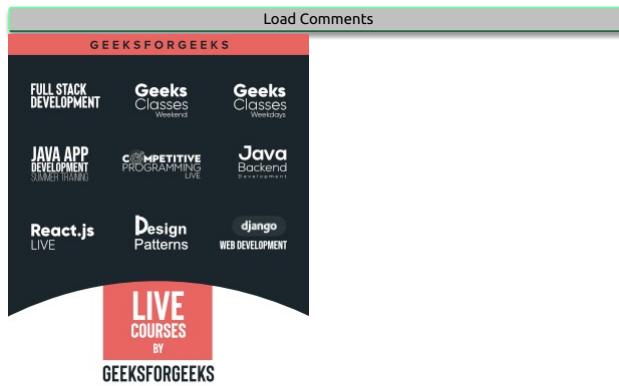
[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
[ctype.h](#) [cctype.h](#) library in C/C++ with Examples

Implementation of Convert in C/C++ Examples

Format specifiers in different Programming Languages

C program to find square root of a given number

More related articles in C

[Predefined Macros in C with Examples](#)

[How to call function within function in C or C++](#)

[C program to print odd line contents of a File followed by even line content](#)

[How to create GUI in C programming using GTK Toolkit](#)

[Introduction to the C99 Programming Language : Part II](#)

Const Qualifier in C

The qualifier const can be applied to the declaration of any variable to specify that its value will not be changed (Which depends upon where const variables are stored, we may change the value of const variable by using pointer). The result is implementation-defined if an attempt is made to change a const.

1) Pointer to variable.

```
filter_none
edit
close
play_arrow
link
brightness_4
code
int *ptr;
chevron_right
filter_none
```

We can change the value of ptr and we can also change the value of object ptr pointing to. Pointer and value pointed by pointer both are stored in the read-write area. See the following code fragment.

```
filter_none
edit
close
play_arrow
link
brightness_4
code
#include <stdio.h>
int main(void)
{
    int i = 10;
    int j = 20;
    int *ptr = &i;
    /* pointer to integer */
    printf("*ptr: %d\n", *ptr);

    /* pointer is pointing to another variable */
    ptr = &j;
    printf("*ptr: %d\n", *ptr);

    /* we can change value stored by pointer */
    *ptr = 100;
    printf("*ptr: %d\n", *ptr);

    return 0;
}
chevron_right
filter_none
```

Output:

```
*ptr: 10  
*ptr: 20  
*ptr: 100
```

2) Pointer to constant.

Pointer to constant can be declared in following two ways.

```
filter_none  
edit  
close  
play_arrow  
link  
brightness_4  
code  
const int *ptr;  
chevron_right  
filter_none
```

or

```
filter_none  
edit  
close  
play_arrow  
link  
brightness_4  
code  
int const *ptr;  
chevron_right  
filter_none
```

We can change the pointer to point to any other integer variable, but cannot change the value of the object (entity) pointed using pointer ptr. The pointer is stored in the read-write area (stack in the present case). The object pointed may be in the read-only or read-write area. Let us see the following examples.

```
filter_none  
edit  
close  
play_arrow  
link  
brightness_4  
code  
#include <stdio.h>  
int main(void)  
{  
    int i = 10;  
    int j = 20;  
    /* ptr is pointer to constant */  
    const int *ptr = &i;  
  
    printf("ptr: %d\n", *ptr);  
    /* error: object pointed cannot be modified  
       using the pointer ptr */  
    *ptr = 100;  
  
    ptr = &j; /* valid */  
    printf("ptr: %d\n", *ptr);  
  
    return 0;  
}  
chevron_right  
filter_none
```

Output:

```
error: assignment of read-only location '*ptr'
```

Following is another example where variable i itself is constant.

```
filter_none  
edit  
close  
play_arrow  
link  
brightness_4  
code  
#include <stdio.h>  
  
int main(void)  
{  
    /* i is stored in read only area*/  
    int const i = 10;  
    int j = 20;  
  
    /* pointer to integer constant. Here i  
       is of type "const int", and &i is of  
       type "const int *". And p is of type  
       "const int", types are matching no issue */  
    int const *ptr = &i;  
  
    printf("ptr: %d\n", *ptr);  
  
    /* error */
```

```
*ptr = 100;

/* valid. We call it up qualification. In
C/C++, the type of "int *" is allowed to up
qualify to the type "const int *". The type of
&j is "int *" and is implicitly up qualified by
the compiler to "const int *" */

ptr = &j;
printf("ptr: %d\n", *ptr);
```

```
return 0;
}
```

```
chevron_right
```

```
filter_none
```

Output:

```
error: assignment of read-only location '*ptr'
```

Down qualification is not allowed in C++ and may cause warnings in C. Following is another example with down qualification.

```
filter_none
```

```
edit
```

```
close
```

```
play_arrow
```

```
link
```

```
brightness_4
```

```
code
```

```
#include <stdio.h>
```

```
int main(void)
{
```

```
    int i = 10;
    int const j = 20;
```

```
    /* ptr is pointing an integer object */
    int *ptr = &i;
```

```
    printf("*ptr: %d\n", *ptr);
```

```
    /* The below assignment is invalid in C++, results in error
       In C, the compiler *may* throw a warning, but casting is
       implicitly allowed */
    ptr = &j;
```

```
    /* In C++, it is called 'down qualification'. The type of expression
       &j is "const int *" and the type of ptr is "int *". The
       assignment "ptr = &j" causes to implicitly remove const-ness
       from the expression &j. C++ being more type restrictive, will not
       allow implicit down qualification. However, C++ allows implicit
       up qualification. The reason being, const qualified identifiers
       are bound to be placed in read-only memory (but not always). If
       C++ allows above kind of assignment (ptr = &j), we can use 'ptr'
       to modify value of j which is in read-only memory. The
       consequences are implementation dependent, the program may fail
       at runtime. So strict type checking helps clean code. */

    printf("*ptr: %d\n", *ptr);
```

```
    return 0;
}
```

```
// Reference:
```

```
// http://www.dansaks.com/articles/1999-02%20const%20T%20vs%20T%20const.pdf
```

```
// More interesting stuff on C/C++ @ http://www.dansaks.com/articles.shtml
```

```
chevron_right
```

```
filter_none
```

3) Constant pointer to variable.

```
filter_none
```

```
edit
```

```
close
```

```
play_arrow
```

```
link
```

```
brightness_4
```

```
code
```

```
int *const ptr;
```

```
chevron_right
```

```
filter_none
```

Above declaration is a constant pointer to an integer variable, means we can change the value of object pointed by pointer, but cannot change the pointer to point another variable.

```
filter_none
```

```
edit
```

```
close
```

```
play_arrow
```

```
link
```

```
brightness_4
```

```
code
```

```
#include <stdio.h>

int main(void)
{
    int i = 10;
    int j = 20;
    /* constant pointer to integer */
    int *const ptr = &i;

    printf("ptr: %d\n", *ptr);

    *ptr = 100;      /* valid */
    printf("ptr: %d\n", *ptr);

    ptr = &j;        /* error */
    return 0;
}
```

chevron_right

filter_none

Output:

```
error: assignment of read-only variable 'ptr'
```

4) constant pointer to constant

filter_none
edit
close

play_arrow
link
brightness_4
code

```
const int *const ptr;
chevron_right
```

filter_none

Above declaration is a constant pointer to a constant variable which means we cannot change value pointed by the pointer as well as we cannot point the pointer to other variables. Let us see with an example.

filter_none
edit
close

play_arrow
link
brightness_4
code

```
#include <stdio.h>
```

```
int main(void)
{
    int i = 10;
    int j = 20;
    /* constant pointer to constant integer */
    const int *const ptr = &i;

    printf("ptr: %d\n", *ptr);

    ptr = &j;        /* error */
    *ptr = 100;     /* error */

    return 0;
}
```

chevron_right

filter_none

Output:

```
error: assignment of read-only variable 'ptr'
error: assignment of read-only location '*ptr'
```

This article is compiled by "**Narendra Kangalkar**". Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes *arrow_drop_up*

Add your personal notes
here! (max 5000 chars)

Recommended Posts:

- [Difference between const int*, const int * const, and int const *](#)
- [Difference between const char *p, char * const p and const char * const p](#)
- [Understanding "volatile" qualifier in C | Set 2 \(Examples\)](#)
- [Understanding "volatile" qualifier in C | Set 1 \(Introduction\)](#)
- [C++ | const keyword | Question 2](#)
- [C++ | const keyword | Question 5](#)
- [C++ | const keyword | Question 3](#)
- [Difference between #define and const in C?](#)
- [How to modify a const variable in C?](#)
- [C++ | const keyword | Question 1](#)

- C++ | const keyword | Question 5
- Function overloading and const keyword
- "static const" vs "#define" vs "enum"
- Why copy constructor argument should be const in C++?

Improved By : NiranjanBSubramanian, akshayna115

Article Tags :

C
C-Storage Classes and Type Qualifiers
pointer
Practice Tags :

C

65

To-do Done
2.8

Based on 135 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

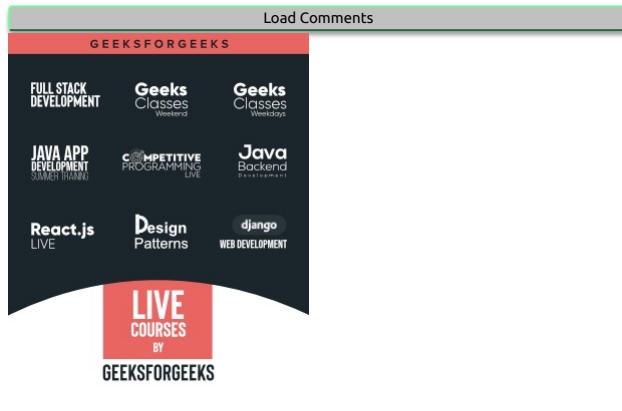
Previous

[first_page](#) Functions that are executed before and after main() in C

Next

[last_page](#) Why is the size of an empty class not zero in C++?

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C
`ctype.h <ctype.h>` library in C/C++ with Examples
Print all possible combinations of the string by replacing '\$' with any other digit from the string
C program to display month by month calendar for a given year
Predefined Macros in C with Examples
Implicit Type Conversion in C with Examples

More related articles in C
How to use the printf() function in C or C++
C program to print odd line contents of a file followed by even line content
Introduction to the C99 Programming Language : Part II
C program to find square root of a given number
Format specifiers in different Programming Languages

Initialization of static variables in C

In C, static variables can only be initialized using constant literals. For example, following program fails in compilation.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

#include<stdio.h>
int initializer(void)
{
    return 50;
}

int main()
{
    static int i = initializer();
    printf(" value of i = %d", i);
    getchar();
    return 0;
}
chevron_right
filter_none
```

If we change the program to following, then it works without any error.

```
filter_none
edit
close
play_arrow
link
brightness_4
code
#include<stdio.h>
int main()
{
    static int i = 50;
    printf(" value of i = %d", i);
    getchar();
    return 0;
}
```

[chevron_right](#)

[filter_none](#)

The reason for this is simple: All objects with static storage duration must be initialized (set to their initial values) before execution of main() starts. So a value which is not known at translation time cannot be used for initialization of static variables.

Thanks to Venki and Prateek for their contribution.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes [arrow_drop_up](#)

Add your personal notes
here! (max 5000 chars)

[Save](#)

Recommended Posts:

- [Initialization of global and static variables in C](#)
- [Difference between Static variables and Register variables in C](#)
- [Implicit initialization of variables with 0 or 1 in C](#)
- [Initialization of variables sized arrays in C](#)
- [Static Variables in C](#)
- [How are variables scoped in C - Static or Dynamic?](#)
- [What are the default values of static variables in C?](#)
- [Internal static variable vs. External static variable with Examples in C](#)
- [Initialization of a multidimensional arrays in C/C++](#)
- [Initialization of data members](#)
- [Static functions in C](#)
- [C++ | Static Keyword | Question 6](#)
- [C++ | Static Keyword | Question 5](#)
- [C++ | Static Keyword | Question 4](#)
- [C++ | Static Keyword | Question 3](#)

Article Tags :

[C](#)
[C-Storage Classes and Type Qualifiers](#)

Practice Tags :

[C](#)

[thumb_up](#)
22

To-do Done
1.9

Based on 61 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) [Result of comma operator as l-value in C and C++](#)

Next

[last_page](#) [How Linkers Resolve Global Symbols Defined at Multiple Places?](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

[Load Comments](#)



Most popular in C
`ctype.h<<ctype.h>` library in C/C++ with Examples
Print all possible combinations of the string by replacing '\$' with any other digit from the string
C program to display month by month calendar for a given year
Predefined Macro in C with Examples
Implicit Type Conversion in C with Examples

More related articles in C
How to call function within function in C or C++
C program to print contents of file followed by even line content
Introduction to the C99 Programming Language - Part II
C program to find square root of a given number
Format specifiers in different Programming Languages

Understanding “register” keyword in C

Registers are faster than memory to access, so the variables which are most frequently used in a C program can be put in registers using `register` keyword. The keyword `register` hints to compiler that a given variable can be put in a register. It's compiler's choice to put it in a register or not. Generally, compilers themselves do optimizations and put the variables in register.

1) If you use `&` operator with a register variable then compiler may give an error or warning (depending upon the compiler you are using), because when we say a variable is a register, it may be stored in a register instead of memory and accessing address of a register is invalid. Try below program.

filter_none
edit
close
play_arrow
link
brightness_4
code
`#include<stdio.h>`

```
int main()
{
    register int i = 10;
    int* a = &i;
    printf("%d", *a);
    getchar();
    return 0;
}
```

2) `register` keyword can be used with pointer variables. Obviously, a register can have address of a memory location. There would not be any problem with the below program.

filter_none
edit
close
play_arrow
link
brightness_4
code
`#include<stdio.h>`

```
int main()
{
    int i = 10;
    register int* a = &i;
    printf("%d", *a);
    getchar();
    return 0;
}
```

3) Register is a storage class, and C doesn't allow multiple storage class specifiers for a variable. So, `register` can not be used with `static`. Try below program.

filter_none
edit
close
play_arrow
link

```
brightness_4
code
#include<stdio.h>

int main()
{
    int i = 10;
    register static int* a = &i;
    printf("%d", *a);
    getchar();
    return 0;
}
```

chevron_right
filter_none

4) Register can only be used within a block (local), it can not be used in the *global* scope (outside main).

```
filter_none
edit
close
play_arrow
link
brightness_4
code
#include <stdio.h>
```

```
// error (global scope)
register int x = 10;
int main()
{
    // works (inside a block)
    register int i = 10;
    printf("%d\n", i);
    printf("%d", x);
    return 0;
}
```

chevron_right
filter_none

Compile Errors:

```
prog.c:3:14: error: register name not specified for 'x'
register int x = 10;//error (global scope)
^
```

5) There is no limit on number of register variables in a C program, but the point is compiler may put some variables in register and some not.

Please write comments if you find anything incorrect in the above article or you want to share more information about register keyword.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up

Save

Recommended Posts:

- Understanding "extern" keyword in C
- Understanding nullptr in C++
- Understanding Hypothesis Testing
- Understanding constexpr specifier in C++
- Understanding "volatile" qualifier in C | Set 1 (Introduction)
- Understanding "volatile" qualifier in C | Set 2 (Examples)
- Understanding Lvalues, PRvalues and Xvalues in C/C++ with Examples
- restrict keyword in C
- _Generic keyword in C
- Use of explicit keyword in C++
- C++ mutable keyword
- C++ | Static Keyword | Question 1
- C++ | const keyword | Question 5
- C++ | Static Keyword | Question 2
- C++ | Static Keyword | Question 5

Improved By : AnasKhedr, Ankur Trivedi

Article Tags :

Articles
C
c-puzzle
C-Storage Classes and Type Qualifiers
Practice Tags :
C

thumb_up
33

To-do Done
1.8

Based on 81 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Previous

[first_page](#) What are Wild Pointers? How can we avoid?

Next

[last_page](#) Return values of printf() and scanf() in C/C++

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Weekend Geeks Classes Weekdays

JAVA APP DEVELOPMENT COMPETITIVE PROGRAMMING LITE Java Backend Development

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS



Most popular in Articles
new vs malloc() and free() vs delete in C++
How to change the default icon of Android App
Difference Between sum of degrees of odd and even degree nodes in an Undirected Graph
When to use each sorting algorithms
Group all co-prime numbers from 1 to N

Most visited in C
P(i) in C++ with Examples
SemiColon Code executions with help of Pragma in C/C++
Program to calculate Electricity Bill
Modulo Operator (%) in C/C++ with Examples
Role of SemiColon in various Programming Languages

C Storage Classes and Type Qualifiers

1234

Question 1

Which of the following is not a storage class specifier in C?

A auto
B register
C static
D extern
E volatile
F typedef

C Storage Classes and Type Qualifiers

Discuss it

Question 1 Explanation:

volatile is not a **storage class specifier**. volatile and const are **type qualifiers**.

Question 2

Output of following program?

```
#include <stdio.h>
int main()
{
    static int i=5;
    if(--i){
        main();
        printf("%d ",i);
    }
}
```

A 4 3 2 1

B 1 2 3 4

C 0 0 0 0

D Compiler Error

C Storage Classes and Type Qualifiers

Discuss it

Question 2 Explanation:

A static variable is shared among all calls of a function. All calls to main() in the given program share the same i. i becomes 0 before the printf() statement in all calls to main().

Question 3

```
#include <stdio.h>
int main()
{
    static int i=5;
    if (--i){
        printf("%d ",i);
        main();
    }
}
```

A 4 3 2 1

B 1 2 3 4

C 4 4 4 4

D 0 0 0 0

C Storage Classes and Type Qualifiers

Discuss it

Question 3 Explanation:

Since i is static variable, it is shared among all calls to main(). So is reduced by 1 by every function call.

Question 4

```
#include <stdio.h>
int main()
{
    int x = 5;
    int * const ptr = &x;
    ++(*ptr);
    printf("%d", x);

    return 0;
}
```

ACompiler Error
BRuntime Error
C6
D5

C Storage Classes and Type Qualifiers

Discuss it

Question 4 Explanation:

See following declarations to know the difference between constant pointer and a pointer to a constant. `int * const ptr` → ptr is constant pointer. You can change the value at the location pointed by pointer p, but you can not change p to point to other location. `int const * ptr` → ptr is a pointer to a constant. You can change ptr to point other variable. But you cannot change the value pointed by ptr. Therefore above program works well because we have a constant pointer and we are not changing ptr to point to any other location. We are only incrementing value pointed by ptr.

Question 5

```
#include <stdio.h>
int main()
{
    int x = 5;
    int const * ptr = &x;
    ++(*ptr);
    printf("%d", x);

    return 0;
}
```

ACompiler Error
BRuntime Error
C6
D5

C Storage Classes and Type Qualifiers

Discuss it

Question 5 Explanation:

See following declarations to know the difference between constant pointer and a pointer to a constant. `int * const ptr` → ptr is constant pointer. You can change the value at the location pointed by pointer p, but you can not change p to point to other location. `int const * ptr` → ptr is a pointer to a constant. You can change ptr to point other variable. But you cannot change the value pointed by ptr. In the above program, ptr is a pointer to a constant. So the value pointed cannot be changed.

Question 6

```
#include<stdio.h>
int main()
{
    typedef static int *i;
    int j;
    i a = &j;
    printf("%d", *a);
    return 0;
}
```

ARuntime Error
B0
CGarbage Value
DCompiler Error

C Storage Classes and Type Qualifiers

Discuss it

Question 6 Explanation:

Compiler Error -> Multiple Storage classes for a. In C, `typedef` is considered as a `storage class`. The Error message may be different on different compilers.

Question 7

Output?

```
#include<stdio.h>
int main()
{
    typedef int i;
    i a = 0;
    printf("%d", a);
    return 0;
}
```

ACompiler Error
BRuntime Error
C0
D1

C Storage Classes and Type Qualifiers

Discuss it

Question 7 Explanation:

There is no problem with the program. It simply creates a user defined type i and creates a variable a of type i.

Question 8

```
#include<stdio.h>
int main()
{
    typedef int *i;
    int j = 10;
    i *a = &j;
    printf("%d", **a);
    return 0;
}
```

ACompiler Error
BGarbage Value
C10
D0

C Storage Classes and Type Qualifiers

Discuss it

Question 8 Explanation:

Compiler Error -> Initialization with incompatible pointer type. The line `typedef int *i` makes i as type `int *`. So, the declaration of a means a is pointer to a pointer. The Error message may be different on different compilers.

Question 9

Output?

```
#include <stdio.h>
int fun()
{
    static int num = 16;
    return num--;
}

int main()
{
    for(fun(); fun(); fun())
        printf("%d ", fun());
    return 0;
}
```

AInfinite loop
B13 10 7 4 1
C14 11 8 5 2
D15 12 8 5 2

C Storage Classes and Type Qualifiers

Discuss it

Question 9 Explanation:

Since `num` is static in `fun()`, the old value of `num` is preserved for subsequent function calls. Also, since the statement `return num-` is postfix, it returns the old value of `num`, and updates the value for next function call.

`fun()` called first time: num = 16 // for loop initialization done;

```

In test condition, compiler checks for non zero value
fun() called again : num = 15
printf("%d \n", fun());:num=14 ->printed
Increment/decrement condition check
fun(); called again : num = 13
-----
fun() called second time: num: 13

In test condition,compiler checks for non zero value
fun() called again : num = 12
printf("%d \n", fun());:num=11 ->printed
fun(); called again : num = 10
-----
fun() called second time : num = 10

In test condition,compiler checks for non zero value
fun() called again : num = 9
printf("%d \n", fun());:num=8 ->printed
fun(); called again : num = 7
-----
fun() called second time: num = 7

In test condition,compiler checks for non zero value
fun() called again : num = 6
printf("%d \n", fun());:num=5 ->printed
fun(); called again : num = 4
-----
fun() called second time: num: 4

In test condition,compiler checks for non zero value
fun() called again : num = 3
printf("%d \n", fun());:num=2 ->printed
fun(); called again : num = 1
-----
fun() called second time: num: 1

In test condition,compiler checks for non zero value
fun() called again : num = 0 => STOP

```

Question 10

```

#include <stdio.h>
int main()
{
    int x = 10;
    static int y = x;

    if(x == y)
        printf("Equal");
    else if(x > y)
        printf("Greater");
    else
        printf("Less");
    return 0;
}

```

ACompiler Error

BEqual

CGreater

DLess

C Storage Classes and Type Qualifiers

Discuss it

Question 10 Explanation:

In C, static variables can only be initialized using constant literals. This is allowed in C++ though. See this GFact for details.

There are 31 questions to complete.

1234

My Personal Notes [arrow_drop_up](#)

Add your personal notes here! (max 5000 chars)

[Save](#)

Understanding “volatile” qualifier in C | Set 1 (Introduction)

In spite of tons of literature on C language, “**volatile**” keyword is somehow not understood well (even by experienced C programmers). We think that the main reason for this is due to not having real-world use-case of a ‘**volatile**’ variable in typical C programs that are written in high level language. Basically, unless you’re doing some low level hardware programming in C, you probably won’t use a variable while is qualified as “**volatile**”. By low level programming, we mean a piece of C code which is dealing with peripheral devices, IO ports (mainly memory mapped IO ports), Interrupt Service Routines (ISRs) which interact with Hardware. That’s why it’s not so straight forward to have a sample working C program which can easily show-case the exact effect of “**volatile**” keyword.

In fact, in this article, if we could explain the meaning and purpose of ‘**volatile**’, it would serve as basic groundwork for further study and use of ‘**volatile**’ in C. To understand ‘**volatile**’, we first need to have some background about what a compiler does to a C program. At high level, we know that a compiler converts C code to Machine code so that the executable can be run without having actual source code. Similar to other technologies, compiler technology had also evolved a lot. While translating Source code to Machine code, compilers typically try to optimize the output so that lesser Machine code needs to be executed finally. One such optimization is removing unnecessary Machine code for accessing variable which is not changing from Compiler’s perspective. Suppose we have the following code:

```

filter_none
edit
close
play_arrow
link

```

```

brightness_4
code

uint32 status = 0;

while (status == 0)
{
    /*Let us assume that status isn't being changed
     in this while loop or may be in our whole program*/
}

chevron_right


---


filter_none


---



```

An optimizing Compiler would see that *status* isn't being changed by *while* loop. So there's no need to access *status* variable again and again after each iteration of loop. So the Compiler would convert this loop to a infinite loop i.e. *(1)* so that the Machine code to read *status* isn't needed. Please note that compiler isn't aware of that *status* is a special variable which can be changed from outside the current program at any point of time e.g. some IO operation happened on a peripheral device for which device IO port was memory mapped to this variable. So in reality, we want compiler to access *status* variable after every loop iteration even though it isn't modified by our program which is being compiled by Compiler.

One can argue that we can turn-off all the compiler optimizations for such programs so that we don't run into this situation. This is not an option due to multiple reasons such as
A) Each compiler implementation is different so it's not a portable solution
B) Just because of one variable, we don't want to turn off all the other optimizations which compiler does at other portions of our program.
C) By turning off all the optimizations, our low level program couldn't work as expected e.g. too much increase in size or delayed execution.

That's where "**volatile**" comes in picture. Basically, we need to instruct Compiler that *status* is special variable so that no such optimization are allowed on this variable. With this, we would define our variable as follows:

```

filter_none
edit
close

play_arrow

link
brightness_4
code

volatile uint32 status = 0;
chevron_right


---


filter_none


---



```

For simplicity of explanation purpose, we chose the above example. But in general, **volatile** is used with pointers such as follows:

```

filter_none
edit
close

play_arrow

link
brightness_4
code

volatile uint32 * statusPtr = 0xF1230000
chevron_right


---


filter_none


---



```

Here, *statusPtr* is pointing to a memory location (e.g. for some IO port) at which the content can change at any point of time from some peripheral device. Please note that our program might not have any control or knowledge about when that memory would change. So we would make it "**volatile**" so that compiler doesn't perform optimization for the *volatile* variable which is pointed by *statusPtr*.

In the context of our discussion about "**volatile**", we quote C language standard i.e. ISO/IEC 9899 C11 - clause 6.7.3

"An object that has volatile-qualified type may be modified in ways unknown to the implementation or have other unknown side effects."

"A volatile declaration may be used to describe an object corresponding to a memory-mapped input/output port or an object accessed by an asynchronously interrupting function. Actions on objects so declared shall not be "optimized out" by an implementation or reordered except as permitted by the rules for evaluating expressions."

Basically, C standard says that "**volatile**" variables can change from outside the program and that's why compilers aren't supposed to optimize their access. Now, you can guess that having too many '**volatile**' variables in your program would also result in lesser & lesser compiler optimization. We hope it gives you enough background about meaning and purpose of "**volatile**".

From this article, we would like you to take-away the concept of "**volatile variable -> don't do compiler optimization for that variable**"!

The following article explains volatile through more examples.

[Understanding "volatile" qualifier in C | Set 2 \(Examples\)](#)

If you like GeeksforGeeks and would like to contribute, you can also write an article using [contribute.geeksforgeeks.org](#) or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes **arrow_drop_up**

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- [Understanding "volatile" qualifier in C | Set 2 \(Examples\)](#)
- [Const Qualifier in C](#)
- [Understanding nullptr in C++](#)
- [Understanding "extern" keyword in C](#)
- [Understanding constexpr specifier in C++](#)
- [Understanding "register" keyword in C](#)
- [Understanding Lvalues, PRvalues and Xvalues in C/C++ with Examples](#)
- [C Language Introduction](#)
- [Namespace in C++ | Set 1 \(Introduction\)](#)
- [Hygienic Macros : An Introduction](#)
- [Introduction of Smart Pointers in C++ and It's Types](#)
- [Pointers in C and C++ | Set 1 \(Introduction, Arithmetic and Array\)](#)
- [OpenMP | Introduction with Installation Guide](#)

- Introduction to the C99 Programming Language : Part II
- Introduction to the C99 Programming Language : Part I

Article Tags :

C
C-Storage Classes and Type Qualifiers
Practice Tags :

thumb_up
36

To-do Done
2.6

Based on 38 vote(s)

Basic | Easy | Medium | Hard | Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

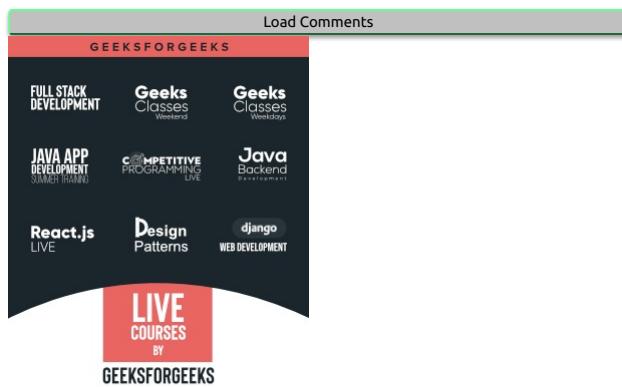
Previous

first_page Printing source code of a C program itself

Next

last_page _attribute_((constructor)) and _attribute_((destructor)) syntaxes in C

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.



Most popular in C
Print all possible combinations of the string by replacing 's' with any other digit from the string
Format specifiers in different Programming Languages
C program to find square root of a given number
Predefined Macros in C with Examples
C program to print odd line contents of a File followed by even line content

More related articles in C
Introduction to the C99 Programming Language : Part II
Features of C Programming Language
Problem in comparing Floating point numbers and how to compare them correctly?
<cmath> float.h in C/C++ with Examples
Getting System and Process Information Using C Programming and Shell in Linux

Return values of printf() and scanf() in C/C++

What values do the **printf()** and **scanf()** functions return ?

- **printf()** : It returns **total number of Characters Printed**, Or negative value if an output error or an encoding error

Example 1: The printf() function in the code written below returns 6. As 'CODING' contains 6 characters.

filter_none

edit

close

play_arrow

link

brightness_4

code

```
// C/C++ program to demonstrate return value
// of printf()
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    char st[] = "CODING";
```

```
    printf("While printing ");
    printf(", the value returned by printf() is : %d",
           printf("%s", st));
```

```
    return 0;
}
```

chevron_right

filter_none

Output: While printing CODING, the value returned by printf() is : 6

Example 2: The printf() function in the code written below returns 9. As '123456789' contains 9 characters.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// C/C++ program to demonstrate return value
// of printf()

#include <stdio.h>
int main()
{
    long int n = 123456789;

    printf("While printing ");
    printf(", the value returned by printf() is : %d",
           printf("%d", n));

    return 0;
}
```

chevron_right

filter_none

Output: While printing 123456789, the value returned by printf() is : 9

- scandf() : It returns **total number of Inputs Scanned successfully**, or EOF if input failure occurs before the first receiving argument was assigned.

Example 1: The first scanf() function in the code written below returns 1, as it is scanning 1 item. Similarly second scanf() returns 2 as it is scanning 2 inputs and third scanf() returns 3 as it is scanning 3 inputs.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// C/C++ program to demonstrate return value
// of printf()

#include <stdio.h>
int main()
{
    // scanf() with one input
    printf("\n First scanf() returns : %d",
           scanf("%s", a));

    // scanf() with two inputs
    printf("\n Second scanf() returns : %d",
           scanf("%s%s", a, b));

    // scanf() with three inputs
    printf("\n Third scanf() returns : %d",
           scanf("%s%s%s", a, b, c));

    return 0;
}
```

chevron_right

filter_none

```
Input:
Hey!
welcome to
geeks for geeks
Output:
First scanf() returns : 1
Second scanf() returns : 2
Third scanf() returns : 3
```

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up

Add your personal notes
here! (max 5000 chars)

Save

Recommended Posts:

- Use of & in scanf() but not in printf()
- Cin-Cout vs Scanf-Printf
- Nested printf (printf inside printf) in C
- C function argument and return values
- How can I return multiple values from a function?
- How to return multiple values from a function in C or C++?
- Problem with scanf() when there is fgets()/gets()/scanf() after it
- scanf("%[^\\n]s", str) Vs gets(str) in C with Examples

- Why to use fgets() over scanf() in C?
- Difference between scanf() and gets() in C
- Why "%s" is not used for strings in scanf() function?
- scanf() and fscanf() in C - Simple Yet Powerful
- What is use of %n in printf() ?
- How to print % using printf()?
- Execution of printf with ++ operators

Improved By : [mdamircoder](#)

Article Tags :

C
GFacts
C-Input and Output Quiz
cpp-input-output
Practice Tags :
C



18

To-do Done
1.6

Based on 91 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) Understanding "register" keyword in C

Next

[last_page](#) What is return type of getchar(), fgetc() and getc() ?

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Weekend Geeks Classes Weekdays

JAVA APP DEVELOPMENT COMPETITIVE PROGRAMMING LIVE Java Backend Development

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS

Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
ctype.h <cctype> library in C/C++ with Examples
Predefined Macros in C with Examples
Implicit Type Conversion in C with Examples
How to call function within function in C or C++

Most visited in GFacts
Interesting and Cool Tricks in Java
Interview Questions Part 1
How to Install and Configure MongoDB in Ubuntu?

What is return type of getchar(), fgetc() and getc() ?

In C, return type of getchar(), fgetc() and getc() is int (not char). So it is recommended to assign the returned values of these functions to an integer type variable.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

char ch; /* May cause problems */
while ((ch = getchar()) != EOF)
{
    putchar(ch);
}
chevron_right
filter_none
```

Here is a version that uses integer to compare the value of getchar().

```
filter_none
edit
close
play_arrow
```

```

link
brightness_4
code

int in;
while ((in = getchar()) != EOF)
{
    putchar(in);
}
chevron_right
filter_none

```

See [this](#) for more details.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes [arrow_drop_up](#)

Add your personal notes here! (max 5000 chars)

[Save](#)

Recommended Posts:

- Difference between `getc()`, `getchar()`, `getch()` and `getche()`
- Implicit return type `int` in C
- `EOF`, `getc()` and `feof()` in C
- `fgetc()` and `fputc()` in C
- Difference between Type Casting and Type Conversion
- What does `main()` return in C and C++?
- Return from void functions in C++
- `return` statement in C/C++ with Examples
- C function argument and return values
- How to return multiple values from a function in C or C++?
- `return` statement vs `exit()` in `main()`
- Return values of `printf()` and `scanf()` in C/C++
- How can I return multiple values from a function?
- Type Conversion in C
- What is data type of `FILE` in C ?

Article Tags :

C
GFacts
C-Input and Output Quiz
Practice Tags :
C

[thumb_up](#)
10

To-do Done
2.1

Based on 50 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) Return values of `printf()` and `scanf()` in C/C++

Next

[last_page](#) Difference Between `malloc()` and `calloc()` with Examples

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

[Load Comments](#)

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Weekend Geeks Classes Weekdays

JAVA APP DEVELOPMENT COMPETITIVE PROGRAMMING DN Java Backend

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS

Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
`ctype.h(<cctype>)` library in C/C++ with Examples
Implicit Type Conversion in C with Examples
How to call function within function in C or C++
Format specifiers in different Programming Languages

Most visited in GFacts
Interesting and Cool Tricks in Java

Scansets in C

scanf family functions support scanset specifiers which are represented by %[]. Inside scanset, we can specify single character or range of characters. While processing scanset, scanf will process only those characters which are part of scanset. We can define scanset by putting characters inside square brackets. Please note that the scansets are case-sensitive.

Let us see with example. Below example will store only capital letters to character array 'str', any other character will not be stored inside character array.

```
filter_none
edit
close

play_arrow

link
brightness_4
code

/* A simple scanset example */
#include <stdio.h>

int main(void)
{
    char str[128];

    printf("Enter a string: ");
    scanf("%[A-Z]s", str);

    printf("You entered: %s\n", str);

    return 0;
}
chevron_right
```

If first character of scanset is '^', then the specifier will stop reading after first occurrence of that character. For example, given below scanset will read all characters but stops after first occurrence of 'o'

```
filter_none
edit
close

play_arrow

link
brightness_4
code

scanf("%[^o]s", str);
chevron_right
```

Let us see with example.

```
filter_none
edit
close

play_arrow

link
brightness_4
code

/* Another scanset example with ^ */
#include <stdio.h>

int main(void)
{
    char str[128];

    printf("Enter a string: ");
    scanf("%[^o]s", str);

    printf("You entered: %s\n", str);

    return 0;
}
chevron_right
```

```
[root@centos-6 ~]# ./scan-set
Enter a string: http://geeks for geeks
You entered: http://geeks f
[root@centos-6 ~]#
```

Let us implement gets() function by using scan set. gets() function reads a line from stdin into the buffer pointed to by s until either a terminating newline or EOF found.

```
filter_none
edit
close

play_arrow

link
brightness_4
code

/* implementation of gets() function using scanset */
#include <stdio.h>
```

```

int main(void)
{
    char str[128];

    printf("Enter a string with spaces: ");
    scanf("%[^\\n]s", str);

    printf("You entered: %s\\n", str);

    return 0;
}

```

chevron_right

filter_none

```
[root@centos-6 ~]# ./gets
Enter a string with spaces: Geeks For Geeks
You entered: Geeks For Geeks
[root@centos-6 ~]#
```

As a side note, using `gets()` may not be a good idea in general. Check below note from Linux man page.

Never use `gets()`. Because it is impossible to tell without knowing the data in advance how many characters `gets()` will read, and because `gets()` will continue to store characters past the end of the buffer, it is extremely dangerous to use. It has been used to break computer security. Use `fgets()` instead. Also see this post.

This article is compiled by "Narendra Kangalkar" and reviewed by GeeksforGeeks team. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes *arrow_drop_up*

Add your personal notes
here! (max 5000 chars)

Save

Recommended Posts:

- [Format specifiers in different Programming Languages](#)
- [C program to find square root of a given number](#)
- [C program to print odd line contents of a File followed by even line content](#)
- [Getting System and Process Information Using C Programming and Shell in Linux](#)
- [Program to calculate Electricity Bill](#)
- [Program to print half Diamond star pattern](#)
- [C/C++ program for calling main\(\) in main\(\)](#)
- [Print all possible combinations of the string by replacing '\\$' with any other digit from the string](#)
- [How to use make utility to build C projects?](#)
- [How to call function within function in C or C++](#)
- [Role of SemiColon in various Programming Languages](#)
- [C program to display month by month calendar for a given year](#)
- [Difference between Type Casting and Type Conversion](#)
- [Pi\(\$\pi\$ \) in C++ with Examples](#)

Improved By : [tornado711](#), [nidhi_biet](#)

Article Tags :

[C](#)
[C-Input and Output Quiz](#)

Practice Tags :

[C](#)

thumb_up
17

To-do Done
1.8

Based on 45 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) [Initialization of variables sized arrays in C](#)

Next

[last_page](#) [Use of explicit keyword in C++](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
`ctype.h <cctype>` library in C/C++ with Examples
Predefined Macros in C with Examples
Implicit Type Conversion in C with Examples
Format specifiers in different Programming Languages

More related articles in C
How to call function within function in C or C++
C program to print contents of file followed by even line content
Introduction to the C99 Programming Language - Part II
C program to find square root of a given number
Problem in comparing Floating point numbers and how to compare them correctly?

puts() vs printf() for printing a string

In C, given a string variable `str`, which of the following two should be preferred to print it to `stdout`?

- 1) `puts(str);`
- 2) `printf(str);`

`puts()` can be preferred for printing a string because it is generally less expensive (implementation of `puts()` is generally simpler than `printf()`), and if the string has formatting characters like '%', then `printf()` would give unexpected results. Also, if `str` is a user input string, then use of `printf()` might cause security issues (see [this](#) for details).

Also note that `puts()` moves the cursor to next line. If you do not want the cursor to be moved to next line, then you can use following variation of `puts()`.

```
fputs(str, stdout)
```

You can try following programs for testing the above discussed differences between `puts()` and `printf()`.

Program 1

```
filter_none
edit
close
play_arrow
link
brightness_4
code

#include<stdio.h>
int main()
{
    puts("Geeksfor");
    puts("Geeks");

    getchar();
    return 0;
}
```

Program 2

```
filter_none
edit
close
play_arrow
link
brightness_4
code

#include<stdio.h>
int main()
{
    fputs("Geeksfor", stdout);
    fputs("Geeks", stdout);

    getchar();
    return 0;
}
```

Program 3

```
filter_none
edit
```

close
play_arrow
link
brightness_4
code

```
#include<stdio.h>
int main()
{
    // % is intentionally put here to show side effects of using printf(str)
    printf("Geek%sforGeek%s");
    getchar();
    return 0;
}
```

chevron_right

filter_none

Program 4

filter_none

edit

close

play_arrow

link

brightness_4

code

chevron_right

filter_none

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes *arrow_drop_up*

Add your personal notes
here! (max 5000 chars)

Save

Recommended Posts:

- Nested printf (printf inside printf) in C
- What is use of %n in printf() ?
- Use of & in scanf() but not in printf()
- How to print % using printf()?
- Cin-Cout vs Scanf-Printf
- Passing NULL to printf in C
- Execution of printf with ++ operators
- What is the difference between printf, sprintf and fprintf?
- Return values of printf() and scanf() in C/C++
- How to change the output of printf() in main() ?
- Printing source code of a C program itself
- Print all possible combinations of the string by replacing '\$' with any other digit from the string
- Print substring of a given string without using any string function and loop in C
- How to find length of a string without string.h and loop in C?
- Tokenizing a string in C++

Article Tags :

C

C-Input and Output Quiz

Practice Tags :

C

thumb_up
27

To-do Done
1.7

Based on 62 vote(s)

Basic **Easy** **Medium** **Hard** **Expert**

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) How Linkers Resolve Global Symbols Defined at Multiple Places?

Next

[last_page](#) CRASH() macro - interpretation

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
ctype.h(<cctype>) library in C/C++ with Examples

Implicit Type Conversion in C with Examples
Format specifiers in different Programming Languages
C program to find square root of a given number

More related articles in C
Predefined Macros in C with Examples
How to call a function with a function in C or C++
C program to print end line contents of a file followed by even line content
Introduction to the C99 Programming Language : Part II
Features of C Programming Language

What is use of %n in printf() ?

In C printf(), %n is a special format specifier which instead of printing something causes printf() to load the variable pointed by the corresponding argument with a value equal to the number of characters that have been printed by printf() before the occurrence of %n.

%n in printf()

```
printf("geeks for %ngeeks ", &c);
```

There are 10 characters
before the %n
inside the printf() method

Hence the value 10
will be stored in c

```
close
play_arrow
link
brightness_4
code
#include<stdio.h>

int main()
{
    int c;
    printf("geeks for %ngeeks ", &c);
    printf("%d", c);
    getchar();
    return 0;
}
chevron_right
filter_none
```

The above program prints "geeks for geeks 10". The first printf() prints "geeks for geeks". The second printf() prints 10 as there are 10 characters printed (the 10 characters are "geeks for ") before %n in first printf() and c is set to 10 by first printf().

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes [arrow_drop_up](#)

Add your personal notes
here! (max 5000 chars)

[Save](#)

Recommended Posts:

- Nested printf (printf inside printf) in C
- Use of & in scanf() but not in printf()
- How to print % using printf()?
- Execution of printf with ++ operators
- Cin-Cout vs Scanf-Printf
- Passing NULL to printf in C
- How to change the output of printf() in main() ?
- puts() vs printf() for printing a string
- Return values of printf() and scanf() in C/C++
- What is the difference between printf, sprintf and fprintf?
- Format specifiers in different Programming Languages
- C program to find square root of a given number
- C program to print odd line contents of a File followed by even line content
- Getting System and Process Information Using C Programming and Shell in Linux

Article Tags :

C
C-Input and Output Quiz

Practice Tags :

C

[thumb_up](#)

18

To-do Done
1.8

Based on 56 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) "delete this" in C++

Next

[last_page](#) Initialization of data members

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

[Load Comments](#)



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
`ctype.h <ctype.h>` library in C/C++ with Examples
Predefined Macros in C with Examples
Implicit Type Conversion in C with Examples
How to call function within function in C or C++
More related articles in C
C program to print odd line contents of a File followed by even line content
Introduction to C/C++ Programs - Lecture 2 : Part II
C program to find square root of a given number
Format specifiers in different Programming Languages
Problem in comparing Floating point numbers and how to compare them correctly?

How to print % using printf()?

Asked by Tanuj

Here is the standard prototype of printf function in C.

```
int printf(const char *format, ...);
```

The format string is composed of zero or more directives: ordinary characters (not %), which are copied unchanged to the output stream; and conversion specifications, each of argument (and it is an error if insufficiently many arguments are given).

The character % is followed by one of the following characters.

The flag character
The field width
The precision
The length modifier
The conversion specifier:

See <http://swoolley.org/man.cgi/3/print> for details of all the above characters. The main thing to note in the standard is the below line about conversion specifier.

```
A '%' is written. No argument is converted. The complete conversion specification is `%%'.
```

So we can print "%" using "%%"

```
filter_none
edit
close
play_arrow
link
brightness_4
code

/* Program to print */
#include<stdio.h>
/* Program to print */
int main()
{
    printf("%%");
    getchar();
    return 0;
}
```

chevron_right

filter_none

We can also print "%" using below.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

printf("%c", '%');
printf("%s", "%");
chevron_right
```

filter_none

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up

Add your personal notes here! (max 5000 chars)

Recommended Posts:

- Nested printf (printf inside printf) in C
- What is use of %n in printf() ?
- Use of & in scanf() but not in printf()
- Execution of printf with ++ operators
- Cin-Cout vs Scanf-Printf
- Passing NULL to printf in C
- How to change the output of printf() in main() ?
- What is the difference between printf, sprint and fprintf?
- Return values of printf() and scanf() in C/C++
- puts() vs printf() for printing a string
- How to print a variable name in C?
- Print Hello World without semicolon in C/C++
- How will you print numbers from 1 to 100 without using loop? | Set-2
- How to Read and Print an Integer value in C
- Print a long int in C using putchar() only

Article Tags :

C
C-Input and Output Quiz
c-puzzle

Practice Tags :

C



18

To-do Done

2

Based on 62 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

first_page Write a C macro PRINT(x) which prints x

Next

last_page How to declare a pointer to a function?

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

[Load Comments](#)

GEEKSFORGEEKS

FULL STACK DEVELOPMENT
Geeks Classes Weekend
Geeks Classes Weekdays

JAVA APP DEVELOPMENT SUMMER TRAINING
COMPETITIVE PROGRAMMING LIVE
Java Backend Development

React.js LIVE
Design Patterns
django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS



Most popular in C
 Print all possible combinations of the string by replacing '\$' with any other digit from the string
`ctype.h<<ctype>>` library in C/C++ with Examples
 C program to display month by month calendar for a given year
 Predefined Macros in C with Examples
 Implicit Type Conversion in C with Examples

More related articles in C
 Introduction to the C99 Programming Language : Part I
 How to call function within function in C or C++
 C program to read the contents of a file followed by even line content
 Nested Loops in C with Examples
 Introduction to the C99 Programming Language : Part II

C Input and Output

123

Question 1

Predict the output of following program?

```
#include "stdio.h"
int main()
{
    char arr[100];
    printf("%d", scanf("%s", arr));
    /* Suppose that input value given
       for above scanf is "GeeksQuiz" */

    return 1;
}
```

A9

B1

C10

D100

C Input and Output

Discuss it

Question 1 Explanation:

In C, scanf returns the no. of inputs it has successfully read. See <http://geeksforgeeks.org/archives/674>

Question 2

Predict output of the following program

```
#include <stdio.h>
int main()
{
    printf("new_c_questionby");
    printf("geeksforgeeks");
    getchar();
    return 0;
}
```

Anew_c_question

geeksforgeeks

Bnew_c_ques

geeksforgeeks

C

geeksforgeeks

DDepends on terminal configuration

C Input and Output

Discuss it

Question 2 Explanation:

See <http://stackoverflow.com/questions/17236242/usage-of-b-and-r-in-c>

Question 3

```
#include <stdio.h>

int main()
{
    printf(" GEEKS %% FOR %% GEEKS");
    getchar();
    return 0;
}
```

A"GEEKS % FOR % GEEKS"

BGEEKS % FOR % GEEKS

CGEEEKS %% FOR %% GEEKS%

DGEEKS %% FOR %% GEEKS

C Input and Output

Discuss it

Question 3 Explanation:

Backslash (\\\\") works as escape character for double quote ("). For explanation of %%, see <http://www.geeksforgeeks.org/how-to-print-using-printf/>

Question 4

```
#include <stdio.h>
// Assume base address of "GeeksQuiz" to be 1000
int main()
{
    printf(5 + "GeeksQuiz");
    return 0;
}
```

AGeeksQuiz

BQuiz

C1005

DCompile-time error

C Input and Output

Discuss it

Question 4 Explanation:

printf is a library function defined under stdio.h header file. The compiler adds 5 to the base address of the string through the expression **5 + "GeeksQuiz"**. Then the string "Quiz" gets passed to the standard library function as an argument.

Question 5

Predict the output of the below program:

```
#include <stdio.h>

int main()
{
    printf("%c ", 5["GeeksQuiz"]);
    return 0;
}
```

ACompile-time error

BRuntime error

CQ

Ds

C Input and Output

Discuss it

Question 5 Explanation:

The crux of the program lies in the expression: **5["GeeksQuiz"]** This expression is broken down by the compiler as: *(**5 + "GeeksQuiz"**). Adding 5 to the base address of the string increments the pointer(let's say a pointer was pointing to the start(**G**) of the string initially) to point to **Q**. Applying **value-of** operator gives the character at the location pointed to by the pointer i.e. Q.

Question 6

Predict the output of below program:

```
#include <stdio.h>
int main()
{
    printf("%c ", "GeeksQuiz"[5]);
    return 0;
}
```

ACompile-time error

BRuntime error

CQ

Ds

C Input and Output

Discuss it

Question 6 Explanation:

The crux of the program lies in the expression: "**GeeksQuiz**[5]. This expression is broken down by the compiler as: *("GeeksQuiz" + 5). Adding 5 to the base address of the string increments the pointer(let's say a pointer was pointing to the start(**G**) of the string initially) to point to Q. Applying **value-of** operator gives the character at the location pointed to by the pointer i.e. Q.

Question 7

Which of the following is true

Agets() can read a string with newline characters but a normal scanf() with %s can not.

Bgets() can read a string with spaces but a normal scanf() with %s can not.

Cgets() can always replace scanf() without any additional code.

DNone of the above

C Input and Output

Discuss it

Question 7 Explanation:

gets() can read a string with spaces but a normal scanf() with %s can not. Consider following program as an example. If we enter "Geeks Quiz" as input in below program, the program prints "Geeks" But in the following program, if we enter "Geeks Quiz", it prints "Geeks Quiz"

Question 8

Which of the following is true

Agets() doesn't do any array bound testing and should not be used.

Bfgets() should be used in place of gets() only for files, otherwise gets() is fine

Cgets() cannot read strings with spaces

DNone of the above

C Input and Output

Discuss it

Question 8 Explanation:

See `gets()` is risky to use!

Question 9

What does the following C statement mean?

`scanf("%4s", str);`

A Read exactly 4 characters from console.

B Read maximum 4 characters from console.

C Read a string str in multiples of 4

D Nothing

C Input and Output**Discuss it**

Question 9 Explanation:

Try following program, enter GeeksQuiz, the output would be "Geek"

`#include <stdio.h>`

```
int main()
{
    char str[50] = {0};
    scanf("%4s", str);
    printf(str);
    getchar();
    return 0;
}
```

Question 10

`#include<stdio.h>`

```
int main()
{
    char *s = "Geeks Quiz";
    int n = 7;
    printf("%.%s", n, s);
    return 0;
}
```

AGeeks Quiz

BNothing is printed

CGeeks Q

DGeeks Qu

C Input and Output**Discuss it**

Question 10 Explanation:

.* means The precision is not specified in the format string, but as an additional integer value argument preceding the argument that has to be formatted.

There are 25 questions to complete.

123

My Personal Notes 

Add your personal notes
here! (max 5000 chars)



What is the difference between printf, sprintf and fprintf?

printf:

printf function is used to print character stream of data on stdout console.

Syntax :`int printf(const char* str, ...);`**Example :**

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// simple print on stdout
#include<stdio.h>
int main()
{
    printf("hello geeksquiz");
    return 0;
}
```

Output :`hello geeksquiz`**sprintf:**

Syntax:

`int sprintf(char *str, const char *string,...);`

String print function instead of printing on console store it on char buffer which are specified in sprintf

Example :

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// Example program to demonstrate sprintf()
#include<stdio.h>
int main()
{
    char buffer[50];
```

```
int a = 10, b = 20, c;
c = a + b;
sprintf(buffer, "Sum of %d and %d is %d", a, b, c);

// The string "sum of 10 and 20 is 30" is stored
// into buffer instead of printing on stdout
printf("%s", buffer);

return 0;
}
```

chevron_right

filter_none

Output :

```
Sum of 10 and 20 is 30
```

printf:

printf is used to print the string content in file but not on stdout console.

```
int fprintf(FILE *fptr, const char *str, ...);
```

Example :

```
filter_none
edit
close

play_arrow
link
brightness_4
code

#include<stdio.h>
int main()
{
    int i, n=2;
    char str[50];

    //open file sample.txt in write mode
    FILE *fptr = fopen("sample.txt", "w");
    if (fptr == NULL)
    {
        printf("Could not open file");
        return 0;
    }

    for (i=0; i<n; i++)
    {
        puts("Enter a name");
        gets(str);
        fprintf(fptr,"%d.%s\n", i, str);
    }
    fclose(fptr);

    return 0;
}
```

chevron_right

filter_none

Input: GeeksforGeeks

GeeksQuiz

Output : sample.txt file now having output as

0. GeeksforGeeks

1. GeeksQuiz

filter_right

filter_none

Thank you for reading, i will soon update with scanf, fscanf, sscanf keep tuned.

This article is contributed by **Vankayala Karunakar**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes **arrow_drop_up**

Add your personal notes
here! (max 5000 chars)

Save

Recommended Posts:

- Nested printf (printf inside printf) in C
- fprintf() in C
- sprintf() in C
- What is use of %n in printf() ?
- How to print % using printf()?
- Use of & in scanf() but not in printf()
- Passing NULL to printf in C
- Cin-Cout vs Scanf-Printf
- Execution of printf with ++ operators
- puts() vs printf() for printing a string
- Return values of printf() and scanf() in C/C++
- How to change the output of printf() in main() ?
- Difference between C and C#
- Difference between C and C++
- Difference between ++*p, *p++ and *++p

Improved By : [mikerua](#)

Article Tags :

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

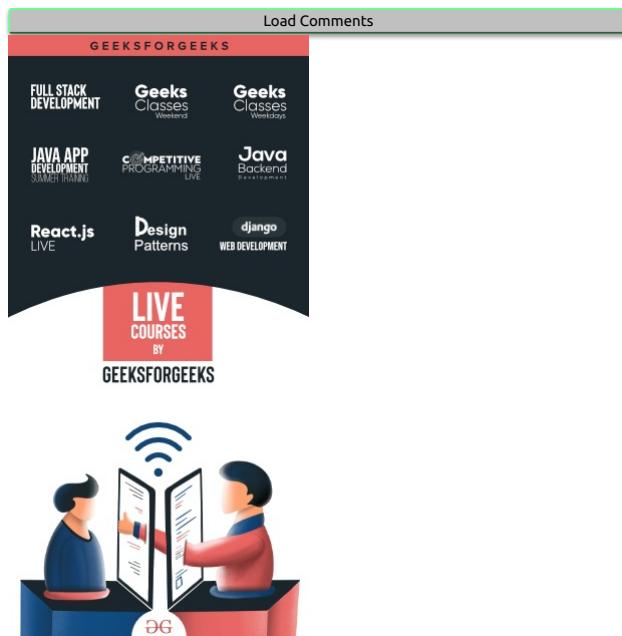
Previous

[first_page](#) Some Interesting facts about default arguments in C++

Next

[last_page](#) Difference between getc(), getchar(), getch() and getche()

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C
C program to display month-by-month calendar for a given year
Prepend all combinations of the string by replacing '\$' with any other digit from the string ctype.h<cctype> library in C/C++ with Examples
Implicit Type Conversion in C with Examples
Introduction to the C99 Programming Language : Part I

More related articles in C

Format specifiers in different Programming Languages

C program to find square root of a given number

Predefined Macros in C with Examples

How to call function within function in C or C++

C program to print odd line contents of a File followed by even line content

Difference between getc(), getchar(), getch() and getche()

All of these functions read a character from input and return an integer value. The integer is returned to accommodate a special value used to indicate failure. The value EOF is generally used for this purpose.

getc():

It reads a single character from a given input stream and returns the corresponding integer value (typically ASCII value of read character) on success. It returns EOF on failure.

Syntax:

```
int getc(FILE *stream);
```

Example:

```
filter_none
edit
close

play_arrow

link
brightness_4
code

// Example for getc() in C
#include <stdio.h>
int main()
{
    printf("%c", getc(stdin));
    return(0);
}
```

chevron_right

filter_none

```
Input: g (press enter key)
Output: g
```

An Example Application : [C program to compare two files and report mismatches](#)

getchar():

The difference between getc() and getchar() is getc() can read from any input stream, but getchar() reads from standard input. So getchar() is equivalent to getc(stdin).

Syntax:

```
int getch(void);
```

Example:

```
filter_none
edit
close

play_arrow
link
brightness_4
code

// Example for getch() in C
#include <stdio.h>
int main()
{
    printf("%c", getch());
    return 0;
}
```

chevron_right

filter_none

```
Input: g(press enter key)
Output: g
```

getch():

getch() is a nonstandard function and is present in conio.h header file which is mostly used by MS-DOS compilers like Turbo C. It is not part of the C standard library or ISO C, nor is it defined by POSIX (Source: <http://en.wikipedia.org/wiki/Conio.h>)

Like above functions, it reads also a single character from keyboard. But it does not use any buffer, so the entered character is immediately returned without waiting for the enter key.

Syntax:

```
int getch();
```

Example:

```
filter_none
edit
close

play_arrow
link
brightness_4
code

// Example for getch() in C
#include <stdio.h>
#include <conio.h>
int main()
{
    printf("%c", getch());
    return 0;
}
```

chevron_right

filter_none

```
Input: g (Without enter key)
Output: Program terminates immediately.
        But when you use DOS shell in Turbo C,
        it shows a single g, i.e., 'g'
```

getche()

Like getch(), this is also a non-standard function present in conio.h. It reads a single character from the keyboard and displays immediately on output screen without waiting for enter key.

Syntax:

```
int getche(void);
```

Example:

```
filter_none
edit
close

play_arrow
link
brightness_4
code

#include <stdio.h>
#include <conio.h>
// Example for getche() in C
int main()
{
    printf("%c", getche());
    return 0;
}
```

chevron_right

filter_none

```
Input: g(without enter key as it is not buffered)
Output: Program terminates immediately.
        But when you use DOS shell in Turbo C,
        double g, i.e., 'gg'
```

This article is contributed by **Vankayala Karunakar**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes 

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- What is return type of getchar(), fgetc() and getc() ?
- EOF, getc() and feof() in C
- getch() function in C with Examples
- Difference between C and C#
- Difference between C and C++
- Difference between ++p, *p++ and *++p
- Difference between while(1) and while(0) in C language
- Difference between scanf() and gets() in C
- Difference between C and Python
- Difference between while and do-while loop in C, C++, Java
- Difference between Java and C language
- Difference between Structure and Array in C
- Difference between fork() and exec()
- Difference between float and double in C/C++
- Difference between Structure and Union in C

Article Tags :

C
C-Input and Output Quiz
c-input-output

Practice Tags :

C



18

To-do Done
2

Based on 32 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) What is the difference between printf, sprintf and fprintf?

Next

[last_page](#) Write your own memcpy() and memmove()

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Weekend Geeks Classes Weekdays

JAVA APP DEVELOPMENT COMPETITIVE PROGRAMMING LIVE Java Backend

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS

Most popular in C
C program to display month by month calendar for a given year
Print all possible combinations of the string by replacing 'S' with any other digit from the string
ctype.h(<cctype>) library in C/C++ with Examples
Implicit Type Conversion in C with Examples
Introduction to the C99 Programming Language : Part I

More related articles in C
Format specifiers in different Programming Languages
C program to find square root of a given number
Preprocessor Macros in C with Examples
How to call function within function in C or C++
C program to print odd line contents of a File followed by even line content

Difference between %d and %i format specifier in C language

A format specifier is a sequence formed by an initial percentage sign (%) indicates a format specifier, which is used to specify the type and format of the data to be retrieved from the stream and stored into the locations pointed by the additional arguments. In short it tell us which type of data to store and which type of data to print.

For example - If we want to read and print integer using scanf() and printf() function, either %i or %d is used but there is subtle difference in both %i and %d format specifier.

%d specifies signed decimal integer while %i specifies integer.

%d and %i behave similar with printf

There is no difference between the %i and %d format specifiers for printf. Consider a following example.

filter_none
edit
close

play_arrow

link

brightness_4

code

```
// C program to demonstrate
// the behavior of %i and %d
// with printf statement
#include <stdio.h>
```

int main()

{

int num = 9;

// print value using %d

printf("Value of num using %d is = %d\n", num);

// print value using %i

printf("Value of num using %i is = %i\n", num);

return 0;

}

chevron_right

filter_none

Output:

Value of num using %d is = 9

Value of num using %i is = 9

%d and %i behavior is different in scanf

%d assume base 10 while %i auto detects the base. Therefore, both specifiers behaves differently while they are used with an input specifier. So, 012 would be 10 with %i but 12 with %d.

- %d takes integer value as signed decimal integer i.e. it takes negative values along with positive values but values should be in decimal otherwise it will print garbage value. Consider a following example.
- %i takes integer value as integer value with decimal, hexadecimal or octal type.

To enter a value in hexadecimal format - value should be provided by preceding "0x" and value in octal format - value should be provided by preceding "0".

Consider a following example.

filter_none

edit

close

play_arrow

link

brightness_4

code

```
// C program to demonstrate the difference
```

```
// between %i and %d specifier
```

```
#include <stdio.h>
```

int main()

{

int a, b, c;

printf("Enter value of a in decimal format:");

scanf("%d", &a);

printf("Enter value of b in octal format: ");

scanf("%i", &b);

printf("Enter value of c in hexadecimal format: ");

scanf("%i", &c);

printf("a = %i, b = %i, c = %i", a, b, c);

return 0;

}

chevron_right

filter_none

Output:

Enter value of a in decimal format:12

Enter value of b in octal format: 012

Enter value of c in hexadecimal format: 0x12

a = 12, b = 10, c = 18

Explanation:

The decimal value of a as 12 is 12

The decimal value of b as 12(octal) is 10

The decimal value of c as 12(hexadecimal) is 18

This article is contributed by Shubham Bansal. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer Placement 100 for details

My Personal Notes *arrow_drop_up*

Add your personal notes
here! (max 5000 chars)

Save

Recommended Posts:

- Difference Between Assembly Language And Machine Language
- Difference Between Machine Language and Assembly Language
- Difference between while(1) and while(0) in C language

- Difference between Java and C language
- Difference Between Go and Python Programming Language
- Difference between Compiled and Interpreted Language
- Difference between Structured Query Language (SQL) and Transact-SQL (T-SQL)
- What is the difference between a language construct and a “built-in” function in PHP ?
- C++ final specifier
- _Noreturn function specifier in C
- Understanding constexpr specifier in C++
- Difference between Procedural and Non-Procedural language
- Format specifiers in C
- Format String Vulnerability and Prevention with Example
- Format specifiers in different Programming Languages

Article Tags :

C
Difference Between
Practice Tags :

C



15

To-do Done
2.2

Based on 15 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

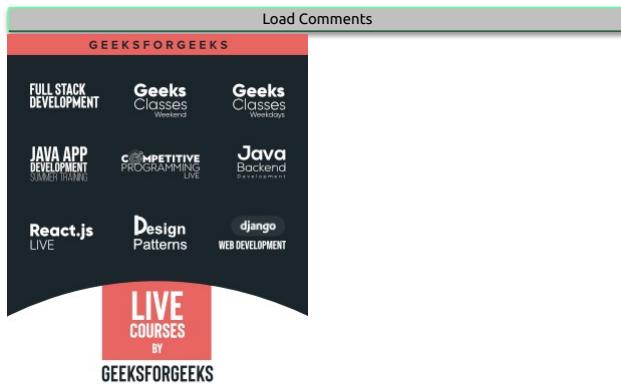
Previous

[first_page](#) C vs BASH Fork bomb

Next

[last_page](#) Bresenham's circle drawing algorithm

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
[ctype.h <ctype.h>](#) library in C/C++ with Examples
C program to display month by month calendar for a given year
Predefined Macros in C with Examples
Implicit Type Conversion in C with Examples

Most visited in Difference Between
[Difference between PostgreSQL and MongoDB](#)
[Difference between COUNT and TRUNCATE](#)
[Difference Between SMO and SEO](#)
[Difference between Prim's and Kruskal's algorithm for MST](#)
[Monolithic vs Microservices architecture](#)

Use of fflush(stdin) in C

fflush() is typically used for output stream only. Its purpose is to clear (or flush) the output buffer and move the buffered data to console (in case of stdout) or disk (in case of file output stream). Below is its syntax.

```
fflush(FILE *ostream);

ostream points to an output stream
or an update stream in which the
most recent operation was not input,
the fflush function causes any
unwritten data for that stream to
be delivered to the host environment
to be written to the file; otherwise,
the behavior is undefined.
```

Can we use it for input stream like stdin?

As per C standard, it is undefined behavior to use fflush(stdin). However some compilers like Microsoft visual studio allows it allow it. How is it used in these in these compilers? While taking an input string with spaces, the buffer does not get cleared for the next input and considers the previous input for the same. To solve this problem fflush(stdin) is used to clear the stream/buffer.

filter_none
edit
close
play_arrow
link
brightness_4
code

```
// C program to illustrate situation
// where flush(stdin) is required only
// in certain compilers.
#include <stdio.h>
#include<stdlib.h>
int main()
{
    char str[20];
    int i;
    for (i=0; i<2; i++)
    {
        scanf("%[^\\n]s", str);
        printf("%s\\n", str);
        // fflush(stdin);
    }
    return 0;
}
```

chevron_right

filter_none

Input:

```
geeks
geeksforgeeks
```

Output:

```
geeks
geeks
```

The code above takes only single input and gives the same result for the second input. Reason is because as the string is already stored in the buffer i.e. stream is not cleared yet as it was expecting string with spaces or new line. So, to handle this situation fflush(stdin) is used.

```
filter_none
edit
close

play_arrow

link
brightness_4
code

// C program to illustrate flush(stdin)
// This program works as expected only
// in certain compilers like Microsoft
// visual studio.
#include <stdio.h>
#include<stdlib.h>
int main()
{
    char str[20];
    int i;
    for (i = 0; i<2; i++)
    {
        scanf("%[^\\n]s", str);
        printf("%s\\n", str);

        // used to clear the buffer
        // and accept the next string
        fflush(stdin);
    }
    return 0;
}
```

chevron_right

filter_none

Input:

```
geeks
geeksforgeeks
```

Output:

```
geeks
geeksforgeeks
```

Is it good to use fflush(stdin)?

Although using "fflush(stdin)" after "scanf()" statement also clears the input buffer in certain compilers, it is not recommended to use it as it is undefined behavior by language standard. In C and C++, we have different methods to clear the buffer discussed in this [post](#).

Reference:

<https://stackoverflow.com/questions/2979209/using-fflushstdin>

This article is contributed by **Sahil Chhabra**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- Format specifiers in different Programming Languages

- C program to find square root of a given number
- C program to print odd line contents of a File followed by even line content
- Getting System and Process Information Using C Programming and Shell in Linux
- Program to calculate Electricity Bill
- Program to print half Diamond star pattern
- C/C++ program for calling main() in main()
- Print all possible combinations of the string by replacing '\$' with any other digit from the string
- How to use make utility to build C projects?
- How to call function within function in C or C++
- Role of SemiColon in various Programming Languages
- C program to display month by month calendar for a given year
- Difference between Type Casting and Type Conversion
- Pi(r) in C++ with Examples
- Output of C programs | Set 66 (Accessing Memory Locations)

Article Tags :

C
C-Input and Output Quiz
c-input-output

Practice Tags :

C

12

To-do Done

2.6

Based on 13 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

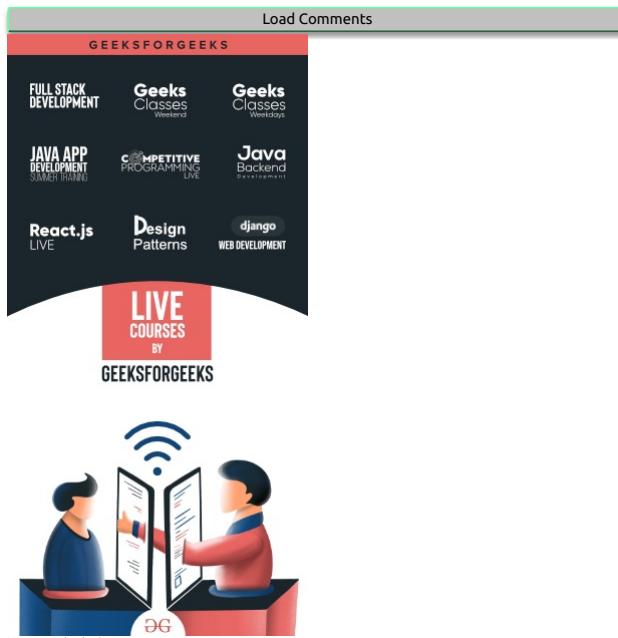
Previous

[first_page strcpy in C/C++](#)

Next

[last_page rand\(\) and srand\(\) in C/C++](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
ctype, cctype library in C/C++ with Examples
C programs display month by month calendar for a given year
Predefined Macros in C with Examples
Implicit Type Conversion in C with Examples

More related articles in C
Introduction to the C99 Programming Language : Part I
How to call function within function in C or C++
C program print odd line contents of a File followed by even line content
Nested Loops in C with Examples
Introduction to the C99 Programming Language : Part II

Clearing The Input Buffer In C/C++

What is a buffer?

A temporary storage area is called buffer. All standard input and output devices contain an input and output buffer. In standard C/C++, streams are buffered, for example in the case of standard input, when we press the key on keyboard, it isn't sent to your program, rather it is buffered by operating system till the time is allotted to that program.

How does it effect Programming?

On various occasions you may need to clear the unwanted buffer so as to get the next input in the desired container and not in the buffer of previous variable. For example, in case of C after encountering "scanf()", if we need to input a character array or character ,and in case of C++, after encountering "cin" statement, we require to input a character array or a string , we require to clear the input buffer or else the desired input is occupied by buffer of previous variable, not by the desired container.On pressing "Enter" (carriage return) on output screen after the first input , as the buffer of previous variable was the space for new container(as we did'n clear it) , the program skips the following input of container.

In case of C Programming

```

code
// C Code to explain why not
// clearing the input buffer
// causes undesired outputs
#include<stdio.h>
int main()
{
    char str[80], ch;

    // Scan input from user -
    // GeeksforGeeks for example
    scanf("%s", str);

    // Scan character from user-
    // 'a' for example
    ch = getchar();

    // Printing character array,
    // prints "GeeksforGeeks"
    printf("%s\n", str);

    // This does not print
    // character 'a'
    printf("%c", ch);

    return 0;
}

```

chevron_right

filter_none

Input:

```
GeeksforGeeks
a
```

Output:

```
GeeksforGeeks
```

In case of C++

```

filter_none
edit
close
play_arrow
link
brightness_4
code

// C++ Code to explain why
// not clearing the input
// buffer causes undesired
// outputs
#include<iostream>
#include<vector>
using namespace std;
```

```

int main()
{
    int a;
    char ch[80];

    // Enter input from user
    // - 4 for example
    cin >> a;

    // Get input from user -
    // "GeeksforGeeks" for example
    cin.getline(ch,80);

    // Prints 4
    cout << a << endl;

    // Printing string : This does
    // not print string
    cout << ch << endl;

    return 0;
}
```

chevron_right

filter_none

Input:

```
4
GeeksforGeeks
```

Output:

```
4
```

In both the above codes, the output is not printed as desired. Reason to this is an occupied Buffer. The "\n" character goes remains there in buffer and read as next input.

How can it be resolved?

In case of C :

- Using “ **while ((getchar()) != '\n');** ” : Typing “while ((getchar()) != '\n');” reads the buffer characters till the end and discards them(including newline) and using it after the “scanf()” statement clears the input buffer and allows the input in the desired container.

```

filter_none
edit
close

play_arrow
link
brightness_4
code

// C Code to explain why adding
// "while ( (getchar() != '\n');");
// after "scanf()" statement
// flushes the input buffer
#include<stdio.h>

int main()
{
    char str[80], ch;

    // scan input from user -
    // GeeksforGeeks for example
    scanf("%s", str);

    // flushes the standard input
    // (clears the input buffer)
    while ((getchar()) != '\n');

    // scan character from user -
    // 'a' for example
    ch = getchar();

    // Printing character array,
    // prints "GeeksforGeeks")
    printf("%s\n", str);

    // Printing character a: It
    // will print 'a' this time
    printf("%c", ch);

    return 0;
}

```

chevron_right

filter_none

Input:

```
GeeksforGeeks
a
```

Output:

```
GeeksforGeeks
a
```

2. **Using “ fflush(stdin) ” :** Typing “fflush(stdin)” after “scanf()” statement also clears the input buffer but use of it is avoided and is termed to be “undefined” for input stream as per the C++11 standards.

In case of C++ :

1. **Using “ cin.ignore(numeric_limits::max(),'\n'); ” :-** Typing “cin.ignore(numeric_limits::max(),'\n');” after the “cin” statement discards everything in the input stream including the newline.

```

filter_none
edit
close

play_arrow
link
brightness_4
code

// C++ Code to explain how
// "cin.ignore(numeric_limits
// <streamsize>::max(),'\\n');");
// discards the input buffer
#include<iostream>

// for <streamsize>
#include<ios>

// for numeric_limits
#include<limits>
using namespace std;

int main()
{
    int a;
    char str[80];

    // Enter input from user
    // - 4 for example
    cin >> a;

    // discards the input buffer
    cin.ignore(numeric_limits<streamsize>::max(),'\\n');

    // Get input from user -
    // GeeksforGeeks for example
}
```

```

cin.getline(str, 80);

// Prints 4
cout << a << endl;

// Printing string : This
// will print string now
cout << str << endl;

return 0;
}

chevron_right
filter_none

```

Input:

4
GeeksforGeeks

Output:

4
GeeksforGeeks

2. **Using " cin.sync() " :** Typing "cin.sync()" after the "cin" statement discards all that is left in buffer. Though "cin.sync()" **does not work** in all implementations (According to C++11 and above standards).

```

filter_none
edit
close
play_arrow
link
brightness_4
code

// C++ Code to explain how " cin.sync(); "
// discards the input buffer
#include<iostream>
#include<iostream>
#include<limits>
using namespace std;

int main()
{
    int a;
    char str[80];

    // Enter input from user
    // - 4 for example
    cin >> a;

    // Discards the input buffer
    cin.sync();

    // Get input from user -
    // GeeksforGeeks for example
    cin.getline(str, 80);

    // Prints 4
    cout << a << endl;

    // Printing string - this
    // will print string now
    cout << str << endl;

    return 0;
}

```

chevron_right

```

filter_none

```

Input:

4
GeeksforGeeks

Output:

4

3. **Using " cin >> ws " :** Typing "cin>>ws" after "cin" statement tells the compiler to ignore buffer and also to discard all the whitespaces before the actual content of string or character array.

```

filter_none
edit
close
play_arrow
link
brightness_4
code

// C++ Code to explain how "cin >> ws"
// discards the input buffer along with
// initial white spaces of string

#include<iostream>
#include<vector>
using namespace std;

int main()

```

```

{
    int a;
    string s;

    // Enter input from user -
    // 4 for example
    cin >> a;

    // Discards the input buffer and
    // initial white spaces of string
    cin >> ws;

    // Get input from user -
    // GeeksforGeeks for example
    getline(cin, s);

    // Prints 4 and GeeksforGeeks :
    // will execute print a and s
    cout << a << endl;
    cout << s << endl;

    return 0;
}

```

chevron_right

filter_none

Input:

```

4
GeeksforGeeks

```

Output:

```

4
GeeksforGeeks

```

This article is contributed by **Manjeet Singh**. If you like GeeksforGeeks and would like to contribute, you can also write an article and mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes *arrow_drop_up*

Add your personal notes
here! (max 5000 chars)

Recommended Posts:

- [std::basic_istream::ignore in C++ with Examples](#)
- [std::basic_istream::getline in C++ with Examples](#)
- [Format specifiers in different Programming Languages](#)
- [C program to find square root of a given number](#)
- [C program to print odd line contents of a File followed by even line content](#)
- [std::is_heap\(\) in C++ with Examples](#)
- [std::basic_istream::gcount\(\) in C++ with Examples](#)
- [new vs malloc\(\) and free\(\) vs delete in C++](#)
- [std::string::rfind in C++ with Examples](#)
- [Sum of factorials of Prime numbers in a Linked list](#)
- [Sum of all Palindrome Numbers present in a Linked list](#)
- [Generate an array of given size with equal count and sum of odd and even numbers](#)
- [Namespaces in C++ | Set 4 \(Overloading, and Exchange of Data in different Namespaces\)](#)
- [Getting System and Process Information Using C Programming and Shell in Linux](#)
- [Program to calculate Electricity Bill](#)

Improved By : [tornado711](#)

Article Tags :

C
C++
cpp-input-output
Practice Tags :
C
CPP

thumb_up
59

To-do Done
3

Based on 86 vote(s)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

first_page [scanf\(\) and fscanf\(\) in C - Simple Yet Powerful](#)

Next

last_page [How to sum two integers without using arithmetic operators in C/C++?](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C
[Print all possible combinations of the string by replacing '\\$' with any other digit from the string ctype.h<ctype.h> library in C/C++ with Examples](#)
[C program to display month by month calendar for a given year](#)
[Predefined Macros in C with Examples](#)
[Implicit Type Conversion in C with Examples](#)

Most visited in C++
[Vector of Vectors in C++ STL with Examples](#)
[C++ STL with Examples](#)
[Which C++ libraries are useful for competitive programming?](#)
[Array of Vectors in C++ STL](#)
[Map of Vectors in C++ STL with Examples](#)

scanf() and fscanf() in C – Simple Yet Powerful

Many of us know the traditional uses of scanf. Well, here are some of the lesser known facts

How to read only a part of the input that we need? For example, consider some input stream that contains only characters followed by an integer or a float. And we need to scan only that integer or float. That is ,
Input: "this is the value 100",
Output: value read is 100
Input : "this is the value 21.2",
Output : value read is 21.2

```
filter_none
edit
close
play_arrow
link
brightness_4
code

/* C program to demonstrate that we can
ignore some string in scanf() */
#include <stdio.h>
int main()
{
    int a;
    scanf("This is the value %d", &a);
    printf("Input value read : a = %d", a);
    return 0;
}
// Input : This is the value 100
// Output : Input value read : a = 100
chevron_right
filter_none
```

Now, assume we don't know what the preceding characters are but we surely know that the last value is an integer. How can we scan the last value as an integer?

Below solution works only if input string has no spaces.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

/* Sample C program to demonstrate use of *s */
#include<stdio.h>
int main()
{
    int a;
    scanf("%*s %d", &a);
    printf("Input value read : a=%d",a);
    return 0;
}

// Input: "blablabla 25"
// Output: Value read : 25
chevron_right
filter_none
```

Explanation: The %s in scanf is used to ignore some input as required. In this case, it ignores the input until the next space or new line. Similarly if you write %d it will ignore integers until the next space or new line.

The above fact may not seem as an useful trick at the first glance. Inorder to understand its usage, let us first see fscanf().

fscanf() : Tired of all the clumsy syntax to read from files? well, fscanf comes to the rescue.

```
int fscanf(FILE *ptr, const char *format, ...)
```

fscanf reads from a file pointed by the FILE pointer (ptr), instead of reading from the input stream.

Consider the following text file abc.txt

NAME	AGE	CITY
abc	12	hyderabad
bef	25	delhi
cce	65	bangalore

Now, we want to read only the city field of the above text file, ignoring all the other fields. A combination of fscanf and the trick mentioned above does this with ease

```
filter_none
edit
close

play_arrow
link
brightness_4
code

/*c program demonstrating fscanf and its usage*/
#include<stdio.h>
int main()
{
    FILE* ptr = fopen("abc.txt","r");
    if (ptr==NULL)
    {
        printf("no such file.");
        return 0;
    }

    /* Assuming that abc.txt has content in below
     * format
     */
    NAME    AGE    CITY
    abc    12    hyderabad
    bef    25    delhi
    cce    65    bangalore */
    char buf[100];
    while (fscanf(ptr,"%*s %*s %s ",buf)==1)
        printf("%s\n", buf);

    return 0;
}
chevron_right
filter_none
```

Output:

CITY
hyderabad
delhi
bangalore

Exercise: Count the number of words, characters and lines in a file using fscanf!

This article is contributed by Nikhil Chakravartula. If you like GeeksforGeeks and would like to contribute, you can also write an article and mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes 

Add your personal notes
here! (max 5000 chars)

Recommended Posts:

- Inbuilt library functions for user Input | scanf, fscanf, sscanf, scanf_s, fscanf_s, sscanf_s
- Problem with scanf() when there is fgets()/gets()/scanf() after it
- Difference between scanf() and gets() in C
- Use of & in scanf() but not in printf()
- Why to use fgets() over scanf() in C?
- scanf("%[^\n]s", str) Vs gets(str) in C with Examples
- Why "%s" is not used for strings in scanf() function?
- Cin-Cout vs Scanf-Printf
- Return values of printf() and scanf() in C/C++
- OpenGL program for simple Animation (Revolution) in C
- OpenGL program for Simple Ball Game
- Format specifiers in different Programming Languages
- C program to find square root of a given number
- C program to print odd line contents of a File followed by even line content

Improved By : Vivek Kumar 56, SubhashKshatri

Article Tags :

C
C-Input and Output Quiz
c-input-output
Practice Tags :
C

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

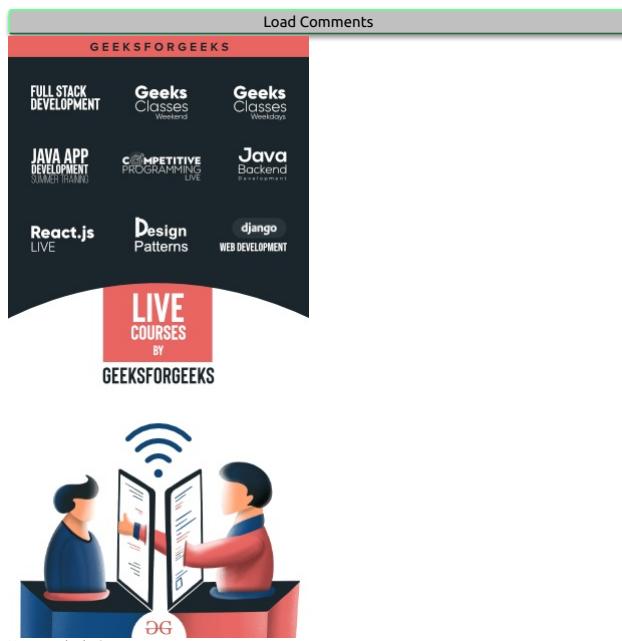
Previous

first_page How to find size of array in C/C++ without using sizeof ?

Next

last_page Clearing The Input Buffer In C/C++

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.



Most popular in C
Print all possible combinations of the string by replacing 's' with any other digit from the string
C program to display month by month calendar for a given year
ctype.h<ccctype> library in C/C++ with Examples

Implicit Type Conversion in C with Examples

Introduction to the C99 Programming Language : Part I

More related articles in C

How to call function within function in C or C++

Format Specifiers in different Programming Languages

Predefined Macros in C with Examples

C program to print odd line contents of a File followed by even line content

Nested Loops in C with Examples

getchar_unlocked() – faster input in C/C++ for Competitive Programming

getchar_unlocked() is similar to getchar() with the exception that it is not thread safe. Below is an example code.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// A simple C program to demonstrate
// working of getchar_unlocked()
#include <stdio.h>
int main()
{
    // Syntax is same as getchar()
    char c = getchar_unlocked();

    printf("Entered character is %c", c);

    return 0;
}
```

Following are some important points:

1. Since it is not thread safe, all overheads of mutual exclusion are avoided and it is faster than getchar().
2. Can be especially useful for competitive programming problems with "Warning: Large I/O data, be careful with certain languages (though most should be OK if the algorithm is well designed)".
3. There is no issue with using getchar_unlocked() even in multithreaded environment as long as the thread using it is the only thread accessing file object.
4. One more difference with getchar() is, it is not a C standard library function, but a POSIX function. It may not work on Windows based compilers.
5. **It is a known fact that scanf() is faster than cin and getchar() is faster than scanf() in general. getchar_unlocked() is faster than getchar(), hence fastest of all.**
6. Similarly, there are getc_unlocked(), putc_unlocked(), and putchar_unlocked() which are non-thread-safe versions of getc(), putc() and putchar() respectively.

filter_none

edit

close

play_arrow

```

link
brightness_4
code

// A simple C program to demonstrate
// working of putchar_unlocked()
#include <stdio.h>
int main()
{
    // Syntax is same as getchar()
    char c = getchar_unlocked();

    putchar_unlocked(c);

    return 0;
}

```

chevron_right

filter_none

Input: g
Output: g

As an exercise, the readers may try solutions given [here](#) with `getchar_unlocked()` and compare performance with `getchar()`.

This article is contributed by **Ayush Saluja**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above



My Personal Notes [arrow_drop_up](#)

Add your personal notes here! (max 5000 chars)

[Save](#)

Recommended Posts:

- Writing code faster during Competitive Programming in C++
- Python Input Methods for Competitive Programming
- Input/Output from external file in C/C++, Java and Python for Competitive Programming
- Input/Output from external file in C/C++, Java and Python for Competitive Programming | Set 2
- Tips and Tricks for Competitive Programmers | Set 2 (Language to be used for Competitive Programming)
- How can competitive programming help you get a job?
- Which C++ libraries are useful for competitive programming?
- How to become a master in competitive programming?
- Python in Competitive Programming
- C++ tricks for competitive programming (for C++ 11)
- Bit Tricks for Competitive Programming
- How to begin with Competitive Programming?
- Must do Math for Competitive Programming
- Fast I/O for Competitive Programming
- A Better Way To Approach Competitive Programming

Article Tags :

C
C++
Competitive Programming
cpp-input-output
CPP-Library
Practice Tags :
C
CPP

[thumb_up](#)
10

To-do Done
2

Based on 19 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) Understanding constexpr specifier in C++

Next

[last_page](#) Different methods to reverse a string in C/C++

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

[Load Comments](#)



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
ctype.h<ctype> library in C/C++ with Examples
C program to display month by month calendar for a given year
Introduction to the C99 Programming Language : Part I
Predefined Macros in C with Examples

Most visited in C++
Vector of Vectors in C++ STL with Examples
C++ STL with Examples
Which C++ libraries are useful for competitive programming?
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

Problem with scanf() when there is fgets()/gets()/scanf() after it

Consider below simple program in C. The program reads an integer using scanf(), then reads a string using fgets().

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// C program to demonstrate the problem when
// fgets()/gets() is used after scanf()
#include<stdio.h>
```

```
int main()
{
    int x;
    char str[100];
    scanf("%d", &x);
    fgets(str, 100, stdin);
    printf("x = %d, str = %s", x, str);
    return 0;
}
```

chevron_right

filter_none

Input

```
10
test
```

Output:

```
x = 10, str =
```

The problem with above code is scanf() reads an integer and leaves a newline character in buffer. So fgets() only reads newline and the string "test" is ignored by the program.

The similar problem occurs when scanf() is used in a loop.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// C program to demonstrate the problem when
// scanf() is used in a loop
#include<stdio.h>
```

```
int main()
{
    char c;
    printf(".....Enter q to quit.....\n");
    do
    {
        printf("Enter a character\n");
        scanf("%c", &c);
        printf("%c\n", c);
```

```

}
while (c != 'q');
return 0;
}

chevron_right
filter_none
Input
a
b
q

```

Output:

```

.....Enter q to quit.....
Enter a character
a
Enter a character

Enter a character
b
Enter a character

Enter a character
q

```

We can notice that above program prints an extra "Enter a character" followed by an extra new line. This happens because every scanf() leaves a newline character in buffer that is read by next scanf.

How to solve above problem?

1. We can make scanf() to read a new line by using an extra "\n", i.e., **scanf("%d\n", &x)** . In fact **scanf("%d ", &x)** also works (Note extra space).
2. We can add a getchar() after scanf() to read an extra newline.

See this and this for corrected programs.

Same problem occurs with Scanner in Java when `nextLine()` is used after `nextXXX()`.

This article is contributed by **Dheeraj Gupta**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up

Save

Recommended Posts:

- Difference between scanf() and gets() in C
- Why to use fgets() over scanf() in C?
- scanf("%[^\\n]s", str) Vs gets(str) in C with Examples
- Use of & in scanf() but not in printf()
- Why "%s" is not used for strings in scanf() function?
- Cin-Cout vs Scanf-Printf
- Return values of printf() and scanf() in C/C++
- scanf() and fscanf() in C - Simple Yet Powerful
- Inbuilt library functions for user Input | scanf, fscanf, sscanf, scanf_s, fscanf_s, sscanf_s
- Problem Solving on Storage Classes and Scoping of Variables
- Problem in comparing Floating point numbers and how to compare them correctly?
- Format specifiers in different Programming Languages
- C program to find square root of a given number
- C program to print odd line contents of a File followed by even line content

Article Tags :

C
C-Input and Output Quiz
c-input-output
C-Library
Practice Tags :
C

23

To-do Done
2.3

Based on 49 vote(s)

Basic **Easy** **Medium** **Hard** **Expert**

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) C | C Quiz – 113 | Question 1

Next

[last_page](#) How to print size of array parameter in C++?

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments



Most popular in C
Print all possible combinations of the string by replacing 'S' with any other digit from the string
C program to display month by month calendar for a given year
`ctype.h(<ctype.h>)` library in C/C++ with Examples
Implicit Type Conversion in C with Examples
Introduction to the C99 Programming Language : Part I

More related articles in C
How to call function within function in C or C++
Format Specifiers in different Programming Languages
Predefined Macros in C with Examples
C program to print odd line contents of a File followed by even line content
Nested Loops in C with Examples

Differentiate printable and control character in C ?

Given a character we need to find if it is printable or not. We also need to find if it is control character or not. A character is known as printable character if it occupies printing space.
For the standard ASCII character set (used by the "C" locale), control characters are those between ASCII codes 0x00 (NUL) and 0x1f (US), plus 0x7f (DEL).

Examples:

```
Input : a
Output : a is printable character
         a is not control character

Input :\r
Output : is not printable character
         is control character
```

To find the difference between a printable character and a control character we can use some predefined functions, which are declared in the "ctype.h" header file.

The **isprint()** function checks whether a character is a printable character or not. isprint() function takes single argument in the form of an integer and returns a value of type int. We can pass a char type argument internally they acts as a int by specifying ASCII value.

The **iscntrl()** function is used to check whether a character is a control character or not. iscntrl() function also take a single argument and return an integer.

```
filter_none
edit
close

play_arrow
link
brightness_4
code

// C program to illustrate isprint() and iscntrl() functions.
#include <stdio.h>
#include <ctype.h>
int main(void)
{
    char ch = 'a';
    if (isprint(ch)) {
        printf("%c is printable character\n", ch);
    } else {
        printf("%c is not printable character\n", ch);
    }

    if (iscntrl(ch)) {
        printf("%c is control character\n", ch);
    } else {
        printf("%c is not control character", ch);
    }
    return (0);
}
chevron_right
filter_none
```

Output:

```
a is printable character
a is not control character
```

This article is contributed by **Bishal Kumar Dubey**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- C | Loops & Control Structure | Question 21
- C | Loops & Control Structure | Question 1
- C | Loops & Control Structure | Question 2
- C | Loops & Control Structure | Question 11
- C | Loops & Control Structure | Question 17
- C | Loops & Control Structure | Question 16
- C | Loops & Control Structure | Question 15
- C | Loops & Control Structure | Question 14
- C | Loops & Control Structure | Question 13
- C | Loops & Control Structure | Question 10
- C | Loops & Control Structure | Question 9
- C | Loops & Control Structure | Question 3
- C | Loops & Control Structure | Question 4
- C | Loops & Control Structure | Question 20
- C | Loops & Control Structure | Question 19

Article Tags :

C

C-Library

Practice Tags :

C



5

To-do Done
1.6

Based on 8 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) Nested functions in C

Next

[last_page](#) Different ways to declare variable as constant in C and C++

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Weekend Geeks Classes Weekdays

JAVA APP DEVELOPMENT COMPETITIVE PROGRAMMING JAVA Backend Development

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS



The screenshot shows the GeeksforGeeks website's header and sidebar. The header has a red bar with 'GEEKSFORGEEKS' and a search bar. Below it is a dark sidebar with various programming categories like Full Stack Development, Java App Development, React.js, etc., each with a small icon. A prominent red button labeled 'LIVE COURSES BY GEEKSFORGEEKS' is centered. At the bottom of the sidebar, there's an illustration of two people looking at a large screen displaying course content.

Most popular in C
Print all possible combinations of the string by replacing 's' with any other digit from the string

Predefined Macros in C with Examples

Format specifiers in different Programming Languages

C program to print odd line contents of a File followed by even line content

Introduction to the C99 Programming Language : Part II

More related articles in C

C program to find square root of a given number

Problems comparing Floating point numbers and how to compare them correctly?

Features of C Programming Language

Pointer Expressions in C with Examples

Introduction to the C99 Programming Language : Part III

rand() and srand() in C/C++

rand ()

rand() function is used in C to generate random numbers. If we generate a sequence of random number with rand() function, it will create the same sequence again and again every time program runs. Say if we are generating 5 random numbers in C with the help of rand() in a loop, then every time we compile and run the program our output must be the same sequence of numbers.

Syntax:

```
int rand(void);  
returns a pseudo-random number in the range of 0 to RAND_MAX.  
RAND_MAX: is a constant whose default value may vary  
between implementations but it is granted to be at least 32767.
```

filter_none
edit

```
close
play_arrow
link
brightness_4
code

// C program to generate random numbers
#include <stdio.h>
#include <stdlib.h>

// Driver program
int main(void)
{
    // This program will create same sequence of
    // random numbers on every program run

    for(int i = 0; i<5; i++)
        printf(" %d ", rand());
    return 0;
}
```

chevron_right

filter_none

NOTE: This program will create same sequence of random numbers on every program run.

Output 1:

```
453 1276 3425 89
```

Output 2:

```
453 1276 3425 89
```

Output n:

```
453 1276 3425 89
```

strand()

The `strand()` function sets the starting point for producing a series of pseudo-random integers. If `strand()` is not called, the `rand()` seed is set as if `rand(1)` were called at program start. Any other value for seed sets the generator to a different starting point.

Syntax:

```
void strand( unsigned seed );
```

Seeds the pseudo-random number generator used by `rand()`

with the `value` seed.

Note: The pseudo-random number generator should only be seeded once, before any calls to `rand()`, and the start of the program. It should not be repeatedly seeded, or reseeded every time you wish to generate a new batch of pseudo-random numbers.

Standard practice is to use the result of a call to `strand(time(0))` as the seed. However, `time()` returns a `time_t` value which vary everytime and hence the pseudo-random number vary for every program call.

```
filter_none
edit
close

play_arrow
link
brightness_4
code

// C program to generate random numbers
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

// Driver program
int main(void)
{
    // This program will create different sequence of
    // random numbers on every program run

    // Use current time as seed for random generator
    srand(time(0));

    for(int i = 0; i<5; i++)
        printf(" %d ", rand());

    return 0;
}
```

chevron_right

filter_none

NOTE: This program will create different sequence of random numbers on every program run.

Output 1:

```
453 1432 325 89
```

Output 2:

```
8976 21234 45 8975
```

Output n:

```
563 9873 12321 24132
```

How `strand()` and `rand()` are related to each other?

`rand()` sets the seed which is used by `rand` to generate "random" numbers. If you don't call `strand` before your first call to `rand`, it's as if you had called `rand(1)` to set the seed to one. In short, **strand() — Set Seed for rand() Function**.

This article is contributed by [Shivam Pradhan \(anuj_charm\)](#). If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes

Add your personal notes here! (max 5000 chars)

Recommended Posts:

- Guess Game using rand() and srand() in C
- Output of C programs | Set 33 (rand() and srand())
- Rust vs C++: Will Rust Replace C++ in Future ?
- Priority queue of pairs in C++ with ordering by first and second element
- Implementation of lower_bound() and upper_bound() in Vector of Pairs in C++
- Generating RGBA portable graphic images through C++
- std::basic_istream::ignore in C++ with Examples
- std::basic_istream::getline in C++ with Examples
- Format specifiers in different Programming Languages
- C program to find square root of a given number
- C program to print odd line contents of a File followed by even line content
- std::is_heap() in C++ with Examples
- std::basic_istream::gcount() in C++ with Examples
- new vs malloc() and free() vs delete in C++

Article Tags :

C
C++

C-Library

CPP-Library

Practice Tags :

C
CPP

 thumb_up

32

To-do Done
2

Based on 21 vote(s)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) new vs operator new in C++

Next

[last_page](#) Quickly check if two STL vectors contain same elements or not

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Weekend Geeks Classes Weekdays

JAVA APP DEVELOPMENT COMPETITIVE PROGRAMMING Java Backend Development

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS



Most popular in C

Print all possible combinations of the string by replacing '\$' with any other digit from the string

How to call function within function in C or C++

Format specifiers in different Programming Languages

Predefined Macros in C with Examples

C program to print odd line contents of a File followed by even line content

Most visited in C++

Vector of Vectors in C++ STL with Examples

C++ STL with Examples

Which C++ libraries are useful for competitive programming?

Array of Vectors in C++ STL

Map of Vectors in C++ STL with Examples

Operators in C | Set 1 (Arithmetic Operators)

Operators in C

	Operator	Type
Unary operator	+ +, - -	Unary operator
Binary operator	+ -, *, /, %	Arithmetic operator
	<, <=, >, >=, !=	Relational operator
	&&, , !	Logical operator
	&, , <<, >>, ~, ^	Bitwise operator
Ternary operator	=, +=, -=, *=, /=, %=	Assignment operator
	?:	Ternary or conditional operator

DG

Operators are the foundation of any programming language. Thus the functionality of C language is incomplete without the use of operators. Operators allow us to perform different kinds of operations on operands. In C, operators in Can be categorized in following categories:

- **Arithmetic Operators** (+, -, *, /, %, post-increment, pre-increment, post-decrement, pre-decrement)
- **Relational Operators** (==, !=, >, <, >=, & <=) Logical Operators (&&, || and !)
- **Bitwise Operators** (&, |, ^, ~, >> and <<)
- **Assignment Operators** (=, +=, -=, *=, etc)
- **Other Operators** (conditional, comma, sizeof, address, redirector)

Arithmetic Operators: These are used to perform arithmetic/mathematical operations on operands. The binary operators falling in this category are:

- **Addition:** The '+' operator adds two operands. For example, **x+y**.
- **Subtraction:** The '-' operator subtracts two operands. For example, **x-y**.
- **Multiplication:** The '*' operator multiplies two operands. For example, **x*y**.
- **Division:** The '/' operator divides the first operand by the second. For example, **x/y**.
- **Modulus:** The '%' operator returns the remainder when first operand is divided by the second. For example, **x%y**.

C

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// C program to demonstrate working of binary arithmetic operators
#include <stdio.h>

int main()
{
    int a = 10, b = 4, res;

    // printing a and b
    printf("a is %d and b is %d\n", a, b);

    res = a + b; // addition
    printf("a+b is %d\n", res);

    res = a - b; // subtraction
    printf("a-b is %d\n", res);

    res = a * b; // multiplication
    printf("a*b is %d\n", res);

    res = a / b; // division
    printf("a/b is %d\n", res);

    res = a % b; // modulus
    printf("a%b is %d\n", res);

    return 0;
}

chevron_right
filter_none
```

C++

```
filter_none
edit
close
play_arrow
link
brightness_4
code

#include <iostream>
using namespace std;
```

```

int main() {
    int a = 10, b = 4, res;

    // printing a and b
    cout << "a is " << a << " and b is " << b << "\n";

    // addition
    res = a + b;
    cout << "a+b is: " << res << "\n";

    // subtraction
    res = a - b;
    cout << "a-b is: " << res << "\n";

    // multiplication
    res = a * b;
    cout << "a*b is: " << res << "\n";

    // division
    res = a / b;
    cout << "a/b is: " << res << "\n";

    // modulus
    res = a % b;
    cout << "a%b is: " << res << "\n";

    return 0;
}

```

chevron_right

filter_none

Output:

```

a is 10 and b is: 4
a+b is: 14
a-b is: 6
a*b is: 40
a/b is: 2
a%b is: 2

```

The ones falling into the category of unary arithmetic operators are:

- **Increment:** The ‘`++`’ operator is used to increment the value of an integer. When placed before the variable name (also called pre-increment operator), its value is incremented instantly. For example, `++x`.
And when it is placed after the variable name (also called post-increment operator), its value is preserved temporarily until the execution of this statement and it gets updated before the execution of the next statement. For example, `x++`.
- **Decrement:** The ‘`--`’ operator is used to decrement the value of an integer. When placed before the variable name (also called pre-decrement operator), its value is decremented instantly. For example, `--x`.
And when it is placed after the variable name (also called post-decrement operator), its value is preserved temporarily until the execution of this statement and it gets updated before the execution of the next statement. For example, `x--`.

C

```

filter_none
edit
close

play_arrow
link
brightness_4
code

// C program to demonstrate working of Unary arithmetic operators
#include <stdio.h>

int main()
{
    int a = 10, b = 4, res;

    // post-increment example:
    // res is assigned 10 only, a is not updated yet
    res = a++;
    printf("a is %d and res is %d\n", a, res); // a becomes 11 now

    // post-decrement example:
    // res is assigned 11 only, a is not updated yet
    res = a--;
    printf("a is %d and res is %d\n", a, res); // a becomes 10 now

    // pre-increment example:
    // res is assigned 11 now since a is updated here itself
    res = ++a;
    // a and res have same values = 11
    printf("a is %d and res is %d\n", a, res);

    // pre-decrement example:
    // res is assigned 10 only since a is updated here itself
    res = --a;
    // a and res have same values = 10
    printf("a is %d and res is %d\n", a, res);

    return 0;
}

```

chevron_right

filter_none

C++

```
filter_none
edit
close
play_arrow
link
brightness_4
code

#include <iostream>
using namespace std;

int main() {
    int a = 10, b = 4, res;

    // post-increment example:
    // res is assigned 10 only, a is not updated yet
    res = a++;
    // a becomes 11 now
    cout << "a is "<<a<<" and res is "<<res<< "\n";

    // post-decrement example:
    // res is assigned 11 only, a is not updated yet
    res = a--;
    // a becomes 10 now
    cout << "a is "<<a<<" and res is "<<res<< "\n";

    // pre-increment example:
    // res is assigned 11 now since a is updated here itself
    res = ++a;
    // a and res have same values = 11
    cout << "a is "<<a<<" and res is "<<res<< "\n";

    // pre-decrement example:
    // res is assigned 10 only since a is updated here itself
    res = --a;
    // a and res have same values = 10
    cout << "a is "<<a<<" and res is "<<res<< "\n";

    return 0;
}
```

chevron_right

filter_none

Output:

```
a is 11 and res is 10
a is 10 and res is 11
a is 11 and res is 11
a is 10 and res is 10
```

We will soon be discussing other categories of operators in different posts.

To know about **Operator Precedence and Associativity**, refer [this](#) link:

Quiz on Operators in C

This article is contributed by Ayush Jaggi. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- [How to sum two integers without using arithmetic operators in C/C++?](#)
- [Operators in C | Set 2 \(Relational and Logical Operators\)](#)
- [# and ## Operators in C](#)
- [Operators in C / C++](#)
- [Assignment Operators in C/C++](#)
- [C | Operators | Question 27](#)
- [C | Operators | Question 26](#)
- [C | Operators | Question 24](#)
- [C | Operators | Question 23](#)
- [C | Operators | Question 22](#)
- [C | Operators | Question 27](#)
- [C | Operators | Question 26](#)
- [C | Operators | Question 18](#)
- [C | Operators | Question 17](#)
- [C | Operators | Question 21](#)

Improved By : [sirraghavgupta](#)

Article Tags :

C

C-Operators

cpp-operator

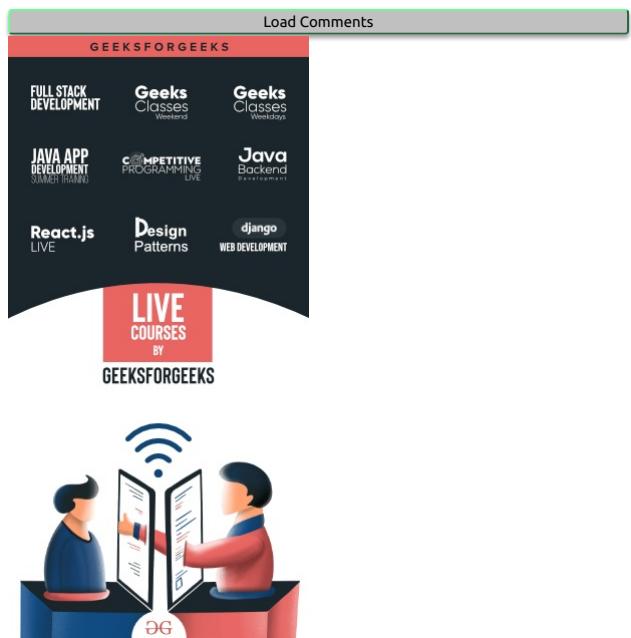
Practice Tags :

cpp-operator

C

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.
Post navigation
Previous
first_page Storage Classes in C
Next
last_page What are near, far and huge pointers?

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.



Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Standard Geeks Classes Standard

JAVA APP DEVELOPMENT SUMMER TRAINING COMPETITIVE PROGRAMMING LIVE Java Backend Development

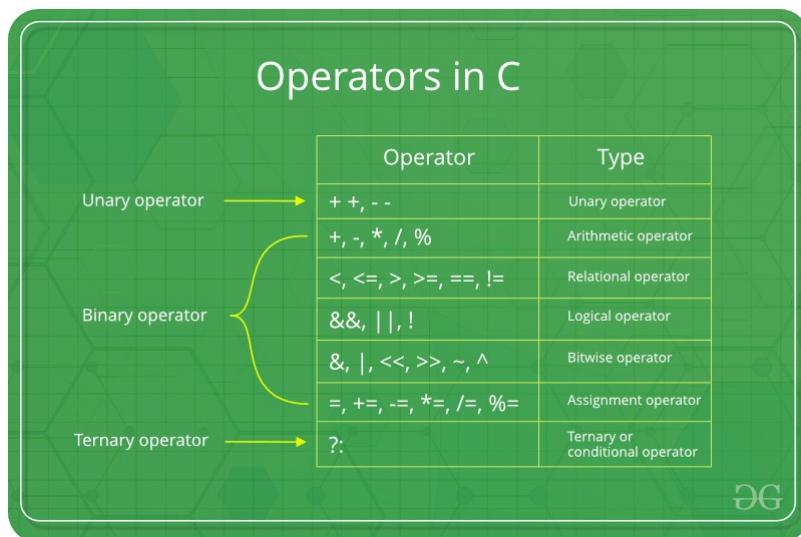
React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS

Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
Format specifiers in different Programming Languages
C program to find square root of a given number
Predefined Macros in C with Examples
How to call function within function in C or C++
More related articles in C
C program to print odd line contents of a File followed by even line content
Introduction to the C99 Programming Language : Part II
Features of C Programming Language
Problem in comparing Floating point numbers and how to compare them correctly?
<cmath> float.h in C/C++ with Examples

Operators in C | Set 2 (Relational and Logical Operators)

We have discussed [Introduction to Operators in C](#) where we got an overall idea of what types of Operators, C and C++ support and its basic implementations. Following that, we studied [Arithmetic Operators](#) where we got a detailed understanding of the types and use of Arithmetic operators in C and C++. In this article, let's try to understand the types and uses of [Relational and Logical Operators](#).



Operators in C

Operator	Type
+ +, - -	Unary operator
+, -, *, /, %	Arithmetic operator
<, <=, >, >=, ==, !=	Relational operator
&&, , !	Logical operator
&, , <<, >>, ~, ^	Bitwise operator
=, +=, -=, *=, /=, %=	Assignment operator
?:	Ternary or conditional operator

Relational Operators

Relational operators are used for comparison of two values to understand the type of relationship a pair of number shares. For example, less than, greater than, equal to etc. Let's see them one by one

- Equal to operator:** Represented as '==', the equal to operator checks whether the two given operands are equal or not. If so, it returns true. Otherwise it returns false. For example, **5==5** will return true.
- Not equal to operator:** Represented as '!=', the not equal to operator checks whether the two given operands are equal or not. If not, it returns true. Otherwise it returns false. It is the exact boolean complement of the '==' operator. For example, **5!=5** will return false.
- Greater than operator:** Represented as '>', the greater than operator checks whether the first operand is greater than the second operand or not. If so, it returns true. Otherwise it returns false. For example, **6>5** will return true.
- Less than operator:** Represented as '<', the less than operator checks whether the first operand is lesser than the second operand. If so, it returns true. Otherwise it returns false. For example, **6<5** will return false.
- Greater than or equal to operator:** Represented as '>=', the greater than or equal to operator checks whether the first operand is greater than or equal to the second operand. If so, it returns true else it returns false. For example, **5>=5** will return true.

6. **Less than or equal to operator: Represented as '`<=`'**, the less than or equal to operator checks whether the first operand is less than or equal to the second operand. If so, it returns true else false. For example, `5<=5` will also return true.

Examples:

C

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// C program to demonstrate working of relational operators
#include <stdio.h>

int main()
{
    int a = 10, b = 4;

    // greater than example
    if (a > b)
        printf("a is greater than b\n");
    else
        printf("a is less than or equal to b\n");

    // greater than equal to
    if (a >= b)
        printf("a is greater than or equal to b\n");
    else
        printf("a is lesser than b\n");

    // less than example
    if (a < b)
        printf("a is less than b\n");
    else
        printf("a is greater than or equal to b\n");

    // lesser than equal to
    if (a <= b)
        printf("a is lesser than or equal to b\n");
    else
        printf("a is greater than b\n");

    // equal to
    if (a == b)
        printf("a is equal to b\n");
    else
        printf("a and b are not equal\n");

    // not equal to
    if (a != b)
        printf("a is not equal to b\n");
    else
        printf("a is equal b\n");

    return 0;
}

chevron_right
filter_none
```

C++

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// C++ program to demonstrate working of logical operators
#include <iostream>
using namespace std;

int main()
{
    int a = 10, b = 4;

    // greater than example
    if (a > b)
        cout << "a is greater than b\n";
    else
        cout << "a is less than or equal to b\n";

    // greater than equal to
    if (a >= b)
        cout << "a is greater than or equal to b\n";
    else
```

```

cout << "a is lesser than b\n";
cout << "a is less than b\n";
if (a < b)
    cout << "a is less than b\n";
else
    cout << "a is greater than or equal to b\n";

cout << "a is lesser than or equal to b\n";
if (a <= b)
    cout << "a is lesser than or equal to b\n";
else
    cout << "a is greater than b\n";

cout << "a is equal to b\n";
if (a == b)
    cout << "a is equal to b\n";
else
    cout << "a and b are not equal\n";

cout << "a is not equal to b\n";
if (a != b)
    cout << "a is not equal to b\n";
else
    cout << "a is equal b\n";

return 0;
}

```

[chevron_right](#)

[filter_none](#)

Output:

```
a is greater than b
a is greater than or equal to b
a is greater than or equal to b
a is greater than b
a and b are not equal
a is not equal to b
```

Logical Operators:

They are used to combine two or more conditions/constraints or to complement the evaluation of the original condition under consideration. They are described below:

- Logical AND operator:** The '`&&`' operator returns true when both the conditions under consideration are satisfied. Otherwise it returns false. For example, `a && b` returns true when both a and b are true (i.e. non-zero).
- Logical OR operator:** The '`||`' operator returns true even if one (or both) of the conditions under consideration is satisfied. Otherwise it returns false. For example, `a || b` returns true if one of a or b or both are true (i.e. non-zero). Of course, it returns true when both a and b are true.
- Logical NOT operator:** The '`!`' operator returns true the condition in consideration is not satisfied. Otherwise it returns false. For example, `!a` returns true if a is false, i.e. when a=0.

Examples:

C

```

filter_none
edit
close

play_arrow
link
brightness_4
code

// C program to demonstrate working of logical operators
#include <stdio.h>

int main()
{
    int a = 10, b = 4, c = 10, d = 20;

    // logical operators

    // logical AND example
    if (a > b && c == d)
        printf("a is greater than b AND c is equal to d\n");
    else
        printf("AND condition not satisfied\n");

    // logical AND example
    if (a > b || c == d)
        printf("a is greater than b OR c is equal to d\n");
    else
        printf("Neither a is greater than b nor c is equal "
               "to d\n");

    // logical NOT example
    if (!a)
        printf("a is zero\n");
    else
        printf("a is not zero");

    return 0;
}

```

[chevron_right](#)

[filter_none](#)

C++

```

filter_none
edit
close
play_arrow
link
brightness_4
code

// C++ program to demonstrate working of
// logical operators
#include <iostream>
using namespace std;

int main()
{
    int a = 10, b = 4, c = 10, d = 20;

    // logical operators

    // logical AND example
    if (a > b && c == d)
        cout << "a is greater than b AND c is equal to d\n";
    else
        cout << "AND condition not satisfied\n";

    // logical AND example
    if (a > b || c == d)
        cout << "a is greater than b OR c is equal to d\n";
    else
        cout << "Neither a is greater than b nor c is equal "
            " to d\n";

    // logical NOT example
    if (!a)
        cout << "a is zero\n";
    else
        cout << "a is not zero";

    return 0;
}
chevron_right

```

filter_none

Output:

```

AND condition not satisfied
a is greater than b OR c is equal to d
a is not zero

```

Short-Circuiting in Logical Operators:

- In case of **logical AND**, the second operand is not evaluated if first operand is false. For example, program 1 below doesn't print "GeeksQuiz" as the first operand of logical AND itself is false.

C

```

filter_none
edit
close
play_arrow
link
brightness_4
code

#include <stdbool.h>
#include <stdio.h>
int main()
{
    int a = 10, b = 4;
    bool res = ((a == b) && printf("GeeksQuiz"));
    return 0;
}
chevron_right

```

filter_none

C++

```

filter_none
edit
close
play_arrow
link
brightness_4
code

#include <iostream>
using namespace std;

int main()
{
    int a = 10, b = 4;
    bool res = ((a == b) && cout << "GeeksQuiz");
    return 0;
}
chevron_right

```

[filter_none](#)

Output:

[No Output](#)

But below program prints "GeeksQuiz" as first operand of logical AND is true.

C

[filter_none](#)

[edit](#)

[close](#)

[play_arrow](#)

[link](#)

[brightness_4](#)

[code](#)

```
#include <stdbool.h>
```

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int a = 10, b = 4;
```

```
    bool res = ((a != b) && printf("GeeksQuiz"));
```

```
    return 0;
```

```
}
```

[chevron_right](#)

[filter_none](#)

C++

[filter_none](#)

[edit](#)

[close](#)

[play_arrow](#)

[link](#)

[brightness_4](#)

[code](#)

```
#include <iostream>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    int a = 10, b = 4;
```

```
    bool res = ((a != b) && cout << "GeeksQuiz");
```

```
    return 0;
```

```
}
```

[chevron_right](#)

[filter_none](#)

Output:

[GeeksQuiz](#)

- In case of **logical OR**, the second operand is not evaluated if first operand is true. For example, program 1 below doesn't print "GeeksQuiz" as the first operand of logical OR itself is true.

C

[filter_none](#)

[edit](#)

[close](#)

[play_arrow](#)

[link](#)

[brightness_4](#)

[code](#)

```
#include <stdbool.h>
```

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int a = 10, b = 4;
```

```
    bool res = ((a != b) || printf("GeeksQuiz"));
```

```
    return 0;
```

```
}
```

[chevron_right](#)

[filter_none](#)

C++

[filter_none](#)

[edit](#)

[close](#)

[play_arrow](#)

[link](#)

[brightness_4](#)

[code](#)

```
#include <stdbool.h>
```

```
#include <stdio.h>
```

```
int main()
```

```

{
    int a = 10, b = 4;
    bool res = ((a != b) || cout << "GeeksQuiz");
    return 0;
}
chevron_right
filter_none

```

Output:

No Output

But below program prints "GeeksQuiz" as first operand of logical OR is false.

C

```

filter_none
edit
close
play_arrow
link
brightness_4
code

#include <stdbool.h>
#include <stdio.h>
int main()
{
    int a = 10, b = 4;
    bool res = ((a == b) || printf("GeeksQuiz"));
    return 0;
}
chevron_right
filter_none

```

C++

```

filter_none
edit
close
play_arrow
link
brightness_4
code

#include <stdbool.h>
#include <stdio.h>
int main()
{
    int a = 10, b = 4;
    bool res = ((a == b) || cout << "GeeksQuiz");
    return 0;
}
chevron_right
filter_none

```

Output:

GeeksQuiz

Quiz on Operators in C

This article is contributed by Ayush Jaggi. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up

Save

Recommended Posts:

- [What are the differences between bitwise and logical AND operators in C/C++?](#)
- [Order of operands for logical operators](#)
- [Operators in C | Set 1 \(Arithmetic Operators\)](#)
- [Operators in C / C++](#)
- [# and ## Operators in C/C++](#)
- [Assignment Operators in C/C++](#)
- [C | Operators | Question 18](#)
- [C | Operators | Question 3](#)
- [C | Operators | Question 4](#)
- [C | Operators | Question 17](#)
- [C | Operators | Question 16](#)
- [C | Operators | Question 15](#)
- [C | Operators | Question 27](#)
- [C | Operators | Question 13](#)
- [C | Operators | Question 11](#)

Article Tags :

Bit Magic
C
C-Operators
cpp-operator
Operators

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

first_page What are near, far and huge pointers?

Next

last_page Walmart Labs Interview Experience | Set 3 (On-Campus)

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Weekdays Geeks Classes Weekdays

JAVA APP DEVELOPMENT SUMMER TRAINING COMPETITIVE PROGRAMMING LIVE Java Backend Development

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS

Most popular in Bit Magic
Range Queries for finding the Sum of all even parity numbers
Minimum flips required to keep all 1s together in a Binary string
Find sum of xor of all unordered triplets of the array
Count of pairs having bit size at most X and Bitwise OR equal to X
Count of distinct XORs formed by rearranging two Binary strings

Most visited in C
P(n) in C++ with Examples
Speed up Code executions with help of Pragma in C/C++
Program to calculate Electricity Bill
Modulo Operator (%) In C/C++ with Examples
Role of SemiColon in various Programming Languages

Bitwise Operators in C/C++

In C, the following 6 operators are bitwise operators (work at bit-level)

Operators in C

Operator	Type
+ +, - -	Unary operator
+, -, *, /, %	Arithmetic operator
<, <=, >, >=, ==, !=	Relational operator
&&, , !	Logical operator
&, , <<, >>, ~, ^	Bitwise operator
=, +=, -=, *=, /=, %=	Assignment operator
?:	Ternary or conditional operator

1. The **&** (**bitwise AND**) in C or C++ takes two numbers as operands and does AND on every bit of two numbers. The result of AND is 1 only if both bits are 1.
2. The **|** (**bitwise OR**) in C or C++ takes two numbers as operands and does OR on every bit of two numbers. The result of OR is 1 if any of the two bits is 1.
3. The **^** (**bitwise XOR**) in C or C++ takes two numbers as operands and does XOR on every bit of two numbers. The result of XOR is 1 if the two bits are different.
4. The **<<** (**left shift**) in C or C++ takes two numbers, left shifts the bits of the first operand, the second operand decides the number of places to shift.
5. The **>>** (**right shift**) in C or C++ takes two numbers, right shifts the bits of the first operand, the second operand decides the number of places to shift.
6. The **~** (**bitwise NOT**) in C or C++ takes one number and inverts all bits of it

Example:

```

filter_none
edit
close
play_arrow
link
brightness_4
code

// C Program to demonstrate use of bitwise operators
#include <stdio.h>
int main()
{
    // a = 5(00000101), b = 9(00001001)
    unsigned char a = 5, b = 9;

    // The result is 00000001
    printf("a = %d\n", a, b);
    printf("a&b = %d\n", a & b);

    // The result is 00001101
    printf("a|b = %d\n", a | b);

    // The result is 00001100
    printf("a^b = %d\n", a ^ b);

    // The result is 11111010
    printf("~a = %d\n", a = ~a);

    // The result is 00010010
    printf("b<<1 = %d\n", b << 1);

    // The result is 00000100
    printf("b>>1 = %d\n", b >> 1);

    return 0;
}

```

chevron_right

filter_none

Output:

```
a = 5, b = 9
a&b = 1
a|b = 13
a^b = 12
~a = 250
b<<1 = 18
b>>1 = 4
```

Interesting facts about bitwise operators

- The left shift and right shift operators should not be used for negative numbers.** If any of the operands is a negative number, it results in undefined behaviour. For example results of both $-1 \ll 1$ and $1 \ll -1$ is undefined. Also, if the number is shifted more than the size of integer, the behaviour is undefined. For example, $1 \ll 33$ is undefined if integers are stored using 32 bits. See [this](#) for more details.
- The bitwise XOR operator is the most useful operator from technical interview perspective.** It is used in many problems. A simple example could be "Given a set of numbers where all elements occur even number of times except one number, find the odd occurring number" This problem can be efficiently solved by just doing XOR of all numbers.

filter_none

edit

close

play_arrow

link

brightness_4

code

```
#include <stdio.h>
```

```
// Function to return the only odd
```

```
// occurring element
```

```
int findOdd(int arr[], int n)
{
    int res = 0, i;
    for (i = 0; i < n; i++)
        res ^= arr[i];
    return res;
}
```

```
// Driver Method
```

```
int main(void)
{
    int arr[] = { 12, 12, 14, 90, 14, 14, 14 };
    int n = sizeof(arr) / sizeof(arr[0]);
    printf("The odd occurring element is %d ",
           findOdd(arr, n));
    return 0;
}
```

chevron_right

filter_none

Output:

```
The odd occurring element is 90
```

The following are many other interesting problems using XOR operator.

- Find the Missing Number
- swap two numbers without using a temporary variable
- A Memory Efficient Doubly Linked List
- Find the two non-repeating elements.

- v. Find the two numbers with odd occurrences in an unsorted-array.
- vi. Add two numbers without using arithmetic operators.
- vii. Swap bits in a given number/.
- viii. Count number of bits to be flipped to convert a to b .
- ix. Find the element that appears once.
- x. Detect if two integers have opposite signs.

3. **The bitwise operators should not be used in place of logical operators.** The result of logical operators (`&&`, `||` and `!`) is either 0 or 1, but bitwise operators return an integer value. Also, the logical operators consider any non-zero operand as 1. For example, consider the following program, the results of `&` and `&&` are different for same operands.

[filter_none](#)

[edit](#)

[close](#)

[play_arrow](#)

[link](#)

[brightness_4](#)

[code](#)

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int x = 2, y = 5;
    (x & y) ? printf("True ") : printf("False ");
    (x && y) ? printf("True ") : printf("False ");
    return 0;
}
```

[chevron_right](#)

[filter_none](#)

Output:

`False True`

4. **The left-shift and right-shift operators are equivalent to multiplication and division by 2 respectively.** As mentioned in point 1, it works only if numbers are positive.

[filter_none](#)

[edit](#)

[close](#)

[play_arrow](#)

[link](#)

[brightness_4](#)

[code](#)

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int x = 19;
    printf("x << 1 = %d\n", x << 1);
    printf("x >> 1 = %d\n", x >> 1);
    return 0;
}
```

[chevron_right](#)

[filter_none](#)

Output:

`x << 1 = 38
x >> 1 = 9`

5. **The & operator can be used to quickly check if a number is odd or even.** The value of expression `(x & 1)` would be non-zero only if x is odd, otherwise the value would be zero.

[filter_none](#)

[edit](#)

[close](#)

[play_arrow](#)

[link](#)

[brightness_4](#)

[code](#)

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int x = 19;
    (x & 1) ? printf("Odd") : printf("Even");
    return 0;
}
```

[chevron_right](#)

[filter_none](#)

Output:

`Odd`

6. **The ~ operator should be used carefully.** The result of `~` operator on a small number can be a big number if the result is stored in an unsigned variable. And the result may be a negative number if the result is stored in a signed variable (assuming that the negative numbers are stored in 2's complement form where the leftmost bit is the sign bit)

[filter_none](#)

[edit](#)

[close](#)

[play_arrow](#)

[link](#)

[brightness_4](#)

[code](#)

```
// Note that the output of the following
// program is compiler dependent
#include <stdio.h>
```

```
int main()
```

```
{  
    unsigned int x = 1;  
    printf("Signed Result %d \n", ~x);  
    printf("Unsigned Result %u \n", ~x);  
    return 0;  
}
```

chevron_right

filter_none

Output:

```
Signed Result -2  
Unsigned Result 4294967294d
```

Important Links:

1. Bits manipulation (Important tactics)
2. Bitwise Hacks for Competitive Programming
3. Bit Tricks for Competitive Programming

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes *arrow_drop_up*

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- Check if a Number is Odd or Even using Bitwise Operators
- Bitwise right shift operators in Java
- What are the differences between bitwise and logical AND operators in C/C++?
- Check if a number is multiple of 9 using bitwise operators
- Toggle case of a string using Bitwise Operators
- Check if a number is divisible by 8 using bitwise operators
- Check if a number is divisible by 17 using bitwise operators
- Russian Peasant (Multiply two numbers using bitwise operators)
- Case conversion (Lower to Upper and Vice Versa) of a string using Bitwise operators in C/C++
- Total pairs in an array such that the bitwise AND, bitwise OR and bitwise XOR of LSB is 1
- Leftover element after performing alternate Bitwise OR and Bitwise XOR operations on adjacent pairs
- Find subsequences with maximum Bitwise AND and Bitwise OR
- Operators in C | Set 2 (Relational and Logical Operators)
- Operators in C | Set 1 (Arithmetic Operators)
- # and ## Operators in C

Improved By : Shubham Dhiman 1, prakash_, AmanRaj1608, VijaySingh8

Article Tags :

Bit Magic

C

C++

Bitwise-XOR

C-Operators

cpp-operator

Practice Tags :

cpp-operator

Bit Magic

C

CPP

thumb_up

98

To-do Done

2.6

Based on 138 vote(s)

Basic **Easy** **Medium** **Hard** **Expert**

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

first_page Interesting Facts about Macros and Preprocessors in C

Next

last_page Print a long int in C using putchar() only

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments



Most popular in Bit Magic
 Range Queries for finding the Sum of all even parity numbers
 Minimum flips required to keep all 1s together in a Binary string
 Number of integers in a range [L, R] which are divisible by exactly K of its digits
 Find sum of xor of all unordered triplets of the array
 Count of pairs having bit size at most X and Bitwise OR equal to X
 Most visited in C
 P(n) in C/C++ with Examples
 Swap up to n numbers with help of Pragma in C/C++
 Program to calculate Electricity Bill
 Modulo Operator (%) in C/C++ with Examples
 Role of SemiColon in various Programming Languages

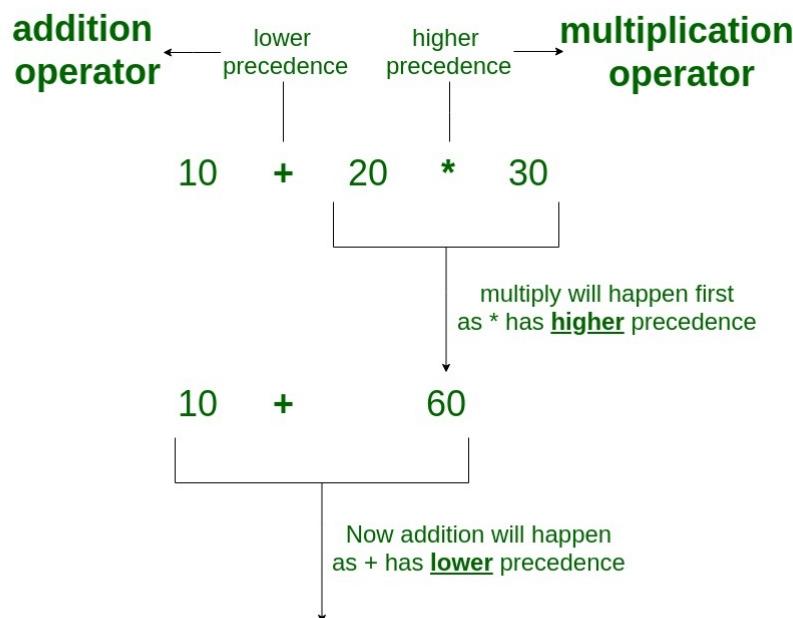
Operator Precedence and Associativity in C

Operator precedence determines which operator is performed first in an expression with more than one operators with different precedence.

For example: Solve

`10 + 20 * 30`

Operator Precedence

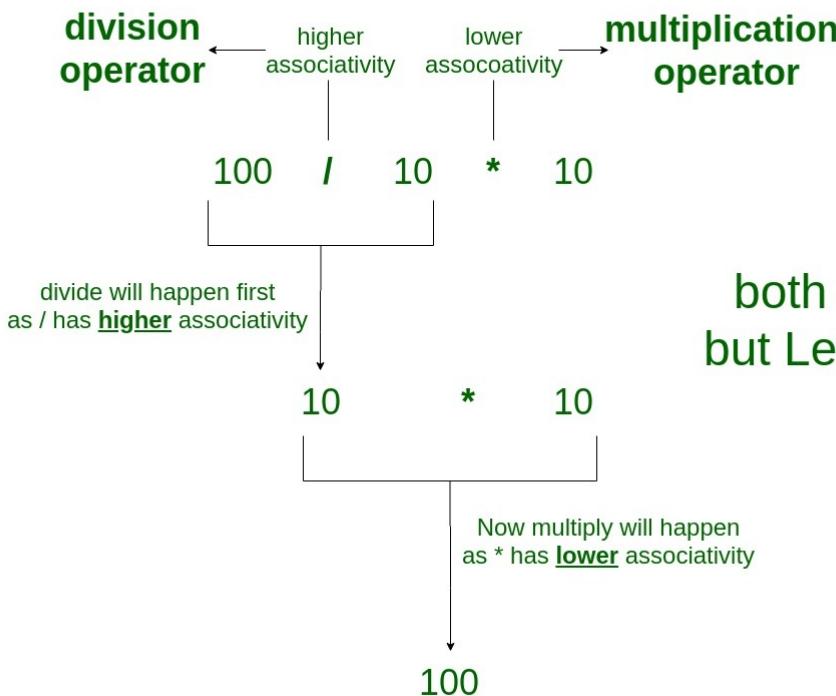


`10 + 20 * 30` is calculated as $10 + (20 * 30)$
 and not as $(10 + 20) * 30$

Operators Associativity is used when two operators of same precedence appear in an expression. Associativity can be either **Left to Right** or **Right to Left**.

For example: '*' and '/' have same precedence and their associativity is **Left to Right**, so the expression " $100 / 10 * 10$ " is treated as " $(100 / 10) * 10$ ".

Operator Associativity



$/$ and $*$

both have the same precedence but Left to Right (**LTR**) association

Operators Precedence and Associativity are two characteristics of operators that determine the evaluation order of sub-expressions in absence of brackets

For example: Solve

$100 + 200 / 10 - 3 * 10$

Operator Precedence

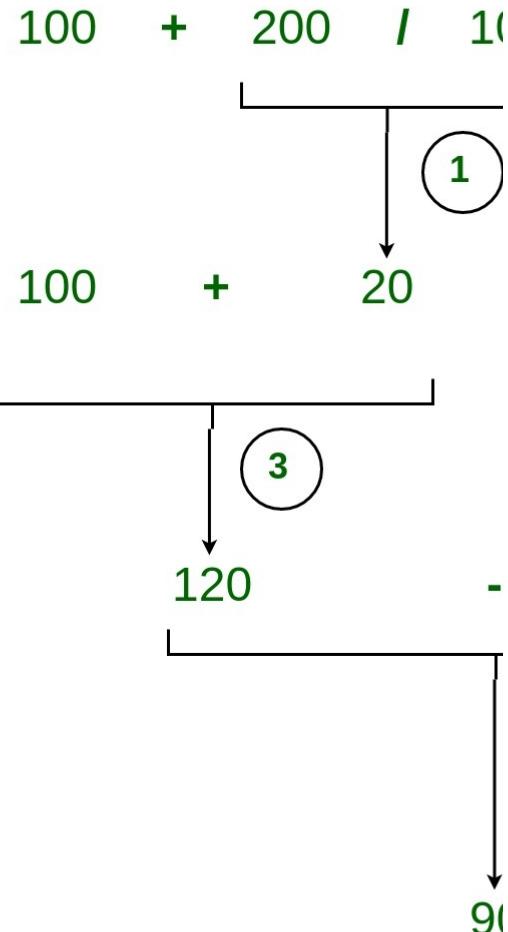
1

Divide ' $/$ ' will happen first
It has **higher precedence** than $+$ and $-$
It has the same precedence as $*$
but **higher associativity**

3

Addition ' $+$ ' will happen third
It has **lower precedence** than $/$ and $*$

It has the same precedence as -
but higher associativity



1) Associativity is only used when there are two or more operators of same precedence.

The point to note is associativity doesn't define the order in which operands of a single operator are evaluated. For example, consider the following program, associativity of the + operator is left to right, but it doesn't mean f1() is always called before f2(). The output of the following program is in-fact compiler dependent. See [this](#) for details.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// Associativity is not used in the below program.
// Output is compiler dependent.
```

```
#include <stdio.h>

int x = 0;
int f1()
{
    x = 5;
    return x;
}
int f2()
{
    x = 10;
    return x;
}
int main()
{
```

```

int p = f1() + f2();
printf("%d ", x);
return 0;
}
chevron_right
filter_none

```

2) All operators with the same precedence have same associativity

This is necessary, otherwise, there won't be any way for the compiler to decide evaluation order of expressions which have two operators of same precedence and different associativity. For example + and - have the same associativity.

3) Precedence and associativity of postfix ++ and prefix ++ are different

Precedence of postfix ++ is more than prefix ++, their associativity is also different. Associativity of postfix ++ is left to right and associativity of prefix ++ is right to left. See [this](#) for examples.

4) Comma has the least precedence among all operators and should be used carefully For example consider the following program, the output is 1. See [this](#) and [this](#) for more details.

```

filter_none
edit
close
play_arrow
link
brightness_4
code

#include <stdio.h>
int main()
{
    int a;
    a = 1, 2, 3; // Evaluated as (a = 1), 2, 3
    printf("%d", a);
    return 0;
}
chevron_right
filter_none

```

5) There is no chaining of comparison operators in C

In Python, expression like "c > b > a" is treated as "c > b and b > a", but this type of chaining doesn't happen in C. For example consider the following program. The output of following program is "FALSE".

```

filter_none
edit
close
play_arrow
link
brightness_4
code

#include <stdio.h>
int main()
{
    int a = 10, b = 20, c = 30;

    // (c > b > a) is treated as ((c > b) > a), associativity of '>' is left to right. Therefore the value becomes ((30 > 20) > 10)
    // which becomes (1 > 20)
    if (c > b > a)
        printf("TRUE");
    else
        printf("FALSE");
    return 0;
}
chevron_right
filter_none

```

Please see the following precedence and associativity table for reference.

OPERATOR	DESCRIPTION	ASSOCIATIVITY
()	Parentheses (function call) (see Note 1)	
[]	Brackets (array subscript)	left-to-right
.	Member selection via object name	
->	Member selection via pointer	
++ -	Postfix increment/decrement (see Note 2)	
++ -	Prefix increment/decrement	
+ -	Unary plus/minus	
! ~	Logical negation/bitwise complement	
(type)	Cast (convert value to temporary value of type)	
*	Dereference	
&	Address (of operand)	
sizeof	Determine size in bytes on this implementation	
* / %	Multiplication/division/modulus	
+ -	Addition/subtraction	
<< >>	Bitwise shift left, Bitwise shift right	
< <=	Relational less than/less than or equal to	
> >=	Relational greater than/greater than or equal to	
== !=	Relational is equal to/is not equal to	
&	Bitwise AND	
^	Bitwise exclusive OR	
	Bitwise inclusive OR	
&&	Logical AND	
	Logical OR	
? :	Ternary conditional	
=	Assignment	
+ = - =	Addition/subtraction assignment	
* = / =	Multiplication/division assignment	
% = & =	Modulus/bitwise AND assignment	
^ = =	Bitwise exclusive/inclusive OR assignment	
<<= >>=	Bitwise shift left/right assignment	
,	Comma (separate expressions)	left-to-right

It is good to know precedence and associativity rules, but the best thing is to use brackets, especially for less commonly used operators (operators other than +, -, *, etc). Brackets increase the readability of the code as the reader doesn't have to see the table to find out the order.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

My Personal Notes

Add your personal notes here! (max 5000 chars)

Recommended Posts:

- Precedence of postfix ++ and prefix ++ in C/C++
- dot (.) operator in C/C++
- sizeof operator in C
- array::operator[] in C++ STL
- typeid operator in C++ with Examples
- Conditional or Ternary Operator (?) in C/C++
- Modulo Operator (%) in C/C++ with Examples
- C++ | Nested Ternary Operator
- Types of Operator Overloading in C++
- Result of comma operator as l-value in C and C++
- Operands for sizeof operator
- Addition of two number using '-' operator
- A comma operator question
- Logical Not ! operator in C with Examples
- To find sum of two numbers without using any operator

Improved By : [sirraghavgupta](#)

Article Tags :

[C Quiz](#)
[C-Operators](#)
[cpp-operator](#)

Practice Tags :

[cpp-operator](#)

 38

To-do Done
2.4

Based on 56 vote(s)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) [C](#) | Input and Output | Question 13

Next

[last_page](#) [C](#) | Advanced Pointer | Question 10

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Weekend Geeks Classes Weekdays

JAVA APP DEVELOPMENT SUMMER TRAINING COMPETITIVE PROGRAMMING LIVE Java Backend Development

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS



Most popular articles
[Must Do Coding Questions for Companies like Amazon, Microsoft, Adobe, ...](#)
[Must Do Coding Questions Company-wise](#)
[10 Projects That Every Developer Should Lay Their Hands-On](#)
[C++ Tutorial](#)
[Map of Vectors in C++ STL with Examples](#)

Most Visited Articles
[Python Tutorial](#)
[Median of sliding window in an array](#)
[Introducing MAX_VALUE and Integer.MIN_VALUE in Java with Examples](#)
[Top 10 Projects For Beginners To Practice HTML and CSS Skills](#)
[Egg dropping puzzle | Set 2](#)

Evaluation order of operands

Consider the below program.

C++

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// C++ implementation
#include <bits/stdc++.h>
using namespace std;
int x = 0;

int f1()
{
    x = 5;
    return x;
}

int f2()
{
    x = 10;
    return x;
}

int main()
{
    int p = f1() + f2();
    cout << ("%d ", x);
    getchar();
    return 0;
}
```

C

```
filter_none
edit
close
play_arrow
link
brightness_4
code

#include <stdio.h>
int x = 0;

int f1()
{
    x = 5;
    return x;
}

int f2()
{
    x = 10;
    return x;
}

int main()
{
    int p = f1() + f2();
    printf("%d ", x);
    getchar();
    return 0;
}
```

Java

```
filter_none
edit
close
play_arrow
link
brightness_4
code

class GFG {

    static int x = 0;

    static int f1()
    {
        x = 5;
        return x;
    }
```

```
static int f2()
{
    x = 10;
    return x;
}

public static void main(String[] args)
{
    int p = f1() + f2();
    System.out.printf("%d ", x);
}
}

// This code is contributed by Rajput-Ji
chevron_right
filter_none
```

Python3

```
filter_none
edit
close
play_arrow
link
brightness_4
code

# Python3 implementation of the above approach
class A():
    x = 0;

    def f1():
        A.x = 5;
        return A.x;

    def f2():
        A.x = 10;
        return A.x;

# Driver Code
p = A.f1() + A.f2();
print(A.x);

# This code is contributed by mits
chevron_right
filter_none
```

C#

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// C# implementation of the above approach
using System;

class GFG {

    static int x = 0;

    static int f1()
    {
        x = 5;
        return x;
    }

    static int f2()
    {
        x = 10;
        return x;
    }

    // Driver code
    public static void Main(String[] args)
    {
        int p = f1() + f2();
        Console.WriteLine("{0} ", x);
    }
}

// This code has been contributed
// by 29AjayKumar
chevron_right
filter_none
```

PHP

```
filter_none
edit
close
play_arrow
link
brightness_4
code

<?php
// PHP implementation of the above approach
$x = 0;

function f1()
{
    global $x;
    $x = 5;
    return $x;
}

function f2()
{
    global $x;
    $x = 10;
    return $x;
}

// Driver Code
$p = f1() + f2();
print($x);

// This code is contributed by mits
?>
chevron_right
filter_none
```

Output:

10

What would the output of the above program - '5' or '10'?

The output is undefined as the order of evaluation of `f1() + f2()` is not mandated by standard. The compiler is free to first call either `f1()` or `f2()`. Only when equal level precedence operators appear in an expression, the associativity comes into picture. For example, `f1() + f2() + f3()` will be considered as `(f1() + f2()) + f3()`. But among first pair, which function (the operand) evaluated first is not defined by the standard.

Thanks to Venki for suggesting the solution.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up

Add your personal notes
here! (max 5000 chars)

Save

Recommended Posts:

- Order of operands for logical operators
- What is evaluation order of function parameters in C?
- Operands for sizeof operator
- Sorting 2D Vector in C++ | Set 2 (In descending order by row and column)
- Program to copy the contents of one array into another in the reverse order
- Sorting Vector of Pairs in C++ | Set 2 (Sort in descending order by first and second)
- Format specifiers in different Programming Languages
- C program to find square root of a given number
- C program to print odd line contents of a File followed by even line content
- Getting System and Process Information Using C Programming and Shell in Linux
- Program to calculate Electricity Bill
- Program to print half Diamond star pattern
- C/C++ program for calling `main()` in `main()`
- Print all possible combinations of the string by replacing '\$' with any other digit from the string

Improved By : Rajput-Ji, 29AjayKumar, Mithun Kumar, Code_Mech

Article Tags :

C

C-Operators

Practice Tags :

C

thumb_up

11

To-do Done
1.9

Based on 50 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

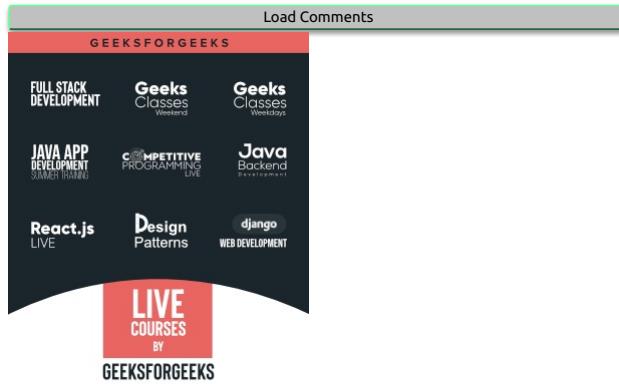
Previous

[first_page](#) Can we call an undeclared function in C++?

Next

[last_page](#) Can static functions be virtual in C++?

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
[ctype.h<ctype.h> library in C/C++ with Examples](#)
Programs with Examples
Format specifiers in different Programming Languages
How to call function within function in C or C++

More related articles in C
C program to print odd line contents of a file followed by even line content
C program to find square root of a given number
Introduction to the C99 Programming Language : Part II
Problem in comparing Floating point numbers and how to compare them correctly?
Features of C Programming Language

Comma in C and C++

In C and C++, comma (,) can be used in two contexts:

1) Comma as an operator:

The comma operator (represented by the token,) is a binary operator that evaluates its first operand and discards the result, it then evaluates the second operand and returns this value (and type). The comma operator has the lowest precedence of any C operator, and acts as a sequence point.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

/* comma as an operator */
int i = (5, 10); /* 10 is assigned to i*/
int j = (f1(), f2()); /* f1() is called (evaluated) first followed by f2().
The returned value of f2() is assigned to j */

chevron_right
```

2) Comma as a separator:

Comma acts as a separator when used with function calls and definitions, function like macros, variable declarations, enum declarations, and similar constructs.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

/* comma as a separator */
int a = 1, b = 2;
void fun(x, y);
chevron_right
```

The use of comma as a separator should not be confused with the use as an operator. For example, in below statement, f1() and f2() can be called in any order.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

/* Comma acts as a separator here and doesn't enforce any sequence.
Therefore, either f1() or f2() can be called first */
void fun(f1(), f2());
chevron_right
```

[filter_none](#)

See [this](#) for C vs C++ differences of using comma operator.

You can try below programs to check your understanding of comma in C.

[filter_none](#)

[edit](#)

[close](#)

[play_arrow](#)

[link](#)

[brightness_4](#)

[code](#)

// PROGRAM 1

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int x = 10;
```

```
    int y = 15;
```

```
    printf("%d", (x, y));
```

```
    getchar();
```

```
    return 0;
```

```
}
```

[chevron_right](#)

[filter_none](#)

[edit](#)

[close](#)

[play_arrow](#)

[link](#)

[brightness_4](#)

[code](#)

// PROGRAM 2: Thanks to Shekhu for suggesting this program

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int x = 10;
```

```
    int y = (x++, ++x);
```

```
    printf("%d", y);
```

```
    getchar();
```

```
    return 0;
```

```
}
```

[chevron_right](#)

[filter_none](#)

[edit](#)

[close](#)

[play_arrow](#)

[link](#)

[brightness_4](#)

[code](#)

// PROGRAM 3: Thanks to Venki for suggesting this program

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int x = 10, y;
```

// The following is equivalent

// to y = x + 2 and x += 3,

// with two printings

```
    y = (x++,
```

```
        printf("x = %d\n", x),
```

```
        ++x,
```

```
        printf("x = %d\n", x),
```

```
        x++);
```

// Note that last expression is evaluated

// but side effect is not updated to y

```
    printf("y = %d\n", y);
```

```
    printf("x = %d\n", x);
```

```
    return 0;
```

```
}
```

[chevron_right](#)

[filter_none](#)

3) Comma operator in place of a semicolon.

We know that in C and C++, every statement is terminated with a semicolon but comma operator also used to terminate the statement after satisfying the following rules.

- The variable declaration statements must be terminated with semicolon.
- The statements after declaration statement can be terminated by comma operator.
- The last statement of the program must be terminated by semicolon.

Examples:

[filter_none](#)

[edit](#)

[close](#)

[play_arrow](#)

```
link  
brightness_4  
code  
  
#include <iostream>  
using namespace std;  
int main()  
{  
    cout << "First Line\n",  
        cout << "Second Line\n",  
        cout << "Third Line\n",  
        cout << "Last line";  
    return 0;  
}
```

[chevron_right](#)

[filter_none](#)

Output:

```
First Line  
Second Line  
Third Line  
Last line
```

References:

http://en.wikipedia.org/wiki/Comma_operator
http://publib.boulder.ibm.com/infocenter/comphelp/v101v121/index.jsp?topic=/com.ibm.xlcpp101.aix.doc/language_ref/co.html
<http://msdn.microsoft.com/en-us/library/zs06xbxh.aspx>

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes [arrow_drop_up](#)

Add your personal notes here! (max 5000 chars)

[Save](#)

Recommended Posts:

- [A comma operator question](#)
- [Result of comma operator as l-value in C and C++](#)
- [Comma operator should be used carefully](#)
- [How to input a comma separated string in C++?](#)
- [Program to Parse a comma separated string in C++](#)
- [How can we use Comma Operator in place of curly braces?](#)
- [std::basic_istream::ignore in C++ with Examples](#)
- [std::basic_istream::getline in C++ with Examples](#)
- [Format specifiers in different Programming Languages](#)
- [C program to find square root of a given number](#)
- [C program to print odd line contents of a File followed by even line content](#)
- [std::is_heap\(\) in C++ with Examples](#)
- [std::basic_istream::gcount\(\) in C++ with Examples](#)
- [new vs malloc\(\) and free\(\) vs delete in C++](#)

Improved By : [NarutoWindy](#), [heart_bleed](#), [olivierpirsonopi](#)

Article Tags :

C
C++
C-Operators
Practice Tags :
C
CPP

[thumb_up](#)
36

To-do Done
2.4

Based on 80 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) [malloc\(\) vs new](#)

Next

[last_page](#) [delete and free\(\) in C++](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

[Load Comments](#)



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
ctype.h<ctype> library in C/C++ with Examples
Predefined Macros in C with Examples
How to call function within function in C or C++
C program to print odd line contents of a file followed by even line content

Most visited in C++
Vector of Vectors in C++ STL with Examples
C++ STL with Examples
Which C++ libraries are useful for competitive programming?
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

sizeof operator in C

sizeof is a much used operator in the **C** or **C++**. It is a compile time unary operator which can be used to compute the size of its operand. The result of sizeof is of unsigned integral type which is usually denoted by `size_t`. sizeof can be applied to any data-type, including primitive types such as integer and floating-point types, pointer types, or compound datatypes such as Structure, union etc.

Usage

`sizeof()` operator is used in different way according to the operand type.

1. When operand is a Data Type.

When `sizeof()` is used with the data types such as int, float, char... etc it simply returns the amount of memory is allocated to that data types.

Let's see example:

C

```
filter_none
edit
close

play_arrow
link
brightness_4
code

#include <stdio.h>
int main()
{
    printf("%lu\n", sizeof(char));
    printf("%lu\n", sizeof(int));
    printf("%lu\n", sizeof(float));
    printf("%lu", sizeof(double));
    return 0;
}
chevron_right
filter_none
```

C++

```
filter_none
edit
close

play_arrow
link
brightness_4
code

#include <iostream>
using namespace std;

int main()
{
    cout << sizeof(char) << "\n";
    cout << sizeof(int) << "\n";
    cout << sizeof(float) << "\n";
    cout << sizeof(double) << "\n";
    return 0;
}
```

chevron_right

filter_none

Output:

```
1  
4  
4  
8
```

Note: `sizeof()` may give different output according to machine, we have run our program on 32 bit gcc compiler.

2. When operand is an expression.

When `sizeof()` is used with the expression, it returns size of the expression. Let see example:

C

filter_none

edit

close

play_arrow

link

brightness_4

code

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int a = 0;
```

```
    double d = 10.21;
```

```
    printf("%lu", sizeof(a + d));
```

```
    return 0;
```

```
}
```

chevron_right

filter_none

C++

filter_none

edit

close

play_arrow

link

brightness_4

code

```
#include <iostream>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    int a = 0;
```

```
    double d = 10.21;
```

```
    cout << sizeof(a + d);
```

```
    return 0;
```

```
}
```

chevron_right

filter_none

Output:

```
8
```

As we know from first case size of int and double is 4 and 8 respectively, a is int variable while d is a double variable. The final result will be a double, Hence the output of our program is 8 bytes.

Need of `sizeof`

1. To find out number of elements in a array.

`sizeof` can be used to calculate number of elements of the array automatically. Let see Example :

C

filter_none

edit

close

play_arrow

link

brightness_4

code

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int arr[] = { 1, 2, 3, 4, 7, 98, 0, 12, 35, 99, 14 };
```

```
    printf("Number of elements:%lu ", sizeof(arr) / sizeof(arr[0]));
```

```
    return 0;
```

```
}
```

chevron_right

filter_none

C++

filter_none

edit

close

play_arrow

link

brightness_4

code

```
#include <iostream>
using namespace std;

int main()
{
    int arr[] = { 1, 2, 3, 4, 7, 98,
    0, 12, 35, 99, 14 };
    cout << "Number of elements: "
    <<(sizeof(arr) / sizeof(arr[0]));
    return 0;
}
```

chevron_right

filter_none

Output:

Number of elements: 11

2. To allocate a block of memory dynamically.

sizeof is greatly used in dynamic memory allocation. For example, if we want to allocate memory for which is sufficient to hold 10 integers and we don't know the sizeof(int) in that particular machine. We can allocate with the help of sizeof.

filter_none
edit
close

play_arrow

link
brightness_4
code

```
int* ptr = (int*)malloc(10 * sizeof(int));
chevron_right
```

filter_none

References
<https://en.wikipedia.org/wiki/Sizeof>

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes *arrow_drop_up*

Add your personal notes
here! (max 5000 chars)

Save

Recommended Posts:

- Operands for sizeof operator
- Is sizeof for a struct equal to the sum of sizeof of each member?
- Implement Your Own sizeof
- Why does sizeof(x++) not increment x in C?
- sizeof() for Floating Constant in C
- Do not use sizeof for array parameters
- Anything written in sizeof() is never executed in C
- How to find size of array in C/C++ without using sizeof ?
- Difference between strlen() and sizeof() for string in C
- dot(.) operator in C/C++
- array::operator[] in C++ STL
- typeid operator in C++ with Examples
- Arrow operator -> in C/C++ with Examples
- Conditional or Ternary Operator (?:) in C/C++
- Modulo Operator (%) in C/C++ with Examples

Improved By : dominicsavio

Article Tags :

C
C Basics
C-Operators
cpp-operator
Practice Tags :
cpp-operator
C

thumb_up
26

To-do Done
1.4

Based on 43 vote(s)

Basic *Easy* *Medium* *Hard* *Expert*

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

first_page Variable Length Arrays in C and C++

Next

last_page Variables and Keywords in C

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
`ctype.h <ctype.h>` library in C/C++ with Examples
Predefined Macros in C with Examples
How to call function within function in C or C++
C program to print odd line contents of a file followed by even line content

More related articles in C
Introduction to the C99 Programming Language : Part II
C program to find square root of a given number
Floating point operators in different Programming Languages
Problem in comparing Floating point numbers and how to compare them correctly
Features of C Programming Language

Operands for sizeof operator

The sizeof operator is used to return the size of its operand, in bytes. This operator always precedes its operand. The operand either may be a data-type or an expression. Let's look at both the operands through proper examples.

1. **type-name:** The type-name must be specified in parentheses.

```
filter_none
edit
close
play_arrow
link
brightness_4
code
sizeof(type - name)
chevron_right
```

Let's look at the code:

C

```
filter_none
edit
close
play_arrow
link
brightness_4
code
#include <stdio.h>
int main()
{
    printf("%lu\n", sizeof(char));
    printf("%lu\n", sizeof(int));
    printf("%lu\n", sizeof(float));
    printf("%lu", sizeof(double));
    return 0;
}
chevron_right
```

filter_none

C++

```
filter_none
edit
close
play_arrow
link
brightness_4
code
#include <iostream>
using namespace std;

int main()
{
    cout << sizeof(char) << "\n";
    cout << sizeof(int) << "\n";
}
```

```
cout << sizeof(float) << "\n";
cout << sizeof(double) << "\n";
return 0;
}
```

chevron_right

filter_none

Output:

```
1
4
4
8
```

2. **expression:** The expression can be specified with or without the parentheses.

filter_none

edit

close

play_arrow

link

brightness_4

code

```
// First type
sizeof(expression)
```

```
// Second type
sizeof(expression)
chevron_right
```

filter_none

The expression is used only for getting the type of operand and not evaluation. For example, below code prints value of i as 5 and the size of i as

C

filter_none

edit

close

play_arrow

link

brightness_4

code

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int i = 5;
    int int_size = sizeof(i++);
```

```
    // Displaying the size of the operand
    printf("\n size of i = %d", int_size);
```

```
    // Displaying the value of the operand
    printf("\n Value of i = %d", i);
```

```
    getchar();
    return 0;
}
```

chevron_right

filter_none

C++

filter_none

edit

close

play_arrow

link

brightness_4

code

```
#include <iostream>
using namespace std;
```

```
int main()
```

```
{
```

```
    int i = 5;
    int int_size = sizeof(i++);
```

```
    // Displaying the size of the operand
    cout << "\n size of i = " << int_size;
```

```
    // Displaying the value of the operand
    cout << "\n Value of i = " << i;
```

```
    return 0;
}
```

// This code is contributed by SHUBHAMSINGH10

chevron_right

filter_none

Output:

```
size of i = 4
Value of i = 5
```

References:

<http://www.gnu.org/software/gnu-c-manual/gnu-c-manual.html#The-sizeof-Operator>

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes 

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- sizeof operator in C
- G-Fact 1 | (Sizeof is an operator)
- Is sizeof for a struct equal to the sum of sizeof of each member?
- Evaluation order of operands
- Order of operands for logical operators
- Why does sizeof(x++) not increment x in C?
- Implement Your Own sizeof
- Anything written in sizeof() is never executed in C
- sizeof() for Floating Constant in C
- Do not use sizeof for array parameters
- How to find size of array in C/C++ without using sizeof ?
- Difference between strlen() and sizeof() for string in C
- Why overriding both the global new operator and the class-specific operator is not ambiguous?
- Operator Overloading '<<' and '>>' operator in a linked list class
- dot(.) operator in C/C++

Improved By : SHUBHAMSINGH10, mohanasrujan

Article Tags :

C
C++
GFacts
C-Operators

Practice Tags :
C
CPP

 10

To-do Done
1.7

Based on 37 vote(s)

Basic **Easy** **Medium** **Hard** **Expert**

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) What is Memory Leak? How can we avoid?

Next

[last_page](#) What are Wild Pointers? How can we avoid?

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Wednesday Geeks Classes Wednesday

JAVA APP DEVELOPMENT COMPETITIVE PROGRAMMING LIVE Java Backend Development

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
getch() function in C with Examples
How to use make utility to build C projects?
`ctype.h <cctype>` library in C/C++ with Examples
C program to display month by month calendar for a given year

Most visited in C++
Vector of Vectors in C++ STL with Examples
C++ Tutorials
Which C++ libraries are useful for competitive programming?
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

A comma operator question

Consider the following C programs.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// PROGRAM 1
#include<stdio.h>
```

```
int main(void)
{
    int a = 1, 2, 3;
    printf("%d", a);
    return 0;
}
```

[chevron_right](#)

[filter_none](#)

The above program fails in compilation, but the following program compiles fine and prints 1.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// PROGRAM 2
#include<stdio.h>
```

```
int main(void)
{
    int a;
    a = 1, 2, 3;
    printf("%d", a);
    return 0;
}
```

[chevron_right](#)

[filter_none](#)

And the following program prints 3, why?

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// PROGRAM 3
#include<stdio.h>
```

```
int main(void)
{
    int a;
    a = (1, 2, 3);
    printf("%d", a);
    return 0;
}
```

[chevron_right](#)

[filter_none](#)

In a C/C++ program, comma is used in two contexts: (1) A separator (2) An Operator. (See [this](#) for more details).

Comma works just as a separator in PROGRAM 1 and we get compilation error in this program.

Comma works as an operator in PROGRAM 2. [Precedence of comma operator is least in operator precedence table](#). So the assignment operator takes precedence over comma and the expression "a = 1, 2, 3" becomes equivalent to "(a = 1), 2, 3". That is why we get output as 1 in the second program.

In PROGRAM 3, brackets are used so comma operator is executed first and we get the output as 3 (See the [Wiki page](#) for more details).

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes [arrow_drop_up](#)

Add your personal notes
here! (max 5000 chars)

[Save](#)

Recommended Posts:

- [Result of comma operator as l-value in C and C++](#)
- [How can we use Comma operator in place of curly braces?](#)

- Comma in C and C++
- C++ | Operator Overloading | Question 4
- C++ | Operator Overloading | Question 10
- C++ | Operator Overloading | Question 2
- C++ | Operator Overloading | Question 3
- C++ | Operator Overloading | Question 7
- C++ | Operator Overloading | Question 9
- C++ | Operator Overloading | Question 10
- C++ | Operator Overloading | Question 5
- C++ | Operator Overloading | Question 10
- C++ | Operator Overloading | Question 6
- dot (.) operator in C/C++
- sizeof operator in C

Article Tags :

C
C-Operators
Practice Tags :

C

21

To-do Done
2.5

Based on 35 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) Catch block and type conversion in C++

Next

[last_page](#) Scope rules in C

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Weekend Geeks Classes Weekdays

JAVA APP DEVELOPMENT SUMMER TRAINING COMPETITIVE PROGRAMMING LIVE Java Backend Development

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS

Most popular in C

[getch\(\)](#) function in C with Examples

[Print all possible combinations of the string by replacing 's' with any other digit from the string](#)

[copy or strcpy\(\) library in C/C++ with Examples](#)

[How to use make utility to build C projects?](#)

[C program to display month by month calendar for a given year](#)

More related articles in C

[Predefined Macros in C with Examples](#)

[Hello World Program : First program while learning Programming](#)

[Implicit Type Conversion in C with Examples](#)

[Introduction to the C99 Programming Language : Part I](#)

[How to call function within function in C or C++](#)

Result of comma operator as l-value in C and C++

Using result of comma operator as l-value is not valid in C. But in C++, result of comma operator can be used as l-value if the right operand of the comma operator is l-value.

For example, if we compile the following program as a C++ program, then it works and prints b = 30. And if we compile the same program as C program, then it gives warning/error in compilation (Warning in Dev C++ and error in Code Blocks).

```
filter_none
edit
close
play_arrow
link
brightness_4
code

#include<stdio.h>

int main()
{
    int a = 10, b = 20;
    (a, b) = 30; // Since b is l-value, this statement is valid in C++, but not in C.
    printf("b = %d", b);
    getchar();
    return 0;
}
```

}

chevron_right

filter_none

C++ Output:

`b = 30`

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes *arrow_drop_up*

Save

Recommended Posts:

- A comma operator question
- Comma operator should be used carefully
- How can we use Comma operator in place of curly braces?
- Ivalue and rvalue in C language
- Comma in C and C++
- How to input a comma separated string in C++?
- Program to Parse a comma separated string in C++
- Why overriding both the global new operator and the class-specific operator is not ambiguous?
- Operator Overloading '<<' and '>' operator in a linked list class
- set operators= in C++ STL
- map::operator[] in C++ STL
- map operator= in C++ STL
- dot(.) operator in C/C++
- new vs operator new in C++
- unordered_multimap operator= in C++

Article Tags :

C
C++
C-Operators

Practice Tags :

C
CPP

thumb_up

5

To-do Done

3

Based on 27 vote(s)

Basic **Easy** **Medium** **Hard** **Expert**

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

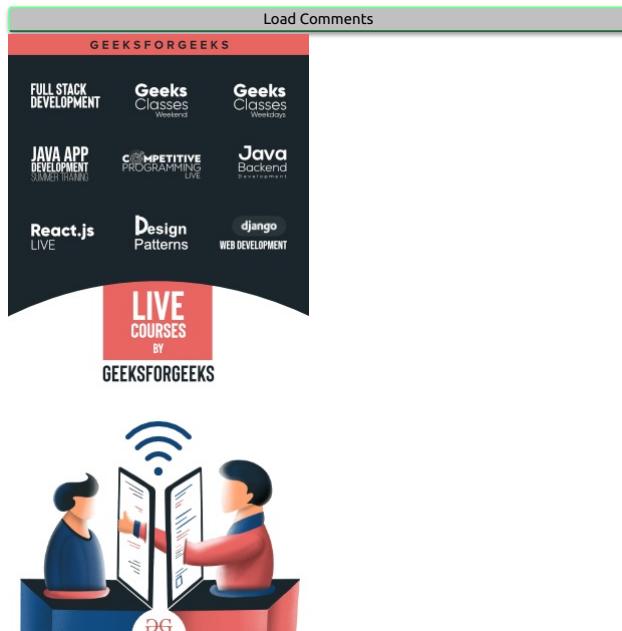
Previous

[first_page](#) Copy constructor vs assignment operator in C++

Next

[last_page](#) Initialization of static variables in C

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C
`getch()` function in C with Examples
Print all possible combinations of the string by replacing '\$' with any other digit from the string
How to use make utility to build C projects?
`ctype.h <ctype>` library in C/C++ with Examples
C program to display month by month calendar for a given year

Most visited in C++
Vector of Vectors in C++ STL with Examples
C++ STL
Which C++ libraries are useful for competitive programming?
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

Order of operands for logical operators

The order of operands of logical operators `&&`, `||` are important in C/C++.

In mathematics, logical AND, OR, etc... operations are commutative. The result will not change even if we swap RHS and LHS of the operator.

In C/C++ (may be in other languages as well) even though these operators are commutative, their order is critical. For example see the following code,

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// Traverse every alternative node
while( pTemp && pTemp->Next )
{
    // Jump over to next node
    pTemp = pTemp->Next->Next;
}
chevron_right
filter_none
```

The first part `pTemp` will be evaluated against NULL and followed by `pTemp->Next`. If `pTemp->Next` is placed first, the pointer `pTemp` will be dereferenced and there will be runtime error when `pTemp` is NULL.

It is mandatory to follow the order. Infact, it helps in generating efficient code. When the pointer `pTemp` is NULL, the second part will not be evaluated since the outcome of AND (`&&`) expression is guaranteed to be 0.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes [arrow_drop_up](#)

[Save](#)

Recommended Posts:

- Evaluation order of operands
- Operators in C | Set 2 (Relational and Logical Operators)
- Written version of Logical operators in C++
- What are the differences between bitwise and logical AND operators in C/C++?
- Operands for sizeof operator
- Code Optimization Technique (logical AND and logical OR)
- Operators in C | Set 1 (Arithmetic Operators)
- Logical Not ! operator in C with Examples
- Operators in C / C++
- # and ## Operators in C
- Assignment Operators in C/C++
- What are the operators that can be and cannot be overloaded in C++?
- C | Operators | Question 16
- C | Operators | Question 3
- C | Operators | Question 2

Article Tags :

C
C++
C-Operators
Practice Tags :
C
CPP

thumb_up
7

To-do Done
2.7

Based on 29 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) [Structure Member Alignment, Padding and Data Packing](#)

Next

[last_page](#) [Function overloading and return type](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

[Load Comments](#)



Most popular in C
getch() function in C with Examples
Print all possible combinations of the string by replacing '\$' with any other digit from the string
ctype.h(<cctype>) library in C/C++ with Examples
How to use make utility to build C projects?
C program to display month by month calendar for a given year

Most visited in C++
Vector of Vectors in C++ STL with Examples
C++ STL with Examples
Which C++ libraries are useful for competitive programming?
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

Increment (Decrement) operators require L-value Expression

What will be the output of the following program?

```
filter_none
edit
close
play_arrow
link
brightness_4
code

#include<stdio.h>
int main()
{
    int i = 10;
    printf("%d", ++(-i));
    return 0;
}
chevron_right
filter_none
```

A) 11 B) 10 C) -9 D) None

Answer: D, None – Compilation Error.

Explanation:

In C/C++ the pre-increment (decrement) and the post-increment (decrement) operators require an L-value expression as operand. Providing an R-value or a *const* qualified variable results in compilation error.

In the above program, the expression `-i` results in R-value which is operand of pre-increment operator. The pre-increment operator requires an L-value as operand, hence the compiler throws an error.

The increment/decrement operators needs to update the operand after the *sequence point*, so they need an L-value. The unary operators such as `-`, `+`, won't need L-value as operand. The expression `-(+i)` is valid.

In C++ the rules are little complicated because of references. We can apply these pre/post increment (decrement) operators on references variables that are not qualified by *const*. References can also be returned from functions.

Puzzle phrased by **Venki**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes

Add your personal notes
here! (max 5000 chars)

Recommended Posts:

- Difference between Increment and Decrement Operators
- Pre-increment (or pre-decrement) in C++
- Increment (++) and Decrement (--) operator overloading in C++
- lvalue and rvalue in C language
- Pre-increment and Post-increment in C/C++
- Operators in C | Set 2 (Relational and Logical Operators)
- Operators in C | Set 1 (Arithmetic Operators)
- Why does sizeof(x++) not increment x in C?
- Lambda expression in C++
- Can we use function on left side of an expression in C and C++?
- std::regex_match, std::regex_replace() | Regex (Regular Expression) In C++
- # and ## Operators in C
- Operators in C / C++
- C | Operators | Question 26
- Bitwise Operators in C/C++

Article Tags :

C
C++
C-Operators
Practice Tags :
C
CPP

thumb_up
9

To-do Done

3

Based on 35 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

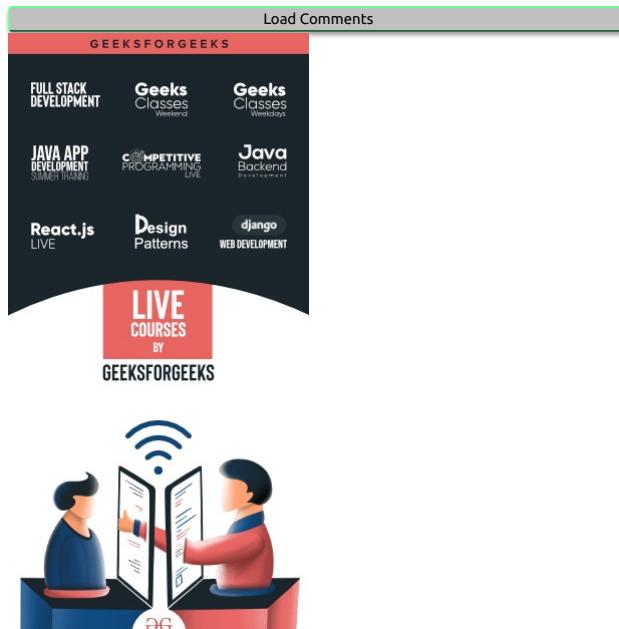
Previous

first_page Templates and Static variables in C++

Next

last_page The OFFSETOF() macro

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C
Vector of Vectors in C with Examples
Print all possible combinations of the string by replacing 'S' with any other digit from the string
ctype.h <<ctype>> library in C/C++ with Examples
How to use make utility to build C projects?
C programs display month by month calendar for a given year

Most visited in C++
Vector of Vectors in C++ STL with Examples
C++ Tutorial
What are STL libraries are useful for competitive programming?
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

Precedence of postfix ++ and prefix ++ in C/C++

In C/C++, precedence of Prefix ++ (or Prefix -) has higher priority than dereference (*) operator, and precedence of Postfix ++ (or Postfix -) is higher than both Prefix ++ and *.

If p is a pointer then *p++ is equivalent to *(p++) and ++*p is equivalent to ++(*p) (both Prefix ++ and * are right associative).

For example, program 1 prints 'h' and program 2 prints 'e'.

```
filter_none
edit
close

play_arrow

link
brightness_4
code

// Program 1
#include<stdio.h>
int main()
{
    char arr[] = "geeksforgeeks";
    char *p = arr;
    ++*p;
    printf(" %c", *p);
    getchar();
    return 0;
}
chevron_right
filter_none
```

Output:

```
h
filter_none
edit
```

```
close
play_arrow
link
brightness_4
code

// Program 2
#include<stdio.h>
int main()
{
    char arr[] = "geeksforgeeks";
    char *p = arr;
    *p++;
    printf(" %c", *p);
    getchar();
    return 0;
}
```

chevron_right
filter_none

Output:

e

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes **arrow_drop_up**

Add your personal notes
here! (max 5000 chars)

Save

Recommended Posts:

- [Operator Precedence and Associativity in C](#)
- [match_results prefix\(\) and suffix\(\) in C++](#)
- [Rust vs C++: Will Rust Replace C++ in Future ?](#)
- [std::basic_istream::ignore in C++ with Examples](#)
- [std::basic_istream::getline in C++ with Examples](#)
- [Format specifiers in different Programming Languages](#)
- [C program to find square root of a given number](#)
- [C program to print odd line contents of a File followed by even line content](#)
- [std::is_heap\(\) in C++ with Examples](#)
- [std::basic_istream::gcount\(\) in C++ with Examples](#)
- [new vs malloc\(\) and free\(\) vs delete in C++](#)
- [std::string::rfind in C++ with Examples](#)
- [Sum of factorials of Prime numbers in a Linked list](#)
- [Sum of all Palindrome Numbers present in a Linked list](#)

Improved By : [sourabh571993](#)

Article Tags :

C
C++
C-Operators
Practice Tags :
C
CPP

thumb_up
8

To-do Done
2.7

Based on 42 vote(s)

Basic **Easy** **Medium** **Hard** **Expert**

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) [Modulus on Negative Numbers](#)

Next

[last_page](#) [Catching base and derived classes as exceptions](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments



Most popular in C
getch() function in C with Examples
Print all possible combinations of the string by replacing '\$' with any other digit from the string
ctype.h(<cctype>) library in C/C++ with Examples
How to use make utility to build C projects?
C program to display month by month calendar for a given year

Most visited in C++
Vector of Vectors in C++ STL with Examples
C++ STL with Examples
Which C++ libraries are useful for competitive programming?
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

Modulus on Negative Numbers

What will be the output of the following C program?

```
filter_none
edit
close
play_arrow
link
brightness_4
code
#include <stdio.h>
int main()
{
    int a = 3, b = -8, c = 2;
    printf("%d", a % b / c);
    return 0;
}
chevron_right
```

filter_none

Output

1

% and / have same precedence and left to right associativity. So % is performed first which results in 3 and / is performed next resulting in 1. The emphasis is, **sign of left operand is appended to result in case of modulus operator in C.**

```
filter_none
edit
close
play_arrow
link
brightness_4
code
#include <stdio.h>
int main()
{
    // a positive and b negative.
    int a = 3, b = -8;
    printf("%d", a % b);
    return 0;
}
chevron_right
```

filter_none

Output

3

```
filter_none
edit
close
play_arrow
link
brightness_4
code
#include <stdio.h>
int main()
```

```
{  
    // a negative and b positive  
    int a = -3, b = 8;  
    printf("%d", a % b);  
    return 0;  
}
```

chevron_right

filter_none

Output

-3

filter_none

edit

close

play_arrow

link

brightness_4

code

```
#include <stdio.h>  
int main()  
{  
    // a and b both negative  
    int a = -3, b = -8;  
    printf("%d", a % b);  
    return 0;  
}
```

chevron_right

filter_none

Output

-3

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes *arrow_drop_up*

Add your personal notes
here! (max 5000 chars)

Save

Recommended Posts:

- Modulus of two float or double numbers
- C program to Check Whether a Number is Positive or Negative or Zero
- C program to invert (making negative) an image content in PGM format
- Add two numbers using ++ and/or --
- Octal numbers in c
- Converting Strings to Numbers in C/C++
- How will you print numbers from 1 to 100 without using loop? | Set-2
- Why variable name does not start with numbers in C ?
- To find sum of two numbers without using any operator
- How will you print numbers from 1 to 100 without using loop?
- Program to print a pattern of numbers
- C Program to print numbers from 1 to N without using semicolon?
- LEX program to add line numbers to a given file
- How to Count Variable Numbers of Arguments in C?
- C Program to Multiply two Floating Point Numbers

Improved By : [sushantgundla](#)

Article Tags :

C

C-Operators

Practice Tags :

C

thumb_up
10

To-do Done
2.5

Based on 30 vote(s)

Basic **Easy** **Medium** **Hard** **Expert**

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

first_page What is the difference between single quoted and double quoted declaration of char array?

Next

last_page Precedence of postfix ++ and prefix ++ in C/C++

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments



Most popular in C
`getch()` function in C with Examples
 Print all possible combinations of the string by replacing '\$' with any other digit from the string
`ctype.h(<cctype>)` library in C/C++ with Examples
 How to use make utility to build C projects?
 C program to display month by month calendar for a given year

More related articles in C
 Predefined Macros in C with Examples
 Hello World Program : First Program while learning Programming
 Implicit Type Conversion in C with Examples
 Introduction to the C99 Programming Language : Part I
 How to call function within function in C or C++

C/C++ Ternary Operator – Some Interesting Observations

Predict the output of following C++ program.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

#include <iostream>
using namespace std;

int main()
{
    int test = 0;
    cout << "First character " << '1' << endl;
    cout << "Second character " << (test ? 3 : '1') << endl;

    return 0;
}
chevron_right
filter_none
```

One would expect the output will be same in both the print statements. However, the output will be,

```
filter_none
edit
close
play_arrow
link
brightness_4
code

First character 1
Second character 49
chevron_right
filter_none
```

Why the second statement printing 49? Read on the ternary expression.

Ternary Operator (C/C++):

A ternary operator has the following form,

`exp1 ? exp2 : exp3`

The expression `exp1` will be evaluated always. Execution of `exp2` and `exp3` depends on the outcome of `exp1`. If the outcome of `exp1` is non zero `exp2` will be evaluated, otherwise `exp3` will be evaluated.

Side Effects:

Any side effects of `exp1` will be evaluated and updated immediately before executing `exp2` or `exp3`. In other words, there is **sequence point** after the evaluation of condition in the ternary expression. If either `exp2` or `exp3` have side effects, only one of them will be evaluated.

Return Type:

It is another interesting fact. The ternary operator has return type. The return type depends on `exp2`, and convertibility of `exp3` into `exp2` as per usual overloaded conversion rules. If they are not convertible, the compiler throws an error. See the examples below,

The following program compiles without any error. The return type of ternary expression is expected to be `float` (as that of `exp2`) and `exp3` (i.e. literal zero – `int` type) is implicitly convertible to `float`.

`filter_none`

edit
close
[play_arrow](#)
[link](#)
[brightness_4](#)
[code](#)

```
#include <iostream>
using namespace std;

int main()
{
    int test = 0;
    float fvalue = 3.111f;
    cout << (test ? fvalue : 0) << endl;

    return 0;
}
```

[chevron_right](#)
[filter_none](#)

The following program will not compile, because the compiler is unable to find return type of ternary expression or implicit conversion is unavailable between `exp2` (`char array`) and `exp3` (`int`).

[filter_none](#)
[edit](#)
[close](#)

[play_arrow](#)
[link](#)
[brightness_4](#)
[code](#)

```
#include <iostream>
using namespace std;
```

```
int main()
{
    int test = 0;
    cout << test ? "A String" : 0 << endl;

    return 0;
}
```

[chevron_right](#)
[filter_none](#)

The following program *may* compile, or but fails at runtime. The return type of ternary expression is bounded to type (`char *`), yet the expression returns `int`, hence the program fails. Literally, the program tries to print string at 0th address at runtime.

[filter_none](#)
[edit](#)
[close](#)

[play_arrow](#)
[link](#)
[brightness_4](#)
[code](#)

```
#include <iostream>
using namespace std;
```

```
int main()
{
    int test = 0;
    cout << (test ? "A String" : 0) << endl;
```

[return 0;](#)
[chevron_right](#)
[filter_none](#)

We can observe that `exp2` is considered as output type and `exp3` will be converted into `exp2` at runtime. If the conversion is implicit the compiler inserts stubs for conversion. If the conversion is explicit the compiler throws an error. If any compiler misses to catch such error, the program may fail at runtime.

Best Practice:

It is the power of C++ type system that avoids such bugs. Make sure both the expressions `exp2` and `exp3` return same type or atleast safely convertible types. We can see other idioms like C++ `convert union` for safe conversion.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above. We will be happy to learn and update from other geeks.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes [arrow_drop_up](#)

Add your personal notes here! (max 5000 chars)

[Save](#)

Recommended Posts:

- [C++ | Nested Ternary Operator](#)
- [Conditional or Ternary Operator \(?:\) in C/C++](#)
- [Implementing ternary operator without any conditional statement](#)
- [Program to Find the Largest Number using Ternary Operator](#)
- [Why overriding both the global new operator and the class-specific operator is not ambiguous?](#)
- [Operator Overloading '<<' and '>>' operator in a linked list class](#)
- [Output of C programs | Set 55 \(Ternary Operators\)](#)
- [Interesting Facts about C++](#)

- Interesting facts about C Language
- C++ bitset interesting facts
- Interesting Facts in C Programming
- Programs to print Interesting Patterns
- Interesting facts about switch statement in C
- Interesting Facts about Macros and Preprocessors in C
- Some Interesting facts about default arguments in C++

Article Tags :

C
C++
C-Operators
Practice Tags :
C
CPP

19

To-do Done
3.9

Based on 70 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

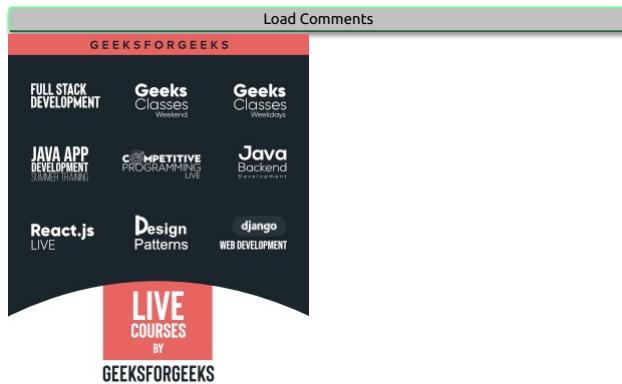
Previous

[first_page](#) Can a C++ class have an object of self type?

Next

[last_page](#) What is the difference between single quoted and double quoted declaration of char array?

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C
getch() function in C with Examples
Print all possible combinations of the string by replacing '\$' with any other digit from the string
ctype.h<ccTypes> library in C/C++ with Examples
How to build a static library and use it in C/C++ projects
C program to display month by month calendar for a given year

Most visited in C++
Vector & Vectors in C++ STL with Examples
C++ Tutorials
Which C++ libraries are useful for competitive programming?
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

Pre-increment (or pre-decrement) in C++

In C++, pre-increment (or pre-decrement) can be used as l-value, but post-increment (or post-decrement) can not be used as l-value.

For example, following program prints `a = 20` (`++a` is used as l-value)

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// CPP program to illustrate
// Pre-increment (or pre-decrement)
#include <cstdio>

int main()
{
    int a = 10;

    ++a = 20; // works

    printf("a = %d", a);
    getchar();
    return 0;
}
```

chevron_right

filter_none

a = 20

The above program works whereas the following program fails in compilation with error “non-lvalue in assignment” (a++ is used as l-value)

filter_none
edit
close

play_arrow
link
brightness_4
code

// CPP program to illustrate
// Post-increment (or post-decrement)
#include <stdio.h>

```
int main()
{
    int a = 10;
    a++ = 20; // error
    printf("a = %d", a);
    getchar();
    return 0;
}
```

chevron_right

filter_none

```
prog.cpp: In function 'int main()':
prog.cpp:6:5: error: lvalue required as left operand of assignment
a++ = 20; // error
^
```

How ++a is different from a++ as lvalue?

It is because ++a returns an *lvalue*, which is basically a reference to the variable to which we can further assign — just like an ordinary variable. It could also be assigned to a reference as follows:

```
int &ref = ++a; // valid
int &ref = a++; // invalid
```

Whereas if you recall how a++ works, it doesn't immediately increment the value it holds. For brevity, you can think of it as getting incremented in the next statement. So what basically happens is that a++ returns an *rvalue*, which is basically just a value like the value of an expression which is not stored. You can think of a++ = 20; as follows after being processed:

```
int a = 10;
// On compilation, a++ is replaced by the value of a which is an rvalue:
10 = 20; // Invalid
// Value of a is incremented
a = a + 1;
```

That should help to understand why a++ = 20; won't work.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes *arrow_drop_up*

Add your personal notes
here! (max 5000 chars)

Recommended Posts:

- std::basic_istream::ignore in C++ with Examples
- std::basic_istream::getline in C++ with Examples
- Format specifiers in different Programming Languages
- C program to find square root of a given number
- C program to print odd line contents of a File followed by even line content
- std::is_heap() in C++ with Examples
- std::basic_istream::gcount() in C++ with Examples
- new vs malloc() and free() vs delete in C++
- std::string::find in C++ with Examples
- Sum of factorials of Prime numbers in a Linked list
- Sum of all Palindrome Numbers present in a Linked list
- Generate an array of given size with equal count and sum of odd and even numbers
- Namespaces in C++ | Set 4 (Overloading, and Exchange of Data in different Namespaces)
- Getting System and Process Information Using C Programming and Shell in Linux
- Program to calculate Electricity Bill

Improved By : [SangeethSudheer](#)

Article Tags :

C

C++

C-Operators

Practice Tags :

C

CPP

thumb_up

11

To-do Done
2.8

Based on 57 vote(s)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) Rules for operator overloading

Next

[last_page](#) Comma operator should be used carefully

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Webinars Geeks Classes Workshops

JAVA APP DEVELOPMENT SUMMER TRAINING COMPETITIVE PROGRAMMING LIVE Java Backend Development

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS

GG

Most popular in C
getchar() function in C with Examples
Print all possible combinations of the string by replacing '\$' with any other digit from the string
ctype.h<<ctype>> library in C/C++ with Examples
How to use make utility to build C projects?
C programs display month by month calendar for a given year

Most visited in C++
Vector of Vectors in C++ STL with Examples
C++ Tutorial
Which C++ libraries are useful for competitive programming?
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

Difference between ++*p, *p++ and *++p

Predict the output of following C programs.

```
filter_none
edit
close

play_arrow
link
brightness_4
code

// PROGRAM 1
#include <stdio.h>
int main(void)
{
    int arr[] = {10, 20};
    int *p = arr;
    ++*p;
    printf("arr[0] = %d, arr[1] = %d, *p = %d",
           arr[0], arr[1], *p);

    return 0;
}
chevron_right
filter_none
filter_none
edit
close

play_arrow
link
brightness_4
code

// PROGRAM 2
#include <stdio.h>
int main(void)
{
    int arr[] = {10, 20};
    int *p = arr;
    *p++;
    printf("arr[0] = %d, arr[1] = %d, *p = %d",
           arr[0], arr[1], *p);

    return 0;
}
chevron_right
filter_none
```

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// PROGRAM 3
#include <stdio.h>
int main(void)
{
    int arr[] = {10, 20};
    int *p = arr;
    *++p;
    printf("arr[0] = %d, arr[1] = %d, *p = %d",
           arr[0], arr[1], *p);
    return 0;
}
chevron_right
```

The output of above programs and all such programs can be easily guessed by remembering following simple rules about postfix ++, prefix ++ and * (dereference) operators
1) Precedence of prefix ++ and * is same. Associativity of both is right to left.
2) Precedence of postfix ++ is higher than both * and prefix ++. Associativity of postfix ++ is left to right.

(Refer: [Precedence Table](#))

The expression `++*p` has two operators of same precedence, so compiler looks for associativity. Associativity of operators is right to left. Therefore the expression is treated as `++(*p)`. Therefore the output of first program is "`arr[0] = 11, arr[1] = 20, *p = 11`".

The expression `*p++` is treated as `(*p++)` as the precedence of postfix ++ is higher than *. Therefore the output of second program is "`arr[0] = 10, arr[1] = 20, *p = 20`".

The expression `*++p` has two operators of same precedence, so compiler looks for associativity. Associativity of operators is right to left. Therefore the expression is treated as `(*++p)`. Therefore the output of third program is "`arr[0] = 10, arr[1] = 20, *p = 20`".

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up

Save

Recommended Posts:

- Difference between C and C++
- Difference between ZIP and RAR
- Difference between PCA VS t-SNE
- Difference between 1G and 2G
- Difference between JSP and ASP
- Difference between T-SQL and PL-SQL
- Difference between TDM and FDM
- Difference between CD-R and CD-RW
- Difference between H.323 and SIP
- Difference between OOP and POP
- Difference between Blu-ray and DVD
- Difference between LAN, MAN and WAN
- Difference between LAN and MAN
- Difference between LAN and WAN
- Difference between RPC and RMI

Improved By : [srinam](#)

Article Tags :

C
Difference Between
C-Operators
cpp-operator
cpp-pointer
Practice Tags :
cpp-operator
C

thumb_up
57

To-do Done
2.7

Based on 114 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) [C | File Handling | Question 2](#)

Next

[last_page](#) [C Program to print environment variables](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments



Most popular in C
getch() function in C with Examples
Print all possible combinations of the string by replacing '\$' with any other digit from the string
ctype.h(<cctype>) library in C/C++ with Examples
How to use make utility to build C projects?
C program to display month by month calendar for a given year

Most visited in Difference Between
Difference between PostgreSQL and MongoDB
Difference Between DELETE and TRUNCATE
Difference Between SMO and SEO
Difference between Prim's and Kruskal's algorithm for MST
Monolithic vs Microservices architecture

Results of comparison operations in C and C++

In C, data type of result of comparison operations is int. For example, see the following program.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

#include<stdio.h>
int main()
{
    int x = 10, y = 10;
    printf("%d \n", sizeof(x == y));
    printf("%d \n", sizeof(x < y));
    return 0;
}
chevron_right
```

Output:

```
4
4
```

Whereas in C++, type of results of comparison operations is bool. For example, see the following program.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

#include<iostream>
using namespace std;

int main()
{
    int x = 10, y = 10;
    cout << sizeof(x == y) << endl;
    cout << sizeof(x < y);
    return 0;
}
chevron_right
```

Output:

```
1
1
```

This article is contributed by **Rajat**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes [arrow_drop_up](#)

Add your personal notes
here! (max 5000 chars)

Save

Recommended Posts:

- Write a program that produces different results in C and C++
- Comparison of a float with a value in C
- Ratio Manipulations in C++ | Set 2 (Comparison)
- Comparison of Python with Other Programming Languages
- Comparison of Java with other programming languages
- Comparison of Exception Handling in C++ and Java
- Assigning an integer to float and comparison in C/C++
- Comparison of static keyword in C++ and Java
- How to check whether a number is in the range[low, high] using one comparison ?
- Comparison of boolean data type in C++ and Java
- Operations on struct variables in C
- Floating Point Operations & Associativity in C, C++ and Java
- Merge operations using STL in C++ | merge(), includes(), set_union(), set_intersection(), set_difference(), .. inplace_merge,
- Rust vs C++: Will Rust Replace C++ in Future ?

Article Tags :

C
C++
C-Operators
cpp-operator
Practice Tags :
cpp-operator
C
CPP



5

To-do Done
2.7

Based on 34 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) Object Slicing in C++

Next

[last_page](#) Does overloading work with Inheritance?

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Weekend Geeks Classes Weekdays

Java APP DEVELOPMENT COMPETITIVE PROGRAMMING LIVE Java Backend Programming

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS

Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
getch() function in C with Examples
How to make use of C/C++ projects?
ctype.h<ctype> library in C/C++ with Examples
C program to display month by month calendar for a given year

Most visited in C++
Vector of Vectors in C++ STL with Examples
C++ Tutorial
Which C++ libraries are useful for competitive programming?
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

To find sum of two numbers without using any operator

Write a program to find sum of positive integers without using any operator. Only use of printf() is allowed. No other library function can be used.

Solution

It's a trick question. We can use printf() to find sum of two numbers as printf() returns the number of characters printed. The width field in printf() can be used to find the sum of two numbers. We can use '*' which indicates the minimum width of output. For example, in the statement "printf("%*d", width, num);", the specified 'width' is substituted in place of *, and 'num' is printed within the minimum width specified. If number of digits in 'num' is smaller than the specified 'width', the output is padded with blank spaces. If number of digits are more, the output is printed as it is (not truncated). In the following program, add() returns sum of x and y. It prints 2 spaces within the width specified using x and y. So total characters printed is equal to sum of x and y. That is why add() returns x+y.

filter_none

edit

close

play_arrow

link

```
brightness_4
code

#include <stdio.h>

int add(int x, int y)
{
    return printf("%c%c", x, ' ', y, ' ');
}
```

```
// Driver code
int main()
{
    printf("Sum = %d", add(3, 4));
    return 0;
}
```

[chevron_right](#)

[filter_none](#)

Output:

Sum = 7

The output is seven spaces followed by "Sum = 7". We can avoid the leading spaces by using carriage return. Thanks to **krazyCoder** and **Sandeep** for suggesting this. The following program prints output without any leading spaces.

```
filter_none
edit
close

play_arrow
link
brightness_4
code

#include <stdio.h>

int add(int x, int y)
{
    return printf("%c%c", x, '\r', y, '\r');
}
```

```
// Driver code
int main()
{
    printf("Sum = %d", add(3, 4));
    return 0;
}
```

[chevron_right](#)

[filter_none](#)

Output:

Sum = 7

Another Method :

C++

```
filter_none
edit
close

play_arrow
link
brightness_4
code

#include <iostream>
using namespace std;

int main()
{
    int a = 10, b = 5;
    if (b > 0) {
        while (b > 0) {
            a++;
            b--;
        }
    }
    if (b < 0) { // when 'b' is negative
        while (b < 0) {
            a--;
            b++;
        }
    }
    cout << "Sum = " << a;
    return 0;
}
```

```
// This code is contributed by SHUBHAMSINGH10
// This code is improved & fixed by Abhijeet Soni.
chevron\_right
```

[filter_none](#)

C

filter_none
edit
close

play_arrow

link
brightness_4
code

```
#include <stdio.h>

int main()
{
    int a = 10, b = 5;
    if (b > 0) {
        while (b > 0) {
            a++;
            b--;
        }
    }
    if (b < 0) { // when 'b' is negative
        while (b < 0) {
            a--;
            b++;
        }
    }
    printf("Sum = %d", a);
    return 0;
}
```

// This code is contributed by Abhijeet Soni
chevron_right

filter_none

Java

filter_none
edit
close

play_arrow

link
brightness_4
code

```
// Java code
class GFG {

    public static void main(String[] args)
    {
        int a = 10, b = 5;
        if (b > 0) {
            while (b > 0) {
                a++;
                b--;
            }
        }
        if (b < 0) { // when 'b' is negative
            while (b < 0) {
                a--;
                b++;
            }
        }
        System.out.println("Sum is: " + a);
    }
}
```

// This code is contributed by Abhijeet Soni
chevron_right

filter_none

Python 3

filter_none
edit
close

play_arrow

link
brightness_4
code

```
# Python 3 Code

if __name__ == '__main__':
    a = 10
    b = 5

    if b > 0:
        while b > 0:
            a = a + 1
            b = b - 1

    if b < 0:
```

```
while b < 0:  
    a = a - 1  
    b = b + 1  
  
print("Sum is: ", a)  
  
# This code is contributed by Akanksha Rai  
# This code is improved & fixed by Abhijeet Soni  
chevron_right  
filter_none
```

C#

```
filter_none  
edit  
close  
  
play_arrow  
link  
brightness_4  
code  
  
// C# code  
using System;  
  
class GFG {  
    static public void Main()  
    {  
        int a = 10, b = 5;  
        if (b > 0) {  
            while (b > 0) {  
                a++;  
                b--;  
            }  
        }  
        if (b < 0) { // when 'b' is negative  
            while (b < 0) {  
                a--;  
                b++;  
            }  
        }  
        Console.WriteLine("Sum is: " + a);  
    }  
}  
  
// This code is contributed by Tushil  
// This code is improved & fixed by Abhijeet Soni.  
chevron_right  
filter_none
```

PHP

```
filter_none  
edit  
close  
  
play_arrow  
link  
brightness_4  
code  
  
<?php  
// PHP Code  
$a = 10;  
$b = 5;  
  
if ($b > 0) {  
    while($b > 0)  
    {  
        $a++;  
        $b--;  
    }  
}  
  
if ($b < 0) {  
    while($b < 0)  
    {  
        $a--;  
        $b++;  
    }  
}  
  
echo "Sum is: ", $a;  
  
// This code is contributed by Dinesh  
// This code is improved & fixed by Abhijeet Soni.  
?>  
chevron_right  
filter_none
```

Output:

sum = 15

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- Program to find remainder without using modulo or % operator
- Find the remainder when N is divided by 4 using Bitwise AND operator
- Program to Find the Largest Number using Ternary Operator
- Given two numbers a and b find all x such that a % x = b
- Find two numbers whose sum and GCD are given
- Program to find LCM of two numbers
- Find XOR of numbers from the range [L, R]
- Program to find GCD or HCF of two numbers
- Find the XOR of first N Prime Numbers
- Find the numbers from 1 to N that contains exactly k non-zero digits
- Find the sum of all amicable numbers up to N
- Find the sum of the first Nth Icosagonal Numbers
- Program to find LCM of 2 numbers without using GCD
- Find two prime numbers with given sum
- Find k numbers which are powers of 2 and have sum N | Set 1

Improved By : [falgunatara](#), [prerna saini](#), [jit_t](#), [Akanksha_Rai](#), [SHUBHAMSINGH10](#), [more](#)

Article Tags :

C

Mathematical

C-Operators

Practice Tags :

Mathematical

C



28

To-do Done
3.6

Based on 121 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) Variable length arguments for Macros

Next

[last_page](#) Copy elision in C++

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes WEEKEND Geeks Classes WEEKEND

JAVA APP DEVELOPMENT COMPETITIVE PROGRAMMING LIVE Java Backend DEVELOPMENT

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS



Most popular in C
getchar function in C with Examples
Print all possible combinations of the string by replacing '\$' with any other digit from the string
ctype.h<ctype> library in C/C++ with Examples
How to use make utility to build C projects?

C program to display month by month calendar for a given year

Most visited in Mathematical
Must do Math for Competitive Programming
Count of subarrays having exactly K perfect square numbers
Product of all Subarrays of an Array
Perfect Sum Problem
Logarithm tricks for Competitive Programming

In this post, we will try to cover many ambiguous questions like following.

Guess the output of following programs.

```
filter_none
edit
close

play_arrow

link
brightness_4
code

// PROGRAM 1
#include <stdio.h>
int f1() { printf ("Geeks"); return 1;}
int f2() { printf ("forGeeks"); return 1;}
int main()
{
    int p = f1() + f2();
    return 0;
}

// PROGRAM 2
#include <stdio.h>
int x = 20;
int f1() { x = x+10; return x;}
int f2() { x = x-5; return x;}
int main()
{
    int p = f1() + f2();
    printf ("p = %d", p);
    return 0;
}

// PROGRAM 3
#include <stdio.h>
int main()
{
    int i = 8;
    int p = i++*i++;
    printf("%d\n", p);
}
chevron_right
```

The output of all of the above programs is **undefined or unspecified**. The output may be different with different compilers and different machines. It is like asking the value of undefined automatic variable.

The reason for undefined behavior in PROGRAM 1 is, the operator '+' doesn't have standard defined order of evaluation for its operands. Either f1() or f2() may be executed first. So output may be either "GeeksforGeeks" or "forGeeksGeeks".

Similar to operator '+', most of the other similar operators like '-', '/', '*', Bitwise AND &, Bitwise OR |, .. etc don't have a standard defined order for evaluation for its operands.

Evaluation of an expression may also produce side effects. For example, in the above program 2, the final values of p is ambiguous. Depending on the order of expression evaluation, if f1() executes first, the value of p will be 55, otherwise 40.

The output of program 3 is also undefined. It may be 64, 72, or may be something else. The subexpression i++ causes a side effect, it modifies i's value, which leads to undefined behavior since i is also referenced elsewhere in the same expression.

Unlike above cases, *at certain specified points in the execution sequence called sequence points, all side effects of previous evaluations are guaranteed to be complete*. A sequence point defines any point in a computer program's execution at which it is guaranteed that all side effects of previous evaluations will have been performed, and no side effects from subsequent evaluations have yet been performed. Following are the sequence points listed in the C standard:

– The end of the first operand of the following operators:

- a) logical AND &&
- b) logical OR ||
- c) conditional ?
- d) comma ,

For example, the output of following programs is guaranteed to be "GeeksforGeeks" on all compilers/machines.

```
filter_none
edit
close

play_arrow

link
brightness_4
code

// Following 3 lines are common in all of the below programs
#include <stdio.h>
int f1() { printf ("Geeks"); return 1;}
int f2() { printf ("forGeeks"); return 1;}

// PROGRAM 4
int main()
{
    // Since && defines a sequence point after first operand, it is
    // guaranteed that f1() is completed first.
    int p = f1() && f2();
    return 0;
}

// PROGRAM 5
int main()
{
    // Since comma operator defines a sequence point after first operand, it is
    // guaranteed that f1() is completed first.
```

```

int p = (f1(), f2());
return 0;
}

// PROGRAM 6
int main()
{
    // Since ? operator defines a sequence point after first operand, it is
    // guaranteed that f1() is completed first.
    int p = f1()? f2(): 3;
    return 0;
}
chevron_right
filter_none

```

— The end of a full expression. This category includes following expression statements

- a) Any full statement ended with semicolon like "a = b;"
- b) return statements
- c) The controlling expressions of if, switch, while, or do-while statements.
- d) All three expressions in a for statement.

The above list of sequence points is partial. We will be covering all remaining sequence points in the next post on Sequence Point.

References:

http://en.wikipedia.org/wiki/Sequence_point
<http://c-faq.com/expr/seqpoints.html>
[http://msdn.microsoft.com/en-us/library/d45c7a5d\(v=vs.110\).aspx](http://msdn.microsoft.com/en-us/library/d45c7a5d(v=vs.110).aspx)
<http://www.open-std.org/jtc1/sc22/wg14/www/docs/n925.htm>

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes 

Add your personal notes
 here! (max 5000 chars)

Recommended Posts:

- Print numbers in sequence using thread synchronization
- Format specifiers in different Programming Languages
- C program to find square root of a given number
- C program to print odd line contents of a File followed by even line content
- Getting System and Process Information Using C Programming and Shell in Linux
- Program to calculate Electricity Bill
- Program to print half Diamond star pattern
- C/C++ program for calling main() in main()
- Print all possible combinations of the string by replacing '\$' with any other digit from the string
- How to use make utility to build C projects?
- How to call function within function in C or C++
- Role of Semicolon in various Programming Languages
- C program to display month by month calendar for a given year
- Difference between Type Casting and Type Conversion

Article Tags :

C
 C-Operators
 Practice Tags :
 C

 16

To-do Done
 3.4

Based on 56 vote(s)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) Importance of function prototype in C

Next

[last_page](#) Variable length arguments for Macros

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
ctype.h(<cctype>) library in C/C++ with Examples
Implicit Type Conversion in C with Examples
Format specifiers in different Programming Languages
C program to find square root of a given number

More related articles in C
Predefined Macros in C with Examples
How to call a function with a function in C or C++
C program to print end line contents of a file followed by even line content
Introduction to the C99 Programming Language : Part II
Features of C Programming Language

Execution of printf with ++ operators

Consider below C++ program and predict its output.

```
filter_none
edit
close
play_arrow
link
brightness_4
code
printf("%d %d", i, ++i, i++);
chevron_right
filter_none
```

The above invokes undefined behaviour by referencing both 'i' and 'i++' in the argument list. It is not defined in which order the arguments are evaluated. Different compilers may choose different orders. A single compiler can also choose different orders at different times.

For example below three printf statements may also cause undefined behavior.

```
filter_none
edit
close
play_arrow
link
brightness_4
code
// All three printf() statements
// in this cause undefined behavior
#include <stdio.h>

int main()
{
    volatile int a = 10;
    printf("\n %d %d", a, a++);
    a = 10;
    printf("\n %d %d", a++, a);
    a = 10;
    printf("\n %d %d %d ", a, a++, ++a);
    return 0;
}
chevron_right
filter_none
```

Therefore, it is not recommended Not to do two or more than two pre or post increment operators in the same statement.
This means that there's absolutely no temporal ordering in this process. The arguments can be evaluated in any order, and the process of their evaluation can be intertwined in any way.

This article is contributed by **Spoorthi Aman**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes [arrow_drop_up](#)
Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- Nested printf (printf inside printf) in C
- What is use of %n in printf() ?
- How to print % using printf()
- Use of & in scanf() but not in printf()
- Passing NULL to printf in C
- Cin-Cout vs Scanf-Printf
- What is the difference between printf, sprintf and fprintf?
- How to change the output of printf() in main() ?
- puts() vs printf() for printing a string
- Return values of printf() and scanf() in C/C++
- Operators in C | Set 2 (Relational and Logical Operators)
- Measure execution time of a function in C++
- Measure execution time with high precision in C/C++
- Operators in C | Set 1 (Arithmetic Operators)
- # and ## Operators in C

Improved By : ArunSivaramakrishnanS

Article Tags :

C
C++
C-Operators
cpp-input-output
cpp-operator
Practice Tags :
cpp-operator
C
CPP



14

To-do Done
2.3

Based on 39 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) Precision of floating point numbers in C++ (floor(), ceil(), trunc(), round() and setprecision())

Next

[last_page](#) Dangling, Void , Null and Wild Pointers

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Weekend Geeks Classes Weekdays

JAVA APP DEVELOPMENT COMPETITIVE PROGRAMMING LIVE Java Backend Development

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
ctype.h Header in C with Examples
Predefined Macros in C with Examples
Implicit Type Conversion in C with Examples
How to call function within function in C or C++

Most visited in C++
Vector of Vectors in C++ STL with Examples
C++ Tutorial
Which C++ libraries are useful for competitive programming?
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

Anything written in sizeof() is never executed in C

In C/C++ sizeof() operator is used to find size of a data type or variable. Expressions written in sizeof() are never executed.

Examples:

filter_none
edit
close
play_arrow
link
brightness_4
code

```

// C program to demonstrate that the
// expressions written in sizeof() are
// never executed
#include <stdio.h>

int main(){

    // The printf in sizeof is not executed
    // Only the return type of printf is
    // considered and its size is evaluated
    // by sizeof,
    int a = sizeof_printf("hey");

    printf("%d", a);

    return 0;
}

```

chevron_right
filter_none

Output:
4

Even if we assign a value inside sizeof(), the changes are not reflected.

```

filter_none
edit
close

play_arrow

link
brightness_4
code

// One more C program to demonstrate that
// the expressions written in sizeof() are
// never executed
#include <stdio.h>

int main() {
    int a = 5;
    int b = sizeof(a = 6);
    printf("a = %d, b = %d\n", a, b);
    return 0;
}

```

chevron_right
filter_none

Output:
a = 5, b = 4

This article is contributed by **Nishu Singh**. If you like GeeksforGeeks and would like to contribute, you can also write an article using [contribute.geeksforgeeks.org](#) or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes [arrow_drop_up](#)

Add your personal notes
here! (max 5000 chars)

[Save](#)

Recommended Posts:

- Is sizeof for a struct equal to the sum of sizeof of each member?
- Functions that are executed before and after main() in C
- Why does sizeof(x++) not increment x in C?
- sizeof operator in C
- Implement Your Own sizeof
- sizeof() for Floating Constant in C
- Operands for sizeof operator
- Do not use sizeof for array parameters
- Difference between strlen() and sizeof() for string in C
- How to find size of array in C/C++ without using sizeof?
- Written version of Logical operators in C++
- Rust vs C++: Will Rust Replace C++ in Future?
- Priority queue of pairs in C++ with ordering by first and second element
- Implementation of lower_bound() and upper_bound() in Vector of Pairs in C++

Article Tags :

C
C++
cpp-sizeof

Practice Tags :

C
CPP

[thumb_up](#)
9

To-do Done
2.5

Based on 9 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Previous
[first_page](#) **strspn()** function in C
Next
[last_page](#) C program to print a string without any quote (single or double) in the program

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Weekend Geeks Classes Weekdays

JAVA APP DEVELOPMENT COMPETITIVE PROGRAMMING LIVE Java Backend Development

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
Predefined Macros in C with Examples
Format specifiers in different Programming Languages
C program to print odd line contents of a File followed by even line content
C program to find square root of a given number

Most visited in C++
Vector of Vectors in C++ STL with Examples
C++ STL
Which C++ libraries are useful for competitive programming?
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

Difference between strlen() and sizeof() for string in C

sizeof()

Sizeof operator is a compile time unary operator which can be used to compute the size of its operand.

- The result of sizeof is of unsigned integral type which is usually denoted by size_t.
- sizeof can be applied to any data-type, including primitive types such as integer and floating-point types, pointer types, or compound datatypes such as Structure, union etc.

strlen()

strlen() is a predefined function in C whose definition is contained in the header file "string.h".

- strlen() accepts a pointer to an array as argument and walks through memory at run time from the address we give it looking for a **NULL** character and counts up how many memory locations it passed before it finds one.
- The main task of strlen() is to count the length of an array or string.

sizeof vs strlen()

- Type:** Sizeof operator is a unary operator whereas strlen() is a predefined function in C
- Data types supported:** Sizeof gives actual size of any type of data (allocated) in bytes (including the null values) whereas get the length of an array of chars/string.
- Evaluation size:** sizeof() is a compile-time expression giving you the size of a type or a variable's type. It doesn't care about the value of the variable.
Strlen on the other hand, gives you the length of a C-style NULL-terminated string.
- Summary:** The two are almost different concepts and used for different purposes.
- In context of C++:** In C++, you do not need any of them as such.
strlen() in C-style strings can be replaced by **C++ std::strings**.
sizeof() in C is as an argument to functions like malloc(), memcpy() or memset() can be replaced by **C++ (use new, std::copy(), and std::fill() or constructors)**.

```
filter_none
edit
close

play_arrow
link
brightness_4
code

// C program to demonstrate difference
// between strlen() and sizeof()
#include<stdio.h>
#include<string.h>

int main()
{
    char str[] = "November";
    printf("Length of String is %lu\n", strlen(str));
    printf("Size of String is %lu\n", sizeof(str));
}
```

chevron_right

filter_none

Output:

```
Length of String is 8
Size of String is 9
```

Since size of char in C is 1 byte but then also we find that strlen() gave one less value than sizeof().

Explanation : We know, that every string terminates with a NULL character ("0").

strlen() searches for that NULL character and counts the number of memory address passed, So it actually counts the number of elements present in the string before the NULL character, here which is 8. **sizeof()** operator returns actual amount of memory allocated for the operand passed to it. Here the operand is an array of characters which contains 9 characters including Null character and size of 1 character is 1 byte. So, here the total size is 9 bytes.

Try to guess the output of following program:

```
filter_none
edit
close

play_arrow
link
brightness_4
code

#include <iostream>
#include <string.h>
using namespace std;

int main()
{
    char a[] = {"Geeks for"};
    char b[] = {'G','e','e','k','s',' ','f','o','r'};
    cout << "sizeof(a) = " << sizeof(a);
    cout << "\nstrlen(a) = " << strlen(a);
    cout << "\nsizeof(b) = " << sizeof(b);
    cout << "\nstrlen(b) = " << strlen(b);

    return 0;
}
chevron_right
```

The strlen function looks for a null character and behaves abnormally if it doesn't find it.

Output:

```
sizeof(a) = 10
strlen(a) = 9
sizeof(b) = 9
strlen(b) = 11
```

This article is contributed by [Aditya Kumar](#). If you like GeeksforGeeks and would like to contribute, you can also write an article using [contribute.geeksforgeeks.org](#) or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up

Add your personal notes
here! (max 5000 chars)

Save

Recommended Posts:

- Write your own strlen() for a long string padded with '\0's
- Is sizeof for a struct equal to the sum of sizeof of each member?
- strlen() function in c
- Implement Your Own sizeof
- Why does sizeof(x++) not increment x in C?
- sizeof operator in C
- sizeof() for Floating Constant in C
- Anything written in sizeof() is never executed in C
- Operands for sizeof operator
- Do not use sizeof for array parameters
- How to find size of array in C/C++ without using sizeof ?
- Difference between Relational operator(==) and std::string::compare() in C++
- Difference between String and Character array in Java
- Print substring of a given string without using any string function and loop in C
- Print all possible combinations of the string by replacing '\$' with any other digit from the string

Improved By : [abhikulshrestha1](#)

Article Tags :

C
C++
Difference Between
Practice Tags :
C
CPP

thumb_up
19

To-do Done
2.4

Based on 19 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) Conversion of whole String to uppercase or lowercase using STL in C++

Next

[last_page](#) Output of C++ programs | Set 24 (C++ vs C)

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Weekend Geeks Classes Weekdays

JAVA APP DEVELOPMENT COMPETITIVE PROGRAMMING DN Java Backend

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
ctype.h(<cctype>) library in C/C++ with Examples
Predefined Macros in C with Examples
Implicit Type Conversion in C with Examples
How to call function within function in C or C++

Most visited in C++
Vector of Vectors in C++ STL with Examples
C++ Tutorials
Which C++ libraries are useful for competitive programming?
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

and ## Operators in C

Stringizing operator (#)

This operator causes the corresponding actual argument to be enclosed in double quotation marks. The # operator, which is generally called the *stringize* operator, turns the argument it precedes into a quoted string. For more on pre-processor directives - refer [this](#)

Examples :

1. The following preprocessor turns the line `printf(mkstr(geeksforgeeks));` into `printf("geeksforgeeks");`

```
filter_none
edit
close

play_arrow
link
brightness_4
code

// CPP program to illustrate (#) operator
#include <stdio.h>
#define mkstr(s) #
int main(void)
{
    printf(mkstr(geeksforgeeks));
    return 0;
}
chevron_right
filter_none
```

Output:

geeksforgeeks

2. In this program, value of a is replaced by macro.

```
filter_none
edit
close

play_arrow
link
brightness_4
code

// CPP program to illustrate (#) operator
#include <iostream>
using namespace std;

#define a 8.3297

int main()
{
    cout << "Value of a is " << a << endl;

    return 0;
}
chevron_right
filter_none
```

Output:

Value of a is 8.3297

3. This program finds out maximum out of two numbers using macro

```
filter_none
edit
close

play_arrow

link
brightness_4
code

// CPP program to illustrate (#) operator
#include <iostream>
using namespace std;

#define MAX(i, j) (((i) > (j)) ? i : j)

int main()
{
    int a, b;

    a = 250;
    b = 25;

    cout << "The maximum is " << MAX(a, b) << endl;

    return 0;
}
```

chevron_right

filter_none

Output:

The maximum is 250

Token-pasting operator (##)

Allows tokens used as actual arguments to be concatenated to form other tokens. It is often useful to merge two tokens into one while expanding macros. This is called token pasting or token concatenation. The '##' pre-processing operator performs token pasting. When a macro is expanded, the two tokens on either side of each '##' operator are combined into a single token, which then replaces the '##' and the two original tokens in the macro expansion.

Examples :

1. The preprocessor transforms printf("%d", concat(x, y)); into printf("%d", xy);

```
filter_none
edit
close

play_arrow

link
brightness_4
code

// CPP program to illustrate (##) operator
#include <stdio.h>
#define concat(a, b) a##b
int main(void)
{
    int xy = 30;
    printf("%d", concat(x, y));
    return 0;
}
```

chevron_right

filter_none

Output:

30

Application: The ## provides a way to concatenate actual arguments during macro expansion. If a parameter in the replacement text is adjacent to a ##, the parameter is replaced by the actual argument, the ## and surrounding white space are removed, and the result is re-scanned.

This article is contributed by **Shivani Ghugtyal**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes 

Add your personal notes
here! (max 5000 chars)

Save

Recommended Posts:

- [Operators in C | Set 2 \(Relational and Logical Operators\)](#)
- [Operators in C | Set 1 \(Arithmetic Operators\)](#)
- [Operators in C / C++](#)
- [C | Operators | Question 8](#)
- [Bitwise Operators in C/C++](#)
- [What are the operators that can be and cannot be overloaded in C++?](#)
- [C | Operators | Question 10](#)
- [Assignment Operators in C/C++](#)
- [C | Operators | Question 9](#)
- [C | Operators | Question 23](#)

- C | Operators | Question 7
- C | Operators | Question 6
- C | Operators | Question 24
- C | Operators | Question 5
- C | Operators | Question 4

Improved By : shrutiss48

Article Tags :
 C
 C-Operators
 cpp-operator
Practice Tags :
 cpp-operator
 C

thumb_up
 7

To-do Done
 1.8

Based on 14 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

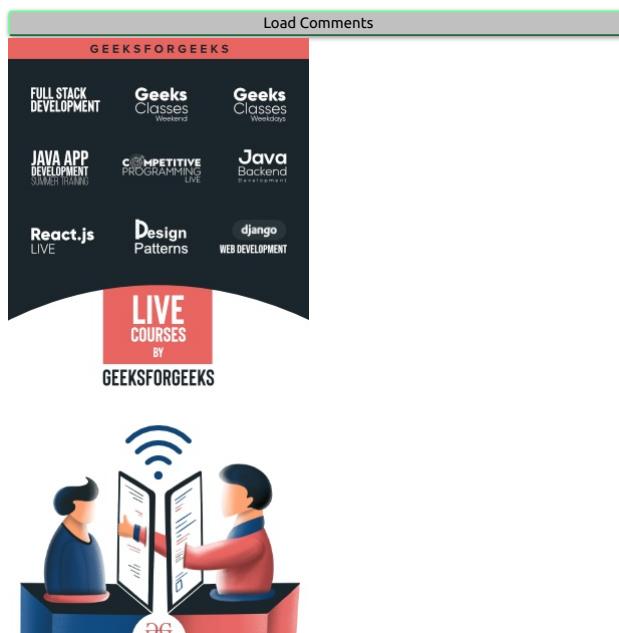
Previous

[first_page](#) [ispunct\(\) function in C](#)

Next

[last_page](#) [strcoll\(\) in C/C++](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C
[Print all possible combinations of the string by replacing '\\$' with any other digit from the string ctype.h\(<cctype>\) library in C/C++ with Examples](#)
[Predefined Macros in C with Examples](#)
[Implicit Conversions in C with Examples](#)
[How to call function within function in C or C++](#)

More related articles in C
[C program to print all line contents of a File followed by even line content](#)
[Introduction to the C99 Programming Language : Part II](#)
[C program to find square root of a given number](#)
[Format specifiers in different Programming Languages](#)
[Problem in comparing Floating point numbers and how to compare them correctly?](#)

Write a C macro PRINT(x) which prints x

At the first look, it seems that writing a C macro which prints its argument is child's play. Following program should work i.e. it should print x

```
filter_none
edit
close
play_arrow
link
brightness_4
code

#define PRINT(x) (x)

int main()
{
    printf("%s",PRINT(x));
    return 0;
}
chevron_right
filter_none
```

But it would issue compile error because the data type of x, which is taken as variable by the compiler, is unknown. Now it doesn't look so obvious. Isn't it? Guess what, the followings also won't work

```
filter_none
edit
close
play_arrow
```

```
link  
brightness_4  
code  
  
#define PRINT(x) ('x')  
#define PRINT(x) ("x")  
chevron_right  
  
filter_none
```

But if we know one of lesser known traits of C language, writing such a macro is really a child's play. In C, there's a # directive, also called 'Stringizing Operator', which does this magic. Basically # directive converts its argument in a string. Voila! it is so simple to do the rest. So the above program can be modified as below.

```
filter_none  
edit  
close  
  
play_arrow  
  
link  
brightness_4  
code  
  
#define PRINT(x) (#x)  
int main()  
{  
    printf("%s",PRINT(x));  
    return 0;  
}  
chevron_right  
  
filter_none
```

Now if the input is `PRINT(x)`, it would print `x`. In fact, if the input is `PRINT(geeks)`, it would print `geeks`.

You may find the details of this directive from Microsoft portal [here](#).

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes 

Add your personal notes here! (max 5000 chars)

Recommended Posts:

- [The OFFSETOF\(\) macro](#)
- [C | Macro & Preprocessor | Question 15](#)
- [C | Macro & Preprocessor | Question 14](#)
- [C | Macro & Preprocessor | Question 1](#)
- [C | Macro & Preprocessor | Question 2](#)
- [C | Macro & Preprocessor | Question 12](#)
- [C | Macro & Preprocessor | Question 11](#)
- [C | Macro & Preprocessor | Question 8](#)
- [C | Macro & Preprocessor | Question 6](#)
- [C | Macro & Preprocessor | Question 5](#)
- [C | Macro & Preprocessor | Question 4](#)
- [C | Macro & Preprocessor | Question 14](#)
- [C | Macro & Preprocessor | Question 9](#)
- [C | Macro & Preprocessor | Question 10](#)
- [C | Macro & Preprocessor | Question 7](#)

Article Tags :

C
C Macro
C-Macro & Preprocessor
c-puzzle

Practice Tags :

C

 10

To-do Done
2.3

Based on 38 vote(s)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) What is the purpose of a function prototype?

Next

[last_page](#) How to print % using printf()?

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C
`getch()` function in C with Examples
Print all possible combinations of the string by replacing '\$' with any other digit from the string
`ctype.h(<cctype>)` library in C/C++ with Examples
C program to display month by month calendar for a given year
Introduction to the C99 Programming Language : Part I

More related articles in C
Predefined Macros in C with Examples
Input type Conversion with Examples
Format specifier in different Programming Languages
How to call function within function in C or C++
C program to print odd line contents of a File followed by even line content

Variable length arguments for Macros

Like functions, we can also pass variable length arguments to macros. For this we will use the following preprocessor identifiers.

To support variable length arguments in macro, we must include ellipses (...) in macro definition. There is also `_VA_ARGS_` preprocessing identifier which takes care of variable length argument substitutions which are provided to macro. Concatenation operator ## (aka paste operator) is used to concatenate variable arguments.

Let us see with example. Below macro takes variable length argument like `"printf()"` function. This macro is for error logging. The macro prints filename followed by line number, and finally it prints info/error message. First argument "prio" determines the priority of message, i.e. whether it is information message or error, "stream" may be "standard output" or "standard error". It displays INFO messages on stdout and ERROR messages on stderr stream.

```
filter_none
edit
close
play_arrow
link
brightness_4
code
#include <stdio.h>

#define INFO    1
#define ERR    2
#define STD_OUT stdout
#define STD_ERR stderr

#define LOG_MESSAGE(prio, stream, msg, ...) do {\
    char *str;\
    if (prio == INFO)\\
        str = "INFO";\
    else if (prio == ERR)\\
        str = "ERR";\
    fprintf(stream, "[%s] : %s : %d : %s\n", \
            str, __FILE__, __LINE__, ##_VA_ARGS_);\
} while (0)
```

```
int main(void)
{
    char *s = "Hello";

    /* display normal message */
    LOG_MESSAGE(ERR, STD_ERR, "Failed to open file");

    /* provide string as argument */
    LOG_MESSAGE(INFO, STD_OUT, "%s Geeks for Geeks", s);

    /* provide integer as arguments */
    LOG_MESSAGE(INFO, STD_OUT, "%d + %d = %d", 10, 20, (10 + 20));

    return 0;
}
```

Compile and run the above program, it produces below result.

```
[narendra@/media/partition/GFG]$ ./variable_length
[ERR] : variable_length.c : 26 : Failed to open file
[INFO] : variable_length.c : 27 : Hello Geeks for Geeks
[INFO] : variable_length.c : 28 : 10 + 20 = 30
[narendra@/media/partition/GFG]$
```

My Personal Notes

Add your personal notes here! (max 5000 chars)

[Save](#)

Recommended Posts:

- How to Count Variable Numbers of Arguments in C?
- Variable Length Argument in C
- Variable Length Arrays in C and C++
- X-Macros in C
- Can we access global variable if there is a local variable with same name?
- Multiline macros in C
- Macros vs Functions
- Predefined Macros in C with Examples
- Branch prediction macros in GCC
- Hygienic Macros : An Introduction
- Interesting Facts about Macros and Preprocessors in C
- Data Type Ranges and their macros in C++
- Internal static variable vs. External static variable with Examples in C
- Command line arguments in C/C++
- Command line arguments example in C

Article Tags :

C
C-Macro & Preprocessor

c-puzzle

Practice Tags :

C



6

To-do Done
4.2

Based on 52 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) Sequence Points in C | Set 1

Next

[last_page](#) To find sum of two numbers without using any operator

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Weekend Geeks Classes Weekdays

JAVA APP DEVELOPMENT COMPETITIVE PROGRAMMING LIVE Java Backend Development

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS



Most popular in C
[getch\(\)](#) function in C with Examples
[Print all possible combinations of the string by replacing '\\$' with any other digit from the string](#)
[ctype.h\(<cctype>\) library in C/C++ with Examples](#)
[C program to display month by month calendar for a given year](#)
[Predefined Macros in C with Examples](#)

More related articles in C
[Implicit Type Conversion in C with Examples](#)
[Introduction to the C99 Programming Language : Part I](#)
[How to call function within function in C or C++](#)
[C program to print odd line contents of a file followed by even line content](#)
[Nested Loops in C with Examples](#)

Multiline macros in C

In this article, we will discuss how to write a multi-line macro. We can write multi-line macro same like function, but each statement ends with "\". Let us see with example. Below is simple macro, which accepts input number from user, and prints whether entered number is even or odd.

[filter_none](#)
[edit](#)
[close](#)

```

play_arrow
link
brightness_4
code

#include <stdio.h>

#define MACRO(num, str) { \
    printf("%d", num); \
    printf(" is"); \
    printf(" %s number", str); \
    printf("\n"); \
}

int main(void)
{
    int num;

    printf("Enter a number: ");
    scanf("%d", &num);

    if (num & 1)
        MACRO(num, "Odd");
    else
        MACRO(num, "Even");

    return 0;
}

```

chevron_right

filter_none

At first look, the code looks OK, but when we try to compile this code, it gives compilation error.

```
[narendra@media/partition/GFG]$ make macro
cc      macro.c -o macro
macro.c: In function 'main':
macro.c:19:2: error: 'else' without a previous 'if'
make: *** [macro] Error 1
[narendra@media/partition/GFG]$
```

Let us see what mistake we did while writing macro. We have enclosed macro in curly braces. According to C-language rule, each C-statement should end with semicolon. That's why we have ended MACRO with semicolon. Here is a mistake. Let us see how compiler expands this macro.

```

if (num & 1)
{
    -----
    ---- Macro expansion ----
    -----
}; /* Semicolon at the end of MACRO, and here is ERROR */

else
{
    -----
    ---- Macro expansion ----
    -----
};

```

We have ended macro with semicolon. When compiler expands macro, it puts semicolon after "if" statement. Because of semicolon between "if and else statement" compiler gives compilation error. Above program will work fine, if we ignore "else" part.

To overcome this limitation, we can enclose our macro in "do-while(0)" statement. Our modified macro will look like this.

```

filter_none
edit
close

play_arrow
link
brightness_4
code

#include <stdio.h>

#define MACRO(num, str) do { \
    printf("%d", num); \
    printf(" is"); \
    printf(" %s number", str); \
    printf("\n"); \
} while(0)

int main(void)
{
    int num;

    printf("Enter a number: ");
    scanf("%d", &num);

    if (num & 1)
        MACRO(num, "Odd");
    else
        MACRO(num, "Even");

    return 0;
}

```

chevron_right

filter_none

Compile and run above code, now this code will work fine.

```
[narendra@media/partition/GFG]$ make macro
```

```

cc    macro.c -o macro
[narendra@media/partition/GFG]$ ./macro
Enter a number: 9
9 is Odd number
[narendra@media/partition/GFG]$ ./macro
Enter a number: 10
10 is Even number
[narendra@media/partition/GFG]$

```

We have enclosed macro in "do - while(0)" loop and at the end of while, we have put condition as "while(0)", that's why this loop will execute only one time.

Similarly, instead of "do - while(0)" loop we can enclose multi-line macro in parenthesis. We can achieve the same result by using this trick. Let us see example.

filter_none
edit
close

play_arrow

link
brightness_4
code

```
#include <stdio.h>
```

```
#define MACRO(num, str) ({\
    printf("%d", num);\
    printf(" is");\
    printf(" %s number", str);\
    printf("\n");\
})
```

```
int main(void)
{
    int num;
```

```
    printf("Enter a number: ");
    scanf("%d", &num);
```

```
    if (num & 1)
        MACRO(num, "Odd");
    else
        MACRO(num, "Even");
```

```
    return 0;
}
```

chevron_right

filter_none

```

[narendra@media/partition/GFG]$ make macro
cc    macro.c -o macro
[narendra@media/partition/GFG]$ ./macro
Enter a number: 10
10 is Even number
[narendra@media/partition/GFG]$ ./macro
Enter a number: 15
15 is Odd number
[narendra@media/partition/GFG]$

```

This article is compiled by **Narendra Kangalkar**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes *arrow_drop_up*

Add your personal notes
here! (max 5000 chars)

Save

Recommended Posts:

- [X-Macros in C](#)
- [Macros vs Functions](#)
- [Predefined Macros in C with Examples](#)
- [Branch prediction macros in GCC](#)
- [Hygienic Macros : An Introduction](#)
- [Variable length arguments for Macros](#)
- [Interesting Facts about Macros and Preprocessors in C](#)
- [Data Type Ranges and their macros in C++](#)
- [Format specifiers in different Programming Languages](#)
- [C program to find square root of a given number](#)
- [C program to print odd line contents of a File followed by even line content](#)
- [Getting System and Process Information Using C Programming and Shell in Linux](#)
- [Program to calculate Electricity Bill](#)
- [Program to print half Diamond star pattern](#)

Article Tags :

[C](#)
[C-Macro & Preprocessor](#)
[Practice Tags :](#)
[C](#)

thumb_up
11

To-do Done

2.4

Based on 31 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Weekend Geeks Classes Weekdays

JAVA APP DEVELOPMENT COMPETITIVE PROGRAMMING DAY Java Backend Development

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS



Most popular in C
C program to display month by month calendar for a given year
Print all possible combinations of the string by replacing '\$' with any other digit from the string
getch() function in C with Examples
ctype.h<ctype> library in C/C++ with Examples
Implicit Type Conversion in C with Examples

More related articles in C
Introduction to the C99 Programming Language : Part I
Formatting in different Programming Languages
C program to find square root of a given number
Predefined Macros in C with Examples
How to call function within function in C or C++

CRASH() macro – interpretation

Given below a small piece of code from an open source project,

```
filter_none
edit
close
play_arrow
link
brightness_4
code
#ifndef __cplusplus

typedef enum BooleanTag
{
    false,
    true
} bool;

#endif

#define CRASH() do { \
    ((void(*)())0)(); \
} while(false)

int main()
{
    CRASH();
    return 0;
}
```

chevron_right

filter_none

Can you interpret above code?

It is simple, a step by step approach is given below,

The statement `while(false)` is meant only for testing purpose. Consider the following operation,

```
((void(*)())0)();
```

It can be achieved as follows,

```
0;          /* literal zero */
(0); ( ()0 );      /* 0 being casted to some type */
( (* ) 0 );        /* 0 casted some pointer type */
( ( * ) 0 );        /* 0 casted as pointer to some function */
( void (*) (void) 0 ); /* Interpret 0 as address of function
taking nothing and returning nothing */
( void (*) (void) 0 )(); /* Invoke the function */
```

So the given code is invoking the function whose code is stored at location zero, in other words, trying to execute an instruction stored at location zero. On systems with memory protection (MMU) the OS will throw an exception (segmentation fault) and on systems without such protection (small embedded systems), it will execute and error will propagate further.

— Venki. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

My Personal Notes [arrow_drop_up](#)

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- Reasons for a C++ program crash
- The OFFSETOF() macro
- C | Macro & Preprocessor | Question 5
- C | Macro & Preprocessor | Question 6
- C | Macro & Preprocessor | Question 7
- C | Macro & Preprocessor | Question 8
- C | Macro & Preprocessor | Question 9
- C | Macro & Preprocessor | Question 10
- C | Macro & Preprocessor | Question 11
- C | Macro & Preprocessor | Question 12
- C | Macro & Preprocessor | Question 13
- Difference between Inline and Macro in C++
- C | Macro & Preprocessor | Question 4
- C | Macro & Preprocessor | Question 15
- C | Macro & Preprocessor | Question 14

Article Tags :

C
C++
C-Macro & Preprocessor

Practice Tags :

C
CPP

4

To-do Done
4.6

Based on 40 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

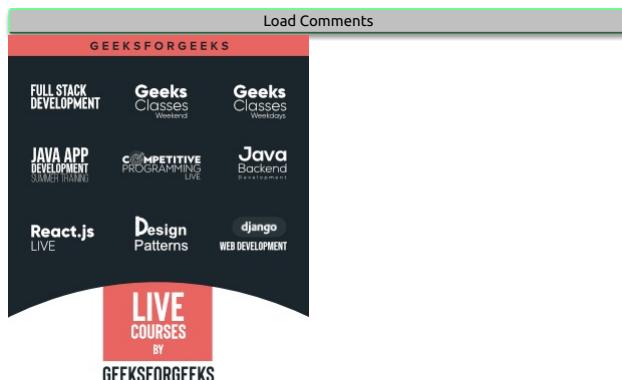
Previous

[first_page](#) [puts\(\)](#) vs [printf\(\)](#) for printing a string

Next

[last_page](#) [return statement](#) vs [exit\(\)](#) in main()

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C
[getch\(\)](#) function in C with Examples
Print multiple combinations of the string by replacing '\$' with any other digit from the string
[ctype.h](#) [cctype.h](#) library in C/C++ with Examples
C program to display month by month calendar for a given year
Introduction to the C99 Programming Language : Part I

Most visited in C++
Vector of Vectors in C++ STL with Examples
C++ Tutorial
Which C++ libraries are useful for competitive programming?
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

The OFFSETOF() macro

We know that the elements in a structure will be stored in sequential order of their declaration.

How to extract the displacement of an element in a structure? We can make use of `offsetof` macro.

Usually we call structure and union types (or *classes with trivial constructors*) as *plain old data (POD) types*, which will be used to *aggregate other data types*. The following non-standard macro can be used to get the displacement of an element in bytes from the base address of the structure variable.

```
#define OFFSETOF(TYPE, ELEMENT) ((size_t)&(((TYPE *)0)->ELEMENT))
```

Zero is casted to type of structure and required element's address is accessed, which is casted to `size_t`. As per standard `size_t` is of type `unsigned int`. The overall expression results in the number of bytes after which the ELEMENT being placed in the structure.

For example, the following code returns 16 bytes (padding is considered on 32 bit machine) as displacement of the character variable `c` in the structure `Pod`.

```
filter_none
edit
close
play_arrow
link
brightness_4
code
#include <stdio.h>

#define OFFSETOF(TYPE, ELEMENT) ((size_t)&(((TYPE *)0)->ELEMENT))

typedef struct PodTag
{
    int i;
    double d;
    char c;
} PodType;

int main()
{
    printf("%d", OFFSETOF(PodType, c) );
    getchar();
    return 0;
}
chevron_right
```

In the above code, the following expression will return the displacement of element `c` in the structure `PodType`.

```
OFFSETOF(PodType, c);
```

After preprocessing stage the above macro expands to

```
filter_none
edit
close
play_arrow
link
brightness_4
code
((size_t)&(((PodType *)0)->c))
chevron_right
```

Since we are considering 0 as address of the structure variable, `c` will be placed after 16 bytes of its base address i.e. `0x00 + 0x10`. Applying `&` on the structure element (in this case it is `c`) returns the address of the element which is `0x10`. Casting the address to `unsigned int (size_t)` results in number of bytes the element is placed in the structure.

Note: We may consider the address operator `&` is redundant. Without address operator in macro, the code de-references the element of structure placed at NULL address. It causes an access violation exception (segmentation fault) at runtime.

Note that there are other ways to implement offsetof macro according to compiler behaviour. The ultimate goal is to extract displacement of the element. We will see practical usage of offsetof macro in liked lists to connect similar objects (for example thread pool) in another article.

Article compiled by **Venki**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

References:

1. [Linux Kernel code](#).
2. <http://msdn.microsoft.com/en-us/library/dz4y9b9a.aspx>
3. [GNU C/C++ Compiler Documentation](#)

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up

Add your personal notes here! (max 5000 chars)

Recommended Posts:

- [C | Macro & Preprocessor | Question 8](#)
- [C | Macro & Preprocessor | Question 9](#)
- [C | Macro & Preprocessor | Question 10](#)
- [C | Macro & Preprocessor | Question 11](#)
- [C | Macro & Preprocessor | Question 12](#)
- [C | Macro & Preprocessor | Question 13](#)
- [C | Macro & Preprocessor | Question 14](#)
- [Difference between Inline and Macro in C++](#)
- [C | Macro & Preprocessor | Question 6](#)
- [C | Macro & Preprocessor | Question 5](#)
- [C | Macro & Preprocessor | Question 4](#)
- [C | Macro & Preprocessor | Question 15](#)
- [CRASH\(\) macro - interpretation](#)
- [C | Macro & Preprocessor | Question 7](#)
- [C | Macro & Preprocessor | Question 2](#)

Article Tags :

C
C++
C-Macro & Preprocessor

Practice Tags :

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) Increment (Decrement) operators require L-value Expression

Next

[last_page](#) Can namespaces be nested in C++?

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Webinar

Java APP DEVELOPMENT SUMMER TRAINING Geeks Classes Webinar

COMPETITIVE PROGRAMMING LIVE Geeks Classes Webinar

Java Backend Development

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS

Most popular in C C program to display month by month calendar for a given year Print all possible combinations of the string by replacing '\$' with any other digit from the string getch() function in C with Examples ctype.h <ctype.h> library in C/C++ with Examples Implicit Type Conversion in C with Examples

Most visited in C+++ Vector of Vectors in C++ STL with Examples C++ Tutorial Which C++ libraries are useful for competitive programming? Array of Vectors in C++ STL Map of Vectors in C++ STL with Examples

Branch prediction macros in GCC

One of the most used optimization techniques in the Linux kernel is “`_builtin_expect`”. When working with conditional code (if-else statements), we often know which branch is true and which is not. If compiler knows this information in advance, it can generate most optimized code.

Let us see macro definition of “`likely()`” and “`unlikely()`” macros from linux kernel code [“http://lxr.linux.no/linux+v3.6.5/include/linux/compiler.h”](http://lxr.linux.no/linux+v3.6.5/include/linux/compiler.h) [line no 146 and 147].

```
filter_none
edit
close
play_arrow
link
brightness_4
code

#define likely(x)    __builtin_expect (!!x, 1)
#define unlikely(x)  __builtin_expect (!!x, 0)
chevron_right
filter_none
```

In the following example, we are marking branch as likely true:

```
filter_none
edit
close
play_arrow
link
brightness_4
code

const char *home_dir ;

home_dir = getenv("HOME");
if (likely(home_dir))
    printf("home directory: %s\n", home_dir);
else
    perror("getenv");
chevron_right
filter_none
```

For above example, we have marked "if" condition as "likely()" true, so compiler will put true code immediately after branch, and false code within the branch instruction. In this way compiler can achieve optimization. But don't use "likely()" and "unlikely()" macros blindly. If prediction is correct, it means there is zero cycle of jump instruction, but if prediction is wrong, then it will take several cycles, because processor needs to flush its pipeline which is worst than no prediction.

Accessing memory is the slowest CPU operation as compared to other CPU operations. To avoid this limitation, CPU uses "CPU caches" e.g L1-cache, L2-cache etc. The idea behind cache is, copy some part of memory into CPU itself. We can access cache memory much faster than any other memory. But the problem is, limited size of "cache memory", we can't copy entire memory into cache. So, the CPU has to guess which memory is going to be used in the near future and load that memory into the CPU cache and above macros are hint to load memory into the CPU cache.

This article is compiled by **Narendra Kangalkar**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes [arrow_drop_up](#)

Add your personal notes here! (max 5000 chars)

[Save](#)

Recommended Posts:

- X-Macros in C
- Multiline macros in C
- Macros vs Functions
- Hygienic Macros : An Introduction
- Predefined Macros in C with Examples
- Data Type Ranges and their macros in C++
- Interesting Facts about Macros and Preprocessors in C
- Variable length arguments for Macros
- Format specifiers in different Programming Languages
- C program to find square root of a given number
- C program to print odd line contents of a File followed by even line content
- Getting System and Process Information Using C Programming and Shell in Linux
- Program to calculate Electricity Bill
- Program to print half Diamond star pattern

Article Tags :

C
AdvanceC
C-Macro & Preprocessor
c-puzzle
GCC

Practice Tags :

C

4

To-do Done
4.3

Based on 26 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

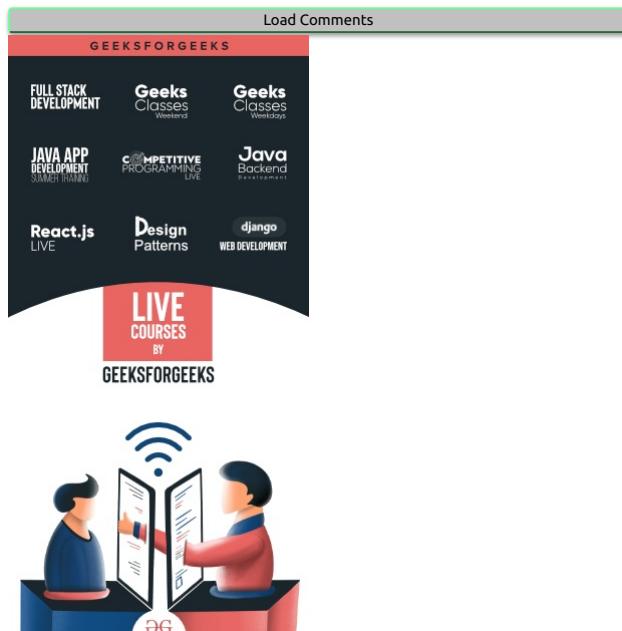
Previous

[first_page](#) Print 1 to 100 in C++, without loop and recursion

Next

[last_page](#) Program to print last 10 lines

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



getch() function in C with Examples
Print all possible combinations of the string by replacing 's' with any other digit from the string
C program to copy a library in C/C++ with Examples
C program to display month by month calendar for a given year
Introduction to the C99 Programming Language : Part I

More related articles in C
Predefined Macros in C with Examples
Implicit Type Conversion in C with Examples
Format specifiers in different Programming Languages
How to call function within function in C or C++
C program to print odd line contents of a file followed by even line content

Difference between #define and const in C?

#define is a [preprocessor directive](#). Things defined by #define are replaced by the preprocessor before compilation begins.

const variables are actual variables like other normal variable.

The big advantage of const over #define is type checking. We can also have pointers to const variables, we can pass them around, typecast them and any other thing that can be done with a normal variable. One disadvantage that one could think of is extra space for variable which is immaterial due to optimizations done by compilers.

In general const is a better option if we have a choice. There are situations when #define cannot be replaced by const. For example, #define can take parameters (See [this](#) for example). #define can also be used to replace some text in a program with another text.

This article is contributed by **Abhay Rathi**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes [arrow_drop_up](#)

Add your personal notes
here! (max 5000 chars)

Save

Recommended Posts:

- "static const" vs "#define" vs "enum"
- Difference between const int*, const int * const, and int const *
- Difference between const char *p, char * const p and const char * const p
- typedef versus #define in C
- Const Qualifier in C
- How to modify a const variable in C?
- C++ | const keyword | Question 5
- C++ | const keyword | Question 5
- C++ | const keyword | Question 3
- C++ | const keyword | Question 2
- C++ | const keyword | Question 1
- Function overloading and const keyword
- Why copy constructor argument should be const in C++?
- Format specifiers in different Programming Languages

Article Tags :

C
C-Macro & Preprocessor
cpp-macros

Practice Tags :

C

9

To-do Done
2.2

Based on 13 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) Comparison of a float with a value in C

Next

[last_page](#) C++ | Class and Object | Question 6

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Weekend Geeks Classes Weekdays

JAVA APP DEVELOPMENT SUMMER TRAINING COMPETITIVE PROGRAMMING LN Java Backend

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS



Most popular in C
getch() function in C with Examples
Print all possible combinations of the string by replacing '\$' with any other digit from the string
ctype.h <cctype> library in C/C++ with Examples
C program to display month by month calendar for a given year
Predefined Macros in C with Examples

More related articles in C
Input type character in C with Examples
Introduction to the C99 Programming Language : Part I
How to call function within function in C or C++
C program to print odd line contents of a File followed by even line content
Nested Loops in C with Examples

A C Programming Language Puzzle

Give $a = 12$ and $b = 36$ write a C function/macro that returns 3612 without using arithmetic, strings and predefined functions.

We strongly recommend you to minimize your browser and try this yourself first.

Below is one solution that uses String Token-Pasting Operator (`##`) of C macros. For example, the expression “`a##b`” prints concatenation of ‘`a`’ and ‘`b`’.

Below is a working C code.

```
filter_none
edit
close

play_arrow

link
brightness_4
code

#include <stdio.h>
#define merge(a, b) b##a
int main(void)
{
    printf("%d ", merge(12, 36));
    return 0;
}
chevron_right
```

filter_none

Output:

3612

Thanks to an anonymous user to suggest this solution.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes 

Add your personal notes here! (max 5000 chars)



Recommended Posts:

- [Features of C Programming Language](#)
- [C Programming Language Standard](#)
- [Introduction to the C99 Programming Language : Part II](#)
- [Introduction to the C99 Programming Language : Part III](#)
- [Benefits of C language over other programming languages](#)
- [Introduction to the C99 Programming Language : Part I](#)
- [Programming puzzle \(Assign value without any control statement\)](#)
- [kbhit in C language](#)
- [Difference between while\(1\) and while\(0\) in C language](#)
- [C Language Introduction](#)
- [fgets\(\) and gets\(\) in C language](#)
- [Stopwatch using C language](#)
- [Signals in C language](#)
- [Constants vs Variables in C language](#)
- [isalnum\(\) function in C Language](#)

Article Tags :

C
C-Macro & Preprocessor
cpp-macros
cpp-puzzle

Practice Tags :

C

 4

To-do Done
2.1

Based on 26 vote(s)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

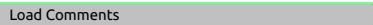
Previous

[first_page](#) [Static Variables in C](#)

Next

[last_page](#) [Storage Classes in C](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.





Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
getch() function in C with Examples
ctype.h(<cctype>) library in C/C++ with Examples
C program to display month by month calendar for a given year
Introduction to the C99 Programming Language : Part I

More related articles in C
Predefined Macros in C with Examples
Input type Conversion with Examples
Format specifier in different Programming Languages
How to call function within function in C or C++
C program to print odd line contents of a File followed by even line content

What's difference between header files "stdio.h" and "stdlib.h" ?

These are two important header files used in C programming. While "<stdio.h>" is header file for **Standard Input Output**, "<stdlib.h>" is header file for **Standard Library**. One easy way to differentiate these two header files is that "<stdio.h>" contains declaration of `printf()` and `scanf()` while "<stdlib.h>" contains declaration of `malloc()` and `free()`. In that sense, the main difference in these two header files can be considered that, while "<stdio.h>" contains header information for 'File related Input/Output' functions, "<stdlib.h>" contains header information for 'Memory Allocation/Freeing' functions.

Wait a minute, you said "<stdio.h>" is for file related IO but `printf()` and `scanf()` don't deal with files... or are they? As a basic principle, in C (due to its association with UNIX history), keyboard and display are also treated as 'files'! In fact keyboard input is the default `stdin` file stream while display output is the default `stdout` file stream. Also, please note that, though "<stdlib.h>" contains declaration of other types of functions as well that aren't related to memory such as `atoi()`, `exit()`, `rand()` etc. yet for our purpose and simplicity, we can remember `malloc()` and `free()` for "<stdlib.h>".

It should be noted that a header file can contain not only function declaration but definition of constants and variables as well. Even macros and definition of new data types can also be added in a header file.

Please do Like/Tweet/G+1 if you find the above useful. Also, please do leave us comment for further clarification or info. We would love to help and learn

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes [arrow_drop_up](#)

Add your personal notes here! (max 5000 chars)

[Save](#)

Recommended Posts:

- [Difference between Header file and Library](#)
- [clocale header file in C++](#)
- [Comment in header file name?](#)
- [How to write your own header file in C?](#)
- [time.h header file in C with Examples](#)
- [Print "Hello World" in C/C++ without using any header file](#)
- [<complex.h> header file in C with Examples](#)
- [accumulate\(\) and partial_sum\(\) in C++ STL : numeric header](#)
- [numeric header in C++ STL | Set 2 \(adjacent_difference\(\), inner_product\(\) and iota\(\)\)](#)
- [Namespace in C++ | Set 3 \(Accessing, creating header, nesting and aliasing\)](#)
- [Linking Files having same variables with different data types in C](#)
- [C program to compare two files and report mismatches](#)
- [C Program to list all files and sub-directories in a directory](#)
- [C Program to merge contents of two files into a third file](#)
- [Difference between JSP and ASP](#)

Article Tags :

[Articles](#)

[C](#)

[Difference Between](#)

[C-Macro & Preprocessor](#)

[CPP-Library](#)

[Practice Tags :](#)

[C](#)

[thumb_up](#)

14

To-do Done
1.7

Based on 21 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

[Post navigation](#)

[Previous](#)

[first_page](#) [Difference between Priority Inversion and Priority Inheritance](#)

[Next](#)

[last_page](#) [Difference between HTML and HTTP](#)



Most popular articles
Must Do Coding Questions for Companies like Amazon, Microsoft, Adobe, ...
Must Do Coding Questions Company-wise
Amazon Interview Experience for SDE2 | 3+ years Experienced
C++ Tutorial
Check if a Binary Tree is BST : Simple and Efficient Approach

Most visited in C
Print Prime Number Examples
Speed up Code execution with help of Pragma in C/C++
Program to calculate Electricity Bill
Modulo Operator (%) in C/C++ with Examples
Role of SemiColon in various Programming Languages

How to print a variable name in C?

How to print and store a variable name in string variable?

We strongly recommend you to minimize your browser and try this yourself first

In C, there's a # directive, also called 'Stringizing Operator', which does this magic. Basically # directive converts its argument in a string.

```
filter_none
edit
close
play_arrow
link
brightness_4
code
#include <stdio.h>
#define getName(var) #var
```

int main()

```
{
```

```
    int myVar;
    printf("%s", getName(myVar));
    return 0;
}
```

```
chevron_right
```

```
filter_none
```

Output:

```
myVar
```

We can also store variable name in a string using `sprintf()` in C.

```
filter_none
edit
close
play_arrow
link
brightness_4
code
#include <stdio.h>
#define getName(var, str) sprintf(str, "%s", #var)
```

```
int main()
```

```
{
```

```
    int myVar;
    char str[20];
    getName(myVar, str);
    printf("%s", str);
    return 0;
}
```

```
chevron_right
```

```
filter_none
```

Output:

```
myVar
```

My Personal Notes 

Add your personal notes
here! (max 5000 chars)

Recommended Posts:

- Can we access global variable if there is a local variable with same name?
- Internal static variable vs. External static variable with Examples in C
- Variable Length Argument in C
- Variable Length Arrays in C and C++
- Different ways to initialize a variable in C/C++
- How to modify a const variable in C?
- Redeclaration of global variable in C
- Why variable name does not start with numbers in C?
- C | Variable Declaration and Scope | Question 1
- C | Variable Declaration and Scope | Question 2
- C | Variable Declaration and Scope | Question 3
- C | Variable Declaration and Scope | Question 4
- C | Variable Declaration and Scope | Question 6
- C | Variable Declaration and Scope | Question 7
- C | Variable Declaration and Scope | Question 8

Article Tags :

C
C-Macro & Preprocessor
c-puzzle
cpp-macros
Practice Tags :
C



6

To-do Done
3

Based on 17 vote(s)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) [fprintf\(\)](#) in C

Next

[last_page](#) C++ | Misc C++ | Question 8

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Weekend Geeks Classes Weekdays

JAVA APP DEVELOPMENT COMPETITIVE PROGRAMMING LIVE Java Backend EXPERTISE

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS



Most popular in C
C program to display month by month calendar for a given year
Print all possible combinations of the string by replacing '\$' with any other digit from the string
getch() function in C with Examples
ctype(ctype.h) library in C/C++ with Examples
Implicit Type Conversion in C with Examples

More related articles in C
Introduction to the C Programming Language : Part I
Format specifiers in different Programming Languages
C program to find square root of a given number
Predefined Macros in C with Examples
How to call function within function in C or C++

Constants in C/C++

As the name suggests the name constants is given to such variables or values in C/C++ programming language which cannot be modified once they are defined. They are fixed values in a program. There can be any types of constants like integer, float, octal, hexadecimal, character constants etc. Every constant has some range. The integers that are too big to fit into an int will be taken as long. Now there are various ranges that differ from unsigned to signed bits. Under the signed bit, the range of an int varies from -128 to +127 and under the unsigned bit, int varies from 0 to 255.

Defining Constants:

In C/C++ program we can define constants in two ways as shown below:

1. Using `#define` preprocessor directive
2. Using a `const` keyword

Literals: The values assigned to each constant variables are referred to as the *literals*. Generally, both terms, constants and literals are used interchangeably. For eg, “`const int = 5;`”, is a constant expression and the value 5 is referred to as constant integer literal.

Refer here for various [Types of Literals in C++](#).

Let us now learn about above two ways in details:

1. **Using `#define` preprocessor directive:** This directive is used to declare an alias name for existing variable or any value. We can use this to declare a constant as shown below:

```
#define identifierName value
```

- **identifierName:** It is the name given to constant.
- **value:** This refers to any value assigned to identifierName.

Example:

C

```
filter_none
edit
close

play_arrow
link
brightness_4
code

#include<stdio.h>
#define val 10
#define floatVal 4.5
#define charVal 'G'

int main()
{
    printf("Integer Constant: %d\n",val);
    printf("Floating point Constant: %.1f\n",floatVal);
    printf("Character Constant: %c\n",charVal);

    return 0;
}
chevron_right
filter_none
```

C++

```
filter_none
edit
close

play_arrow
link
brightness_4
code

#include <iostream>
using namespace std;

#define val 10
#define floatVal 4.5
#define charVal 'G'

int main() {
    cout << "Integer Constant: " << val << "\n";
    cout << "Floating point Constant: " << floatVal << "\n";
    cout << "Character Constant: "<< charVal << "\n";

    return 0;
}
chevron_right
filter_none
```

Output:

```
Integer Constant: 10
Floating point Constant: 4.5
Character Constant: G
```

Refer [Macros and Preprocessors in C](#) for details.

2. **using a `const` keyword:** Using `const` keyword to define constants is as simple as defining variables, the difference is you will have to precede the definition with a `const` keyword.

How to declare constants

const int var;



const int var;
var = 5;



const int var = 5;



✓ DG

Below program shows how to use const to declare constants of different data types:

C

```
filter_none  
edit  
close  
play_arrow  
link  
brightness_4  
code  
  
#include <stdio.h>  
  
int main()  
{  
    // int constant  
    const int intValue = 10;  
  
    // Real constant  
    const float floatVal = 4.14;  
  
    // char constant  
    const char charVal = 'A';  
  
    // string constant  
    const char stringVal[10] = "ABC";  
  
    printf("Integer constant: %d \n", intValue );  
    printf("Floating point constant: %.2f\n", floatVal );  
    printf("Character constant: %c\n", charVal );  
    printf("String constant: %s\n", stringVal);  
  
    return 0;  
}  
chevron_right
```

filter_none

C++

```
filter_none  
edit  
close  
play_arrow  
link  
brightness_4  
code  
  
#include <iostream>  
using namespace std;  
  
int main() {  
    // int constant  
    const int intValue = 10;  
  
    // Real constant  
    const float floatVal = 4.14;  
  
    // char constant  
    const char charVal = 'A';  
  
    // string constant  
    const string stringVal = "ABC";  
  
    cout << "Integer Constant: " << intValue << "\n";  
    cout << "Floating point Constant: " << floatVal << "\n";  
    cout << "Character Constant: "<< charVal << "\n";  
    cout << "String Constant: "<< stringVal << "\n";  
  
    return 0;  
}
```

chevron_right

filter_none

Output:

```
Integer constant: 10
Floating point constant: 4.14
Character constant: A
String constant: ABC
```

Refer [Const Qualifier in C](#) for details.

This article is contributed by **Chinmoy Lenka**. If you like GeeksforGeeks and would like to contribute, you can also write an article using [contribute.geeksforgeeks.org](#) or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes *arrow_drop_up*

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- [Constants vs Variables in C language](#)
- [Data type of character constants in C and C++](#)
- [Format specifiers in different Programming Languages](#)
- [C program to find square root of a given number](#)
- [C program to print odd line contents of a File followed by even line content](#)
- [Getting System and Process Information Using C Programming and Shell in Linux](#)
- [Program to calculate Electricity Bill](#)
- [Program to print half Diamond star pattern](#)
- [C/C++ program for calling main\(\) in main\(\)](#)
- [Count of integral coordinates that lies inside a Square](#)
- [Identical Splitting in a rectangular grid](#)
- [Program to print window pattern](#)
- [Print all possible combinations of the string by replacing '\\$' with any other digit from the string](#)
- [How to use make utility to build C projects?](#)

Article Tags :

C
CBSE - Class 11
school-programming
Practice Tags :

C

thumb_up
9

To-do Done
2

Based on 12 vote(s)

Basic **Easy** **Medium** **Hard** **Expert**

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) Write a C program to print "GfG" repeatedly without using loop, recursion and any control structure?

Next

[last_page](#) C program to detect tokens in a C program

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT **Geeks Classes** Weekend **Geeks Classes** Weekdays

JAVA APP DEVELOPMENT COMPETITIVE PROGRAMMING LIVE Java Backend EXPERTISE

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS

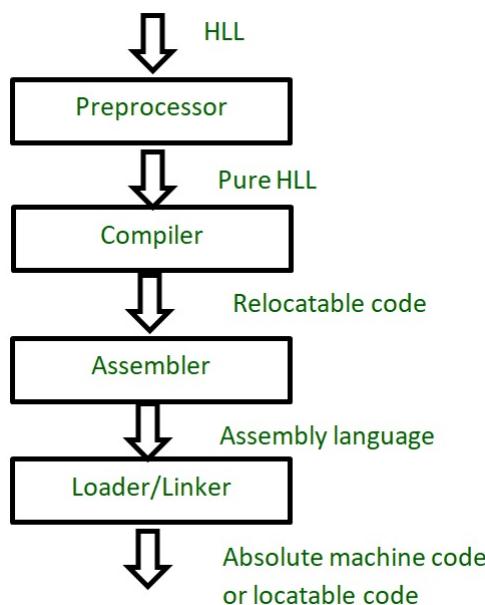
Most popular in C
[Print all possible combinations of the string by replacing '\\$' with any other digit from the string](#)
[Predefined Macros in C with Examples](#)
[C program to print odd line contents of a File followed by even line content](#)
[Introduction to the C99 Programming Language : Part II](#)
[C program to find square root of a given number](#)

How a Preprocessor works in C?

Compiling a C program - Behind the Scene

A Preprocessor is a system software (a computer program that is designed to run on computer's hardware and application programs). It performs preprocessing of the High Level Language(HLL). Preprocessing is the first step of the language processing system. Language processing system translates the high level language to machine level language or absolute machine code(i.e. to the form that can be understood by machine).

- The preprocessor doesn't know about the scope rules of C. Preprocessor directives like #define come into effect as soon as they are seen and remain in effect until the end of the file that contains them; the program's block structure is irrelevant.



A Preprocessor mainly performs three tasks on the HLL code :

- Removing comments :** It removes all the comments. A comment is written only for the humans to understand the code. So, it is obvious that they are of no use to a machine. So, preprocessor removes all of them as they are not required in the execution and won't be executed as well.

This is how to see a file with removed comments in linux :

Write a C code (let the file name be `prog.c`). Preprocess it using the command `gcc -E prog.c`

```
diksha@diksha-VirtualBox: ~
//this is my program
#include<stdio.h>
int main(){
printf("hiii");//i love coding
return 0;
}
"prog.c" 6 lines, 94 characters
```

You will see the output with the code having no comments.

```
diksha@diksha-VirtualBox: ~

extern char *ctermid (char *_s) __attribute__ ((__nothrow__ , __leaf__));
# 912 "/usr/include/stdio.h" 3 4
extern void flockfile (FILE *_stream) __attribute__ ((__nothrow__ , __leaf__));

extern int ftrylockfile (FILE *_stream) __attribute__ ((__nothrow__ , __leaf__)) ;

extern void funlockfile (FILE *_stream) __attribute__ ((__nothrow__ , __leaf__));
# 942 "/usr/include/stdio.h" 3 4
# 3 "prog.c" 2

# 3 "prog.c"
int main(){
printf("hiii");
return 0;
}
```

This file is saved with a '.' extension (`prog.i`) which will be input to the compiler.

- File inclusion :** Including all the files from library that our program needs. In HLL we write `#include` which is a directive for the preprocessor that tells it to include the contents of the library file specified.

For example, `#include` will tell the preprocessor to include all the contents in the library file `stdio.h`.

This can also be written using double quotes - `#include "stdio.h"`

Note: If the filename is enclosed within angle brackets, the file is searched for in the standard compiler include paths. If the filename is enclosed within double quotes, the search path is expanded to

include the current source directory.

3. **Macro expansion :** Macros can be called as small functions that are not as overhead to process. If we have to write a function (having a small definition) that needs to be called recursively (again and again), then we should prefer a macro over a function.

So, defining these macros is done by preprocessor.

`#define SI 1000`

is a simple example of a macro.

- There are two types of macros: Object-like (do not take parameters) and function-like (Can take parameters)

```
// object-like macro  
#define  
// function-like macro  
#define ()
```

- You can delete a macro definition with #undef:

```
// delete the macro  
#undef
```

- We can write multi-line macro same like function, but each statement ends with "\".

```
filter_none  
edit_close  
play_arrow  
link brightness_4 code  
#include <stdio.h>
```

```
#define MACRO(num, str) {\  
    printf("%d", num);\  
    printf(" is");\  
    printf(" %s number", str);\  
    printf("\n");\  
}
```

chevron_right

filter_none

This article is contributed by **Diksha**. If you like GeeksforGeeks and would like to contribute, you can also write an article using [contribute.geeksforgeeks.org](#) or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up

Add your personal notes
here! (max 5000 chars)

Save

Recommended Posts:

- [C/C++ Preprocessor directives | Set 2](#)
- [C | Macro & Preprocessor | Question 2](#)
- [C | Macro & Preprocessor | Question 9](#)
- [C | Macro & Preprocessor | Question 13](#)
- [C | Macro & Preprocessor | Question 11](#)
- [C | Macro & Preprocessor | Question 10](#)
- [C | Macro & Preprocessor | Question 8](#)
- [C | Macro & Preprocessor | Question 7](#)
- [C | Macro & Preprocessor | Question 6](#)
- [C | Macro & Preprocessor | Question 1](#)
- [C | Macro & Preprocessor | Question 5](#)
- [C | Macro & Preprocessor | Question 14](#)
- [C | Macro & Preprocessor | Question 14](#)
- [C | Macro & Preprocessor | Question 4](#)
- [C | Macro & Preprocessor | Question 15](#)

Article Tags :

C
C-Macro & Preprocessor

Practice Tags :

C

thumb_up
12

To-do Done
2

Based on 9 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) C program that does not suspend when Ctrl+Z is pressed

Next

[last_page](#) Convert C/C++ code to assembly language

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

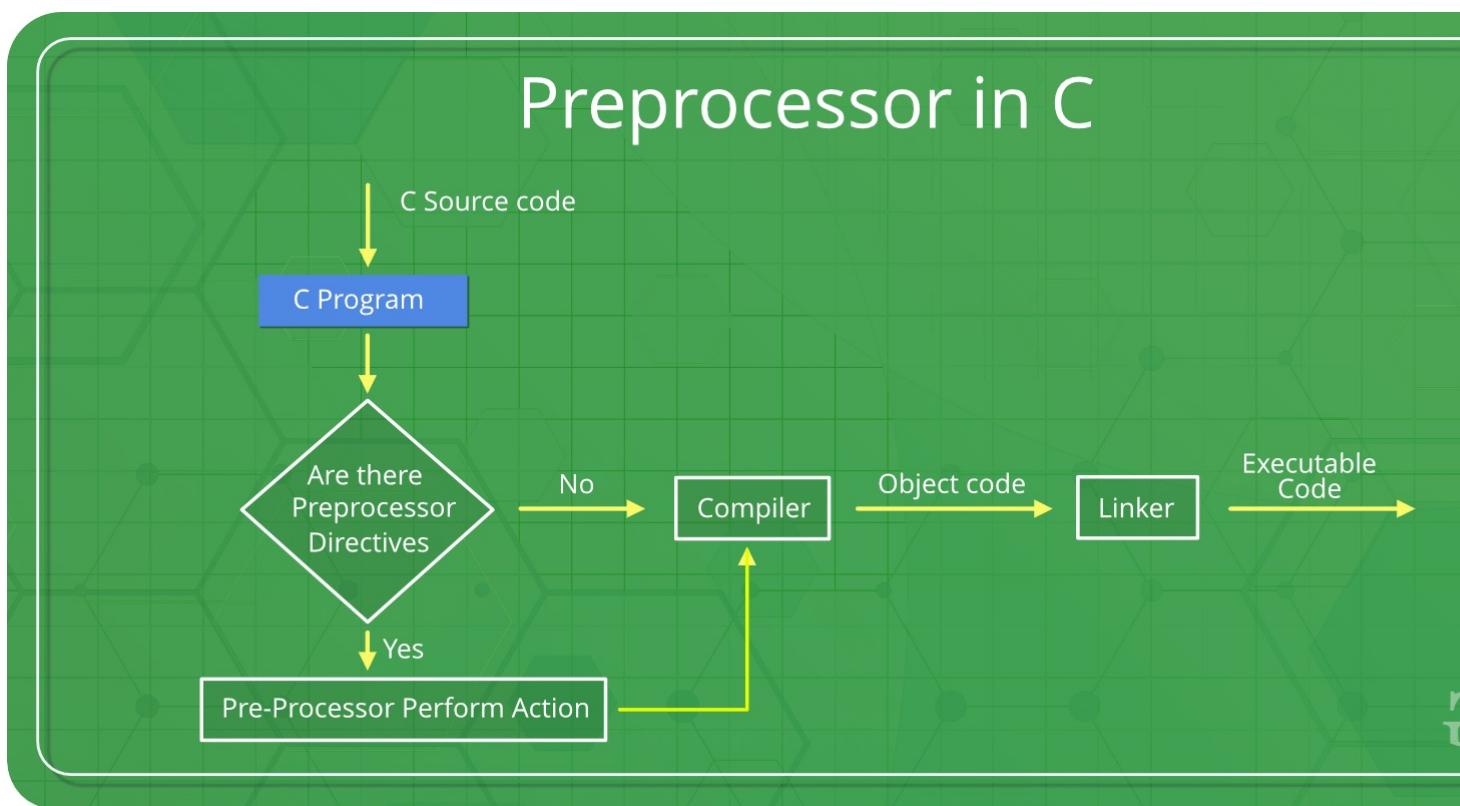


Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
Predefined Macros in C with Examples
Format specifiers in different Programming Languages
C program to print odd line contents of a file followed by even line content
Introduction to the C99 Programming Language : Part II

More related articles in C
C program to find square root of a given number
Program for comparing floating point numbers and how to compare them correctly?
Features of C Programming Language
Pointer Expressions in C with Examples
Introduction to the C99 Programming Language : Part III

C/C++ Preprocessors

As the name suggests Preprocessors are programs that process our source code before compilation. There are a number of steps involved between writing a program and executing a program in C / C++. Let us have a look at these steps before we actually start learning about Preprocessors.



You can see the intermediate steps in the above diagram. The source code written by programmers is stored in the file `program.c`. This file is then processed by preprocessors and an expanded source code file is generated named `program`. This expanded file is compiled by the compiler and an object code file is generated named `program.obj`. Finally, the linker links this object code file to the object code of the library functions to generate the executable file `program.exe`.

Preprocessor programs provide preprocessors directives which tell the compiler to preprocess the source code before compiling. All of these preprocessor directives begin with a '#' (hash) symbol. The '#' symbol indicates that, whatever statement start with #, is going to preprocessor program, and preprocessor program will execute this statement. Examples of some preprocessor directives are: `#include`, `#define`, `#ifndef` etc. Remember that # symbol only provides a path that it will go to preprocessor, and command such as `include` is processed by preprocessor program. For example `include` will include extra code to your program. We can place these preprocessor directives anywhere in our program.

There are 4 main types of preprocessor directives:

1. Macros
2. File Inclusion
3. Conditional Compilation
4. Other directives

Let us now learn about each of these directives in details.

- **Macros:** Macros are a piece of code in a program which is given some name. Whenever this name is encountered by the compiler the compiler replaces the name with the actual piece of code. The '#define' directive is used to define a macro. Let us now understand the macro definition with the help of a program:

C

```
filter_none
edit
close
play_arrow
link
brightness_4
code

#include <stdio.h>

// macro definition
#define LIMIT 5
int main()
{
    for (int i = 0; i < LIMIT; i++) {
        printf("%d \n", i);
    }

    return 0;
}
chevron_right
filter_none
```

C++

```
filter_none
edit
close
play_arrow
link
brightness_4
code

#include <iostream>

// macro definition
#define LIMIT 5
int main()
{
    for (int i = 0; i < LIMIT; i++) {
        std::cout << i << "\n";
    }

    return 0;
}
chevron_right
filter_none
```

Output:

```
0
1
2
3
4
```

In the above program, when the compiler executes the word LIMIT it replaces it with 5. The word 'LIMIT' in the macro definition is called a macro template and '5' is macro expansion.
Note: There is no semi-colon(';) at the end of macro definition. Macro definitions do not need a semi-colon to end.

Macros with arguments: We can also pass arguments to macros. Macros defined with arguments works similarly as functions. Let us understand this with a program:

C

```
filter_none
edit
close
play_arrow
link
brightness_4
code

#include <stdio.h>

// macro with parameter
#define AREA(l, b) (l * b)
int main()
{
    int l1 = 10, l2 = 5, area;

    area = AREA(l1, l2);

    printf("Area of rectangle is: %d", area);

    return 0;
}
chevron_right
filter_none
```

C++

```

filter_none
edit
close

play_arrow

link
brightness_4
code

#include <iostream>

// macro with parameter
#define AREA(l, b) (l * b)
int main()
{
    int l1 = 10, l2 = 5, area;

    area = AREA(l1, l2);

    std::cout << "Area of rectangle is: " << area;

    return 0;
}
chevron_right
filter_none

```

Output:

```
Area of rectangle is: 50
```

We can see from the above program that whenever the compiler finds AREA(l, b) in the program it replaces it with the statement (l*b) . Not only this, the values passed to the macro template AREA(l, b) will also be replaced in the statement (l*b). Therefore AREA(10, 5) will be equal to 10*5.

- **File Inclusion:** This type of preprocessor directive tells the compiler to include a file in the source code program. There are two types of files which can be included by the user in the program:

1. **Header File or Standard files:** These files contains definition of pre-defined functions like printf(), scanf() etc. These files must be included for working with these functions. Different function are declared in different header files. For example standard I/O funtions are in 'iostream' file whereas functions which perform string operations are in 'string' file.

Syntax:

```
#include< file_name >
```

where *file_name* is the name of file to be included. The '<' and '>' brackets tells the compiler to look for the file in standard directory.

2. **user defined files:** When a program becomes very large, it is good practice to divide it into smaller files and include whenever needed. These types of files are user defined files. These files can be included as:

```
#include"filename"
```

- **Conditional Compilation:** Conditional Compilation directives are type of directives which helps to compile a specific portion of the program or to skip compilation of some specific part of the program based on some conditions. This can be done with the help of two preprocessing commands '**ifdef**' and '**endif**'.

Syntax:

```
#ifdef macro_name
    statement1;
    statement2;
    statement3;
    .
    .
    .
    statementN;
#endif
```

If the macro with name as 'macroname' is defined then the block of statements will execute normally but if it is not defined, the compiler will simply skip this block of statements.

- **Other directives:** Apart from the above directives there are two more directives which are not commonly used. These are:

1. **#undef Directive:** The #undef directive is used to undefine an existing macro. This directive works as:

```
#undef LIMIT
```

Using this statement will undefine the existing macro LIMIT. After this statement every "#ifdef LIMIT" statement will evaluate to false.

2. **#pragma Directive:** This directive is a special purpose directive and is used to turn on or off some features. This type of directives are compiler-specific i.e., they vary from compiler to compiler. Some of the #pragma directives are discussed below:

- **#pragma startup** and **#pragma exit:** These directives helps us to specify the functions that are needed to run before program startup(before the control passes to main()) and just before program exit (just before the control returns from main()).

Note: Below program will not work with GCC compilers.

Look at the below program:

```
filter_none
edit
close
```

```
play_arrow
```

```
link
```

```
brightness_4
```

```
code
```

```
#include <stdio.h>
```

```
void func1();
void func2();
```

```
#pragma startup func1
#pragma exit func2
```

```
void func1()
{
    printf("Inside func1()\n");
}
```

```
void func2()
{
    printf("Inside func2()\n");
}
```

```
int main()
{
    void func1();
```

```
void func2();
printf("Inside main()\n");
```

```
return 0;
}
```

```
chevron_right
```

```
filter_none
```

Output:

```
Inside func1()
Inside main()
Inside func2()
```

The above code will produce the output as given below when run on GCC compilers:

```
Inside main()
```

This happens because GCC does not support #pragma startup or exit. However you can use the below code for a similar output on GCC compilers.

```
filter_none
edit
close

play_arrow

link
brightness_4
code

#include <stdio.h>
```

```
void func1();
void func2();
```

```
void __attribute__((constructor)) func1();
void __attribute__((destructor)) func2();
```

```
void func1()
{
    printf("Inside func1()\n");
}
```

```
void func2()
{
    printf("Inside func2()\n");
}
```

```
int main()
{
    printf("Inside main()\n");

    return 0;
}
```

```
chevron_right
```

```
filter_none
```

- **#pragma warn Directive:** This directive is used to hide the warning message which are displayed during compilation.

We can hide the warnings as shown below:

- **#pragma warn -rvl:** This directive hides those warning which are raised when a function which is supposed to return a value does not returns a value.

- **#pragma warn -par:** This directive hides those warning which are raised when a function does not uses the parameters passed to it.

- **#pragma warn -rch:** This directive hides those warning which are raised when a code is unreachable. For example: any code written after the `return` statement in a function is unreachable.

This article is contributed by **Harsh Agarwal**. If you like GeeksforGeeks and would like to contribute, you can also write an article using [contribute.geeksforgeeks.org](#) or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes

Add your personal notes here! (max 5000 chars)

Recommended Posts:

- [Interesting Facts about Macros and Preprocessors in C](#)
- [Rust vs C++ Will Rust Replace C++ in Future ?](#)
- [Priority queue of pairs in C++ with ordering by first and second element](#)
- [Implementation of lower_bound\(\) and upper_bound\(\) in Vector of Pairs in C++](#)
- [Generating RGBA portable graphic images through C++](#)
- [std::basic_istream::ignore in C++ with Examples](#)
- [std::basic_istream::getline in C++ with Examples](#)
- [Format specifiers in different Programming Languages](#)
- [C program to find square root of a given number](#)
- [C program to print odd line contents of a File followed by even line content](#)
- [std::is_heap\(\) in C++ with Examples](#)
- [std::basic_istream::gcount\(\) in C++ with Examples](#)
- [new vs malloc\(\) and free\(\) vs delete in C++](#)
- [std::string::rfind in C++ with Examples](#)

Improved By : RandomGuy7, Mohit Sehgal, mnnt4abhishek

Article Tags :

C

C++

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) [fgetc\(\) and fputc\(\) in C](#)

Next

[last_page](#) [C program to compare two files and report mismatches](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Wednesday
JAVA APP DEVELOPMENT COMPETITIVE PROGRAMMING Java Backend Backend
React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS



Most popular in C
P(i) in C++ with Examples
Sieve Of Eratosthenes with help of Pragma in C/C++
Module Operator (%) in C/C++ with Examples
Program to calculate Electricity Bill
Role of SemiColon in various Programming Languages

Most visited in C++
Vector of Vectors in C++ STL with Examples
P(i) in C++ with Examples
Which C++ libraries are useful for competitive programming?
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

C/C++ Preprocessor directives | Set 2

[C/C++ Preprocessor directives basics](#)

Preprocessor directives: In almost every program we come across in C/C++, we see a few lines at the top of the program preceded by a hash (#) sign. These lines are preprocessed by the compiler before actual compilation begins. The end of these lines are identified by the newline character '\n', no semicolon ';' is needed to terminate these lines.

Preprocessor directives are mostly used in defining macros, evaluating conditional statements, source file inclusion, pragma directive, line control, error detection etc.

In this post, we will discuss about some more types of preprocessor directives given below:

1. Conditional Compilation
2. Line control
3. Error directive

Let us now have a look about each one of these directives in details:

▪ **Conditional Compilation:** Conditional Compilation directives help to compile a specific portion of the program or let us skip compilation of some specific part of the program based on some conditions. In our previous article, we have discussed about two such directives '**#ifdef**' and '**#endif**'. In this post we will discuss **#ifndef**, **#if**, **#else** and **#elif**.

1. **#ifdef:** This directive is the simplest conditional directive. This block is called a conditional group. The controlled text will get included in the preprocessor output iff the macroname is defined. The controlled text inside a conditional will embrace preprocessing directives. They are executed only if the conditional succeeds. You can nest these in multiple layers, but they must be completely nested. In other words, '#endif' always matches the nearest '#ifdef' (or '#ifndef', or '#if'). Also, you can't begin a conditional group in one file and finish it in another.

Syntax:

```
#ifdef MACRO
    controlled text
#endif /* macroname */
```

2. **#ifndef:** We know that the in #ifdef directive if the macroname is defined, then the block of statements following the #ifdef directive will execute normally but if it is not defined, the compiler will simply skip this block of statements. The #ifndef directive is simply opposite to that of the #ifdef directive. In case of #ifndef , the block of statements between #ifndef and #endif will be executed only if the macro or the identifier with #ifndef is not defined.

Syntax :

```
#ifndef macro_name
    statement1;
    statement2;
    statement3;
    .
    .
    .
```

```
statementN;
#endif
```

If the macro with name as 'macroName' is not defined using the #define directive then only the block of statements will execute.

- 3. **#if, #else and #elif:** These directives works together and control compilation of portions of the program using some conditions. If the condition with the #if directive evaluates to a non zero value, then the group of line immediately after the #if directive will be executed otherwise if the condition with the #elif directive evaluates to a non zero value, then the group of line immediately after the #elif directive will be executed else the lines after #else directive will be executed.

Syntax:

```
#if macro_condition
    statements
#elif macro_condition
    statements
#else
    statements
#endif
```

Example:

```
filter_none
edit
close

play_arrow

link
brightness_4
code

#include<iostream>

#define gfg 7

#if gfg > 200
    #undef gfg
    #define gfg 200
#elif gfg < 50
    #undef gfg
    #define gfg 50
#else
    #undef gfg
    #define gfg 100
#endif
```

```
int main()
{
    std::cout << gfg; // gfg = 50
}
```

chevron_right

filter_none

Output:

50

Notice how the entire structure of #if, #elif and #else chained directives ends with #endif.

- **Line control (#line):** Whenever we compile a program, there are chances of occurrence of some error in the program. Whenever compiler identifies error in the program it provides us with the filename in which error is found along with the list of lines and with the exact line numbers where the error is. This makes easy for us to find and rectify error.

However we can control what information should the compiler provide during errors in compilation using the #line directive.

Syntax:

```
#line number "filename"
```

number - line number that will be assigned to the next code line. The line numbers of successive lines will be increased one by one from this point on.

"**filename**" - optional parameter that allows to redefine the file name that will be shown.

- **Error directive (#error):** This directive aborts the compilation process when it is found in the program during compilation and produces an error which is optional and can be specified as a parameter.

Syntax:

```
#error optional_error
```

Here, **optional_error** is any error specified by the user which will be shown when this directive is found in the program.

Example:

```
filter_none
edit
close

play_arrow

link
brightness_4
code
```

```
#ifndef GeeksforGeeks
#error GeeksforGeeks not found !
#endif
```

chevron_right

filter_none

Output:

error: #error GeeksforGeeks not found !

References:

- [ppd_better_practice_cs.auckland.ac.nz](http://ppd.better-practice.cs.auckland.ac.nz)
- <http://www.cplusplus.com/doc/tutorial/preprocessor/>

This article is contributed by **Amartya Ranjan Saikia** and **Harsh Agarwal**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes

Add your personal notes here! (max 5000 chars)

Recommended Posts:

- How a Preprocessor works in C?
- C | Macro & Preprocessor | Question 9
- C | Macro & Preprocessor | Question 8
- C | Macro & Preprocessor | Question 7
- C | Macro & Preprocessor | Question 6
- C | Macro & Preprocessor | Question 5
- C | Macro & Preprocessor | Question 14
- C | Macro & Preprocessor | Question 4
- C | Macro & Preprocessor | Question 1
- C | Macro & Preprocessor | Question 2
- C | Macro & Preprocessor | Question 10
- C | Macro & Preprocessor | Question 11
- C | Macro & Preprocessor | Question 15
- C | Macro & Preprocessor | Question 14
- C | Macro & Preprocessor | Question 13

Improved By : [RishabhPrabhu](#)

Article Tags :

C
C++
C Basics
C-Macro & Preprocessor
cpp-macros
Macro & Preprocessor

Practice Tags :

C
CPP

 5

To-do Done
2.5

Based on 6 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) `std::string::resize()` in C++

Next

[last_page](#) Quick way to check if all the characters of a string are same

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Weekend Geeks Classes Weekdays

JAVA APP DEVELOPMENT COMPETITIVE PROGRAMMING LIVE Java Backend Development

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
ctype.h <cctype> library in C/C++ with Examples
C programs display month by month calendar for a given year
Program to find Max sum of C with Examples
Implicit Type Conversion in C with Examples

Most visited in C++
Vector of Vectors in C++ STL with Examples
C++ Tutorial
Which C++ libraries are useful for competitive programming?
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

isgraph() C library function

The C library function **isgraph()** checks whether a character is a graphic character or not.

Characters that have graphical representation are known as **graphic characters**. For example: `:` `;` `?` `@` etc.

Syntax -

```
#include <ctype.h>
int isgraph(int ch);
```

Return Value - function returns **nonzero** if ch is any printable character other than a space, else it returns 0.

For ASCII environments, printable characters are in the range of **0X21** through **0X7E**.

Code -

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// code to check graphical character
#include <stdio.h>
#include <ctype.h>

int main()
{
    char var1 = 'g';
    char var2 = ' ';
    char var3 = '1';

    if (isgraph(var1))
        printf("var1 = [%c] can be printed\n", var1);
    else
        printf("var1 = [%c] can't be printed\n", var1);

    if (isgraph(var2))
        printf("var2 = [%c] can be printed\n", var2);
    else
        printf("var2 = [%c] can't be printed\n", var2);

    if (isgraph(var3))
        printf("var3 = [%c] can be printed\n", var3);
    else
        printf("var3 = [%c] can't be printed\n", var3);

    return (0);
}
```

chevron_right

filter_none

Output -

```
var1 = [g] can be printed
var2 = [ ] can't be printed
var3 = [1] can be printed
```

Code -

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// code to print all Graphical Characters
#include <stdio.h>
#include <ctype.h>

int main()
{
    int i;
    printf("In C programming All graphic "
           "characters are: \n");

    for (i = 0; i <= 127; ++i)
        if (isgraph(i) != 0)
            printf("%c ", i);

    return 0;
}
```

chevron_right

filter_none

Output -

```
In C programming All graphic characters are:
! " # $ % & ' ( ) * +, - . / 0 1 2 3 4 5 6 7 8 9
: ; ? @ A B C D E F G H I J K L M N O P Q R S T U
V W X Y Z [ \ ] ^ _ ` a b c d e f g h i j k l m n
o p q r s t u v w x y z { | } ~
```

This article is contributed by **Shivani Ghugtyal**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- wcstof function in C library
- difftime() C library function
- How to print range of basic data types without any library function and constant in C?
- snprintf() in C library
- SDL library in C/C++ with Examples
- wprintf() and wscanf in C Library
- C Library math.h functions
- How to add "graphics.h" C/C++ library to gcc compiler in Linux
- ctype.h(<cctype>) library in C/C++ with Examples
- Difference between Header file and Library
- Unordered Sets in C++ Standard Template Library
- Inbuilt library functions for user Input | scanf, fscanf, sscanf, scanf_s, fscanf_s, sscanf_s
- How to call function within function in C or C++
- arc function in C
- putchar() function in C

Article Tags :

C

C-Library

Practice Tags :

C



3

To-do Done
1.8

Based on 5 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) How to write your own header file in C?

Next

[last_page](#) Internal Linkage and External Linkage in C

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Weekend Geeks Classes Weekdays

JAVA APP DEVELOPMENT COMPETITIVE PROGRAMMING JAVA Backend Development

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
ctype.h(<cctype>) library in C/C++ with Examples
C program to display month by month calendar for a given year
Predefined Macros in C with Examples
Implicit Type Conversion in C with Examples

More related articles in C
Format specifiers in different Programming Languages
How to call function within function in C or C++
C program to read line contents of a file followed by even line content
Introduction to the C99 Programming Language : Part II
C program to find square root of a given number

How to write your own header file in C?

As we all know that files with .h extension are called **header files** in C. These header files generally contain function declarations which we can be used in our main C program, like for e.g. there is need to include stdio.h in our C program to use function printf() in the program. So the question arises, is it possible to create your own header file?

The answer to the above is **yes**. header files are simply files in which you can declare your own functions that you can use in your main program or these can be used while writing large C programs.

NOTE:Header files generally contain definitions of data types, function prototypes and C preprocessor commands.

Below is the short example of creating your own header file and using it accordingly.

1. **Creating myhead.h :** Write the below code and then save the file as **myhead.h** or you can give any name but the extension should be .h indicating its a header file.

```
filter_none
edit
close
```

```

play_arrow
link
brightness_4
code

// It is not recommended to put function definitions
// in a header file. Ideally there should be only
// function declarations. Purpose of this code is
// to only demonstrate working of header files.
void add(int a, int b)
{
    printf("Added value=%d\n", a + b);
}
void multiply(int a, int b)
{
    printf("Multiplied value=%d\n", a * b);
}
chevron_right
filter_none

```

2. **Including the .h file in other program :** Now as we need to include stdio.h as #include in order to use printf() function. We will also need to include the above header file myhead.h as #include "myhead.h". The " " here are used to instructs the **preprocessor** to look into the present folder and into the standard folder of all header files if not found in present folder. So, if you wish to use angular brackets instead of " " to include your header file you can save it in the standard folder of header files otherwise. If you are using " " you need to ensure that the header file you created is saved in the same folder in which you will save the C file using this header file.

3. **Using the created header file :**

```

filter_none
edit
close

play_arrow
link
brightness_4
code

// C program to use the above created header file
#include <stdio.h>
#include "myhead.h"
int main()
{
    add(4, 6);

    /*This calls add function written in myhead.h
     and therefore no compilation error.*/
    multiply(5, 5);

    // Same for the multiply function in myhead.h
    printf("BYE!See you Soon");
    return 0;
}
chevron_right
filter_none

```

Output:

```

Added value:10
Multiplied value:25
BYE!See you Soon

```

NOTE : The above code compiles successfully and prints the above output only if you have created the header file and saved it in the same folder the above c file is saved.

Important Points:

The creation of header files are needed generally while writing large C programs so that the modules can share the function definitions, prototypes etc.

- Function and type declarations, global variables, structure declarations and in some cases, inline functions; definitions which need to be centralized in one file.
- In a header file, do not use redundant or other header files; only minimal set of statements.
- Don't put function definitions in a header. Put these things in a separate .c file.
- Include Declarations for functions and variables whose definitions will be visible to the linker. Also, definitions of data structures and enumerations that are shared among multiple source files.
- In short, Put only what is necessary and keep the header file concise.

This article is merely to give you idea about the creation of header files and using the same but this is not what actually happens when you write a large C program. The creation of header files are needed generally while writing large C programs so that the modules can share the function definitions, prototypes etc.

This article is contributed by **Dimpay Varshni**. If you like GeeksforGeeks and would like to contribute, you can also write an article using [contribute.geeksforgeeks.org](#) or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes 

Recommended Posts:

- [Comment in header file name?](#)
- [<complex.h> header file in C with Examples](#)
- [Print "Hello World" in C/C++ without using any header file](#)
- [Difference between Header file and Library](#)
- [time.h header file in C with Examples](#)
- [Read/Write structure to a file in C](#)
- [fopen\(\) for an existing file in write mode](#)
- [lseek\(\) in C/C++ to read the alternate nth byte and write it in another file](#)
- [Write a C program that displays contents of a given file like 'more' utility in Linux](#)
- [C program to copy contents of one file to another file](#)
- [accumulate\(\) and partial_sum\(\) in C++ STL : numeric header](#)

- numeric header in C++ STL | Set 2 (adjacent_difference(), inner_product() and iota())
- What's difference between header files "stdio.h" and "stdlib.h" ?
- Namespace in C++ | Set 3 (Accessing, creating header, nesting and aliasing)
- When should we write our own copy constructor?

Article Tags :

C
File Handling
Practice Tags :
C
File Handling

thumb_up
18

To-do Done
1.9

Based on 11 vote(s)

Basic Easy Medium Hard Expert

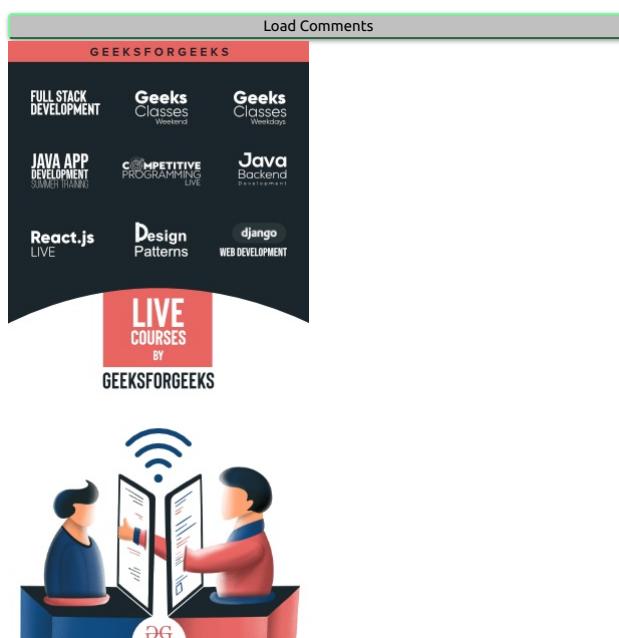
Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous [first_page kbit in C language](#)

Next [last_page isgraph\(\) C library function](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Print all possible combinations of the string by replacing '\$' with any other digit from the string
`ctype.h(<cctype>)` library in C/C++ with Examples
 C program to display month by month calendar for a given year
 Predefined Macros in C with Examples
 Implicit Type Conversion in C with Examples

More related articles in C
 How to print main function in C or C++
 C program to print odd line contents of a file followed by even line content
 Introduction to the C99 Programming Language : Part II
 C program to find square root of a given number
 Format specifiers in different Programming Languages

difftime() C library function

The C library function **difftime()** returns the difference, in **seconds** between **starting time** and **ending time**.(**ending time-starting time**)

```
// It is present in time.h header file
#include

Syntax :
double difftime(time_t time2, time_t time1);

Parameters:
time1 : Lower bound of the time interval
whose length is calculated.
time2 : Higher bound of the time interval
whose length is calculated.

Return value :
Returns the difference between time1 and
time2 (as measured in seconds).

filter_none
edit
close

play_arrow
link
brightness_4
code

// C program to demonstrate working of
// difftime()
#include <time.h>
#include <stdio.h>
#include <unistd.h>
int main()
{
```

```

int sec;
time_t time1, time2;

// Current time
time(&time1);
for (sec = 1; sec <= 6; sec++)
    sleep(1);

// time after sleep in loop.
time(&time2);
printf("Difference is %.2f seconds",
       difftime(time2, time1));

return 0;
}

```

[chevron_right](#)

[filter_none](#)

Output:

Difference is 6.00 seconds

This article is contributed by **Shivani Ghugtyal**. If you like GeeksforGeeks and would like to contribute, you can also write an article using [contribute.geeksforgeeks.org](#) or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes [arrow_drop_up](#)

Add your personal notes
here! (max 5000 chars)

[Save](#)

Recommended Posts:

- [difftime\(\) function in C++](#)
- [wcstof function in C library](#)
- [isgraph\(\) C library function](#)
- [How to print range of basic data types without any library function and constant in C?](#)
- [<strings> library in C++ STL](#)
- [snprintf\(\) in C library](#)
- [<iterator> library in C++ STL](#)
- [<algorithms> library in C++ STL](#)
- [<numeric> library in C++ STL](#)
- [<regex> library in C++ STL](#)
- [SDL library in C/C++ with examples](#)
- [boost::split in C++ library](#)
- [Pattern Searching using C++ library](#)
- [wprintf\(\) and wscanf in C Library](#)
- [Any datatype in C++ boost library](#)

Article Tags :

C
C++
C-Library
date-time-program

Practice Tags :

C
CPP

[thumb_up](#)
3

To-do Done
2.5

Based on 4 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) [strpbrk\(\) in C](#)

Next

[last_page](#) [Callbacks in C](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

[Load Comments](#)



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
ctype.h<<ctype>> library in C/C++ with Examples

C program to display month by month calendar for a given year

Predefined Macro in C with Examples

Implicit Type Conversion in C with Examples

Most visited in C++

Vector of Vectors in C++ STL with Examples

C++ STL libraries are useful for competitive programming?

Array of Vectors in C++ STL

Map of Vectors in C++ STL with Examples

tmpnam() function in C

The tmpnam() function is a special function which is declared inside "stdio.h" header file. It generates a different temporary file name each time it is called up to at least TMP_MAX names. Here TMP_MAX represents maximum number of different file names that can be produced by tmpnam() function. If it is called more than TMP_MAX times, the behavior is implementation dependent. Here, L_tmpnam define the size needed for an array of char to hold the result of tmpnam.

Syntax :

```
char *tmpnam(char *str)
s : The character array to copy the file name.
It generates and returns a valid temporary
filename which does not exist.
If str is null then it simply returns the tmp file name.
```

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// C program to generate random temporary file names.

#include <stdio.h>
int main(void)
{
    // L_tmpnam declared in the stdio.h file.
    // L_tmpnam define length of the generated file name.
    char generate[L_tmpnam + 1]; // Add +1 for the null character.
    tmpnam(generate);
    puts(generate);
    return 0;
}
```

chevron_right

filter_none

Output:

```
The file names are dependent on running machine, which can be anything.
Example: /tmp/fileRTOA0m
        \s260.
        \s3ok.
        \s5gg. etc
```

This article is contributed by **Bishal Kumar Dubey**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- How to call function within function in C or C++
- arc function in C
- Inline function in C
- putchar() function in C
- atexit() function in C/C++
- strcspn() function in C/C++

- `strchr()` function in C/C++
- `gmtime()` Function in C/C++
- `toupper()` function in C/C++
- `wcstoll()` function in C/C++
- `iswalnum()` function in C++ STL
- `ldecp()` function in C/C++
- `iswpunct()` function in C/C++
- `iswblank()` function in C/C++
- `strtoumax()` function in C++

Article Tags :

C
C-Library
Practice Tags :

C

thumb_up

2

To-do Done
2.6

Based on 3 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

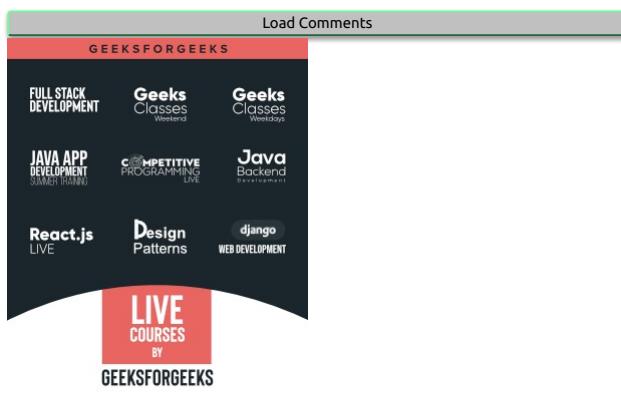
Previous

[first_page](#) Different ways to declare variable as constant in C and C++

Next

[last_page](#) `_Noreturn` function specifier in C

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
`cctype.h(<cctype.h>)` library in C/C++ with Examples
C program to display month by month calendar for a given year
Predefined Macros in C with Examples
Implicit Type Conversion in C with Examples

More related articles in C
Format specifiers in different Programming Languages
How to call function within function in C or C++
C program to print odd line contents of a file followed by even line content
Introduction to the C99 Programming Language : Part II
C program to find square root of a given number

Generic keyword in C

A major drawback of Macro in C/C++ is that the arguments are strongly typed checked i.e. a macro can operate on different types of variables(like char, int ,double,...) without type checking.

```
filter_none
edit
close

play_arrow
link
brightness_4
code

// C program to illustrate macro function.
#include<stdio.h>
#define INC(P) ++P
int main()
{
    char *p = "Geeks";
    int x = 10;
    printf("%s ", INC(p));
    printf("%d", INC(x));
    return 0;
}
```

chevron_right

filter_none

Output:

Therefore we avoid to use Macro. But after the implementation of C11 standard in C programming, we can use Macro with the help of a new keyword i.e. "_Generic". We can define MACRO for the different types of data types. For example, the following macro INC(x) translates to INCi(x), INC(x) or INCf(x) depending on the type of x:

```
#define INC(x) _Generic((x), long double: INCf, \
                      default: INC, \
                      float: INCf)(x)
```

Example:-

```
filter_none
edit
close

play_arrow
link
brightness_4
code

// C program to illustrate macro function.
#include <stdio.h>
int main(void)
{
    // _Generic keyword acts as a switch that chooses
    // operation based on data type of argument.
    printf("%d\n", _Generic( 1.0L, float:1, double:2,
                           long double:3, default:0));
    printf("%d\n", _Generic( 1L, float:1, double:2,
                           long double:3, default:0));
    printf("%d\n", _Generic( 1.0L, float:1, double:2,
                           long double:3));
    return 0;
}
```

[chevron_right](#)

[filter_none](#)

Output:

Note: If you are running C11 compiler then the below mentioned output will be come.

```
3
0
3
```

[filter_none](#)

[edit](#)

[close](#)

[play_arrow](#)

[link](#)

[brightness_4](#)

[code](#)

// C program to illustrate macro function.

```
#include <stdio.h>
#define geeks(T) _Generic( (T), char: 1, int: 2, long: 3, default: 0)
int main(void)
{
    // char returns ASCII value which is int type.
    printf("%d\n", geeks('A'));

    // Here A is a string.
    printf("%d",geeks("A"));

    return 0;
}
```

[chevron_right](#)

[filter_none](#)

Output:

```
2
0
```

This article is contributed by **Bishal Kumar Dubey**. If you like GeeksforGeeks and would like to contribute, you can also write an article using [contribute.geeksforgeeks.org](#) or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes [arrow_drop_up](#)

Add your personal notes
here! (max 5000 chars)

[Save](#)

Recommended Posts:

Article Tags :

[C](#)

Practice Tags :

[C](#)

[thumb_up](#)

3

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#)

[exit\(\) vs _Exit\(\) in C and C++](#)

Next

[last_page](#)

Why variable name does not start with numbers in C ?

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

The screenshot shows the GeeksforGeeks homepage with a navigation bar at the top. Below the navigation, there are several course categories: FULL STACK DEVELOPMENT, JAVA APP DEVELOPMENT (SUMMER TRAINING), React.js (LIVE), Geeks Classes (Webinars), COMPETITIVE PROGRAMMING (LIVE), Design Patterns, Java Backend, and django WEB DEVELOPMENT. A prominent red banner in the center says "LIVE COURSES BY GEEKSFORGEEKS". Below the banner, there's an illustration of two people looking at a screen with a Wi-Fi signal icon above it. At the bottom left, there's a section titled "Most popular in C" with links to articles like "Print all possible combinations of the string by replacing '\$' with any other digit from the string", "Copy & Paste a library in C/C++ with Examples", "Implicit Type Conversion in C with Examples", "Format Specifiers in different Programming Languages", and "C program to find square root of a given number". On the right side, there's a section titled "More related articles in C" with links to "Predefined Macros in C with Examples", "How to call function within function in C or C++", "C program to print odd line contents of a file followed by even line content", "How to create GUI in C programming using GTK Toolkit", and "Introduction to the C99 Programming Language : Part II".

C Library math.h functions

The **math.h** header defines various mathematical functions and one macro. All the functions available in this library take **double** as an argument and return **double** as the result. Let us discuss some important functions one by one.

1. **double ceil(double x):** The C library function double ceil(double x) returns the smallest integer value greater than or equal to x.

Syntax : `double ceil(double x)`

```
filter_none
edit
close

play_arrow

link
brightness_4
code

// C code to illustrate
// the use of ceil function.
#include <stdio.h>
#include <math.h>

int main ()
{
float val1, val2, val3, val4;

val1 = 1.6;
val2 = 1.2;
val3 = -2.8;
val4 = -2.3;

printf ("value1 = %.lf\n", ceil(val1));
printf ("value2 = %.lf\n", ceil(val2));
printf ("value3 = %.lf\n", ceil(val3));
printf ("value4 = %.lf\n", ceil(val4));

return(0);
}
```

Output:

```
value1 = 2.0
value2 = 2.0
value3 = -2.0
value4 = -2.0
```

2. **double floor(double x):** The C library function double floor(double x) returns the largest integer value less than or equal to x.

```
syntax : double floor(double x)
```

filter_none

edit

close

play_arrow

link

brightness_4

code

// C code to illustrate

// the use of floor function

```
#include <stdio.h>
```

```
#include <math.h>
```

```
int main ()
```

```
{
```

```
    float val1, val2, val3, val4;
```

```
    val1 = 1.6;
```

```
    val2 = 1.2;
```

```
    val3 = -2.8;
```

```
    val4 = -2.3;
```

```
    printf("Value1 = %.1lf\n", floor(val1));
```

```
    printf("Value2 = %.1lf\n", floor(val2));
```

```
    printf("Value3 = %.1lf\n", floor(val3));
```

```
    printf("Value4 = %.1lf\n", floor(val4));
```

```
    return(0);
```

}

chevron_right

filter_none

Output:

```
Value1 = 1.0
```

```
Value2 = 1.0
```

```
Value3 = -3.0
```

```
Value4 = -3.0
```

3. **double fabs(double x):** The C library function double fabs(double x) returns the absolute value of x.

```
syntax : double fabs(double x)
```

filter_none

edit

close

play_arrow

link

brightness_4

code

// C code to illustrate

// the use of fabs function

```
#include <stdio.h>
```

```
#include <math.h>
```

```
int main ()
```

```
{
```

```
    int a, b;
```

```
    a = 1234;
```

```
    b = -344;
```

```
    printf("The absolute value of %d is %lf\n", a, fabs(a));
```

```
    printf("The absolute value of %d is %lf\n", b, fabs(b));
```

```
    return(0);
```

}

chevron_right

filter_none

Output:

```
The absolute value of 1234 is 1234.000000
```

```
The absolute value of -344 is 344.000000
```

4. **double log(double x):** The C library function double log(double x) returns the natural logarithm (base-e logarithm) of x.

```
syntax : double log(double x)
```

filter_none

edit

close

play_arrow

link

brightness_4

code

// C code to illustrate

// the use of log function

```
#include <stdio.h>
```

```
#include <math.h>
```

```
int main ()
```

```
{
```

```
    double x, ret;
```

```
    x = 2.7;
```

```
/* finding log(2.7) */
ret = log(x);
printf("log(%lf) = %lf", x, ret);

return(0);
}
```

chevron_right

filter_none

Output:

```
log(2.700000) = 0.993252
```

5. **double log10(double x)**: The C library function double log10(double x) returns the common logarithm (base-10 logarithm) of x.

Syntax : **double log10(double x)**

filter_none

edit

close

play_arrow

link

brightness_4

code

```
// C code to illustrate
// the use of log10 function
#include <stdio.h>
#include <math.h>
```

```
int main ()
```

```
{
```

```
    double x, ret;
    x = 10000;
```

```
    /* finding value of log10(10000) */
    ret = log10(x);
    printf("log10(%lf) = %lf\n", x, ret);
```

```

    return(0);
}
```

chevron_right

filter_none

Output:

```
log10(10000.000000) = 4.000000
```

6. **double fmod(double x, double y)** : The C library function double fmod(double x, double y) returns the remainder of x divided by y.

Syntax : **double fmod(double x, double y)**

filter_none

edit

close

play_arrow

link

brightness_4

code

```
// C code to illustrate
// the use of fmod function
#include <stdio.h>
#include <math.h>
```

```
int main ()
```

```
{
```

```
    float a, b;
    int c;
    a = 8.2;
    b = 5.7;
    c = 3;
    printf("Remainder of %f / %d is %lf\n", a, c, fmod(a, c));
    printf("Remainder of %f / %f is %lf\n", a, b, fmod(a, b));
```

```

    return(0);
}
```

chevron_right

filter_none

Output:

```
Remainder of 8.200000 / 3 is 2.200000
Remainder of 8.200000 / 5.700000 is 2.500000
```

7. **double sqrt(double x)**: The C library function double sqrt(double x) returns the square root of x.

Syntax : **double sqrt(double x)**

filter_none

edit

close

play_arrow

link

brightness_4

code

```
// C code to illustrate
```

```
// the use of sqrt function
#include <stdio.h>
#include <math.h>

int main ()
{
    printf("Square root of %lf is %lf\n", 225.0, sqrt(225.0) );
    printf("Square root of %lf is %lf\n", 300.0, sqrt(300.0) );

    return(0);
}
```

[chevron_right](#)

[filter_none](#)

Output:

```
Square root of 225.000000 is 15.000000
Square root of 300.000000 is 17.320508
```

8. **double pow(double x, double y):** The C library function double pow(double x, double y) returns x raised to the power of y i.e. x^y .

Syntax : `double pow(double x, double y)`

[filter_none](#)
[edit](#)
[close](#)

[play_arrow](#)

[link](#)
[brightness_4](#)
[code](#)

```
// C code to illustrate
// the use of pow function
#include <stdio.h>
#include <math.h>
```

```
int main ()
{
    printf("Value 8.0 ^ 3 = %lf\n", pow(8.0, 3));

    printf("Value 3.05 ^ 1.98 = %lf", pow(3.05, 1.98));

    return(0);
}
```

[chevron_right](#)

[filter_none](#)

Output:

```
Value 8.0 ^ 3 = 512.000000
Value 3.05 ^ 1.98 = 9.097324
```

9. **double modf(double x, double *integer):** The C library function double modf(double x, double *integer) returns the fraction component (part after the decimal), and sets integer to the integer component.

Syntax : `double modf(double x, double *integer)`

[filter_none](#)
[edit](#)
[close](#)

[play_arrow](#)

[link](#)
[brightness_4](#)
[code](#)

```
// C code to illustrate
// the use of modf function
#include<stdio.h>
#include<math.h>
```

```
int main ()
{
    double x, fractpart, intpart;

    x = 8.123456;
    fractpart = modf(x, &intpart);

    printf("Integral part = %lf\n", intpart);
    printf("Fraction Part = %lf \n", fractpart);

    return(0);
}
```

[chevron_right](#)

[filter_none](#)

Output:

```
Integral part = 8.000000
Fraction Part = 0.123456
```

10. **double exp(double x):** The C library function double exp(double x) returns the value of e raised to the x th power.

Syntax : `double exp(double x)`

Following code represents

[filter_none](#)
[edit](#)
[close](#)

[play_arrow](#)

[link](#)

```

brightness_4
code

// C code to illustrate
// the use of exp function
#include <stdio.h>
#include <math.h>

int main ()
{
    double x = 0;

    printf("The exponential value of %lf is %lf\n", x, exp(x));
    printf("The exponential value of %lf is %lf\n", x+1, exp(x+1));
    printf("The exponential value of %lf is %lf\n", x+2, exp(x+2));

    return(0);
}

```

chevron_right

filter_none

Output:

```

The exponential value of 0.000000 is 1.000000
The exponential value of 1.000000 is 2.718282
The exponential value of 2.000000 is 7.389056

```

11. **double cos(double x)** : The C library function double cos(double x) returns the cosine of a radian angle x.

syntax : double cos(double x)

Note : Same syntax can be used for other trigonometric functions like sin, tan etc.

filter_none
edit
close

play_arrow

link
brightness_4
code

```

// C code to illustrate
// the use of cos function
#include <stdio.h>
#include <math.h>

```

```
#define PI 3.14159265
```

```

int main ()
{
    double x, ret, val;

    x = 60.0;
    val = PI / 180.0;
    ret = cos( x*val );
    printf("The cosine of %lf is %lf degrees\n", x, ret);

    x = 90.0;
    val = PI / 180.0;
    ret = cos( x*val );
    printf("The cosine of %lf is %lf degrees\n", x, ret);
}
```

chevron_right

filter_none

Output:

```

The cosine of 60.000000 is 0.500000 degrees
The cosine of 90.000000 is 0.000000 degrees

```

12. **double acos(double x)** : The C library function double acos(double x) returns the arc cosine of x in radians.

syntax : double acos(double x)

Note : Same syntax can be used for other arc trigonometric functions like asin, atan etc.

filter_none
edit
close

play_arrow

link
brightness_4
code

```

// C code to illustrate
// the use of acos function
#include <stdio.h>
#include <math.h>

```

```
#define PI 3.14159265
```

```

int main ()
{
    double x, ret, val;

    x = 0.9;
}
```

```
val = 180.0 / PI;

ret = acos(x) * val;
printf("The arc cosine of %lf is %lf degrees", x, ret);

return(0);
}
```

[chevron_right](#)

[filter_none](#)

Output:

The arc cosine of 0.900000 is 25.855040 degrees

13. **double tanh(double x):** The C library function double tanh(double x) returns the hyperbolic tangent of x.

[syntax : double tanh\(double x\)](#)

Note : Same syntax can be used for other hyperbolic trigonometric functions like sinh, cosh etc.

[filter_none](#)
[edit](#)
[close](#)

[play_arrow](#)

[link](#)
[brightness_4](#)
[code](#)

```
// C code to illustrate
// the use of tanh function
#include <stdio.h>
#include <math.h>

int main ()
{
    double x, ret;
    x = 0.5;

    ret = tanh(x);
    printf("The hyperbolic tangent of %lf is %lf degrees", x, ret);

    return(0);
}
```

[chevron_right](#)

[filter_none](#)

Output:

The hyperbolic tangent of 0.500000 is 0.462117 degrees

This article is contributed by **Avinash Kumar Singh**. If you like GeeksforGeeks and would like to contribute, you can also write an article using [contribute.geeksforgeeks.org](#) or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes [arrow_drop_up](#)

Add your personal notes here! (max 5000 chars)

[Save](#)

Recommended Posts:

- [Wide char and library functions in C++](#)
- [Inbuilt library functions for user input | scanf, fscanf, sscanf, scanf_s, fscanf_s, sscanf_s](#)
- [SDL library in C/C++ with examples](#)
- [<iterator> library in C++ STL](#)
- [snprintf\(\) in C library](#)
- [<numeric> library in C++ STL](#)
- [difftime\(\) C library function](#)
- [isgraph\(\) C library function](#)
- [wcstof function in C library](#)
- [wprintf\(\) and wscanf in C Library](#)
- [boost::split in C++ library](#)
- [Difference between Header file and Library](#)
- [ctype.h\(<cctype>\) library in C/C++ with Examples](#)
- [How to add "graphics.h" C/C++ library to gcc compiler in Linux](#)
- [Unordered Sets in C++ Standard Template Library](#)

Article Tags :

[C](#)
[CPP-Library](#)
[cpp-math](#)

Practice Tags :

[C](#)

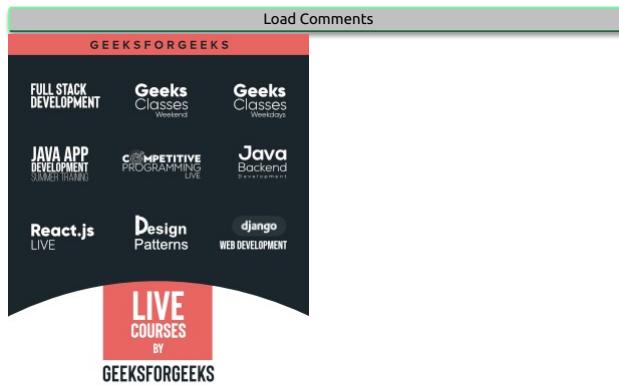
[thumb_up](#)
4

To-do Done
3.2

Based on 5 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
Predefined Macros in C with Examples
Format Specifiers in different Programming Languages
C program to print odd line contents of a File followed by even line content
C program to find square root of a given number

More related articles in C
Introduction to the C99 Programming Language : Part II
Problem in comparing Floating point numbers and how to compare them correctly?
Features of C Programming Language
Pointer Expressions in C with Examples
Introduction to the C99 Programming Language : Part III

typedef versus #define in C

typedef: The typedef is used to give data type a new name. For example,

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// C program to demonstrate typedef
#include <stdio.h>

// After this line BYTE can be used
// in place of unsigned char
typedef unsigned char BYTE;
```

```
int main()
{
    BYTE b1, b2;
    b1 = 'c';
    printf("%c ", b1);
    return 0;
}
```

chevron_right

filter_none

Output:

c

#define in C is a directive which is used to #define alias.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// C program to demonstrate #define
#include <stdio.h>

// After this line HYD is replaced by
// "Hyderabad"
#define HYD "Hyderabad"
```

```
int main()
{
    printf("%s ", HYD);
```

```
    return 0;
```

```
}
```

```
chevron_right
```

```
filter_none
```

Output:

```
Hyderabad
```

Difference between `typedef` and `#define`:

1. `typedef` is limited to giving symbolic names to types only, whereas `#define` can be used to define an alias for values as well, e.g., you can define 1 as ONE, 3.14 as PI, etc.
2. `typedef` interpretation is performed by the compiler where `#define` statements are performed by preprocessor.
3. `#define` should not be terminated with a semicolon, but `typedef` should be terminated with semicolon.
4. `#define` will just copy-paste the definition values at the point of use, while `typedef` is the actual definition of a new type.
5. `typedef` follows the scope rule which means if a new type is defined in a scope (inside a function), then the new type name will only be visible till the scope is there. In case of `#define`, when preprocessor encounters `#define`, it replaces all the occurrences, after that (No scope rule is followed).

```
filter_none
```

```
edit
```

```
close
```

```
play_arrow
```

```
link
```

```
brightness_4
```

```
code
```

```
// C program to demonstrate importance  
// of typedef over #define for data types  
#include <stdio.h>  
typedef char* ptr;  
#define PTR char*  
int main()  
{  
    ptr a, b, c;  
    PTR x, y, z;  
    printf("sizeof a:%u\n", sizeof(a) );  
    printf("sizeof b:%u\n", sizeof(b) );  
    printf("sizeof c:%u\n", sizeof(c) );  
    printf("sizeof x:%u\n", sizeof(x) );  
    printf("sizeof y:%u\n", sizeof(y) );  
    printf("sizeof z:%u\n", sizeof(z) );  
    return 0;  
}
```

```
chevron_right
```

```
filter_none
```

Output:

```
sizeof a:8  
sizeof b:8  
sizeof c:8  
sizeof x:8  
sizeof y:1  
sizeof z:1
```

From the output of the above program size of "a" which is a pointer is 8 (on a machine where pointers are stored using 8 bytes). In the above program, when the compiler comes to

```
typedef char* ptr;  
ptr a, b, c;
```

the statement effectively becomes

```
char *a, *b, *c;
```

This declares a, b, c as `char*`.

In contrast, `#define` works like this:

```
#define PTR char*  
PTR x, y, z;
```

the statement effectively becomes

```
char *x, *y, *z;
```

This makes x, y and z different, as, x is pointer-to-a `char`, whereas, y and z are `char` variables. When we declare macros with pointers while defining if we declare more than one identifier then the actual definition is given to the first identifier and for the rest non-pointer definition is given. In the above case x will be declared as `char*`, so its size is the size of a pointer, whereas, y and z will be declared as `char` so, their size will be 1 byte.

This article is contributed by **HARISH KUMAR**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes 

Add your personal notes
here! (max 5000 chars)

Recommended Posts:

- [Is there any equivalent to `typedef` of C/C++ in Java ?](#)
- [For Versus While](#)
- [Scope Resolution Operator Versus this pointer in C++?](#)
- [Difference between `#define` and `const` in C?](#)
- ["static const" vs "#define" vs "enum"](#)
- [Format specifiers in different Programming Languages](#)
- [C program to find square root of a given number](#)
- [C program to print odd line contents of a File followed by even line content](#)
- [Getting System and Process Information Using C Programming and Shell in Linux](#)
- [Program to calculate Electricity Bill](#)
- [Program to print half Diamond star pattern](#)

- C/C++ program for calling main() in main()
- Print all possible combinations of the string by replacing '\$' with any other digit from the string
- How to use make utility to build C projects?

Improved By : code_master5, yash_09

Article Tags :

C
C-Macro & Preprocessor
Practice Tags :

thumb_up
39

To-do Done
2.7

Based on 29 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

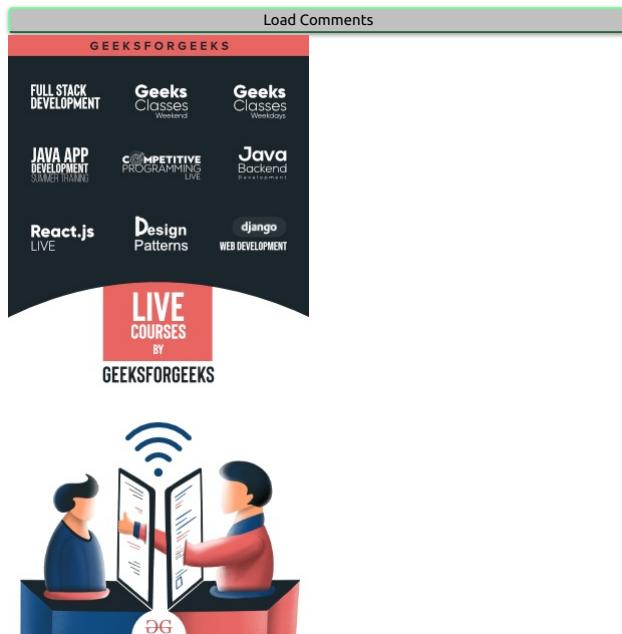
Previous

[first_page](#) Program to print Happy Birthday

Next

[last_page](#) Print colored message with different fonts and sizes in C

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
Format specifiers in different Programming Languages
C program to find square root of a given number
Predefined Macros in C with Examples
C program to print odd line contents of a File followed by even line content

More related articles in C
Introduction to the C99 Programming Language : Part II
Format of C Programming Language
Problem in comparing Floating point numbers and how to compare them correctly?
<cfloat> float.h in C/C++ with Examples
Getting System and Process Information Using C Programming and Shell in Linux

strftime() function in C/C++

strftime() is a function in C which is used to format date and time. It comes under the header file time.h, which also contains a structure named struct tm which is used to hold the time and date. The syntax of strftime() is as shown below :

```
size_t strftime(char *s, size_t max, const char *format,
               const struct tm *tm);
```

strftime() function formats the broken-down time tm according to the formatting rules specified in format and store it in character array s.

Some format specifiers for strftime() are shown as follows :

%x = Preferred date representation
%l = Hour as a decimal number (12-hour clock).
%M = Minutes in decimal ranging from 00 to 59.
%p = Either "AM" or "PM" according to the given time value, etc.
%a = Abbreviated weekday name
%A = Full weekday name
%b = Abbreviated month name
%B = Full month name March
%c = Date and time representation
%d = Day of the month (01-31)
%H = Hour in 24h format (00-23)
%l = Hour in 12h format (01-12)
%j = Day of the year (001-366)
%m = Month as a decimal number (01-12)
%M = Minute (00-59)

Structure struct tm is defined in time.h as follows :

```
struct tm
{
    int tm_sec;           // seconds
    int tm_min;          // minutes
```

```
int tm_hour;           // hours
int tm_mday;          // day of the month
int tm_mon;           // month
int tm_year;          // The number of years since 1900
int tm_wday;          // day of the week
int tm_yday;          // day in the year
int tm_isdst;         // daylight saving time
};
```

filter_none

edit

close

play_arrow

link

brightness_4

code

// C program to demonstrate the

// working of strftime()

```
#include <stdlib.h>
```

```
#include <stdio.h>
```

```
#include <time.h>
```

```
#define Size 50
```

```
int main ()
```

```
{
```

```
    time_t t;
```

```
    struct tm *tmp;
```

```
    char MY_TIME[Size];
```

```
    time( &t );
```

```
//localtime() uses the time pointed by t ,
```

```
// to fill a tm structure with the
```

```
// values that represent the
```

```
// corresponding local time.
```

```
tmp = localtime( &t );
```

```
// using strftime to display time
```

```
strftime(MY_TIME, sizeof(MY_TIME), "%x - %I:%M%p", tmp);
```

```
printf("Formatted date & time : %s\n", MY_TIME );
```

```
return(0);
```

chevron_right

filter_none

Formatted date & time : 03/20/17 - 02:55PM

Why and when do we use strftime() ?

When we are making a software/application which will output the current time and most important in many different formats on the user's demand. Then in that case we will use this function. Its specialty is that we can display date and time in many **different formats**.

Reference: <http://man7.org/linux/man-pages/man3/strftime.3.html>>Linux Man Page

This article is contributed by MAZHAR IMAM KHAN. If you like GeeksforGeeks and would like to contribute, you can also write an article using [contribute.geeksforgeeks.org](#) or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes *arrow_drop_up*

Add your personal notes
here! (max 5000 chars)

Save

Recommended Posts:

- [Function Overloading vs Function Overriding in C++](#)
- [How to call function within function in C or C++](#)
- [What happens when a virtual function is called inside a non-virtual function in C++](#)
- [Difference between Virtual function and Pure virtual function in C++](#)
- [fma\(\) function in C++](#)
- [log\(\) function in C++](#)
- [exp\(\) function C++](#)
- [div\(\) function in C++](#)
- [arc function in C](#)
- [valarray sin\(\) function in C++](#)
- [valarray exp\(\) function in C++](#)
- [valarray cos\(\) function in C++](#)
- [valarray tan\(\) function in C++](#)
- [valarray pow\(\) function in C++](#)
- [valarray endl\(\) function in C++](#)

Article Tags :

C

C++

CPP-Library

Practice Tags :

C

CPP

thumb_up

1

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

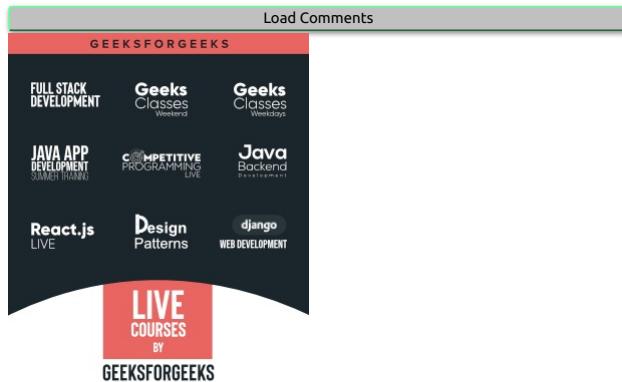
Previous

[first_page](#) Print "Hello World" in C/C++ without using any header file

Next

[last_page](#) Amazing stuff with system() in C / C++

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
Formatting operators in different Programming Languages
C program to find square root of a given number
Predefined Macros in C with Examples
C program print odd line contents of a File followed by even line content

Most visited in C++
Vector of Vectors in C++ STL with Examples
C++ Tutorial
Which C++ libraries are useful for competitive programming?
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

exec family of functions in C

The exec family of functions replaces the current running process with a new process. It can be used to run a C program by using another C program. It comes under the header file **unistd.h**. There are many members in the exec family which are shown below with examples.

- **execvp** : Using this command, the created child process does not have to run the same program as the parent process does. The **exec** type system calls allow a process to run any program files, which include a binary executable or a shell script. **Syntax:**

```
int execvp (const char *file, char *const argv[]);
```

file: points to the file name associated with the file being executed.

argv: is a null terminated array of character pointers.

Let us see a small example to show how to use execvp() function in C. We will have two .C files , **EXEC.c** and **execDemo.c** and we will replace the execDemo.c with EXEC.c by calling execvp() function in execDemo.c .

```
filter_none
edit
close

play_arrow

link
brightness_4
code

//EXEC.c

#include<stdio.h>
#include<unistd.h>

int main()
{
    int i;

    printf("I am EXEC.c called by execvp() ");
    printf("\n");

    return 0;
}

chevron_right
```

Now,create an executable file of EXEC.c using command

```
gcc EXEC.c -o EXEC
```

```
filter_none
edit
close
```

```
play_arrow
```

```
link
brightness_4
code

//execDemo.c

#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
int main()
{
    //A null terminated array of character
    //pointers
    char *args[]={".EXEC",NULL};
    execvp(args[0],args);

    /*All statements are ignored after execvp() call as this whole
process(execDemo.c) is replaced by another process (EXEC.c)
*/
    printf("Ending-----");
}

return 0;
}
chevron_right
```

filter_none

Now, create an executable file of execDemo.c using command

```
gcc execDemo.c -o execDemo
```

After running the executable file of execDemo.cby using command ./execDemo, we get the following output:

```
I AM EXEC.c called by execvp()
```

When the file execDemo.c is compiled, as soon as the statement execvp(args[0],args) is executed, this very program is replaced by the program EXEC.c. "Ending---" is not printed because because as soon as the execvp() function is called, this program is replaced by the program EXEC.c.

- **execv** : This is very similar to execvp() function in terms of syntax as well. The syntax of **execv()** is as shown below:**Syntax:**

```
int execv(const char *path, char *const argv[]);
```

path: should point to the path of the file being executed.

argv[]: is a null terminated array of character pointers.

Let us see a small example to show how to use execv() function in C. This example is similar to the example shown above for execvp() . We will have two .C files , **EXEC.c** and **execDemo.c** and we will replace the execDemo.c with EXEC.c by calling execv() function in execDemo.c .

filter_none
edit
close

play_arrow

link
brightness_4
code

//EXEC.c

```
#include<stdio.h>
#include<unistd.h>

int main()
{
    int i;

    printf("I am EXEC.c called by execv() ");
    printf("\n");
    return 0;
}
```

chevron_right

filter_none

Now,create an executable file of EXEC.c using command

```
gcc EXEC.c -o EXEC
```

filter_none
edit
close

play_arrow

link
brightness_4
code

//execDemo.c

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
int main()
{
    //A null terminated array of character
    //pointers
    char *args[]={".EXEC",NULL};
    execv(args[0],args);

    /*All statements are ignored after execvp() call as this whole
process(execDemo.c) is replaced by another process (EXEC.c)
*/
    printf("Ending-----");
}
```

```
    return 0;
}
chevron_right
filter_none
```

Now, create an executable file of execDemo.c using command

```
gcc execDemo.c -o execDemo
```

After running the executable file of execDemo.c by using command ./execDemo, we get the following output:

```
I AM EXEC.C CALLED BY EXECV()
```

- **execvp and execl :** These two also serve the same purpose but the syntax of them are a bit different which is as shown below:**Syntax:**

```
int execvp(const char *file, const char *arg, .../* (char *) NULL */);
int execl(const char *path, const char *arg, .../* (char *) NULL */);
```

file: file name associated with the file being executed

const char *arg and ellipses : describe a list of one or more pointers to null-terminated strings that represent the argument list available to the executed program.

The same C programs shown above can be executed with execvp() or execl() functions and they will perform the same task i.e. replacing the current process with a new process.

- **execvpe and execle :** These two also serve the same purpose but the syntax of them are a bit different from all the above members of exec family. The syntaxes of both of them are shown below :

Syntax:

```
int execvpe(const char *file, char *const argv[], char *const envp[]);
```

Syntax:
int execle(const char *path, const char *arg, ..., /* (char *) NULL,
char * const envp[] */);

The syntaxes above shown has one different argument from all the above exec members, i.e.

char * const envp[]: allow the caller to specify the environment of the executed program via the argument envp.

envp:This argument is an array of pointers to null-terminated strings and must be terminated by a null pointer. The other functions take the environment for the new process image from the external variable environ in the calling process.

Reference: exec(3) man page

This article is contributed by **Mazhar Imam Khan**. If you like GeeksforGeeks and would like to contribute, you can also write an article using [contribute.geeksforgeeks.org](#) or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes 

Add your personal notes
here! (max 5000 chars)

Recommended Posts:

- [Difference between fork\(\) and exec\(\)](#)
- [Functions in C/C++](#)
- [Thread functions in C/C++](#)
- [C | Functions | Question 8](#)
- [C | Functions | Question 7](#)
- [C | Functions | Question 9](#)
- [Macros vs Functions](#)
- [C | Functions | Question 11](#)
- [C | Functions | Question 10](#)
- [C | Functions | Question 4](#)
- [C | Functions | Question 3](#)
- [C | Functions | Question 11](#)
- [Nested functions in C](#)
- [Static functions in C](#)
- [C | Functions | Question 2](#)

Improved By : [AkashkumarGoryan](#), [ShubhamMaurya3](#)

Article Tags :

[C](#)

[C-Library](#)

Practice Tags :

[C](#)

 [16](#)

To-do Done
4.1

Based on 19 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) [Inbuilt library functions for user Input | scanf, fscanf, sscanf, scanf_s, fscanf_s, sscanf_s](#)

Next

[last_page](#) [strupl\(\) and strdup\(\) functions in C/C++](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
ctype.h<ctype.h> library in C/C++ with Examples
Predefined Macros in C with Examples

How to call function within function in C or C++
C program to print odd line contents of a file followed by even line content
More related articles in C

Introduction to the C99 Programming Language : Part II
C program to find square root of a given number
Fibonacci series in different Programming Languages
Problem in comparing Floating point numbers and how to compare them correctly
Features of C Programming Language

Arrays in C/C++

An array in C or C++ is a collection of items stored at contiguous memory locations and elements can be accessed randomly using indices of an array. They are used to store similar type of elements as in the data type must be the same for all elements. They can be used to store collection of primitive data types such as int, float, double, char, etc of any particular type. To add to it, an array in C or C++ can store derived data types such as the structures, pointers etc. Given below is the picturesque representation of an array.

40	55	63	17	22	68	89	97	89
0	1	2	3	4	5	6	7	8

<- Array Indices

Array Length = 9
First Index = 0
Last Index = 8

Why do we need arrays?

We can use normal variables (v1, v2, v3, ..) when we have a small number of objects, but if we want to store a large number of instances, it becomes difficult to manage them with normal variables. The idea of an array is to represent many instances in one variable.

Array declaration in C/C++:

There are various ways in which we can declare an array. It can be done by specifying its type and size, by initializing it or both.

1. Array declaration by specifying size

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// Array declaration by specifying size
int arr1[10];
```

```
// With recent C/C++ versions, we can also
// declare an array of user specified size
int n = 10;
int arr2[n];
chevron_right
```

filter_none

2. Array declaration by initializing elements

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// Array declaration by initializing elements
int arr[] = { 10, 20, 30, 40 }

// Compiler creates an array of size 4.
// above is same as "int arr[4] = {10, 20, 30, 40}"
chevron_right
```

filter_none

3. Array declaration by specifying size and initializing elements

```
filter_none
edit
close

play_arrow
link
brightness_4
code

// Array declaration by specifying size and initializing
// elements
int arr[6] = { 10, 20, 30, 40 }

// Compiler creates an array of size 6, initializes first
// 4 elements as specified by user and rest two elements as 0.
// above is same as "int arr[] = {10, 20, 30, 40, 0, 0}"
chevron_right
filter_none
```

Advantages of an Array in C/C++:

1. Random access of elements using array index.
2. Use of less line of code as it creates a single array of multiple elements.
3. Easy access to all the elements.
4. Traversal through the array becomes easy using a single loop.
5. Sorting becomes easy as it can be accomplished by writing less line of code.

Disadvantages of an Array in C/C++:

1. Allows a fixed number of elements to be entered which is decided at the time of declaration. Unlike a linked list, an array in C is not dynamic.
2. Insertion and deletion of elements can be costly since the elements are needed to be managed in accordance with the new memory allocation.

Facts about Array in C/C++:

▪ Accessing Array Elements:

Array elements are accessed by using an integer index. Array index starts with 0 and goes till size of array minus 1.

Example:

C

```
filter_none
edit
close

play_arrow
link
brightness_4
code

#include <stdio.h>

int main()
{
    int arr[5];
    arr[0] = 5;
    arr[2] = -10;
    arr[3 / 2] = 2; // this is same as arr[1] = 2
    arr[3] = arr[0];

    printf("%d %d %d %d", arr[0], arr[1], arr[2], arr[3]);

    return 0;
}
chevron_right
filter_none
```

C++

```
filter_none
edit
close

play_arrow
link
brightness_4
code

#include <iostream>
using namespace std;

int main()
{
    int arr[5];
    arr[0] = 5;
    arr[2] = -10;

    // this is same as arr[1] = 2
    arr[3 / 2] = 2;
    arr[3] = arr[0];

    cout << arr[0] << " " << arr[1]
        << " " << arr[2] << " " << arr[3];

    return 0;
}
```

```
}
```

chevron_right

```
filter_none
```

Output:

```
5 2 -10 5
```

▪ **No Index Out of bound Checking:**

There is no index out of bounds checking in C/C++, for example, the following program compiles fine but may produce unexpected output when run.

C

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// This C program compiles fine
// as index out of bound
// is not checked in C.
```

```
#include <stdio.h>
```

```
int main()
{
    int arr[2];

    printf("%d ", arr[3]);
    printf("%d ", arr[-2]);

    return 0;
}
```

chevron_right

```
filter_none
```

C++

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// This C++ program compiles fine
// as index out of bound
// is not checked in C.
```

```
#include <iostream>
using namespace std;
```

```
int main()
{
    int arr[2];

    cout << arr[3] << " ";
    cout << arr[-2] << " ";

    return 0;
}
```

chevron_right

```
filter_none
```

Output:

```
2008101287 4195777
```

- In C, it is not compiler error to initialize an array with more elements than the specified size. For example, the below program compiles fine and shows just Warning.

```
filter_none
edit
close
play_arrow
link
brightness_4
code
```

```
#include <stdio.h>
int main()
{

    // Array declaration by initializing it with more
    // elements than specified size.
    int arr[2] = { 10, 20, 30, 40, 50 };

    return 0;
}
```

chevron_right

```
filter_none
```

Warnings:

```
prog.c: In function 'main':
prog.c:7:25: warning: excess elements in array initializer
  int arr[2] = { 10, 20, 30, 40, 50 };
               ^
prog.c:7:25: note: (near initialization for 'arr')
prog.c:7:29: warning: excess elements in array initializer
  int arr[2] = { 10, 20, 30, 40, 50 };
               ^
prog.c:7:29: note: (near initialization for 'arr')
prog.c:7:33: warning: excess elements in array initializer
  int arr[2] = { 10, 20, 30, 40, 50 };
               ^
prog.c:7:33: note: (near initialization for 'arr')
```

Note: The program won't compile in C++. If we save the above program as a .cpp, the program generates compiler error "error: too many initializers for 'int [2]'".

▪ The elements are stored at contiguous memory locations

Example:

C

```
filter_none
edit
close

play_arrow
link
brightness_4
code

// C program to demonstrate that array elements are stored
// contiguous locations

#include <stdio.h>
int main()
{
    // an array of 10 integers. If arr[0] is stored at
    // address x, then arr[1] is stored at x + sizeof(int)
    // arr[2] is stored at x + sizeof(int) + sizeof(int)
    // and so on.
    int arr[5], i;

    printf("Size of integer in this compiler is %lu\n", sizeof(int));

    for (i = 0; i < 5; i++)
        // The use of '&' before a variable name, yields
        // address of variable.
        printf("Address arr[%d] is %p\n", i, &arr[i]);

    return 0;
}
chevron_right
```

C++

```
filter_none
edit
close

play_arrow
link
brightness_4
code

// C++ program to demonstrate that array elements
// are stored contiguous locations

#include <iostream>
using namespace std;

int main()
{
    // an array of 10 integers. If arr[0] is stored at
    // address x, then arr[1] is stored at x + sizeof(int)
    // arr[2] is stored at x + sizeof(int) + sizeof(int)
    // and so on.
    int arr[5], i;

    cout << "Size of integer in this compiler is " << sizeof(int) << "\n";

    for (i = 0; i < 5; i++)
        // The use of '&' before a variable name, yields
        // address of variable.
        cout << "Address arr[" << i << "] is " << &arr[i] << "\n";

    return 0;
}
chevron_right
```

Output:

```
Size of integer in this compiler is 4
Address arr[0] is 0x7ffd636b4260
Address arr[1] is 0x7ffd636b4264
Address arr[2] is 0x7ffd636b4268
```

Address arr[3] is 0x7ffd636b426c
Address arr[4] is 0x7ffd636b4270

Array vs Pointers

Arrays and pointer are two different things (we can check by applying sizeof). The confusion happens because array name indicates the address of first element and arrays are always passed as pointers (even if we use square bracket). Please see [Difference between pointer and array in C](#) for more details.

What is vector in C++?

Vector in C++ is a class in STL that represents an array. The advantages of vector over normal arrays are,

- We do not need pass size as an extra parameter when we declare a vector i.e., Vectors support dynamic sizes (we do not have to initially specify size of a vector). We can also resize a vector.
- Vectors have many in-built function like, removing an element, etc.
- To know more about functionalities provided by vector, please refer [vector in C++](#) for more details.

Related Articles:

- [Array is C/C++ | Set 2](#)
- [Initialization of multidimensional arrays in C/C++](#)
- [How to print size of array parameter in C++?](#)
- [Properties of arrays in C language](#)
- [Maximum element in an array](#)
- [Array sum in C++ STL](#)
- [Sort an array in C++](#)

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes 

Add your personal notes here! (max 5000 chars)

Recommended Posts:

- [Split the given array into K sub-arrays such that maximum sum of all sub arrays is minimum](#)
- [Merge k sorted arrays | Set 2 \(Different Sized Arrays\)](#)
- [Count of possible arrays from prefix-sum and suffix-sum arrays](#)
- [Find sub-arrays from given two arrays such that they have equal sum](#)
- [Generate all possible sorted arrays from alternate elements of two given sorted arrays](#)
- [Maximum OR sum of sub-arrays of two different arrays](#)
- [Why is a\[i\] == i\[a\] in C/C++ arrays?](#)
- [Introduction to Arrays](#)
- [Sum of XOR of all sub-arrays of length K](#)
- [How to join two Arrays using STL in C++?](#)
- [Associative arrays in C++](#)
- [Minimum LCM and GCD possible among all possible sub-arrays](#)
- [Number of sub arrays with odd sum](#)
- [C | Arrays | Question 5](#)
- [C | Arrays | Question 6](#)

Improved By : [sameer2209](#), [RishabhPrabhu](#), [AnkurVineet](#), [jsanjana96](#)

Article Tags :

Arrays
C
C++
School Programming
C Array and String
C Basics
CBSE - Class 11
cpp-array
CPP-Basics

Practice Tags :

Arrays
C
CPP



179

To-do Done
1.4

Based on 247 vote(s)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) What are the data types for which it is not possible to create an array?

Next

[last_page](#) OLA Cabs Interview Experience | Set 4 (For SDE 2)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in Arrays
Count of subarrays having exactly K perfect square numbers

Product of all Subarrays of an Array
Sliding Window Maximum : Set 2

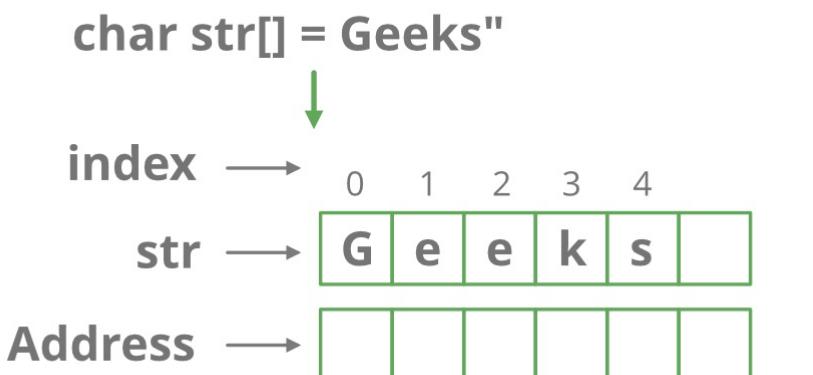
Perfect Sum Problem
Minimize the maximum difference between adjacent elements in an array

Most visited in C
Print in C++ with Examples
Swap two elements with help of Pragma in C/C++
Module Operator (%) in C/C++ with Examples
Program to calculate Electricity Bill
Role of SemiColon in Various Programming Languages

Strings in C

Strings are defined as an array of characters. The difference between a character array and a string is the string is terminated with a special character '\0'.

String in C



Declaration of strings: Declaring a string is as simple as declaring a one dimensional array. Below is the basic syntax for declaring a string.

```
char str_name[size];
```

In the above syntax str_name is any name given to the string variable and size is used define the length of the string, i.e the number of characters strings will store. Please keep in mind that there is an extra terminating character which is the Null character ('\0') used to indicate termination of string which differs strings from normal character arrays.

Initializing a String: A string can be initialized in different ways. We will explain this with the help of an example. Below is an example to declare a string with name as str and initialize it with "GeeksforGeeks".

```
1. char str[] = "GeeksforGeeks";
2. char str[50] = "GeeksforGeeks";
3. char str[] = {'G', 'e', 'e', 'k', 's', 'f', 'o', 'r', 'G', 'e', 'e', 'k', 's', '\0'};
4. char str[14] = {'G', 'e', 'e', 'k', 's', 'f', 'o', 'r', 'G', 'e', 'e', 'k', 's', '\0'};
```

Below is the memory representation of a string "Geeks".

0	1	2	3	4	5	
str	G	e	e	k	s	
Address	0x23452 0x23453 0x23454 0x23455 0x23456 0x23457					

Let us now look at a sample program to get a clear understanding of declaring and initializing a string in C and also how to print a string.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// C program to illustrate strings

#include<stdio.h>

int main()
{
    // declare and initialize string
    char str[] = "Geeks";

    // print string
    printf("%s",str);

    return 0;
}
chevron_right
```

Output:

Geeks

We can see in the above program that strings can be printed using a normal printf statements just like we print any other variable. Unlike arrays we do not need to print a string, character by character. The C language does not provide an inbuilt data type for strings but it has an access specifier "%s" which can be used to directly print and read strings.

Below is a sample program to read a string from user:

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// C program to read strings

#include<stdio.h>

int main()
{
    // declaring string
    char str[50];

    // reading string
    scanf("%s",str);

    // print string
    printf("%s",str);

    return 0;
}
chevron_right
```

You can see in the above program that string can also be read using a single scanf statement. Also you might be thinking that why we have not used the '&' sign with string name 'str' in scanf statement! To understand this you will have to recall your knowledge of scanf. We know that the '&' sign is used to provide the address of the variable to the scanf() function to store the value read in memory. As str[] is a character array so using str without braces '[' and ']' will give the base address of this string. That's why we have not used '&' in this case as we are already providing the base address of the string to scanf.

Passing strings to function: As strings are character arrays, so we can pass strings to function in a same way we pass an array to a function. Below is a sample program to do this:

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// C program to illustrate how to
// pass string to functions
#include<stdio.h>

void printStr(char str[])
{
    printf("String is : %s",str);
}

int main()
{
    // declare and initialize string
    char str[] = "GeeksforGeeks";

    // print string by passing string
    // to a different function
    printStr(str);

    return 0;
}
chevron_right
```

Output:

String is : GeeksforGeeks

Related Articles:

- puts() vs printf() to print a string
- Swap strings in C
- Storage for strings in C
- gets() is risky to use!

This article is contributed by **Harsh Agarwal**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

My Personal Notes arrow_drop_up

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- Number of common base strings for two strings
- <strings> library in C++ STL
- Comparing two strings in C++
- Output of C++ programs | Set 29 (Strings)
- Converting Strings to Numbers in C/C++
- Why "&" is not used for strings in scanf() function?
- C Program to Reverse Array of Strings
- C Program to concatenate two strings without using strcat
- Array of Strings in C++ (5 Different Ways to Create)
- How to write long strings in Multi-lines C/C++?
- How to convert C style strings to std::string and vice versa?
- C/C++ Program to Count number of binary strings without consecutive 1's
- Print all distinct circular strings of length M in lexicographical order
- C program to implement Adjacency Matrix of a given Graph

Article Tags :

C Programs
C-String
cpp-string

47

To-do Done

1.3

Based on 74 vote(s)

Basic **Easy** **Medium** **Hard** **Expert**

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) C Program for Decimal to Binary Conversion

Next

[last_page](#) C Program to Find minimum sum of factors of number

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Weekend Geeks Classes Weekdays

JAVA APP DEVELOPMENT COMPETITIVE PROGRAMMING LITE Java Backend

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS

Most popular in C Programs
C program to implement Adjacency Matrix of a given Graph
Find all comparable and non comparable edges of a machine
Print 12 Days Of Christmas in a few lines of code | The xmas.c code
C/C++ program to add N distances given in inch-feet system using Structures
Hardy's Rule

Most Visited Articles

Python Tutorial
Calculate the Sum of GCD over all subarrays
10 Projects That Every Developer Should Lay Their Hands-On
Amazon Interview Experience for SDE2 | 3+ years Experienced
C++ Tutorial

Arrays in C Language | Set 2 (Properties)

We have introduced arrays in set 1 ([Introduction to arrays in C](#)).

In this post array properties in C are discussed.

1) In C, it is possible to have array of all types except void and functions. See [this](#) for details.

2) In C, array and pointer are different. They seem similar because array name gives address of first element and array elements are accessed using pointer arithmetic. See [array vs pointer in C](#) for details.

3) Arrays are always passed as pointer to functions. See [this](#) for details.

4) A character array initialized with double quoted string has last element as '\0'. See [this](#) for details.

5) Like other variables, arrays can be allocated memory in any of the three segments, data, heap, and stack (See [this](#) for details). Dynamically allocated arrays are allocated memory on heap, static or global arrays are allocated memory on data segment and local arrays are allocated memory on stack segment.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes [arrow_drop_up](#)

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- [kbhit in C language](#)
- [Difference between while\(1\) and while\(0\) in C language](#)
- [Signals in C language](#)
- [Stopwatch using C language](#)
- [C Language Introduction](#)
- [fgets\(\) and gets\(\) in C language](#)
- [A C Programming Language Puzzle](#)
- [Constants vs Variables in C language](#)
- [C Programming Language Standard](#)
- [isalnum\(\) function in C Language](#)
- [Interesting facts about C Language](#)
- [isupper\(\) function in C Language](#)
- [isxdigit\(\) function in C Language](#)
- [Difference between Java and C language](#)
- [Features of C Programming Language](#)

Article Tags :

C
C Array and String

C Basics

cpp-array

Practice Tags :

C

thumb_up
25

To-do Done
1.8

Based on 49 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) How arrays are passed to functions in C/C++

Next

[last_page](#) Write your own strlen() for a long string padded with '\0's

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Weekend Geeks Classes Weekdays

JAVA APP DEVELOPMENT COMPETITIVE PROGRAMMING DN Java Backend Development

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS



Most popular in C
Program to calculate Electricity Bill

Program to print half Diamond star pattern
C/C++ program for calling main() in main()
exit(0) vs exit(1) in C/C++ with Examples
C/C++ #include directive with Examples

More related articles in C
Output of C programs | Set 66 (Accessing Memory Locations)
Chain of Pointers in C with Examples
Difference between Type Casting and Type Conversion
C program to display month by month calendar for a given year
Print all possible combinations of the string by replacing 'S' with any other digit from the string

Do not use sizeof for array parameters

Consider the below program.

```
filter_none
edit
close

play_arrow
link
brightness_4
code

#include<stdio.h>
void fun(int arr[])
{
    int i;

    /* sizeof should not be used here to get number
     * of elements in array*/
    int arr_size = sizeof(arr)/sizeof(arr[0]); /* incorrect use of sizeof*/
    for (i = 0; i < arr_size; i++)
    {
        arr[i] = i; /*executed only once */
    }
}

int main()
{
    int i;
    int arr[4] = {0, 0 ,0, 0};
    fun(arr);

    /* use of sizeof is fine here*/
    for(i = 0; i < sizeof(arr)/sizeof(arr[0]); i++)
        printf(" %d ",arr[i]);

    getchar();
    return 0;
}
chevron_right
filter_none
```

Output: 0 0 0 on a IA-32 machine.

The function fun() receives an array parameter arr[] and tries to find out number of elements in arr[] using sizeof operator.

In C, array parameters are treated as pointers (See [this](#) for details). So the expression sizeof(arr)/sizeof(arr[0]) becomes sizeof(int*)/sizeof(int) which results in 1 for IA 32 bit machine (size of int and int * is 4) and the for loop inside fun() is executed only once irrespective of the size of the array.

Therefore, sizeof should not be used to get number of elements in such cases. A separate parameter for array size (or length) should be passed to fun(). So the **corrected program is:**

```
filter_none
edit
close

play_arrow
link
brightness_4
code

#include<stdio.h>
void fun(int arr[], size_t arr_size)
{
    int i;
    for (i = 0; i < arr_size; i++)
    {
        arr[i] = i;
    }
}

int main()
{
    int i;
    int arr[4] = {0, 0 ,0, 0};
    fun(arr, 4);

    for(i = 0; i < sizeof(arr)/sizeof(arr[0]); i++)
        printf(" %d ", arr[i]);

    getchar();
    return 0;
}
chevron_right
filter_none
```

Please write comments if you find anything incorrect in the above article or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes

Add your personal notes here! (max 5000 chars)

Recommended Posts:

- Is sizeof for a struct equal to the sum of sizeof of each member?
- How to find size of array in C/C++ without using sizeof ?
- Why C treats array parameters as pointers?
- sizeof operator in C
- Why does sizeof(x++) not increment x in C?
- Implement Your Own sizeof
- sizeof() for Floating Constant in C
- Anything written in sizeof() is never executed in C
- Operands for sizeof operator
- Difference between strlen() and sizeof() for string in C
- Class template with multiple parameters
- What is evaluation order of function parameters in C?
- Jagged Array or Array of Arrays in C with Examples
- Difference between pointer to an array and array of pointers
- array::fill() and array::swap() in C++ STL

Improved By : shohamziner

Article Tags :

Articles
C
C++
C Array and String

Practice Tags :

C
CPP



37

To-do Done
2.6

Based on 105 vote(s)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

first_page [gets\(\) is risky to use!](#)

Next

last_page [When is copy constructor called?](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Online Geeks Classes YouTube

JAVA APP DEVELOPMENT COMPETITIVE PROGRAMMING LIVE Java Backend Development

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS



Most popular in Article
[Interview and Coding Tricks in Java](#)
[Design a Queue data structure to get minimum or maximum in O\(1\) time](#)
[Count number of pairs with positive sum in an array](#)
[Count ways to reach the Nth stair using multiple 1 or 2 steps and a single step](#)
[Find a number such that maximum in array is minimum possible after XOR](#)

Most visited in C
[P\(n\) in C++ with Examples](#)
[Speed up Code executions with help of Pragma in C/C++](#)
[Program to calculate Electricity Bill](#)
[Modulo Operator \(%\) in C/C++ with Examples](#)
[Role of SemiColon in various Programming Languages](#)

Initialization of variables sized arrays in C

The C99 standard allows variable sized arrays (see [this](#)). But, unlike the normal arrays, variable sized arrays cannot be initialized.

For example, the following program compiles and runs fine on a C99 compatible compiler.

```
close
play_arrow
link
brightness_4
code
#include<stdio.h>

int main()
{
    int M = 2;
    int arr[M][M];
    int i, j;
    for (i = 0; i < M; i++)
    {
        for (j = 0; j < M; j++)
        {
            arr[i][j] = 0;
            printf ("%d ", arr[i][j]);
        }
        printf("\n");
    }
    return 0;
}
chevron_right
```

[filter_none](#)

Output:

```
0 0
0 0
```

But the following fails with compilation error.

```
filter_none
edit
close
play_arrow
link
brightness_4
code
#include<stdio.h>
```

```
int main()
{
    int M = 2;
    int arr[M][M] = {0}; // Trying to initialize all values as 0
    int i, j;
    for (i = 0; i < M; i++)
    {
        for (j = 0; j < M; j++)
        {
            printf ("%d ", arr[i][j]);
            printf("\n");
        }
    }
    return 0;
}
```

[chevron_right](#)

[filter_none](#)

Output:

Compiler Error: variable-sized object may not be initialized

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes [arrow_drop_up](#)

Add your personal notes here! (max 5000 chars)

[Save](#)

Recommended Posts:

- Initialization of static variables in C
- Implicit initialization of variables with 0 or 1 in C
- Initialization of global and static variables in C
- Initialization of a multidimensional arrays in C/C++
- Difference between Static variables and Register variables in C
- Initialization of data members
- Static Variables in C
- Variables and Keywords in C
- Can Global Variables be dangerous ?
- Constants vs Variables in C language
- Operations on struct variables in C
- C Program to print environment variables
- How are variables scoped in C - Static or Dynamic?
- What are the default values of static variables in C?
- How will you show memory representation of C variables?

Article Tags :

C
C Array and String

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) Complicated declarations in C

Next

[last_page](#) Scansets in C

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Are array members deeply copied?

In C/C++, we can assign a struct (or class in C++ only) variable to another variable of same type. When we assign a struct variable to another, all members of the variable are copied to the other struct variable. But what happens when the structure contains pointer to dynamically allocated memory and what if it contains an array?

In the following C++ program, struct variable st1 contains pointer to dynamically allocated memory. When we assign st1 to st2, str pointer of st2 also start pointing to same memory location. This kind of copying is called **Shallow Copy**.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

# include <iostream>
# include <string.h>

using namespace std;

struct test
{
    char *str;
};

int main()
{
    struct test st1, st2;

    st1.str = new char[20];
    strcpy(st1.str, "GeeksforGeeks");

    st2 = st1;

    st1.str[0] = 'X';
    st1.str[1] = 'Y';

    /* Since copy was shallow, both strings are same */
}
```

/* Since copy was shallow, both strings are same */

```
cout << "st1's str = " << st1.str << endl;
cout << "st2's str = " << st2.str << endl;
```

```
return 0;
}
```

```
chevron_right
```

```
filter_none
```

Output:

```
st1's str = XYeksforGeeks
st2's str = XYeksforGeeks
```

Now, what about arrays? *The point to note is that the array members are not shallow copied, compiler automatically performs Deep Copy for array members.* In the following program, struct test contains array member str[]. When we assign st1 to st2, st2 has a new copy of the array. So st2 is not changed when we change str[] of st1.

```
filter_none
```

```
edit
```

```
close
```

```
play_arrow
```

```
link
```

```
brightness_4
```

```
code
```

```
# include <iostream>
# include <string.h>
```

```
using namespace std;
```

```
struct test
```

```
{
```

```
    char str[20];
```

```
};
```

```
int main()
```

```
{
```

```
    struct test st1, st2;
```

```
    strcpy(st1.str, "GeeksforGeeks");
```

```
    st2 = st1;
```

```
    st1.str[0] = 'X';
    st1.str[1] = 'Y';
```

```
/* Since copy was Deep, both arrays are different */
```

```
cout << "st1's str = " << st1.str << endl;
cout << "st2's str = " << st2.str << endl;
```

```
return 0;
}
```

```
chevron_right
```

```
filter_none
```

Output:

```
st1's str = XYeksforGeeks
st2's str = GeeksforGeeks
```

Therefore, for C++ classes, we don't need to write our own copy constructor and assignment operator for array members as the default behavior is Deep copy for arrays.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes [arrow_drop_up](#)

Add your personal notes
here! (max 5000 chars)

Save

Recommended Posts:

- [Different ways to Initialize all members of an array to the same value in C](#)
- [Flexible Array Members in a structure in C](#)
- [Static data members in C++](#)
- [Initialization of data members](#)
- [Can we access private data members of a class without using a member or a friend function?](#)
- [Difference between pointer to an array and array of pointers](#)
- [Jagged Array or Array of Arrays in C with Examples](#)
- [What's difference between "array" and "&array" for "int array\[5\]" ?](#)
- [Sum of an array using MPI](#)
- [How to pass an array by value in C ?](#)
- [Array class in C++](#)
- [Pointer to an Array | Array Pointer](#)
- [Pointer vs Array in C](#)
- [4 Dimensional Array in C/C++](#)
- [Why array index starts from zero ?](#)

Article Tags :

C
C Array and String
pointer

Practice Tags :

C

thumb_up

11

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

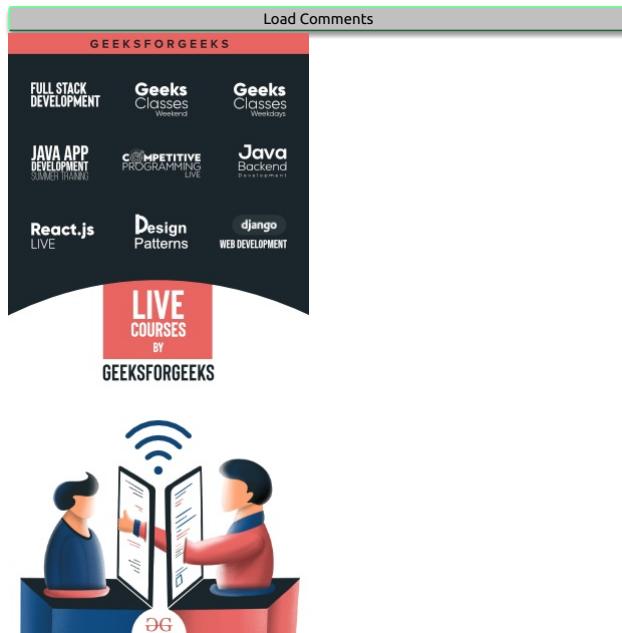
Previous

[first_page When do we use Initializer List in C++?](#)

Next

[last_page C++ Internals | Default Constructors | Set 1](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C
Program to calculate Electricity Bill
Program to print half Diamond star pattern
C/C++ program for calling main() from main()
C/C++ #include directive with Examples
Output of C programs | Set 66 (Accessing Memory Locations)

More related articles in C
Difference between Type Casting and Type Conversion
Chain of Pointers in C with Examples
Print all possible combinations of the string by replacing '\$' with any other digit from the string
getch() function in C with Examples
Jagged Array or Array of Arrays in C with Examples

What is the difference between single quoted and double quoted declaration of char array?

In C/C++, when a character array is initialized with a double quoted string and array size is not specified, compiler automatically allocates one extra space for string terminator '0'. For example, following program prints 6 as output.

```
filter_none
edit
close
play_arrow
link
brightness_4
code
#include<stdio.h>
int main()
{
    // size of arr[] is 6 as it is '\0' terminated
    char arr[] = "geeks";

    printf("%lu", sizeof(arr));

    return 0;
}
```

If array size is specified as 5 in the above program then the program works without any warning/error and prints 5 in C, but causes compilation error in C++.

```
filter_none
edit
close
play_arrow
link
brightness_4
code
// Works in C, but compilation error in C++
#include<stdio.h>
int main()
```

```
{  
// arr[] is not terminated with '\0'  
// and its size is 5  
char arr[5] = "geeks";  
  
printf("%lu", sizeof(arr));  
  
return 0;  
}  
chevron_right
```

filter_none
Output :

5

When character array is initialized with comma separated list of characters and array size is not specified, compiler doesn't create extra space for string terminator '\0'. For example, following program prints 5.

```
filter_none  
edit  
close  
play_arrow  
link  
brightness_4  
code  
  
#include<stdio.h>  
int main()  
{  
// arr[] is not terminated with '\0'  
// and its size is 5  
char arr[]={ 'g', 'e', 'e', 'k', 's' };  
  
printf("%lu", sizeof(arr));  
  
return 0;  
}  
chevron_right
```

filter_none
Output :

5

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up

Add your personal notes
here! (max 5000 chars)

Save

Recommended Posts:

- [C/C++ program to find the size of int, float, double and char](#)
- [size of char datatype and char array in C](#)
- [What is the difference between "char a" and "char a\[1\]"?](#)
- [What's difference between char s\[\] and char *s in C?](#)
- [Difference between float and double in C/C++](#)
- [Difference between const char *p, char * const p and const char * const p](#)
- [What happens when a function is called before its declaration in C?](#)
- [C | Variable Declaration and Scope | Question 3](#)
- [C | Variable Declaration and Scope | Question 2](#)
- [C | Variable Declaration and Scope | Question 4](#)
- [C | Variable Declaration and Scope | Question 1](#)
- [C | Variable Declaration and Scope | Question 8](#)
- [C | Variable Declaration and Scope | Question 7](#)
- [C | Variable Declaration and Scope | Question 6](#)
- [Difference between pointer to an array and array of pointers](#)

Article Tags :

C
C Array and String

Practice Tags :

C

thumb_up

20

To-do Done
1.7

Based on 66 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) [C/C++ Ternary Operator - Some Interesting Observations](#)

Next

[last_page](#) [Modulus on Negative Numbers](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

[Load Comments](#)



Most popular in C
Program to print half Diamond star pattern
C/C++ program for calling main() in main()
C/C++ #include directive with Examples
Output of C programs | Set 66 (Accessing Memory Locations)
Difference between Type Casting and Type Conversion

More related articles in C
[getch\(\) function in C with Examples](#)
[Character Pointers in C with Examples](#)
[Jagged Array | Array of Arrays in C with Examples](#)
[Print all possible combinations of the string by replacing '\\$' with any other digit from the string type.h<cctype> library in C/C++ with Examples](#)

Initialization of a multidimensional arrays in C/C++

In C/C++, initialization of a multidimensional arrays can have left most dimension as optional. Except the left most dimension, all other dimensions must be specified.

For example, following program fails in compilation because two dimensions are not specified.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

#include<stdio.h>
int main()
{
    int a[][][2] = { {{1, 2}, {3, 4}},
                    {{5, 6}, {7, 8}}
                }; // error
    printf("%d", sizeof(a));
    getchar();
    return 0;
}
chevron_right
filter_none
```

Following 2 programs work without any error.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// Program 1
#include<stdio.h>
int main()
{
    int a[][2] = {{1,2},{3,4}}; // Works
    printf("%lu", sizeof(a)); // prints 4*sizeof(int)
    getchar();
    return 0;
}
chevron_right
filter_none
edit
close
play_arrow
link
brightness_4
code

// Program 2
#include<stdio.h>
int main()
{
```

```

int a[][][2] = { {{1, 2}, {3, 4}},
                {{5, 6}, {7, 8}}
            }; // Works
printf("%lu", sizeof(a)); // prints 8*sizeof(int)
getchar();
return 0;
}

```

[chevron_right](#)

[filter_none](#)

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes [arrow_drop_up](#)

Add your personal notes here! (max 5000 chars)

[Save](#)

Recommended Posts:

- Multidimensional Arrays in C / C++
- Initialization of variables sized arrays in C
- Zero Initialization in C++
- Uniform Initialization in C++
- Implicit initialization of variables with 0 or 1 in C
- Initialization of data members
- Initialization of static variables in C
- Initialization of global and static variables in C
- Multidimensional Pointer Arithmetic in C/C++
- How to print dimensions of multidimensional array in C++
- Arrays in C/C++
- Why is `a[i] == i[a]` in C/C++ arrays?
- How to join two Arrays using STL in C++?
- Associative arrays in C++
- C | Arrays | Question 1

Improved By : HassanAli, cheto96

Article Tags :

C
C++
C Array and String
Practice Tags :
C
CPP

[thumb_up](#)
12

To-do Done
1.9

Based on 53 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) Comma operator should be used carefully

Next

[last_page](#) Data type of case labels of switch statement in C++?

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

[Load Comments](#)

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Weekend Geeks Classes Weekdays

JAVA APP DEVELOPMENT SUMMER TRAINING COMPETITIVE PROGRAMMING LIVE Java Backend Part-time

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS



Most popular in C

Write one line functions for strcat() and strcmp()

Recursion can be used to do both tasks in one line. Below are one line implementations for strcat() and strcmp().

```
filter_none
edit
close

play_arrow

link
brightness_4
code

/* my_strcat(dest, src) copies data of src to dest. To do so, it first reaches end of the string dest using recursive calls my_strcat(++dest, src). Once end of dest is reached, data is copied using
(*dest++ = *src++)? my_strcat(dest, src). */
void my_strcat(char *dest, char *src)
{
    (*dest)? my_strcat(++dest, src): (*dest++ = *src++)? my_strcat(dest, src): 0 ;
}

/* driver function to test above function */
int main()
{
    char dest[100] = "geeksfor";
    char *src = "geeks";
    my_strcat(dest, src);
    printf(" %s ", dest);
    getchar();
}
```

The function my_strcmp() is simple compared to my_strcmp().

```
filter_none
edit
close

play_arrow

link
brightness_4
code

/* my_strcmp(a, b) returns 0 if strings a and b are same, otherwise 1. It recursively increases a and b pointers. At any point if *a is not equal to *b then 1 is returned. If we reach end of both strings at the same time then 0 is returned. */
int my_strcmp(char *a, char *b)
{
    return (*a == *b && *b == '\0')? 0 : (*a == *b)? my_strcmp(++a, ++b): 1;
}

/* driver function to test above function */
int main()
{
    char *a = "geeksforgeeks";
    char *b = "geeksforgeeks";
    if(my_strcmp(a, b) == 0)
        printf(" String are same ");
    else
        printf(" String are not same ");

    getchar();
    return 0;
}
```

The above functions do very basic string concatenation and string comparison. These functions do not provide same functionality as standard library functions.

Please write comments if you find the above code incorrect, or find better ways to solve the same problem.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes 

Add your personal notes here! (max 5000 chars)

Recommended Posts:

- Write a one line C function to round floating point numbers
- strcat() vs strncat() in C++
- strcmp() in C/C++
- Difference between strncmp() and strcmp in C/C++
- C program to print odd line contents of a File followed by even line content
- Write your own memcpy() and memmove()

- How to write your own header file in C?
- Write a C program that won't compile in C++
- When should we write our own copy constructor?
- When should we write our own assignment operator in C++?
- Read/Write structure to a file in C
- Write a C macro PRINT(x) which prints x
- C program to write an image in PGM format
- Write a program that produces different results in C and C++
- How to write a running C code without main()?

Article Tags :

C
C Array and String

Practice Tags :

C

thumb_up

5

To-do Done
4

Based on 50 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

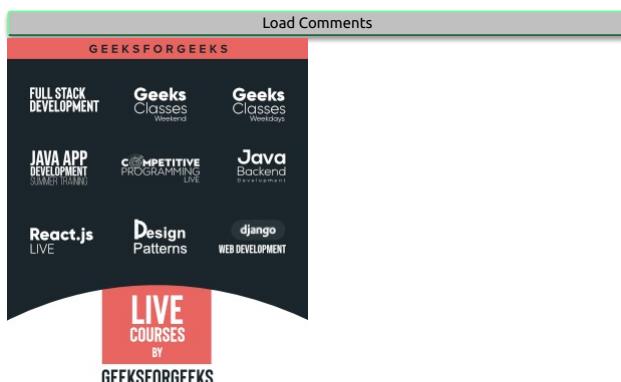
Previous

[first_page](#) A nested loop puzzle

Next

[last_page](#) Data type of character constants in C and C++

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C
C/C++ program for calling main() in main()
C/C++ #include directive with Examples
Difference between Type Casting and Type Conversion
Chain of Pointers in C with Examples
Jagged Array or Array of Arrays in C with Examples

More related articles in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
getch() function in C with Examples
How to use make utility to build C projects?
ctype.h<cctype> library in C/C++ with Examples
C program to display month by month calendar for a given year

What's difference between char s[] and char *s in C?

Consider below two statements in C. What is the difference between the two?

```
char s[] = "geeksquiz";
char *s = "geeksquiz";
```

Below are the key differences:

Char a[10] = "geek";	Char *p = "geek";
1) a is an array	1) p is a pointer variable
2) sizeof(a) = 10 bytes	2) sizeof(p) = 4 bytes
3) a and &a are same	3) p and &p aren't same
4) geek is stored in stack section of memory	4) p is stored at stack but geek is stored at code section of memory
5) char a[10] = "geek"; a = "hello"; //invalid > a, itself being an address and string constant is also an address, so not possible.	5) char *p = "geek"; p = "india"; //valid
6) a++ is invalid	6) p++ is valid
7) char a[10] = "geek"; a[0] = 'b'; //valid	7) char *p = "geek"; p[0] = 'k'; //invalid > Code section is r- only.

The statements '**char s[] = "geeksquiz"**' creates a character array which is like any other array and we can do all array operations. The only special thing about this array is, although we have initialized it with 9 elements, its size is 10 (Compiler automatically adds '\0')

```
filter_none
edit
close

play_arrow

link
brightness_4
code

#include <stdio.h>
int main()
{
    char s[] = "geeksquiz";
    printf("%lu", sizeof(s));
    s[0] = 'j';
    printf("\n%s", s);
    return 0;
}
```

chevron_right

filter_none

Output:

```
10
geeksquiz
```

The statement '**char *s = "geeksquiz"**' creates a string literal. The string literal is stored in the read-only part of memory by most of the compilers. The C and C++ standards say that string literals have static storage duration, any attempt at modifying them gives undefined behaviour.

s is just a pointer and like any other pointer stores address of string literal.

```
filter_none
edit
close

play_arrow

link
brightness_4
code

#include <stdio.h>
int main()
{
    char *s = "geeksquiz";
    printf("%lu", sizeof(s));

    // Uncommenting below line would cause undefined behaviour
    // (Caused segmentation fault on gcc)
    // s[0] = 'j';
    return 0;
}
```

chevron_right

filter_none

Output:

```
8
```

Running above program may generate a warning also "warning: deprecated conversion from string constant to 'char*'". This warning occurs because s is not a const pointer, but stores address of the read-only location. The warning can be avoided by the pointer to const.

```
filter_none
edit
close

play_arrow

link
brightness_4
code

#include <stdio.h>
int main()
{
    const char *s = "geeksquiz";
}
```

```
printf("%lu", sizeof(s));
return 0;
}
```

[chevron_right](#)

[filter_none](#)

This article is contributed by Abhay Rathi. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes [arrow_drop_up](#)

Add your personal notes
here! (max 5000 chars)

[Save](#)

Recommended Posts:

- What is the difference between "char a" and "char a[1]"?
- size of char datatype and char array in C
- Difference between const char *p, char * const p and const char * const p
- char* vs std::string vs char[] in C++
- What is the difference between single quoted and double quoted declaration of char array?
- Convert string to char array in C++
- Wide char and library functions in C++
- C/C++ program to find the size of int, float, double and char
- Difference between WCF and Web API
- Difference between MP4 and MP3
- Difference between URL and URI
- Difference between LAN and MAN
- Difference between CD and DVD
- Difference between USB 2.0 and USB 3.0
- Difference between Tor and VPN

Article Tags :

C
Difference Between
C Array and String
cpp-string
Practice Tags :
C

[thumb_up](#)

27

To-do Done
2.8

Based on 28 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

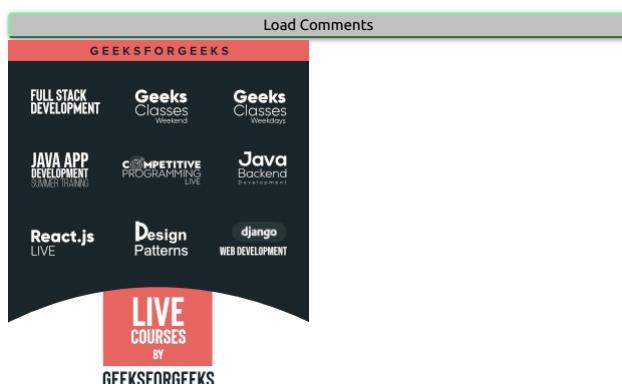
Previous

[first_page](#) What's difference between Microcontroller (μ C) and Microprocessor (μ P)?

Next

[last_page](#) How to check whether a number is in the range[low, high] using one comparison ?

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C
C/C++ program for calling main() in main()
Difference between Type Casting and Type Conversion
Chain of Pointers in C with Examples
Print all possible combinations of the string by replacing 's' with any other digit from the string
getch() function in C with Examples

Most visited in Difference Between
Difference between PostgreSQL and MongoDB
Difference between LEFT and TRUNCATE
Difference Between SMO and SEO
Difference between Prim's and Kruskal's algorithm for MST
Monolithic vs Microservices architecture

gets() is risky to use!

Consider the below program.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

void read()
{
    char str[20];
    gets(str);
    printf("%s", str);
    return;
}
chevron_right
filter_none
```

The code looks simple, it reads string from standard input and prints the entered string, but it suffers from [Buffer Overflow](#) as `gets()` doesn't do any array bound testing. `gets()` keeps on reading until it sees a newline character.

To avoid Buffer Overflow, `fgets()` should be used instead of `gets()` as `fgets()` makes sure that not more than `MAX_LIMIT` characters are read.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

#define MAX_LIMIT 20
void read()
{
    char str[MAX_LIMIT];
    fgets(str, MAX_LIMIT, stdin);
    printf("%s", str);

    getchar();
    return;
}
chevron_right
filter_none
```

Please write comments if you find anything incorrect in the above article, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes [arrow_drop_up](#)

[Save](#)

Recommended Posts:

- Find distance between two nodes in the given Binary tree for Q queries
- Query to count odd and even parity elements in subarray after XOR with K
- Count of the non-prime divisors of a given number
- Priority queue of pairs in C++ with ordering by first and second element
- How to build a simple music player app using Android Studio
- Extends vs Implements in Java
- Format specifiers in different Programming Languages
- C program to find square root of a given number
- C program to print odd line contents of a File followed by even line content
- new vs malloc() and free() vs delete in C++
- When to use each sorting algorithms
- Minimum Bottleneck Spanning Tree(MBST)
- Logarithm tricks for Competitive Programming
- Step by Step guide to install IntelliJ Idea

Article Tags :

[Articles](#)
[C](#)
[C Array and String](#)
[C-programming](#)
[gets](#)
[secure-coding](#)

Practice Tags :

[C](#)

[thumb_up](#)

23

To-do Done
2.3

Based on 74 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) How does “void *” differ in C and C++?

Next

[last_page](#) Do not use sizeof for array parameters

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Weekend Geeks Classes Weekdays

Java APP DEVELOPMENT SUMMER TRAINING COMPETITIVE PROGRAMMING LIVE Java Backend Part-time

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS



Most popular in Articles
Design a Queue data structure to get minimum or maximum in O(1) time
Find a number such that maximum in array is minimum possible after XOR
Count the number of stairs in a staircase using multiple 1 or 2 steps and a single step 3
new vs malloc() and free() vs delete in C++
Dockerizing a simple Django app

Most visited in C
P(i) in C++ with Examples
Speed up Code executions with help of Pragma in C/C++
Modulo Operator (%) in C/C++ with Examples
Program to calculate Electricity Bill
Role of SemiColon in various Programming Languages

C function to Swap strings

Let us consider the below program.

```
filter_none
edit
close

play_arrow

link
brightness_4
code

#include<stdio.h>
void swap(char *str1, char *str2)
{
    char *temp = str1;
    str1 = str2;
    str2 = temp;
}

int main()
{
    char *str1 = "geeks";
    char *str2 = "forgeeks";
    swap(str1, str2);
    printf("str1 is %s, str2 is %s", str1, str2);
    getchar();
    return 0;
}
chevron_right
filter_none
```

Output of the program is *str1 is geeks, str2 is forgeeks*. So the above swap() function doesn't swap strings. The function just changes local pointer variables and the changes are not reflected outside the function.

Let us see the correct ways for swapping strings:

Method 1(Swap Pointers)

If you are **using character pointer for strings** (not arrays) then change str1 and str2 to point each other's data. i.e., swap pointers. In a function, if we want to change a pointer (and obviously we want changes to be reflected outside the function) then we need to pass a pointer to the pointer.

```
filter_none
edit
close

play_arrow

link
brightness_4
code

#include<stdio.h>

/* Swaps strings by swapping pointers */
void swap1(char **str1_ptr, char **str2_ptr)
{
    char *temp = *str1_ptr;
```

```

*str1_ptr = *str2_ptr;
*str2_ptr = temp;
}

int main()
{
    char *str1 = "geeks";
    char *str2 = "forgeeks";
    swap1(&str1, &str2);
    printf("str1 is %s, str2 is %s", str1, str2);
    getchar();
    return 0;
}

```

chevron_right

filter_none

This method cannot be applied if strings are stored using character arrays.

Method 2(Swap Data)

If you are [using character arrays to store strings](#) then preferred way is to swap the data of both arrays.

filter_none

edit

close

play_arrow

link

brightness_4

code

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
```

/ Swaps strings by swapping data*/*

```
void swap2(char *str1, char *str2)
{
    char *temp = (char *)malloc((strlen(str1) + 1) * sizeof(char));
    strcpy(temp, str1);
    strcpy(str1, str2);
    strcpy(str2, temp);
    free(temp);
}
```

int main()

{

```
    char str1[10] = "geeks";
    char str2[10] = "forgeeks";
    swap2(str1, str2);
    printf("str1 is %s, str2 is %s", str1, str2);
    getchar();
    return 0;
}
```

chevron_right

filter_none

This method cannot be applied for strings stored in read-only block of memory.

Please write comments if you find anything incorrect in the above article, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes *arrow_drop_up*

Add your personal notes
here! (max 5000 chars)

Recommended Posts:

- Why "&" is not used for strings in `scanf()` function?
- Count of strings that become equal to one of the two strings after one removal
- Swap two variables in one line in C/C++, Python, PHP and Java
- Storage for Strings in C
- Converting Strings to Numbers in C/C++
- Program to check if two strings are same or not
- Multiply N complex numbers given as strings
- Check whether an array of strings can correspond to a particular number X
- How to write long strings in Multi-lines C/C++?
- How to convert C style strings to `std::string` and vice versa?
- How to call function within function in C or C++
- `arc` function in C
- `strftime()` function in C/C++
- Inline function in C
- Function Pointer in C

Article Tags :

Articles

C

C Array and String

Practice Tags :

C

thumb_up

22

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

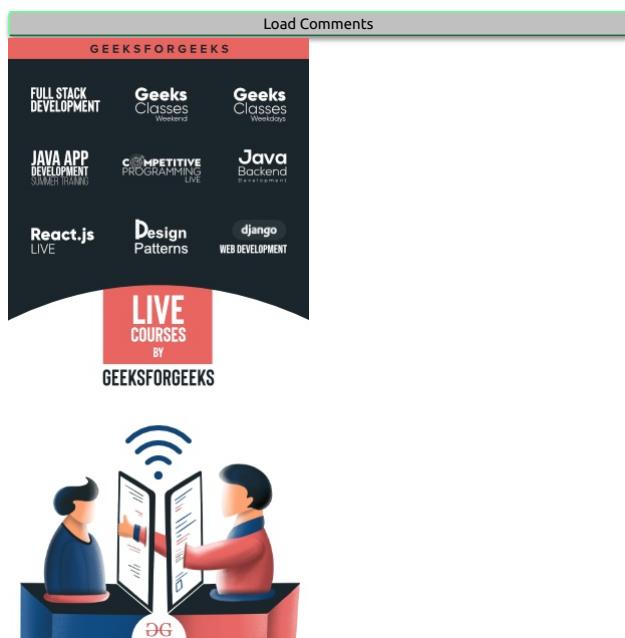
Previous

[first_page An Uncommon representation of array elements](#)

Next

[last_page How does "void *" differ in C and C++?](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in Articles
Find a number such that maximum in array is minimum possible after XOR
Count ways to reach the Nth stair using multiple 1 or 2 steps and a single step 3
new vs malloc() and free() vs delete in C++
Dockerizing a simple Django app
How to build a simple music player app using Android Studio

Most visited in C
Print in C/C++ with Examples
String Case executions with help of Pragma in C/C++
Modulo Operator (%) in C/C++ with Examples
Program to calculate Electricity Bill
Role of SemiColon in various Programming Languages

Storage for Strings in C

In C, a string can be referred to either using a character pointer or as a character array.

Strings as character arrays

```
filter_none
edit
close
play_arrow
link
brightness_4
code

char str[4] = "GfG"; /*One extra for string terminator*/
/* OR */
char str[4] = {'G', 'f', 'G', '\0'}; /* '\0' is string terminator */
chevron_right
filter_none
```

When strings are declared as character arrays, they are stored like other types of arrays in C. For example, if str[] is an **auto variable** then string is stored in stack segment, if it's a global or static variable then stored in **data segment**, etc.

Strings using character pointers

Using character pointer strings can be stored in two ways:

1) Read only string in a shared segment.

When a string value is directly assigned to a pointer, in most of the compilers, it's stored in a read-only block (generally in data segment) that is shared among functions.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

char *str = "GfG";
chevron_right
filter_none
```

In the above line "GfG" is stored in a shared read-only location, but pointer str is stored in a read-write memory. You can change str to point something else but cannot change value at present str. So this kind of string should only be used when we don't want to modify string at a later stage in the program.

2) Dynamically allocated in heap segment.

Strings are stored like other dynamically allocated things in C and can be shared among functions.

```
filter_none
```

```

edit
close
play_arrow

link
brightness_4
code

char *str;
int size = 4; /*one extra for '\0'*/
str = (char *)malloc(sizeof(char)*size);
*(str+0) = 'G';
*(str+1) = 'f';
*(str+2) = 'G';
*(str+3) = '\0';
chevron_right
filter_none

```

Let us see some examples to better understand the above ways to store strings.

Example 1 (Try to modify string)

The below program may crash (gives segmentation fault error) because the line *(str+1) = 'n' tries to write a read only memory.

```

filter_none
edit
close
play_arrow

link
brightness_4
code

int main()
{
    char *str;
    str = "GfG"; /* Stored in read only part of data segment */
    *(str+1) = 'n'; /* Problem: trying to modify read only memory */
    getchar();
    return 0;
}
chevron_right
filter_none

```

Below program works perfectly fine as str[] is stored in writable stack segment.

```

filter_none
edit
close
play_arrow

link
brightness_4
code

int main()
{
    char str[] = "GfG"; /* Stored in stack segment like other auto variables */
    *(str+1) = 'n'; /* No problem: String is now GnG */
    getchar();
    return 0;
}
chevron_right
filter_none

```

Below program also works perfectly fine as data at str is stored in writable heap segment.

```

filter_none
edit
close
play_arrow

link
brightness_4
code

int main()
{
    int size = 4;

    /* Stored in heap segment like other dynamically allocated things */
    char *str = (char *)malloc(sizeof(char)*size);
    *(str+0) = 'G';
    *(str+1) = 'f';
    *(str+2) = 'G';
    *(str+3) = '\0';
    *(str+1) = 'n'; /* No problem: String is now GnG */
    getchar();
    return 0;
}
chevron_right
filter_none

```

Example 2 (Try to return string from a function)

The below program works perfectly fine as the string is stored in a shared segment and data stored remains there even after return of getString()

```

filter_none
edit
close
play_arrow

```

```
link  
brightness_4  
code  
  
char *getString()  
{  
    char *str = "GfG"; /* Stored in read only part of shared segment */  
  
    /* No problem: remains at address str after getString() returns*/  
    return str;  
}  
  
int main()  
{  
    printf("%s", getString());  
    getchar();  
    return 0;  
}
```

[chevron_right](#)
[filter_none](#)

The below program also works perfectly fine as the string is stored in heap segment and data stored in heap segment persists even after the return of getString()

```
filter_none  
edit  
close  
  
play_arrow  
  
link  
brightness_4  
code  
  
char *getString()  
{  
    int size = 4;  
    char *str = (char *)malloc(sizeof(char)*size); /*Stored in heap segment*/  
    *(str+0) = 'G';  
    *(str+1) = 'f';  
    *(str+2) = 'G';  
    *(str+3) = 'O';  
  
    /* No problem: string remains at str after getString() returns */  
    return str;  
}
```

```
int main()  
{  
    printf("%s", getString());  
    getchar();  
    return 0;  
}  
chevron\_right  
filter\_none
```

But, the below program may print some garbage data as string is stored in stack frame of function getString() and data may not be there after getString() returns.

```
filter_none  
edit  
close  
  
play_arrow  
  
link  
brightness_4  
code  
  
char *getString()  
{  
    char str[] = "GfG"; /* Stored in stack segment */  
  
    /* Problem: string may not be present after getString() returns */  
    return str;  
}  
  
int main()  
{  
    printf("%s", getString());  
    getchar();  
    return 0;  
}  
chevron\_right  
filter\_none
```

Please write comments if you find anything incorrect in the above article, or you want to share more information about storage of strings

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes [arrow_drop_up](#)

Add your personal notes
here! (max 5000 chars)

Save

Recommended Posts:

- [Will we ever run out of cloud storage?](#)
- [Storage Classes in C](#)
- [Storage of integer and character values in C](#)
- [C | Storage Classes and Type Qualifiers | Question 19](#)

- C | Storage Classes and Type Qualifiers | Question 19
- C | Storage Classes and Type Qualifiers | Question 18
- C | Storage Classes and Type Qualifiers | Question 14
- C | Storage Classes and Type Qualifiers | Question 19
- C | Storage Classes and Type Qualifiers | Question 19
- C | Storage Classes and Type Qualifiers | Question 19
- C | Storage Classes and Type Qualifiers | Question 19
- C | Storage Classes and Type Qualifiers | Question 6
- C | Storage Classes and Type Qualifiers | Question 7
- C | Storage Classes and Type Qualifiers | Question 8
- C | Storage Classes and Type Qualifiers | Question 3

Improved By : Akanksha_Rai

Article Tags :

Articles

C

C Array and String

Practice Tags :

C



68

To-do Done

3

Based on 133 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

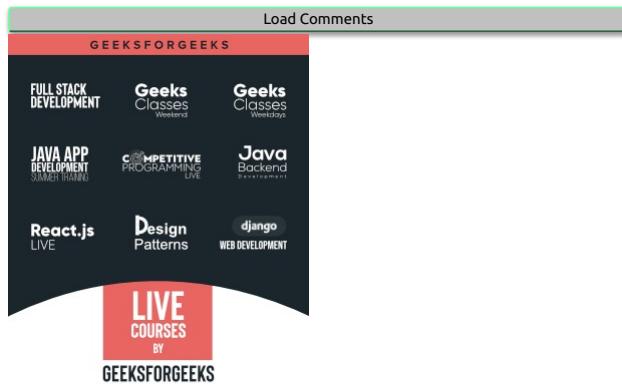
Previous

[first_page](#) Difference Between malloc() and calloc() with Examples

Next

[last_page](#) An Uncommon representation of array elements

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in Articles
[Count ways to reach the Nth stair using multiple 1 or 2 steps and a single step 3](#)
[new vs malloc\(\) and free\(\) vs delete in C++](#)
[Dockerizing a simple Django app](#)
[How to build a simple music player app using Android Studio](#)
[Transportation Problem | Set 6 \(MODI Method - UV Method\)](#)

Most visited in C
[P\(i\) in C/C++ with Examples](#)
[Sprintf/Calloc executions with help of Pragma in C/C++](#)
[Modulo Operator \(%\) in C/C++ with Examples](#)
[Program to calculate Electricity Bill](#)
[Role of SemiColon in various Programming Languages](#)

Difference between pointer and array in C?

Pointers are used for storing address of dynamically allocated arrays and for arrays which are passed as arguments to functions. In other contexts, arrays and pointer are two different things, see the following programs to justify this statement.

Behavior of sizeof operator

C

```
filter_none
edit
close

play_arrow
link
brightness_4
code

// 1st program to show that array and pointers are different
#include <stdio.h>

int main()
{
```

```

int arr[] = {10, 20, 30, 40, 50, 60};
int *ptr = arr;

// sizeof(int) * (number of element in arr[]) is printed
printf("Size of arr[] %ld\n", sizeof(arr));

// sizeof a pointer is printed which is same for all type
// of pointers (char *, void *, etc)
printf("Size of ptr %ld", sizeof(ptr));
return 0;
}
chevron_right
filter_none

```

C++

```

filter_none
edit
close

play_arrow
link
brightness_4
code

// 1st program to show that array and pointers are different
#include <iostream>
using namespace std;

int main() {
    int arr[] = {10, 20, 30, 40, 50, 60};
    int *ptr = arr;

    // sizeof(int) * (number of element in arr[]) is printed
    cout << "Size of arr[] << sizeof(arr)<<\n";

    // sizeof a pointer is printed which is same for all type
    // of pointers (char *, void *, etc)
    cout << "Size of ptr "<< sizeof(ptr);
    return 0;
}
chevron_right
filter_none

```

Output:

```

Size of arr[] 24
Size of ptr 8

```

Assigning any address to an array variable is not allowed.

```

filter_none
edit
close

play_arrow
link
brightness_4
code

// 2nd program to show that array and pointers are different
#include <stdio.h>

int main()
{
    int arr[] = {10, 20}, x = 10;
    int *ptr = &x; // This is fine
    arr = &x; // Compiler Error
    return 0;
}
chevron_right

```

Output:

```

Compiler Error: incompatible types when assigning to
type 'int[2]' from type 'int *'

```

See the [previous post](#) on this topic for more differences.

Although array and pointer are different things, following properties of array make them look similar.

1. Array name gives address of first element of array.
Consider the following program for example.

C

```

filter_none
edit
close

play_arrow
link
brightness_4
code

```

```
#include <stdio.h>

int main()
{
    int arr[] = {10, 20, 30, 40, 50, 60};
    // Assigns address of array to ptr
    int *ptr = arr;
    printf("Value of first element is %d", *ptr);
    return 0;
}
chevron_right
filter_none
```

C++

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// 1st program to show that array and pointers are different
#include <iostream>
using namespace std;

int main() {
    int arr[] = {10, 20, 30, 40, 50, 60};

    // Assigns address of array to ptr
    int *ptr = arr;
    cout << "Value of first element is " << *ptr;
    return 0;
}
chevron_right
filter_none
```

Output:

```
Value of first element is 10
```

2. Array members are accessed using pointer arithmetic.

Compiler uses pointer arithmetic to access array element. For example, an expression like "arr[i]" is treated as *(arr + i) by the compiler. That is why the expressions like *(arr + i) work for array arr, and expressions like ptr[i] also work for pointer ptr.

C

```
filter_none
edit
close
play_arrow
link
brightness_4
code

#include <stdio.h>

int main()
{
    int arr[] = {10, 20, 30, 40, 50, 60};
    int *ptr = arr;
    printf("arr[2] = %d\n", arr[2]);
    printf("*(arr + 2) = %d\n", *(arr + 2));
    printf("ptr[2] = %d\n", ptr[2]);
    printf("*(ptr + 2) = %d\n", *(ptr + 2));
    return 0;
}
chevron_right
filter_none
```

C++

```
filter_none
edit
close
play_arrow
link
brightness_4
code

#include <iostream>
using namespace std;

int main() {
    int arr[] = {10, 20, 30, 40, 50, 60};
    int *ptr = arr;
    cout << "arr[2] = " << arr[2] << "\n";
    cout << "*(arr + 2) = " << *(arr + 2) << "\n";
```

```
cout << "ptr[2] = "<< ptr[2]<< "\n";
cout << *(ptr + 2) = "<< *(ptr + 2)<< "\n";
return 0;
}
```

chevron_right

filter_none

Output:

```
arr[2] = 30
*(arr + 2) = 30
ptr[2] = 30
*(ptr + 2) = 30
```

3. Array parameters are always passed as pointers, even when we use square brackets.

filter_none

edit

close

play_arrow

link

brightness_4

code

```
#include <stdio.h>
```

```
int fun(int ptr[])
{
    int x = 10;

    // size of a pointer is printed
    printf("sizeof(ptr) = %d\n", sizeof(ptr));

    // This allowed because ptr is a pointer, not array
    ptr = &x;

    printf("*ptr = %d ", *ptr);

    return 0;
}
```

// Driver code

```
int main()
{
    int arr[] = {10, 20, 30, 40, 50, 60};
    fun(arr);
    return 0;
}
```

chevron_right

filter_none

Output:

```
sizeof(ptr) = 8
*ptr = 10
```

Please refer [Pointer vs Array in C](#) for more details.

This article is contributed by **Abhay Rathi**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes *arrow_drop_up*

Add your personal notes
here! (max 5000 chars)

Save

Recommended Posts:

- Difference between pointer to an array and array of pointers
- [Pointer vs Array in C](#)
- [Pointer to an Array | Array Pointer](#)
- Sum of array using pointer arithmetic
- Double Pointer (Pointer to Pointer) in C
- What is a Pointer to a Null pointer
- Difference between Structure and Array in C
- 'this' pointer in C++
- C++ | this pointer | Question 3
- void pointer in C / C++
- C++ | this pointer | Question 1
- NULL pointer in C
- C++ | this pointer | Question 2
- Opaque Pointer
- C++ | this pointer | Question 4

Improved By : [rambabu](#)

Article Tags :

C
C Array and String
cpp-array
cpp-pointer

Practice Tags :

C

thumb_up

77

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

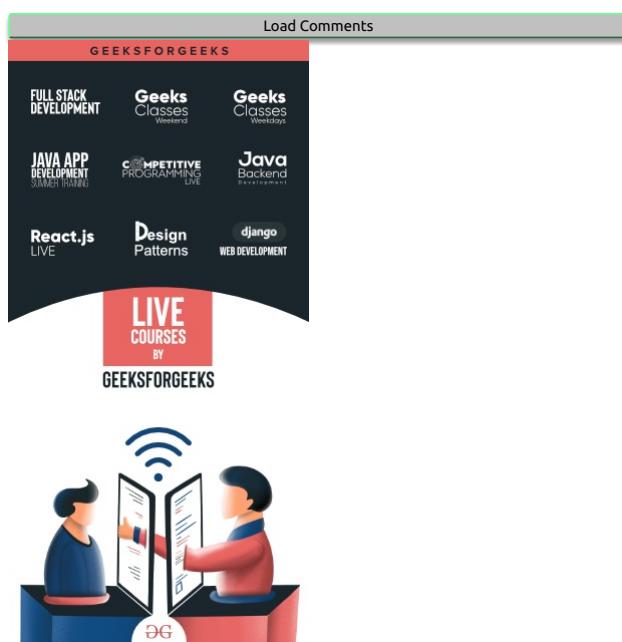
Post navigation

Previous [first_page Commonly Asked C Programming Interview Questions | Set 2](#)

Next

[last_page C Language Introduction](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



How to dynamically allocate a 2D array in C?

Following are different ways to create a 2D array on heap (or dynamically allocate a 2D array).

In the following examples, we have considered 'r' as number of rows, 'c' as number of columns and we created a 2D array with r = 3, c = 4 and following values

```
1 2 3 4
5 6 7 8
9 10 11 12
```

1) Using a single pointer:

A simple way is to allocate memory block of size r*c and access elements using simple pointer arithmetic.

```
filter_none
edit
close

play_arrow

link
brightness_4
code

#include <stdio.h>
#include <stdlib.h>

int main()
{
    int r = 3, c = 4;
    int *arr = (int *)malloc(r * c * sizeof(int));

    int i, j, count = 0;
    for (i = 0; i < r; i++)
        for (j = 0; j < c; j++)
            *(arr + i*c + j) = ++count;

    for (i = 0; i < r; i++)
        for (j = 0; j < c; j++)
            printf("%d ", *(arr + i*c + j));

    /* Code for further processing and free the
     * dynamically allocated memory */

    return 0;
}
```

chevron_right

filter_none

Output:

```
1 2 3 4 5 6 7 8 9 10 11 12
```

2) Using an array of pointers

We can create an array of pointers of size r. Note that from C99, C language allows variable sized arrays. After creating an array of pointers, we can dynamically allocate memory for every row.

filter_none

edit

close

play_arrow

link

brightness_4

code

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
int main()
```

```
{
```

```
    int r = 3, c = 4, i, j, count;
```

```
    int *arr[r];
```

```
    for (i=0; i<r; i++)
```

```
        arr[i] = (int *)malloc(c * sizeof(int));
```

```
// Note that arr[i][j] is same as *((arr+i)+j)
```

```
    count = 0;
```

```
    for (i = 0; i < r; i++)
```

```
        for (j = 0; j < c; j++)
```

```
            arr[i][j] = ++count; // Or *((arr+i)+j) = ++count
```

```
    for (i = 0; i < r; i++)
```

```
        for (j = 0; j < c; j++)
```

```
            printf("%d ", arr[i][j]);
```

```
/* Code for further processing and free the
```

```
dynamically allocated memory */
```

```
    return 0;
```

chevron_right

filter_none

Output:

```
1 2 3 4 5 6 7 8 9 10 11 12
```

3) Using pointer to a pointer

We can create an array of pointers also dynamically using a double pointer. Once we have an array pointers allocated dynamically, we can dynamically allocate memory and for every row like method 2.

filter_none

edit

close

play_arrow

link

brightness_4

code

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
int main()
```

```
{
```

```
    int r = 3, c = 4, i, j, count;
```

```
    int **arr = (int **)malloc(r * sizeof(int *));
```

```
    for (i=0; i<r; i++)
```

```
        arr[i] = (int *)malloc(c * sizeof(int));
```

```
// Note that arr[i][j] is same as *((arr+i)+j)
```

```
    count = 0;
```

```
    for (i = 0; i < r; i++)
```

```
        for (j = 0; j < c; j++)
```

```
            arr[i][j] = ++count; // OR *((arr+i)+j) = ++count
```

```
    for (i = 0; i < r; i++)
```

```
        for (j = 0; j < c; j++)
```

```
            printf("%d ", arr[i][j]);
```

```
/* Code for further processing and free the
```

```
dynamically allocated memory */
```

```
    return 0;
```

chevron_right

filter_none

Output:

```
1 2 3 4 5 6 7 8 9 10 11 12
```

4) Using double pointer and one malloc call

filter_none
edit
close
play_arrow
link
brightness_4
code

```
#include<stdio.h>
#include<stdlib.h>
```

```
int main()
{
    int r=3, c=4, len=0;
    int *ptr, **arr;
    int count = 0,i,j;

    len = sizeof(int *) * r + sizeof(int) * c * r;
    arr = (int **)malloc(len);

    // ptr is now pointing to the first element in of 2D array
    ptr = (int *)(arr + r);

    // for loop to point rows pointer to appropriate location in 2D array
    for(i = 0; i < r; i++)
        arr[i] = (ptr + c * i);

    for (i = 0; i < r; i++)
        for (j = 0; j < c; j++)
            arr[i][j] = ++count; // OR *(*(arr+i)+j) = ++count

    for (i = 0; i < r; i++)
        for (j = 0; j < c; j++)
            printf("%d ", arr[i][j]);

    return 0;
}
```

chevron_right

filter_none

Output:

1 2 3 4 5 6 7 8 9 10 11 12

Thanks to [Trishansh Bhardwaj](#) for suggesting this 4th method.

This article is contributed by **Abhay Rathi**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes *arrow_drop_up*

Add your personal notes
here! (max 5000 chars)

Save

Recommended Posts:

- [How to make a C++ class whose objects can only be dynamically allocated?](#)
- [Jagged Array or Array of Arrays in C with Examples](#)
- [Difference between pointer to an array and array of pointers](#)
- [array::front\(\) and array::back\(\) in C++ STL](#)
- [array::fill\(\) and array::swap\(\) in C++ STL](#)
- [array::crbegin\(\) and array::crend\(\) in C++ STL](#)
- [array::cbegin\(\) and array::cend\(\) in C++ STL](#)
- [array::rbegin\(\) and array::rend\(\) in C++ STL](#)
- [What's difference between "array" and "&array" for "int array\[5\]" ?](#)
- [array::begin\(\) and array::end\(\) in C++ STL](#)
- [array::at\(\) in C++ STL](#)
- [Sum of an array using MPI](#)
- [STD::array in C++](#)
- [array::max_size\(\) in C++ STL](#)
- [All permutations of an array using STL in C++](#)

Improved By : [NileshAwate](#), [dnccdd](#)

Article Tags :

C
C Array and String
cpp-array
cpp-pointer

Practice Tags :

C

thumb_up
38

To-do Done
3.5

Based on 107 vote(s)

Basic **Easy** **Medium** **Hard** **Expert**

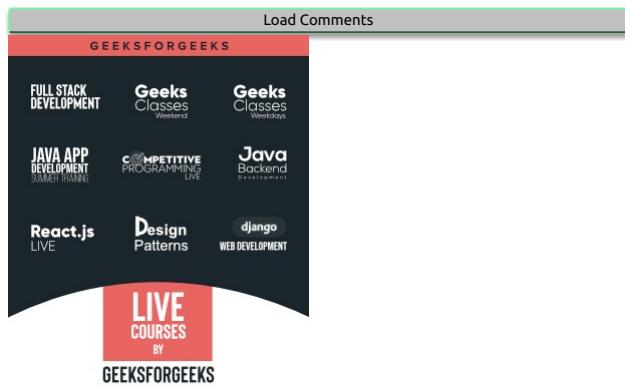
Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

first_page Convert a floating point number to string in C

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C
Print all combinations of the string by replacing 's' with any other digit from the string
getchar() function in C with Examples
Jagged Array or Array of Arrays in C with Examples
C program to display month by month calendar for a given year
ctype.h <ctype> library in C/C++ with Examples

More related articles in C
How to use make utility to build C projects?
Implicit Type Conversion in C with Examples
Introduction to the C99 Programming Language : Part I
C program to count number of vowels and consonants in a String
Hello World Program : First program while learning Programming

How to pass a 2D array as a parameter in C?

This post is an extension of [How to dynamically allocate a 2D array in C?](#)

A one dimensional array can be easily passed as a pointer, but syntax for passing a 2D array to a function can be difficult to remember. One important thing for passing multidimensional arrays is, first array dimension does not have to be specified. The second (and any subsequent) dimensions must be given

1) When both dimensions are available globally (either as a macro or as a global constant).

```
filter_none
edit
close
play_arrow
link
brightness_4
code
#include <stdio.h>
const int M = 3;
const int N = 3;

void print(int arr[M][N])
{
    int i, j;
    for (i = 0; i < M; i++)
        for (j = 0; j < N; j++)
            printf("%d ", arr[i][j]);
}

int main()
{
    int arr[][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
    print(arr);
    return 0;
}
```

chevron_right

filter_none

Output:

1 2 3 4 5 6 7 8 9

2) When only second dimension is available globally (either as a macro or as a global constant).

```
filter_none
edit
close
play_arrow
link
brightness_4
code
#include <stdio.h>
const int N = 3;
```

```

void print(int arr[][N], int m)
{
    int i, j;
    for (i = 0; i < m; i++)
        for (j = 0; j < N; j++)
            printf("%d ", arr[i][j]);
}

int main()
{
    int arr[][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
    print(arr, 3);
    return 0;
}

```

chevron_right

filter_none

Output:

```
1 2 3 4 5 6 7 8 9
```

The above method is fine if second dimension is fixed and is not user specified. The following methods handle cases when second dimension can also change.

3) If compiler is C99 compatible

From C99, C language supports variable sized arrays to be passed simply by specifying the variable dimensions (See [this](#) for an example run)

filter_none

edit

close

play_arrow

link

brightness_4

code

```
// The following program works only if your compiler is C99 compatible.
#include <stdio.h>
```

```
// n must be passed before the 2D array
void print(int m, int n, int arr[][n])
{
    int i, j;
    for (i = 0; i < m; i++)
        for (j = 0; j < n; j++)
            printf("%d ", arr[i][j]);
}
```

```
int main()
{
    int arr[][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
    int m = 3, n = 3;
    print(m, n, arr);
    return 0;
}
```

chevron_right

filter_none

Output on a C99 compatible compiler:

```
1 2 3 4 5 6 7 8 9
```

If compiler is not C99 compatible, then we can use one of the following methods to pass a variable sized 2D array.

4) Using a single pointer

In this method, we must typecast the 2D array when passing to function.

filter_none

edit

close

play_arrow

link

brightness_4

code

```
#include <stdio.h>
void print(int *arr, int m, int n)
{
    int i, j;
    for (i = 0; i < m; i++)
        for (j = 0; j < n; j++)
            printf("%d ", *((arr+i*n) + j));
}

int main()
{
    int arr[][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
    int m = 3, n = 3;

    // We can also use "print(&arr[0][0], m, n);"
    print((int *)arr, m, n);
    return 0;
}
```

chevron_right

filter_none

Output:

```
1 2 3 4 5 6 7 8 9
```

References:

My Personal Notes [arrow_drop_up](#)

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- How to print size of array parameter in C++?
- How to pass an array by value in C ?
- Parameter Passing Techniques in C/C++
- Difference between Argument and Parameter in C/C++ with Examples
- When do we pass arguments by reference or pointer?
- Jagged Array or Array of Arrays in C with Examples
- Difference between pointer to an array and array of pointers
- `array::front()` and `array::back()` in C++ STL
- `array::rbegin()` and `array::rend()` in C++ STL
- `array::cbegin()` and `array::cend()` in C++ STL
- `array::crbegin()` and `array::crend()` in C++ STL
- `array::fill()` and `array::swap()` in C++ STL
- What's difference between "array" and "&array" for "int array[5]" ?
- `array::begin()` and `array::end()` in C++ STL
- Sum of an array using MPI

Improved By : [BabisSarantoglou](#)

Article Tags :

C
C Array and String
cpp-array
cpp-parameter-passing
cpp-pointer

Practice Tags :
C

40

To-do Done
3.1

Based on 60 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) [How to dynamically allocate a 2D array in C?](#)

Next

[last_page](#) [Can virtual functions be inlined?](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Weekend Geeks Classes Weekend

JAVA APP DEVELOPMENT SUMMER TRAINING COMPETITIVE PROGRAMMING LIVE Java Backend Part-time

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS



Most popular in C
Print all possible combinations of the string by replacing 'S' with any other digit from the string
getch() function in C with Examples
Jagged Array or Array of Arrays in C with Examples
C program to display month by month calendar for a given year
ctype.h(<cctype>) library in C/C++ with Examples

More related articles in C
How to use make utility to build C projects?
Implicit Type Conversion in C with Examples
Introduction to the C99 Programming Language : Part I
Hello World Program : First program while learning Programming
How to call function within function in C or C++

How to write long strings in Multi-lines C/C++?

Image a situation where we want to use or print a long long string in C or C++, how to do this?

In C/C++, we can break a string at any point in the middle using two double quotes in the middle. Below is a simple example to demonstrate the same.

```
filter_none
edit
close

play_arrow

link
brightness_4
code

#include<stdio.h>
int main()
{
    // We can put two double quotes anywhere in a string
    char *str1 = "geeks""quiz";

    // We can put space line break between two double quotes
    char *str2 = "Qeeks" "Quiz";
    char *str3 = "Qeeks"
                 "Quiz";

    puts(str1);
    puts(str2);
    puts(str3);

    puts("Geeks"      // Breaking string in multiple lines
         "forGeeks");
    return 0;
}
```

chevron_right

filter_none

Output:
geeksquiz
QeeksQuiz
QeeksQuiz
GeeksforGeeks

Below are few examples with long long strings broken using two double quotes for better readability.

```
filter_none
edit
close

play_arrow

link
brightness_4
code

#include<stdio.h>
int main()
{
    char *str = "These are reserved words in C language are int, float, "
               "if, else, for, while etc. An Identifier is a sequence of"
               "letters and digits, but must start with a letter. "
               "Underscore ( _ ) is treated as a letter. Identifiers are "
               "case sensitive. Identifiers are used to name variables,"
               "functions etc.';

    puts(str);
    return 0;
}
```

chevron_right

filter_none

Output: These are reserved words in C language are int, float, if, else, for, while etc. An Identifier is a sequence of letters and digits, but must start with a letter. Underscore (_) is treated as a letter. Identifiers are case sensitive. Identifiers are used to name variables,functions etc.

Similarly, we can write long strings in printf and or cout.

```
filter_none
edit
close

play_arrow

link
brightness_4
code

#include<stdio.h>
int main()
{
    char *str = "An Identifier is a sequence of"
               "letters and digits, but must start with a letter. "
               "Underscore ( _ ) is treated as a letter. Identifiers are "
               "case sensitive. Identifiers are used to name variables,"
               "functions etc.';

    printf ("These are reserved words in C language are int, float, "
           "if, else, for, while etc. %s ", str);
    return 0;
}
```

chevron_right

filter_none

Output: These are reserved words in C language are int, float, if, else, for, while etc. An Identifier is a sequence of letters and digits, but must start with a letter. Underscore (_) is treated as a letter. Identifiers are case sensitive. Identifiers are used to name variables,functions etc.

My Personal Notes

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- Write your own strlen() for a long string padded with '\0's
- Number of common base strings for two strings
- Print a long int in C using putchar() only
- Is there any need of "long" data type in C and C++?
- Write a URL in a C++ program
- Working of Keyword long in C programming
- When should we write our own copy constructor?
- Write your own memcpy() and memmove()
- How to write your own header file in C?
- When should we write our own assignment operator in C++?
- Write a C program that won't compile in C++
- Read/Write structure to a file in C
- Write a program that produces different results in C and C++
- How to write a running C code without main()?
- C program to write an image in PGM format

Improved By : [rahul1642](#), [nidhi_biet](#)

Article Tags :

C
C++
C Array and String
cpp-string

Practice Tags :

C
CPP

 5

To-do Done
2.3

Based on 13 vote(s)

Basic **Easy** **Medium** **Hard** **Expert**

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) Function Pointer in C

Next

[last_page](#) Comment in header file name?

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT **Geeks Classes** front-end **Geeks Classes** front-end

JAVA APP DEVELOPMENT COMPETITIVE PROGRAMMING LIVE Java Backend development

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
Format specifiers in different Programming Languages
C program to find square root of a given number
Predefined Macros in C with Examples
How to call function within function in C or C++

Most visited in C+++
Vector of Vectors in C++ STL with Examples
C++ Tutorial
Which C++ libraries are useful for competitive programming?
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

What are the data types for which it is not possible to create an array?

In C, it is possible to have array of all types except following.

- 1) void.
- 2) functions.

For example, below program throws compiler error

```
int main()
{
    void arr[100];
}
```

Output:

```
error: declaration of 'arr' as array of voids
```

But we can have array of void pointers and function pointers. The below program works fine.

```
int main()
{
    void *arr[100];
}
```

See examples of function pointers for details of array function pointers.

This article is contributed by **Shiva**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes 

Add your personal notes
here! (max 5000 chars)

Recommended Posts:

- Difference between fundamental data types and derived data types
- Data Types in C
- C | Data Types | Question 1
- C | Data Types | Question 2
- C | Data Types | Question 9
- C | Data Types | Question 6
- C | Data Types | Question 5
- User defined Data Types in C++
- Uninitialized primitive data types in C/C++
- Calculate range of data types using C++
- Interesting facts about data-types and modifiers in C/C++
- Linking Files having same variables with different data types in C
- What happen when we exceed valid range of built-in data types in C++?
- How to print range of basic data types without any library function and constant in C?
- How to create a dynamic 2D array inside a class in C++ ?

Article Tags :

C
C Array and String
cpp-array
cpp-data-types

Practice Tags :


 10

To-do Done
1.8

Based on 30 vote(s)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) Comment in header file name?

Next

[last_page](#) Arrays in C/C++

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
Format specifiers in different Programming Languages

Predefined Macros in C with Examples
How to call function within function in C or C++
C program to print odd line contents of a file followed by even line content

More related articles in C
C program to find square root of a given number
Introduction to the C99 Programming Language : Part II
Problem in Comparing Floating point Numbers and how to compare them correctly?
Features of C Programming Language
Getting System and Process Information Using C Programming and Shell in Linux

Variable Length Arrays in C and C++

Variable length arrays is a feature where we can allocate an auto array (on stack) of variable size. C supports variable sized arrays from C99 standard. For example, the below program compiles and runs fine in C.

Also note that in C99 or C11 standards, there is feature called "flexible array members", which works same as the above.

```
filter_none
edit
close
play_arrow
link
brightness_4
code
void fun(int n)
{
    int arr[n];
    // .....
}
int main()
{
    fun(6);
}
chevron_right
filter_none
```

But C++ standard (till C++11) doesn't support variable sized arrays. The C++11 standard mentions array size as a constant-expression See (See 8.3.4 on page 179 of N3337). So the above program may not be a valid C++ program. The program may work in GCC compiler, because GCC compiler provides an extension to support them.

As a side note, the latest C++14 (See 8.3.4 on page 184 of N3690) mentions array size as a simple expression (not constant-expression).

Implementation

```
filter_none
edit
close
play_arrow
link
brightness_4
code
//C program for variable length members in structures in GCC before C99.
#include<string.h>
#include<stdio.h>
#include<stdlib.h>

//A structure of type student
struct student
{
    int stud_id;
    int name_len;
    int struct_size;
    char stud_name[0]; //variable length array must be last.
};

//Memory allocation and initialisation of structure
struct student *createStudent(struct student *s, int id, char a[])
{
    s = malloc( sizeof(*s) + sizeof(char) * strlen(a) );
    s->
```

```

s->stud_id = id;
s->name_len = strlen(a);
strcpy(s->stud_name, a);

s->struct_size = ( sizeof(*s) + sizeof(char) * strlen(s->stud_name) );

return s;
}

// Print student details
void printStudent(struct student *s)
{
    printf("Student_id : %d\n"
           "Stud_Name : %s\n"
           "Name_Length: %d\n"
           "Allocated_Struct_size: %d\n\n",
           s->stud_id, s->stud_name, s->name_len, s->struct_size);

    //Value of Allocated_Struct_size here is in bytes.
}

//Driver Code
int main()
{
    struct student *s1, *s2;

    s1=createStudent(s1, 523, "Sanjayulsha");
    s2=createStudent(s2, 535, "Cherry");

    printStudent(s1);
    printStudent(s2);

    //size in bytes
    printf("Size of Struct student: %lu\n", sizeof(struct student));
    //size in bytes
    printf("Size of Struct pointer: %lu", sizeof(s1));

    return 0;
}

```

chevron_right

filter_none

Output:

```

Student_id : 523
Stud_Name : Sanjayulsha
Name_Length: 11
Allocated_Struct_size: 23

Student_id : 535
Stud_Name : Cherry
Name_Length: 6
Allocated_Struct_size: 18

Size of Struct student: 12
Size of Struct pointer: 8

```

References:

<http://stackoverflow.com/questions/1887097/variable-length-arrays-in-c>
<https://gcc.gnu.org/onlinedocs/gcc/Variable-Length.html>

This article is contributed by **Abhay Rathi** and **Sanjay Kanna**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes **arrow_drop_up**

Add your personal notes
here! (max 5000 chars)

Save

Recommended Posts:

- [Variable Length Argument in C](#)
- [Variable length arguments for Macros](#)
- [Can we access global variable if there is a local variable with same name?](#)
- [Internal static variable vs. External static variable with Examples in C](#)
- [How to print a variable name in C?](#)
- [Redeclaration of global variable in C](#)
- [Different ways to initialize a variable in C/C++](#)
- [Why variable name does not start with numbers in C ?](#)
- [How to modify a const variable in C?](#)
- [Using a variable as format specifier in C](#)
- [How to Count Variable Numbers of Arguments in C?](#)
- [C | Variable Declaration and Scope | Question 8](#)
- [Different ways to declare variable as constant in C and C++](#)
- [C | Variable Declaration and Scope | Question 7](#)
- [C | Variable Declaration and Scope | Question 6](#)

Article Tags :

[C](#)
[C++](#)
[C Array and String](#)
[cpp-array](#)
Practice Tags :
[C](#)
[CPP](#)

thumb_up

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.
Post navigation
Previous
[first_page Functions in C/C++](#)
Next
[last_page sizeof operator in C](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

[Load Comments](#)

GEEKSFORGEEKS

FULL STACK DEVELOPMENT

Geeks Classes Weekend

Geeks Classes Weekdays

JAVA APP DEVELOPMENT

COMPETITIVE PROGRAMMING LVL

Java Backend

React.js LIVE

Design Patterns

django WEB DEVELOPMENT



LIVE COURSES BY GEEKSFORGEEKS

Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
Format specifiers in different Programming Languages
Predefined Macros in C with Examples
C program to print odd line contents of a File followed by even line content
C program to find square root of a given number

Most visited in C++
Vector of Vectors in C++ STL with Examples
C++ iterator
Which C++ libraries are useful for competitive programming?
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

A shorthand array notation in C for repeated values

In C, when there are many repeated values, we can use a shorthand array notation to define array. Below program demonstrates same.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// C program to demonstrate working of shorthand
// array rotation.
#include <stdio.h>

int main()
{
    // This line is same as
    // int array[10] = {1, 1, 1, 1, 0, 0, 2, 2, 2, 2};

    int array[10] = {[0 ... 3]=1, [6 ... 9]=2};

    for (int i = 0; i < 10; i++)
        printf("%d ", array[i]);
    return 0;
}
chevron_right
filter_none
```

Output:

```
1 1 1 0 0 2 2 2 2
```

Note that middle gap of 2 is automatically filled with 0.

This article is contributed by **Kaushik Annangi**. If you like GeeksforGeeks and would like to contribute, you can also write an article using [contribute.geeksforgeeks.org](#) or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes [arrow_drop_up](#)

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- C function argument and return values
- How to return multiple values from a function in C or C++?
- Storage of integer and character values in C
- Return values of printf() and scanf() in C/C++
- What are the default values of static variables in C?
- How can I return multiple values from a function?
- std::tuple, std::pair | Returning multiple values from a function using Tuple and Pair in C++
- Jagged Array or Array of Arrays in C with Examples
- Difference between pointer to an array and array of pointers
- What's difference between "array" and "&array" for "int array[5]" ?
- Sum of an array using MPI
- 4 Dimensional Array in C/C++
- How to pass an array by value in C ?
- Array class in C++
- Pointer to an Array | Array Pointer

Article Tags :

C

C-array

Practice Tags :

C



5

To-do Done
1.6

Based on 6 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

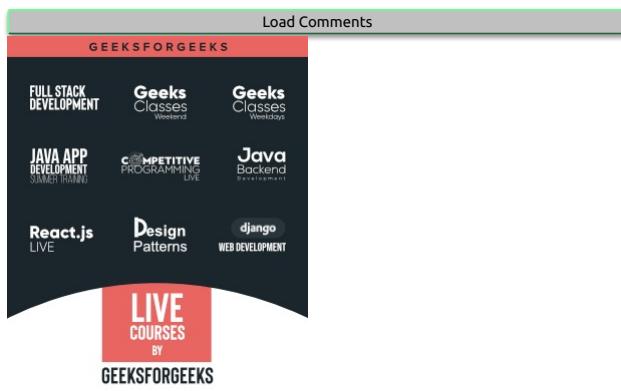
Previous

[first_page](#) What is the size_t data type in C?

Next

[last_page](#) How to compile 32-bit program on 64-bit gcc in C and C++

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C
Print all possible combinations of the string by replacing 's' with any other digit from the string
Predefined Macros in C with Examples
Format specifiers in different Programming Languages
C program to print odd line contents of a File followed by even line content
Introduction to the C99 Programming Language : Part II

More related articles in C
C program to find square root of a given number
Problems comparing Floating point numbers and how to compare them correctly
Features of C Programming Language
Pointer Expressions in C with Examples
Introduction to the C99 Programming Language : Part III

Accessing array out of bounds in C/C++

Prerequisite : [Arrays in C/C++](#)

In high level languages such as Java, there are functions which prevent you from accessing array out of bound by generating an exception such as `java.lang.ArrayIndexOutOfBoundsException`. But in case of C, there is no such functionality, so programmer need to take care of this situation.

What if programmer accidentally accesses any index of array which is out of bound ?

C doesn't provide any specification which deal with problem of accessing invalid index. As per ISO C standard it is called **Undefined Behavior**.

An undefined behavior (UB) is a result of executing computer code whose behavior is not prescribed by the language specification to which the code can adhere to, for the current state of the program (e.g. memory). This generally happens when the translator of the source code makes certain assumptions, but these assumptions are not satisfied during execution.

Examples of Undefined Behavior while accessing array out of bounds

1. **Access non allocated location of memory:** The program can access some piece of memory which is owned by it.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// Program to demonstrate
// accessing array out of bounds
#include <stdio.h>
int main()
{
    int arr[] = {1,2,3,4,5};
    printf("arr [0] is %d\n", arr[0]);

    // arr[10] is out of bound
    printf("arr[10] is %d\n", arr[10]);
    return 0;
}
chevron_right
```

filter_none

Output :

```
arr [0] is 1
arr[10] is -1786647872
```

It can be observed here, that arr[10] is accessing a memory location containing a garbage value.

2. **Segmentation fault:** The program can access some piece of memory which is not owned by it, which can cause crashing of program such as segmentation fault.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// Program to demonstrate
// accessing array out of bounds
#include <stdio.h>
int main()
{
    int arr[] = {1,2,3,4,5};
    printf("arr [0] is %d\n",arr[0]);
    printf("arr[10] is %d\n",arr[10]);

    // allocation memory to out of bound
    // element
    arr[10] = 11;
    printf("arr[10] is %d\n",arr[10]);
    return 0;
}
chevron_right
```

filter_none

Output :

Runtime Error : Segmentation Fault (SIGSEGV)

Important Points:

- Stay inside the bounds of the array in C programming while using arrays to avoid any such errors.
- C++ however offers the `std::vector` class template, which does not require to perform bounds checking. A vector also has the `std::at()` member function which can perform bounds-checking.

This article is contributed by **Mandeep Singh**. If you like GeeksforGeeks and would like to contribute, you can also write an article using [contribute.geeksforgeeks.org](#) or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes 

Add your personal notes
here! (max 5000 chars)

Recommended Posts:

- Last element of vector in C++ (Accessing and updating)
- Output of C programs | Set 66 (Accessing Memory Locations)
- Namespace in C++ | Set 3 (Accessing, creating header, nesting and aliasing)
- Difference between pointer to an array and array of pointers
- Jagged Array or Array of Arrays in C with Examples
- `array::front()` and `array::back()` in C++ STL
- `array::rbegin()` and `array::rend()` in C++ STL
- `array::fill()` and `array::swap()` in C++ STL
- `array::crbegin()` and `array::crend()` in C++ STL
- `array::cbegin()` and `array::cend()` in C++ STL
- What's difference between "array" and "&array" for "int array[5]" ?
- `array::begin()` and `array::end()` in C++ STL
- Sum of an array using MPI
- Array sum in C++ STL
- `array::at()` in C++ STL

Article Tags :

C
C++
c-array
cpp-array
Practice Tags :
C
CPP

thumb_up
7

To-do Done
1.6

Based on 9 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

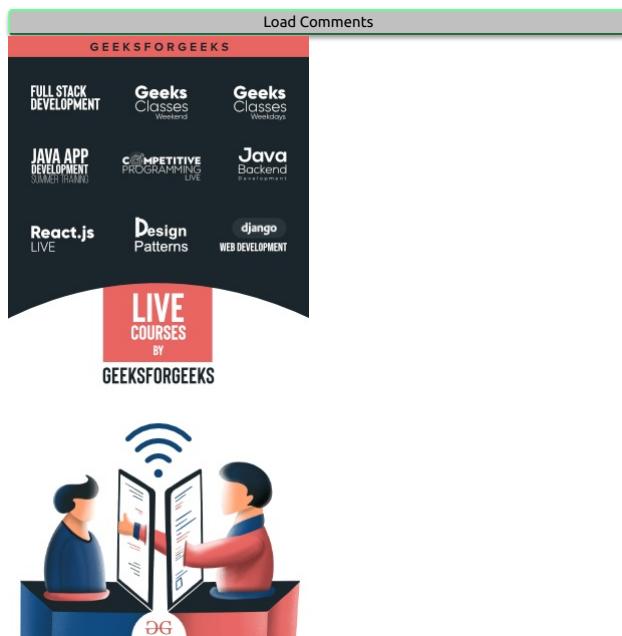
Previous

[first_page](#) Understanding ShellExecute function and it's application to open a list of URLs present in a file using C++ code

Next

[last_page](#) rotate in C++ STL

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C
Print all possible combinations of the string by replacing 'S' with any other digit from the string
Program to print Matrix with Examples
C program to print odd line contents of a File followed by even line content
Introduction to the C99 Programming Language : Part II
C program to find square root of a given number

Most visited in C++
Vector of Vectors in C++ STL with Examples
C++ Tutorial
Which C++ libraries are useful for competitive programming?
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

strcpy in C/C++

strcpy() is a standard library function in C/C++ and is used to copy one string to another. In C it is present in **string.h** header file and in C++ it is present in **cstring** header file.

Syntax:

```
char* strcpy(char* dest, const char* src);
```

Parameters: This method accepts following parameters:

- **dest:** Pointer to the destination array where the content is to be copied.
- **src:** string which will be copied.

Return Value: After copying the source string to the destination string, the strcpy() function returns a pointer to the destination string.

Below program explains different usages of this library function:

```
filter_none
edit
close

play_arrow

link
brightness_4
code

// C program to illustrate
// strcpy() function in C/C++
#include<stdio.h>
#include<string.h>

int main ()
{
    char str1[]="Hello Geeks!";
    char str2[] = "GeeksforGeeks";
    char str3[40];
```

```

char str4[40];
char str5[] = "GfG";

strcpy(str2, str1);
strcpy(str3, "Copy successful");
strcpy(str4, str5);
printf ("str1: %s\nstr2: %s\nstr3: %s\nstr4: %s\n", str1, str2, str3, str4);
return 0;
}

```

chevron_right

filter_none

Output:

```

str1: Hello Geeks!
str2: Hello Geeks!
str3: copy successful
str4: GfG

```

Important Points

- This function copies the entire string to the destination string. It doesn't append the source string to the destination string. In other words, we can say that it replaces the content of destination string by the content of source string.
- It does not affect the source string. The source string remains same after copying.
- This function only works with C style strings and not C++ style strings i.e. it only works with strings of type **char str[];** and not **string s1;** which are created using standard string data type available in C++ and not C.

This article is contributed by **Harsh Agarwal**. If you like GeeksforGeeks and would like to contribute, you can also write an article using [contribute.geeksforgeeks.org](#) or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes *arrow_drop_up*

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- Why strcpy and strncpy are not safe to use?
- C program to copy string without using strcpy() function
- Rust vs C++: Will Rust Replace C++ in Future ?
- Priority queue of pairs in C++ with ordering by first and second element
- Implementation of lower_bound() and upper_bound() in Vector of Pairs in C++
- Generating RGBA portable graphic images through C++
- std::basic_istream::ignore in C++ with Examples
- std::basic_istream::getline in C++ with Examples
- Format specifiers in different Programming Languages
- C program to find square root of a given number
- C program to print odd line contents of a File followed by even line content
- std::is_heap() in C++ with Examples
- std::basic_istream::gcount() in C++ with Examples
- new vs malloc() and free() vs delete in C++

Article Tags :

C
C++
C-String
cpp-string
Practice Tags :
C
CPP

thumb_up
16

To-do Done
2.5

Based on 9 vote(s)

Basic **Easy** **Medium** **Hard** **Expert**

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

first_page OpenGL program for simple Animation (Revolution) in C

Next

last_page Virtual Function in C++

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
Format specifiers in different Programming Languages
C program to find square root of a given number
Predefined Macros in C with Examples
C program to print odd line contents of a File followed by even line content

Most visited in C++
Vector of Vectors in C++ STL with Examples
C++ STL with Examples
Which C++ libraries are useful for competitive programming?
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

strcmp() in C/C++

strcmp() is a built-in library function and is declared in `<string.h>` header file. This function takes two strings as arguments and compare these two strings lexicographically.

Syntax::

```
int strcmp(const char *leftStr, const char *rightStr);
```

In the above prototype, function strcmp takes two strings as parameters and returns an integer value based on the comparison of strings.

- strcmp() compares the **two strings lexicographically** means it starts comparison character by character starting from the first character until the characters in both strings are equal or a NULL character is encountered.
- If first character in both strings are equal, then this function will check the second character, if this is also equal then it will check the third and so on
- This process will be continued until a character in either string is NULL or the characters are unequal.

What does strcmp() return?

This function can return **three different integer values** based on the comparison:

1. **Zero (0)**: A value equal to zero when both strings are found to be identical. That is, All of the characters in both strings are same.

All characters of strings are same

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// C program to illustrate
// strcmp() function
#include<stdio.h>
#include<string.h>

int main()
{
    char leftStr[] = "g f g";
    char rightStr[] = "g f g";

    // Using strcmp()
    int res = strcmp(leftStr, rightStr);

    if (res==0)
        printf("Strings are equal");
    else
        printf("Strings are unequal");

    printf("\nValue returned by strcmp() is: %d", res);
    return 0;
}
```

chevron_right

filter_none

Output:

```
Strings are equal
Value returned by strcmp() is: 0
```

2. **Greater than zero (>0)**: A value greater than zero is returned when the first not matching character in leftStr have the greater ASCII value than the corresponding character in rightStr or we can also say

If character in leftStr is lexicographically

after the character **of** rightStr

filter_none
edit
close
play_arrow
link
brightness_4
code

```
// C program to illustrate
// strcmp() function
#include<stdio.h>
#include<string.h>
int main()
{
    // z has greater ASCII value than g
    char leftStr[] = "zfg";
    char rightStr[] = "gfg";

    int res = strcmp(leftStr, rightStr);

    if (res==0)
        printf("Strings are equal");
    else
        printf("Strings are unequal");

    printf("\nValue of result: %d", res);

    return 0;
}
```

chevron_right

filter_none

Output:

```
Strings are unequal
Value returned by strcmp() is: 19
```

3. **Less than Zero (<0)**: A value less than zero is returned when the first not matching character in leftStr have lesser ASCII value than the corresponding character in rightStr.

If character in leftStr is lexicographically before the character of rightStr

filter_none
edit
close
play_arrow
link
brightness_4
code

```
// C program to illustrate
// strcmp() function
#include<stdio.h>
#include<string.h>
int main()
{
    // b has less ASCII value than g
    char leftStr[] = "bfb";
    char rightStr[] = "gfg";

    int res = strcmp(leftStr, rightStr);

    if (res==0)
        printf("Strings are equal");
    else
        printf("Strings are unequal");

    printf("\nValue returned by strcmp() is: %d", res);

    return 0;
}
```

chevron_right

filter_none

Output:

```
Strings are unequal
Value returned by strcmp() is: -5
```

Important point : When the strings are not same, you will find that the value returned by the strcmp() function is the difference between the ASCII values of first unmatched character in leftStr and rightStr in both the cases.

This article is contributed by **Harsh Agarwal**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes **arrow_drop_up**

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- Difference between `strcmp()` and `strncpy()` in C/C++
- Write one line functions for `strcat()` and `strcmp()`
- `std::basic_istream::ignore` in C++ with Examples
- `std::basic_istream::getline` in C++ with Examples
- Format specifiers in different Programming Languages
- C program to find square root of a given number
- C program to print odd line contents of a File followed by even line content
- `std::is_heap()` in C++ with Examples
- `std::basic_istream::gcount()` in C++ with Examples
- new vs `malloc()` and `free()` vs `delete` in C++
- `std::string::rfind` in C++ with Examples
- Sum of factorials of Prime numbers in a Linked list
- Sum of all Palindrome Numbers present in a Linked list
- Generate an array of given size with equal count and sum of odd and even numbers

Article Tags :

C
C++
C-String
cpp-string

Practice Tags :

C
CPP



17

To-do Done

2

Based on 11 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) [Fork\(\)](#) Bomb

Next

[last_page](#) Integer literal in C/C++ (Prefixes and Suffixes)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

[Load Comments](#)



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
Predefined Macros in C with Examples
C program to print odd line contents of a File followed by even line content
Introduction to the C99 Programming Language : Part II
C program to find square root of a given number

Most visited in C++
Vector of Vectors in C++ STL with Examples
C++ Thread API
Which C++ libraries are useful for competitive programming?
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

strupr() and strndup() functions in C/C++

The `strupr()` and `strndup()` functions are used to duplicate a string.

`strupr()` :

Syntax : `char *strupr(const char *s);`

This function returns a pointer to a null-terminated byte string, which is a duplicate of the string pointed to by `s`. The memory obtained is done dynamically using `malloc` and hence it can be freed using `free()`. It returns a pointer to the duplicated string `s`.

Below is the C implementation to show the use of `strupr()` function in C:

[filter_none](#)
[edit](#)
[close](#)

```
play_arrow
link
brightness_4
code

// C program to demonstrate strdup()
#include<stdio.h>
#include<string.h>

int main()
{
    char source[] = "GeeksForGeeks";

    // A copy of source is created dynamically
    // and pointer to copy is returned.
    char* target = strdup(source);

    printf("%s", target);
    return 0;
}
```

chevron_right

filter_none

Output:

GeeksForGeeks

strndup():

syntax: `char *strndup(const char *s, size_t n);`

This function is similar to strdup(), but copies at most **n** bytes.

Note: If s is longer than n, then only n bytes are copied, and a NULL ("") is added at the end.

Below is the C implementation to show the use of strndup() function in C:

filter_none

edit

close

play_arrow

link

brightness_4

code

```
// C program to demonstrate strndup()
#include<stdio.h>
#include<string.h>
```

```
int main()
{
    char source[] = "GeeksForGeeks";
```

```
    // 5 bytes of source are copied to a new memory
    // allocated dynamically and pointer to copied
    // memory is returned.
    char* target = strndup(source, 5);
```

```
    printf("%s", target);
    return 0;
}
```

chevron_right

filter_none

Output:

Geeks

Reference: [Linux man\(7\)](#)

This article is contributed by **MAZHAR IMAM KHAN**. If you like GeeksforGeeks and would like to contribute, you can also write an article using [contribute.geeksforgeeks.org](#) or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up

Add your personal notes
here! (max 5000 chars)

Save

Recommended Posts:

- [Functions in C/C++](#)
- [Nested functions in C](#)
- [C++ Mathematical Functions](#)
- [Searching in a map using std::map functions in C++](#)
- [C | Functions | Question 6](#)
- [Inline Functions in C++](#)
- [C | Functions | Question 5](#)
- [C | Functions | Question 4](#)
- [C | Functions | Question 3](#)
- [C | Functions | Question 2](#)
- [C | Functions | Question 1](#)
- [Macros vs Functions](#)
- [List in C++ | Set 2 \(Some Useful Functions\)](#)
- [Pure Functions](#)
- [Functions that cannot be overloaded in C++](#)

Article Tags :

C
C++
C-Library
CPP-Library
Practice Tags :
C
CPP

3

To-do Done
2.5

Based on 12 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

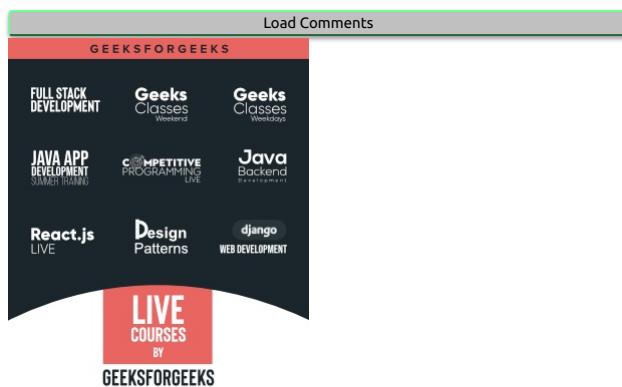
Previous

[first_page](#) Complex numbers in C++ | Set 2

Next

[last_page](#) C program to find Decagonal Number

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
Format specifier in different programming Languages
Prefetching Macros in C with Examples
C program to print odd line contents of a file followed by even line content
C program to find square root of a given number

Most visited in C++
Vector of Vectors in C++ STL with Examples
C++ Tutorial
Which C++ libraries are useful for competitive programming?
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

How to pass an array by value in C ?

In C, array name represents address and when we pass an array, we actually pass address and the parameter receiving function always accepts them as pointers (even if we use [], refer [this](#) for details).

How to pass array by value, i.e., how to make sure that we have a new copy of array when we pass it to function?

This can be done by wrapping the array in a structure and creating a variable of type of that structure and assigning values to that array. After that, passing the variable to some other function and modifying it as per requirements. Note that **array members are copied when passed as parameter, but dynamic arrays are not**. So this solution works only for non-dynamic arrays (created without new or malloc).

Let's see an example to demonstrate the above fact using a C program:

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// C program to demonstrate passing an array
// by value using structures.
#include<stdio.h>
#include<stdlib.h>

#define SIZE 5

// A wrapper for array to make sure that array
// is passed by value.
struct ArrayWrapper
{
    int arr[SIZE];
};
```

```

// An array is passed by value wrapped in temp
void modify(struct ArrayWrapper temp)
{
    int *ptr = temp.arr;
    int i;

    // Display array contents
    printf("In 'modify()', before modification\n");
    for (i = 0; i < SIZE; ++i)
        printf("%d ", ptr[i]);

    printf("\n");

    // Modify the array
    for (i = 0; i < SIZE; ++i)
        ptr[i] = 100; // OR *(ptr + i)

    printf("\nIn 'modify()', after modification\n");
    for (i = 0; i < SIZE; ++i)
        printf("%d ", ptr[i]); // OR *(ptr + i)
}

// Driver code
int main()
{
    int i;
    struct ArrayWrapper obj;
    for (i=0; i<SIZE; i++)
        obj.arr[i] = 10;

    modify(obj);

    // Display array contents
    printf("\n\nIn 'Main', after calling modify() \n");
    for (i = 0; i < SIZE; ++i)
        printf("%d ", obj.arr[i]); // Not changed

    printf("\n");

    return 0;
}

```

[chevron_right](#)

[filter_none](#)

Output:

```

In 'modify()', before modification
10 10 10 10 10

In 'modify()', after modification
100 100 100 100 100

In 'Main', after calling modify()
10 10 10 10 10

```

Reference:

<http://stackoverflow.com/questions/11158858/c-pass-array-by-value>

This article is contributed by **MAZHAR IMAM KHAN**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes [arrow_drop_up](#)

Add your personal notes
here! (max 5000 chars)

[Save](#)

Recommended Posts:

- [How to pass a 2D array as a parameter in C?](#)
- [When do we pass arguments by reference or pointer?](#)
- [Difference between pointer to an array and array of pointers](#)
- [Jagged Array or Array of Arrays in C with Examples](#)
- [What's difference between "array" and "&array" for "int array\[5\]"?](#)
- [Sum of an array using MPI](#)
- [Array class in C++](#)
- [Pointer vs Array in C](#)
- [4 Dimensional Array in C/C++](#)
- [Pointer to an Array | Array Pointer](#)
- [Difference between Structure and Array in C](#)
- [Why array index starts from zero ?](#)
- [Array Type Manipulation in C++](#)
- [What is Array Decay in C++? How can it be prevented?](#)
- [C program to traverse an Array](#)

Article Tags :

[C](#)

[C-Arrays](#)

Practice Tags :

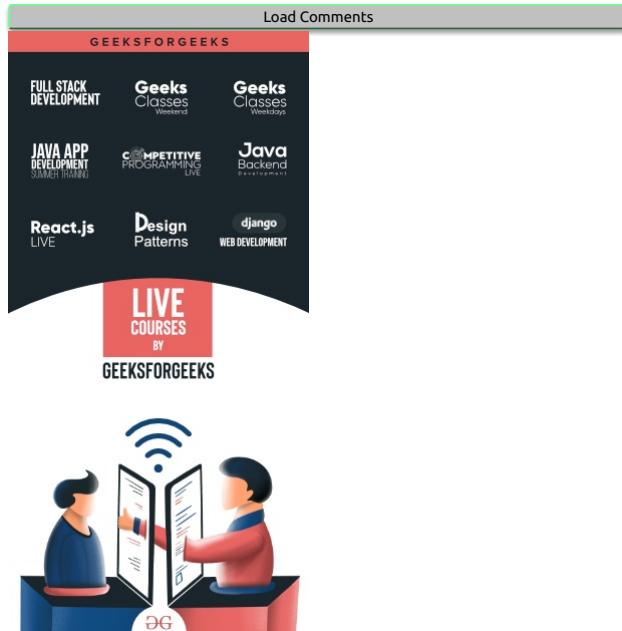
[C](#)



9

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.
Post navigation
Previous
[first_page](#) Pointers in C and C++ | Set 1 (Introduction, Arithmetic and Array)
Next
[last_page](#) Inbuilt library functions for user Input | scanf, fscanf, sscanf, scanf_s, fscanf_s, sscanf_s

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
Predefined Macros in C with Examples
Format Specifiers in different Programming Languages
C program to print odd line contents of a File followed by even line content
Introduction to the C99 Programming Language : Part II

More related articles in C
C program to find square root of a given number
Problem in comparing Floating point numbers and how to compare them correctly?
Features of C Programming Language
Pointer Expressions in C with Examples
Introduction to the C99 Programming Language : Part III

Different methods to reverse a string in C/C++

Given a string, write a C/C++ program to reverse it.

Input : s = "abc"
Output : s = "cba"

Input : s = "geeksforgeeks"
Output : s = "skeegrofskeeg"

1. **Write own reverse function by swapping characters:** One simple solution is to write our own reverse function to reverse a string in C++.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// A Simple C++ program to reverse a string
#include <bits/stdc++.h>
using namespace std;

// Function to reverse a string
void reverseStr(string& str)
{
    int n = str.length();

    // Swap character starting from two
    // corners
    for (int i = 0; i < n / 2; i++)
        swap(str[i], str[n - i - 1]);
}

// Driver program
int main()
{
    string str = "geeksforgeeks";
    reverseStr(str);
    cout << str;
    return 0;
}
```

```
}
```

chevron_right

```
filter_none
```

Output :

```
skeegrofskeeg
```

2. **Using inbuilt "reverse" function:** There is a direct function in "algorithm" header file for doing reverse that saves our time when programming.

```
// Reverses elements in [begin, end]
void reverse (BidirectionalIterator begin,
BidirectionalIterator end);
```

filter_none
edit
close

play_arrow

link
brightness_4
code

// A quickly written program for reversing a string
// using reverse()
#include <bits/stdc++.h>
using namespace std;
int main()
{
 string str = "geeksforgeeks";

 // Reverse str[begin..end]
 reverse(str.begin(), str.end());

 cout << str;
 return 0;
}

chevron_right

```
filter_none
```

Output :

```
skeegrofskeeg
```

3. **Only printing reverse:**

filter_none
edit
close

play_arrow

link
brightness_4
code

// C++ program to print reverse of a string
#include <bits/stdc++.h>
using namespace std;

// Function to reverse a string
void reverse(string str)
{
 for (int i=str.length()-1; i>=0; i--)
 cout << str[i];
}

Driver code

```
int main(void)
{
    string s = "GeeksforGeeks";
    reverse(s);
    return (0);
}
```

chevron_right

```
filter_none
```

Output :

```
skeegrofskeeg
```

4. **Getting reverse of a const string:**

filter_none
edit
close

play_arrow

link
brightness_4
code

// C++ program to get reverse of a const string
#include <bits/stdc++.h>
using namespace std;

// Function to reverse string and return
// reverse string pointer of that
char* reverseConstString(char const* str)
{

```

// find length of string
int n = strlen(str);

// create dynamic pointer char array
char *rev = new char[n+1];

// copy of string to ptr array
strcpy(rev, str);

// Swap character starting from two
// corners
for (int i=0, j=n-1; i<j; i++,j--)
    swap(rev[i], rev[j]);

// return pointer of reversed string
return rev;
}

// Driver code
int main(void)
{
    const char *s = "GeeksforGeeks";
    printf("%s", reverseConstString(s));
    return (0);
}

```

chevron_right

filter_none

Output:

skeeGrofskeeG

This article is contributed by **Priyam kakati, Ranju Kumari, Somesh Awasthi** and improved by **Supratik Mitra**. If you like GeeksforGeeks and would like to contribute, you can also write an article and mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes *arrow_drop_up*

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- Create a new string by alternately combining the characters of two halves of the string in reverse
- 5 Different methods to find length of a string in C++
- Methods to concatenate string in C/C++ with Examples
- Taking String input with space in C (3 Different Methods)
- Count consonants in a string (Iterative and recursive methods)
- Print all distinct characters of a string in order (3 Methods)
- PHP | Reverse a String
- Reverse the given string in the range [L, R]
- Reverse words in a given string
- Reverse a string in Java
- Reverse vowels in a given string
- Reverse words in a given String in Python
- Reverse middle words of a string
- Reverse every word of the string except the first and the last character
- Add index to characters and reverse the string

Improved By : Omkar Goulay, supratik_mitra

Article Tags :

C
C++
School Programming
Strings

cpp-string

Reverse

STL

Practice Tags :

Strings

STL

Reverse

C

CPP

thumb_up

35

To-do Done
2.2

Based on 53 vote(s)

Basic **Easy** **Medium** **Hard** **Expert**

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) [getchar_unlocked\(\)](#) – faster input in C/C++ for Competitive Programming

Next

[last_page](#) [std::transform\(\)](#) in C++ + STL (Perform an operation on all elements)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



**LIVE
COURSES**
BY
GEEKSFORGEEKS



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
Format specifier in different Programming Languages
Prefetch Macro in C with Examples
C program to print odd line contents of a File followed by even line content
C program to find square root of a given number

Most visited in C++
Vector of Vectors in C++ STL with Examples
C++ Tutorial
Which C++ libraries are useful for competitive programming?
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

strpbrk() in C

This function **finds the first character in the string s1 that matches any character specified in s2** (it excludes terminating null-characters).

Syntax :
char *strpbrk(const char *s1, const char *s2)

Parameters :
s1 : **string** to be scanned.
s2 : **string** containing the characters to match.

Return Value :
It returns a pointer to the character **in s1** that matches one **of** the characters **in s2**, **else** returns NULL.

filter_none

edit

close

play_arrow

link

brightness_4

code

// C code to demonstrate the working of
// strpbrk

```
#include <stdio.h>
#include <string.h>
```

// Driver function

int main()

{

// Declaring three strings

```
char s1[] = "geeksforgeeks";
char s2[] = "app";
char s3[] = "kite";
char* r, *t;
```

// Checks for matching character

// no match found

```
r = strpbrk(s1, s2);
```

if (r != 0)

```
printf("First matching character: %c\n", *r);
```

else

```
printf("Character not found");
```

// Checks for matching character

// first match found at "e"

```
t = strpbrk(s1, s3);
```

if (t != 0)

```
printf("\nFirst matching character: %c\n", *t);
```

else

```
printf("Character not found");
```

return (0);

}

chevron_right

filter_none

Output:

```
Character not found
```

First matching character: e

Practical Application

This function can be used in game of lottery where the person having string with letter coming first in victory wins, i.e. this can be used at any place where first person wins.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// C code to demonstrate practical application
// of strpbrk

#include <stdio.h>
#include <string.h>

// Driver function
int main()
{
    // Initializing victory string
    char s1[] = "victory";

    // Declaring lottery strings
    char s2[] = "a23";
    char s3[] = "i22";
    char* r, *t;

    // Use of strpbrk()
    r = strpbrk(s1, s2);
    t = strpbrk(s1, s3);

    // Checks if player 1 has won lottery
    if (r != 0)
        printf("Congrats u have won");
    else
        printf("Better luck next time");

    // Checks if player 2 has won lottery
    if (t != 0)
        printf("\nCongrats u have won");
    else
        printf("Better luck next time");

    return (0);
}
```

chevron_right

filter_none

Output:

```
Better luck next time
Congrats u have won
```

This article is contributed by **Shantanu Singh**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up

Save

Recommended Posts:

- [Rust vs C++: Will Rust Replace C++ in Future?](#)
- [Priority queue of pairs in C++ with ordering by first and second element](#)
- [Implementation of lower_bound\(\) and upper_bound\(\) in Vector of Pairs in C++](#)
- [Generating RGBA portable graphic images through C++](#)
- [std::basic_istream::ignore in C++ with Examples](#)
- [std::basic_istream::getline in C++ with Examples](#)
- [Format specifiers in different Programming Languages](#)
- [C program to find square root of a given number](#)
- [C program to print odd line contents of a File followed by even line content](#)
- [std::is_heap\(\) in C++ with Examples](#)
- [std::basic_istream::gcount\(\) in C++ with Examples](#)
- [new vs malloc\(\) and free\(\) vs delete in C++](#)
- [std::string::rfind in C++ with Examples](#)
- [Sum of factorials of Prime numbers in a Linked list](#)

Improved By : SACHIN KUNTE

Article Tags :

C

C++

C-Library

cpp-string

Practice Tags :

C

CPP

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

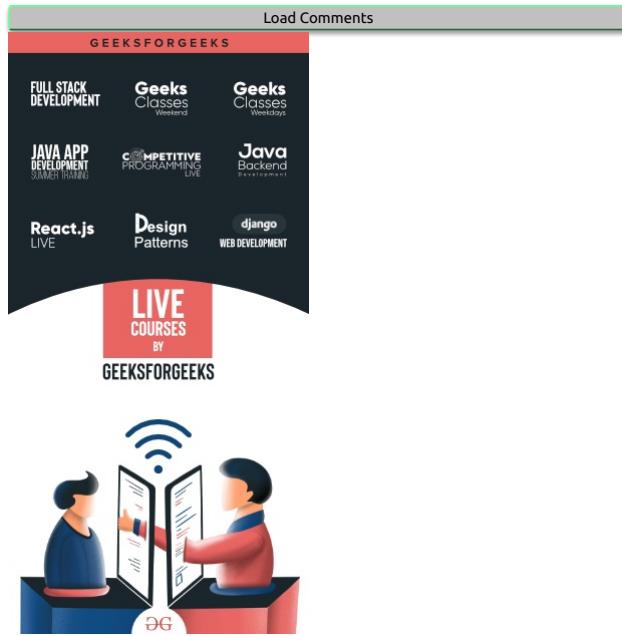
Previous

[first_page](#) [stringstream in C++ and its applications](#)

Next

[last_page](#) [difftime\(\) C library function](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C
C program to display month by month calendar for a given year
Print all possible combinations of the string by replacing '\$' with any other digit from the string
ctype (<cctype.h>) library in C/C++ With Examples
Implicit Type Conversion in C With Examples
Format specifiers in different Programming Languages

Most visited in C++
Vector of Vectors in C++ STL with Examples
C++ Tutorial
Which C++ libraries are useful for competitive programming?
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

strcoll() in C/C++

strcoll() is a built-in library function and is declared in `<string.h>` header file. This function compares the string pointed to by `str1` with the one pointed by `str2`. The **strcoll()** function performs the comparison based on the rules of the current locale's **LC_COLLATE** category.

Syntax:

```
int strcoll(const char *str1, const char *str2)
```

Parameters: Function `strcoll()` takes two strings as parameters and returns an integer value.

Value	Meaning
less than zero	<code>str1</code> is less than <code>str2</code>
zero	<code>str1</code> is equal to <code>str2</code>
greater than zero	<code>str1</code> is greater than <code>str2</code>

1. **less than zero** : When `str1` is less than `str2`

`filter_none`
`edit`
`close`

`play_arrow`

`link`
`brightness_4`
`code`

// C program to illustrate strcoll()

```
#include <stdio.h>
#include <string.h>
```

```
int main()
{
    char str1[10];
    char str2[10];
    int ret;
```

```
strcpy(str1, "geeksforgeeks");
strcpy(str2, "GEEKSFORGEEKS");
```

```
ret = strcoll(str1, str2);
```

```
if (ret > 0) {
    printf("str1 is greater than str2");
} else if (ret < 0) {
    printf("str1 is lesser than str2");
```

```
    } else {
        printf("str1 is equal to str2");
    }

    return (0);
}
```

chevron_right

filter_none

Output:

str1 is greater than str2

2. **greater than zero** :when str1 is greater than str2

filter_none

edit

close

play_arrow

link

brightness_4

code

```
// C program to illustrate strcoll()
#include <stdio.h>
#include <string.h>
```

```
int main()
{
    char str1[10];
    char str2[10];
    int ret;
```

```
    strcpy(str1, "GEEKSFORGEEKS");
    strcpy(str2, "geeksforgeeks");

    ret = strcoll(str1, str2);
```

```
    if (ret > 0) {
        printf("str1 is greater than str2");
    } else if (ret < 0) {
        printf("str1 is lesser than str2");
    } else {
        printf("str1 is equal to str2");
    }
```

```
    return (0);
}
```

chevron_right

filter_none

Output:

str1 is lesser than str2

3. **Is equal to zero** : when str1 is equal to str2

filter_none

edit

close

play_arrow

link

brightness_4

code

```
// C program to illustrate strcoll()
```

```
#include <stdio.h>
#include <string.h>
```

```
int main()
{
```

```
    char str1[10];
    char str2[10];
    int ret;
```

```
    strcpy(str1, "GEEKSFORGEEKS");
    strcpy(str2, "GEEKSFORGEEKS");

    ret = strcoll(str1, str2);
```

```
    if (ret > 0) {
        printf("str1 is greater than str2");
    } else if (ret < 0) {
        printf("str1 is lesser than str2");
    } else {
        printf("str1 is equal to str2");
    }
```

```
    return (0);
}
```

chevron_right

filter_none

Output:

str1 is equal to str2

Related Function : `strcmp()`, `memcmp()`

This article is contributed by **Shivani Ghugtyal**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes [arrow_drop_up](#)

Add your personal notes here! (max 5000 chars)

[Save](#)**Recommended Posts:**

- Rust vs C++: Will Rust Replace C++ in Future ?
- Priority queue of pairs in C++ with ordering by first and second element
- Implementation of `lower_bound()` and `upper_bound()` in Vector of Pairs in C++
- Generating RGBA portable graphic images through C++
- `std::basic_istream::ignore` in C++ with Examples
- `std::basic_istream::getline` in C++ with Examples
- Format specifiers in different Programming Languages
- C program to find square root of a given number
- C program to print odd line contents of a File followed by even line content
- `std::is_heap()` in C++ with Examples
- `std::basic_istream::gcount()` in C++ with Examples
- new vs malloc() and free() vs delete in C++
- `std::string::rfind` in C++ with Examples
- Sum of factorials of Prime numbers in a Linked list

Improved By : [kevray](#), [dvas325](#)**Article Tags :**

C
C++
C-Library
C-String
CPP-Library
cpp-string
Practice Tags :
C
CPP

[thumb_up](#)
4

To-do Done
2

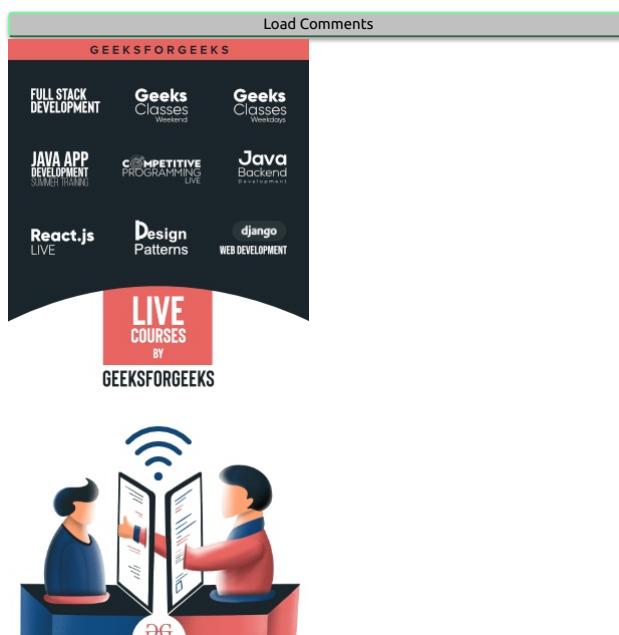
Based on 3 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation
Previous
[first_page](#) 2D vector in C++ with user defined size
Next
[last_page](#) `std::get_temporary_buffer` in C++

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



ispunct() function in C

The **ispunct()** function checks whether a character is a punctuation character or not.

The term "**punctuation**" as defined by this function includes all printable characters that are neither alphanumeric nor a space. For example '@', '\$', etc.

This function is defined in **ctype.h** header file.

syntax:

```
int ispunct(int ch);  
ch: character to be checked.  
Return Value : function return nonzero  
if character is a punctuation character;  
otherwise zero is returned.
```

filter_none
edit
close

play_arrow
link
brightness_4
code

```
// Program to check punctuation  
#include <stdio.h>  
#include <ctype.h>  
int main()  
{  
    // The punctuations in str are '!' and ','  
    char str[] = "welcome! to GeeksForGeeks, ";  
  
    int i = 0, count = 0;  
    while (str[i]) {  
        if (ispunct(str[i]))  
            count++;  
        i++;  
    }  
    printf("Sentence contains %d punctuation"  
          " characters.\n", count);  
    return 0;  
}
```

chevron_right

filter_none

Output:

```
Sentence contains 2 punctuation characters.
```

filter_none
edit
close

play_arrow
link
brightness_4
code

```
// C program to print all Punctuations  
#include <stdio.h>  
#include <ctype.h>  
int main()  
{  
    int i;  
    printf("All punctuation characters in C"  
          " programming are: \n");  
    for (i = 0; i <= 255; ++i)  
        if (ispunct(i) != 0)  
            printf("%c ", i);  
    return 0;  
}
```

chevron_right

filter_none

Output:

```
All punctuation characters in C programming are:  
! " # $ % & ' ( ) * +, - . / ; : ? @ [ \ ] ^ _ ` { | } ~
```

This article is contributed by **Shivani Ghugtya**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up

Add your personal notes
here! (max 5000 chars)

Save

Recommended Posts:

- How to call function within function in C or C++
- arc function in C
- Inline function in C
- putchar() function in C
- atexit() function in C/C++
- strcspn() function in C/C++
- strrchr() function in C/C++
- gmtime() Function in C/C++
- towupper() function in C/C++
- iswpunct() function in C/C++
- wcstoll() function in C/C++
- iswalnum() function in C++ STL
- ldepx() function in C/C++
- iswblank() function in C/C++
- strtoumax() function in C++

Article Tags :

C
C-Library
Practice Tags :

C

3

To-do Done
1.5

Based on 4 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

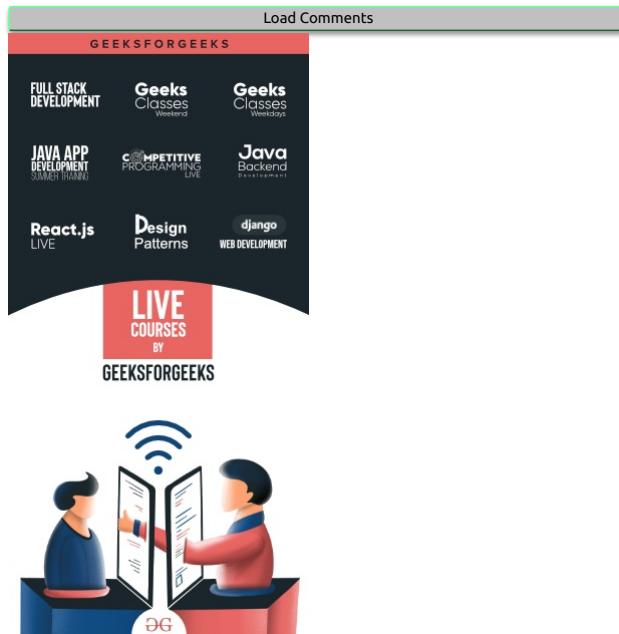
Previous

[first_page](#) C program to print a string without any quote (single or double) in the program

Next

[last_page](#) # and ## Operators in C

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
ctype.h<<ctype>> library in C/C++ with Examples
C program to display month by month calendar for a given year
Predefined Macros in C with Examples
Implicit Type Conversion in C with Examples

More related articles in C
For Loops in C with Examples
How to call function within function in C or C++
C program to print odd line contents of a File followed by even line content
Nested Loops in C with Examples
Introduction to the C99 Programming Language : Part II

strspn() function in C

The **strspn()** function returns the length of the initial substring of the string pointed to by **str1** that is made up of only those character contained in the string pointed to by **str2**.

Syntax :

```
size_t strspn(const char *str1, const char *str2)
str1 : string to be scanned.
str2 : string containing the
       characters to match.
Return Value : This function
       returns the number of characters
       in the initial segment of str1
       which consist only of characters
       from str2.
```

[filter_none](#)

[edit](#)

[close](#)

[play_arrow](#)

[link](#)

[brightness_4](#)

code

```
// C program to illustrate strspn() function
#include <stdio.h>
#include <string.h>

int main () {
    int len = strspn("geeks for geeks", "geek");
    printf("Length of initial segment matching : %d\n", len );
    return(0);
}
```

chevron_right

filter_none

Output:

```
Length of initial segment matching 4
```

filter_none

edit

close

play_arrow

link

brightness_4

code

```
// C program to illustrate strspn() function
#include <stdio.h>
#include <string.h>
```

```
int main () {
    int len = strspn("i am","xyz");
    printf("Length of initial segment matching : %d\n", len );
    return(0);
}
```

chevron_right

filter_none

Output:

```
Length of initial segment matching 0
```

This article is contributed by **Shivani Ghugtyal**. If you like GeeksforGeeks and would like to contribute, you can also write an article using [contribute.geeksforgeeks.org](#) or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- [How to call function within function in C or C++](#)
- [arc function in C](#)
- [mbstowcs\(\) function in C/C++](#)
- [strcmpi\(\) function in C](#)
- [vswprintf\(\) function in C/C++](#)
- [wcstoul\(\) function in C/C++](#)
- [wmemcmp\(\) function in C/C++](#)
- [wmemchr\(\) function in C/C++](#)
- [wcrtombs\(\) function in C/C++](#)
- [fgetenv\(\) function in C/C++](#)
- [fgetexceptflag\(\) function in C/C++](#)
- [wcsncat\(\) function in C/C++](#)
- [wcstol\(\) function in C/C++](#)
- [wcsncmp\(\) function in C/C++](#)
- [wcsrtombs\(\) function in C/C++](#)

Article Tags :

C

C-Library

C-String

Practice Tags :

C

thumb_up

4

To-do Done
1.5

Based on 4 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) Difference between while(1) and while(0) in C language

Next

[last_page](#) Anything written in sizeof() is never executed in C

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



LIVE COURSES
BY
GEEKSFORGEEKS



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
ctype.h ctype.c library in C/C++ with Examples
C program to print month by month calendar for a given year
Predefined Macros in C with Examples
Implicit Type Conversion in C with Examples

More related articles in C
Format specifiers in different Programming Languages
How to call function within function in C or C++
C program to print odd line contents of a File followed by even line content
Nested Loops in C with Examples
Introduction to the C99 Programming Language : Part II

isalpha() and isdigit() functions in C with cstring examples.

isalpha(c) is a function in C which can be used to check if the passed character is an alphabet or not. It returns a non-zero value if it's an alphabet else it returns 0. For example, it returns non-zero values for 'a' to 'z' and 'A' to 'Z' and zeroes for other characters.

Similarly, **isdigit(c)** is a function in C which can be used to check if the passed character is a digit or not. It returns a non-zero value if it's a digit else it returns 0. For example, it returns a non-zero value for '0' to '9' and zero for others.

Avoiding common errors : It is important to note this article does not cover strings! Only Cstrings. Cstrings are an array of single characters (char) in their behaviour. There are advantages and disadvantages to this.

Example Problem : Given a cstring str, find the number of alphabetic letters and number of decimal digits in that cstring.

Examples:

```
Input: 12abc12
Output: Alphabetic_letters = 3, Decimal_digits = 4

Input: 123 GeeksForGeeks is Number 1
Output: Alphabetic_letters = 21, Decimal_digits = 4
```

Recommended: Please try your approach on {IDE} first, before moving on to the solution.

Explanation And Approach:

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// C program to demonstrate working of isalpha() and
// isdigit().
#include<stdio.h>
#include<stdlib.h>

int main()
{
    char str[] = "12abc12";

    int alphabet = 0, number = 0, i;
    for (i=0; str[i]!='\0'; i++)
    {
        // check for alphabets
        if (isalpha(str[i]) != 0)
            alphabet++;

        // check for decimal digits
        else if (isdigit(str[i]) != 0)
            number++;
    }

    printf("Alphabetic_letters = %d, "
           "Decimal_digits = %d\n", alphabet, number);

    return 0;
}
```

chevron_right

Output:

```
Alphabetic_letters = 3, Decimal_digits = 4
```

This article is contributed by **Mazhar Imam Khan**. If you like GeeksforGeeks and would like to contribute, you can also write an article using [contribute.geeksforgeeks.org](#) or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes 

Add your personal notes here! (max 5000 chars)

Recommended Posts:

- [asctime\(\) and asctime_s\(\) functions in C with Examples](#)
- [Commonly used String functions in C/C++ with Examples](#)
- [strtok\(\) and strtok_r\(\) functions in C with examples](#)
- [Functions in C/C++](#)
- [Searching in a map using std::map functions in C++](#)
- [Thread functions in C/C++](#)
- [Nested functions in C](#)
- [List in C++ | Set 2 \(Some Useful Functions\)](#)
- [Inline Functions in C++](#)
- [C++ Mathematical Functions](#)
- [C | Functions | Question 11](#)
- [C | Functions | Question 10](#)
- [C | Functions | Question 9](#)
- [Pure Functions](#)
- [C | Functions | Question 7](#)

Improved By : [BrenlyDrake](#)

Article Tags :

C
C++
C-Library
CPP-Library

Practice Tags :

C
CPP

 5

To-do Done
1.8

Based on 16 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) Nested printf (printf inside printf) in C

Next

[last_page](#) Printing source code of a C program itself

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

[Load Comments](#)

GEEKSFORGEEKS

FULL STACK DEVELOPMENT

Geeks Classes WEEKEND

Geeks Classes WEEKDAYS

JAVA APP DEVELOPMENT SUMMER TRAINING

COMPETITIVE PROGRAMMING LIVE

Java Backend Development

React.js LIVE

Design Patterns

django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
Format specifiers in different Programming Languages
Predefined Macros in C with Examples
C program to print odd line contents of a file followed by even line content
C program to find square root of a given number

Most visited in C++
Vector of Vectors in C++ STL with Examples
C++ Tutorial
Which C++ libraries are useful for competitive programming?
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

Data type of case labels of switch statement in C++?

In C++ switch statement, the expression of each case label must be an integer constant expression.

For example, the following program fails in compilation.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

/* Using non-const in case label */
#include<stdio.h>
int main()
{
    int i = 10;
    int c = 10;
    switch(c)
    {
        case i: // not a "const int" expression
            printf("Value of c = %d", c);
            break;
        /*Some more cases */
    }
    return 0;
}
chevron_right
```

Putting `const` before `i` makes the above program work.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

#include<stdio.h>
int main()
{
    const int i = 10;
    int c = 10;
    switch(c)
    {
        case i: // Works fine
            printf("Value of c = %d", c);
            break;
        /*Some more cases */
    }
    return 0;
}
```

Note : The above fact is only for C++. In C, both programs produce an error. In C, using an integer literal does not cause an error.

Program to find the largest number between two numbers using switch case:

```
filter_none
edit
close
play_arrow
link
brightness_4
code

#include<stdio.h>
int main()
{
    int n1=10,n2=11;

    // n1 > n2 (10 > 11) is false so using
    // logical operator '>', n1 > n2 produces 0
    // (0 means false, 1 means true) So, case 0
    // is executed as 10 > 11 is false. Here we
    // have used type cast to convert boolean to int,
    // to avoid warning.

    switch((int)(n1 > n2))
    {
        case 0:
            printf("%d is the largest\n", n2);
            break;
        default:
            printf("%d is the largest\n", n1);
    }
}
```

```
// n1 < n2 (10 < 11) is true so using logical
// operator '<', n1 < n2 produces 1 (1 means true,
// 0 means false) So, default is executed as we
// don't have case 1 to be executed.
```

```
switch((int)(n1 < n2))
{
    case 0:
        printf("%d is the largest\n", n1);
        break;
    default:
        printf("%d is the largest\n", n2);
}

return 0;
}
```

//This code is contributed by Santanu
chevron_right

filter_none

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- [Switch Statement in C/C++](#)
- [Interesting facts about switch statement in C](#)
- [Using range in switch case in C/C++](#)
- [Nested switch case](#)
- [Output of C programs | Set 30 \(Switch Case\)](#)
- [Is there any need of "long" data type in C and C++?](#)
- [What is the size_t data type in C?](#)
- [What is data type of FILE in C ?](#)
- [Data type of character constants in C and C++](#)
- [Data Type Ranges and their macros in C++](#)
- [Conversion of Struct data type to Hex String and vice versa](#)
- [Local Labels in C](#)
- [Difference between Type Casting and Type Conversion](#)
- [Print individual digits as words without using if or switch](#)
- [goto statement in C/C++](#)

Improved By : [SantanuBasak](#), [HanishGupta1](#)

Article Tags :

C
C-Loops & Control Statements

Practice Tags :

C

thumb_up
7

To-do Done
1.8

Based on 41 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) [Initialization of a multidimensional arrays in C/C++](#)

Next

[last_page](#) [What are the default values of static variables in C?](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
Predefined Macros in C with Examples
Format specifiers in different Programming Languages
C program to print odd line contents of a File followed by even line content
C program to find square root of a given number

More related articles in C
Introduction to the C99 Programming Language : Part II
Pointers in C programming: Floating point numbers and how to compare them correctly?
Features of C Programming Language
Pointer Expressions in C with Examples
Introduction to the C99 Programming Language : Part III

For Versus While

Question: Is there any example for which the following two loops will not work same way?

```
filter_none
edit
close
play_arrow
link
brightness_4
code

/*Program 1 -> For loop*/
for (<init-stmnt>; <boolean-expr>; <incr-stmnt>)
{
    <body-statements>
}

/*Program 2 -> While loop*/
<init-stmnt>;
while (<boolean-expr>)
{
    <body-statements>
    <incr-stmnt>
}
chevron_right
filter_none
```

Solution:

If the body-statements contains continue, then the two programs will work in different ways

See the below examples: Program 1 will print "loop" 3 times but Program 2 will go in an infinite loop.

Example for program 1

```
filter_none
edit
close
play_arrow
link
brightness_4
code

int main()
{
    int i = 0;
    for(i = 0; i < 3; i++)
    {
        printf("loop ");
        continue;
    }
    getchar();
    return 0;
}
chevron_right
filter_none
```

Example for program 2

filter_none

```
edit
close
play_arrow
link
brightness_4
code

int main()
{
    int i = 0;
    while(i < 3)
    {
        printf("loop"); /* printed infinite times */
        continue;
        i++; /*This statement is never executed*/
    }
    getchar();
    return 0;
}
chevron_right
filter_none
```

Please write comments if you want to add more solutions for the above question.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes 

Add your personal notes here! (max 5000 chars)



Recommended Posts:

- [typedef versus #define in C](#)
- [Scope Resolution Operator Versus this pointer in C++?](#)
- [Format specifiers in different Programming Languages](#)
- [C program to find square root of a given number](#)
- [C program to print odd line contents of a File followed by even line content](#)
- [Getting System and Process Information Using C Programming and Shell in Linux](#)
- [Program to calculate Electricity Bill](#)
- [Program to print half Diamond star pattern](#)
- [C/C++ program for calling main\(\) in main\(\)](#)
- [Print all possible combinations of the string by replacing '\\$' with any other digit from the string](#)
- [How to use make utility to build C projects?](#)
- [How to call function within function in C or C++](#)
- [Role of Semicolon in various Programming Languages](#)
- [C program to display month by month calendar for a given year](#)

Article Tags :

[C](#)
[C-Loops & Control Statements](#)
[c-puzzle](#)

Practice Tags :

[C](#)

 9

To-do Done
1.5

Based on 35 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

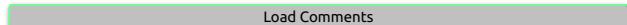
Previous

[first_page](#) [How to declare a pointer to a function?](#)

Next

[last_page](#) [Why C treats array parameters as pointers?](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.





Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
`ctype.h <ctype.h>` library in C/C++ with Examples
Predefined Macros in C with Examples
Format specifiers in different Programming Languages
How to call function within function in C or C++
More related articles in C
C program to print odd line contents of a File followed by even line content
C program to sort strings in ascending order
Introduction to the C99 Programming Language : Part II
Problem in comparing Floating point numbers and how to compare them correctly?
Features of C Programming Language

A nested loop puzzle

Which of the following two code segments is faster? Assume that compiler makes no optimizations.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

/* FIRST */
for(i=0;i<10;i++)
    for(j=0;j<100;j++)
        //do something
chevron_right
filter_none
edit
close
play_arrow
link
brightness_4
code

/* SECOND */
for(i=0;i<100;i++)
    for(j=0;j<10;j++)
        //do something
chevron_right
filter_none
```

Both code segments provide same functionality, and the code inside the two for loops would be executed same number of times in both code segments.
If we take a closer look then we can see that the SECOND does more operations than the FIRST. It executes all three parts (assignment, comparison and increment) of the for loop more times than the corresponding parts of FIRST:

1. The SECOND executes assignment operations ($j = 0$ or $i = 0$) 101 times while FIRST executes only 11 times.
2. The SECOND does $101 + 1100$ comparisons ($i < 100$ or $j < 10$) while the FIRST does $11 + 1010$ comparisons ($i < 10$ or $j < 100$).
3. The SECOND executes 1100 increment operations ($i++$ or $j++$) while the FIRST executes 1010 increment operation.

Below C++ code counts the number of increment operations executed in FIRST and SECOND, and prints the counts.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

//program to count number of increment
//operations in FIRST and SECOND
#include<iostream>

using namespace std;

int main()
{
```

```

int c1 = 0, c2 = 0;

/* FIRST */
for(int i=0; i<10; i++, c1++);
    for(int j=0; j<100; j++, c1++);
        //do something

/* SECOND */
for(int i=0; i<100; i++, c2++);
    for(int j=0; j<10; j++, c2++);
        //do something

cout << " Count in FIRST = " <<c1 << endl;
cout << " Count in SECOND = " <<c2 << endl;

getchar();
return 0;
}

```

[chevron_right](#)

[filter_none](#)

Output:

```

Count in FIRST = 1010
Count in SECOND = 1100

```

Below C++ code counts the number of comparison operations executed by FIRST and SECOND

```

filter\_none
edit
close
play\_arrow
link
brightness\_4
code

//program to count the number of comparison
//operations executed by FIRST and SECOND */
#include<iostream>

using namespace std;

int main()
{
    int c1 = 0, c2 = 0;

    /* FIRST */
    for(int i=0; ++c1&&i<10; i++)
        for(int j=0; ++c1&&j<100; j++);
            //do something

    /* SECOND */
    for(int i=0; ++c2&&i<100; i++)
        for(int j=0; ++c2&&j<10; j++);
            //do something

    cout << " Count fot FIRST " <<c1 << endl;
    cout << " Count fot SECOND " <<c2 << endl;
    getchar();
    return 0;
}

```

[chevron_right](#)

[filter_none](#)

Output:

```

Count fot FIRST 1021
Count fot SECOND 1201

```

Thanks to [Dheeraj](#) for suggesting the solution.

Please write comments if you find any of the answers/codes incorrect, or you want to share more information about the topics discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

[My Personal Notes](#) [arrow_drop_up](#)

Add your personal notes
here! (max 5000 chars)

[Save](#)

Recommended Posts:

- [Nested functions in C](#)
- [C++ | Nested Ternary Operator](#)
- [Nested Loops in C with Examples](#)
- [Nested switch case](#)
- [C/C++ For loop with Examples](#)
- [C/C++ while loop with Examples](#)
- [C/C++ do while loop with Examples](#)
- [How will you print numbers from 1 to 100 without using loop?](#)
- [Output of C programs | Set 57 \(for loop\)](#)
- [How will you print numbers from 1 to 100 without using loop? | Set-2](#)
- [Difference between while and do-while loop in C, C++, Java](#)
- [Output of C programs | Set 56 \(While loop\)](#)
- [Print 1 to 100 in C++, without loop and recursion](#)
- [How to concatenate two integer arrays without using loop in C ?](#)

- What happens if loop till Maximum of Signed and Unsigned in C/C++?

Improved By : nidhi_biet

Article Tags :

C
C-Loops & Control Statements
Practice Tags :
C

thumb_up
5

To-do Done
2.7

Based on 39 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

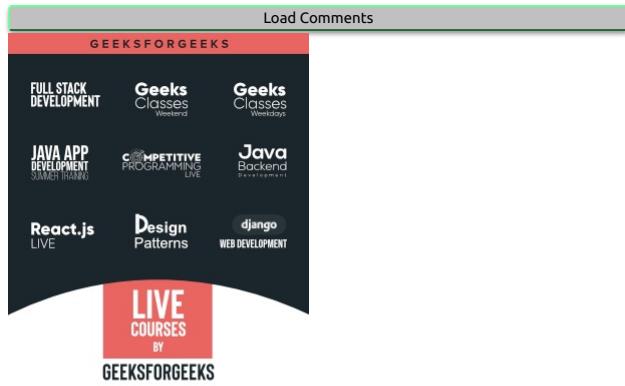
Previous

[first_page](#) How are variables scoped in C - Static or Dynamic?

Next

[last_page](#) Write one line functions for strcat() and strcmp()

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
ctype.h(<cctype>) library in C/C++ with Examples
Format specifier in different Programming Languages
Predefining Macros in C with Examples
How to call function within function in C or C++

More related articles in C
C program to print odd line contents of a file followed by even line content
C program to find square root of a given number
Introduction to the C99 Programming Language : Part II
Problem in comparing Floating point numbers and how to compare them correctly?
Features of C Programming Language

Interesting facts about switch statement in C

Prerequisite - [Switch Statement in C](#)

Switch is a control statement that allows a value to change control of execution.

filter_none
edit
close

play_arrow
link
brightness_4
code

```
// Following is a simple program to demonstrate syntax of switch.
#include <stdio.h>
int main()
{
    int x = 2;
    switch (x)
    {
        case 1: printf("Choice is 1");
        break;
        case 2: printf("Choice is 2");
        break;
        case 3: printf("Choice is 3");
        break;
        default: printf("Choice other than 1, 2 and 3");
        break;
    }
    return 0;
}
```

chevron_right

[filter_none](#)

Output:

Choice is 2

Following are some interesting facts about switch statement.

1) The expression used in switch must be integral type (int, char and enum). Any other type of expression is not allowed.

```
filter_none
edit
close

play_arrow
link
brightness_4
code

// float is not allowed in switch
#include <stdio.h>
int main()
{
    float x = 1.1;
    switch (x)
    {
        case 1.1: printf("Choice is 1");
                    break;
        default: printf("Choice other than 1, 2 and 3");
                    break;
    }
    return 0;
}
```

[chevron_right](#)

[filter_none](#)

Output:

Compiler Error: switch quantity not an integer

In Java, String is also allowed in switch (See [this](#))

2) All the statements following a matching case execute until a break statement is reached.

```
filter_none
edit
close

play_arrow
link
brightness_4
code

// There is no break in all cases
#include <stdio.h>
int main()
{
    int x = 2;
    switch (x)
    {
        case 1: printf("Choice is 1\n");
        case 2: printf("Choice is 2\n");
        case 3: printf("Choice is 3\n");
        default: printf("Choice other than 1, 2 and 3\n");
    }
    return 0;
}
```

[chevron_right](#)

[filter_none](#)

Output:

**Choice is 2
Choice is 3
Choice other than 1, 2 and 3**

```
filter_none
edit
close

play_arrow
link
brightness_4
code

// There is no break in some cases
#include <stdio.h>
int main()
{
    int x = 2;
    switch (x)
    {
        case 1: printf("Choice is 1\n");
        case 2: printf("Choice is 2\n");
        case 3: printf("Choice is 3\n");
        case 4: printf("Choice is 4\n");
                    break;
        default: printf("Choice other than 1, 2, 3 and 4\n");
                    break;
    }
    printf("After Switch");
}
```

```
    return 0;
```

```
}
```

```
chevron_right
```

```
filter_none
```

Output:

```
Choice is 2  
Choice is 3  
Choice is 4  
After Switch
```

3) The default block can be placed anywhere. The position of default doesn't matter, it is still executed if no match found.

```
filter_none
```

```
edit
```

```
close
```

```
play_arrow
```

```
link
```

```
brightness_4
```

```
code
```

```
// The default block is placed above other cases.
```

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int x = 4;  
    switch (x)  
    {  
        default: printf("Choice other than 1 and 2");  
        break;  
        case 1: printf("Choice is 1");  
        break;  
        case 2: printf("Choice is 2");  
        break;  
    }
```

```
    return 0;
```

```
}
```

```
chevron_right
```

```
filter_none
```

Output:

```
Choice other than 1 and 2
```

4) The integral expressions used in labels must be a constant expressions

```
filter_none
```

```
edit
```

```
close
```

```
play_arrow
```

```
link
```

```
brightness_4
```

```
code
```

```
// A program with variable expressions in labels
```

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int x = 2;  
    int arr[] = {1, 2, 3};  
    switch (x)  
    {  
        case arr[0]: printf("Choice 1\n");  
        case arr[1]: printf("Choice 2\n");  
        case arr[2]: printf("Choice 3\n");  
    }
```

```
    return 0;
```

```
}
```

```
chevron_right
```

```
filter_none
```

Output:

```
Compiler Error: case label does not reduce to an integer constant
```

5) The statements written above cases are never executed After the switch statement, the control transfers to the matching case, the statements written before case are not executed.

```
filter_none
```

```
edit
```

```
close
```

```
play_arrow
```

```
link
```

```
brightness_4
```

```
code
```

```
// Statements before all cases are never executed
```

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int x = 1;  
    switch (x)  
    {  
        x = x + 1; // This statement is not executed  
        case 1: printf("Choice is 1");  
        break;  
        case 2: printf("Choice is 2");  
    }
```

```
        break;
    default: printf("Choice other than 1 and 2");
        break;
}
chevron_right
```

filter_none

Output:

Choice is 1

6) Two case labels cannot have same value

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// Program where two case labels have same value
#include <stdio.h>
int main()
{
    int x = 1;
    switch (x)
    {
        case 2: printf("Choice is 1");
            break;
        case 1+1: printf("Choice is 2");
            break;
    }
    return 0;
}
```

chevron_right

filter_none

Output:

Compiler Error: duplicate case value

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- [Interesting Facts about C++](#)
- [Interesting facts about C Language](#)
- [C++ bitset interesting facts](#)
- [Interesting Facts in C Programming](#)
- [Switch Statement in C/C++](#)
- [Interesting Facts about Macros and Preprocessors in C](#)
- [Some Interesting facts about default arguments in C++](#)
- [Nested switch statement in C++](#)
- [Interesting facts about data-types and modifiers in C/C++](#)
- [Some interesting facts about static member functions in C++](#)
- [Data type of case labels of switch statement in C++?](#)
- [C++ programming and STL facts](#)
- [Nested switch case](#)
- [Using range in switch case in C/C++](#)
- [Facts and Question related to Style of writing programs in C/C++](#)

Improved By : [Yuvraj Patil](#)

Article Tags :

C
C++
C-Loops & Control Statements
cpp-switch
Practice Tags :
C
CPP

thumb_up

34

To-do Done
2.2

Based on 62 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) [C](#) | Operators | Question 27

Next

[last_page](#) [C](#) | Input and Output | Question 13

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Weekend

JAVA APP DEVELOPMENT SUMMER TRAINING COMPETITIVE PROGRAMMING LIVE

React.js LIVE Design Patterns

Geeks Classes Weekdays Java Backend Development

django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS



Most popular in C
ctype.h<ctype.h> library in C/C++ with Examples
Print all possible combinations of the string by replacing '\$' with any other digit from the string
Predefined Macros in C with Examples
How to call function within function in C or C++
C program to print odd line contents of a file followed by even line content
Most visited in C++
Vector of Vectors in C++ STL with Examples
C++ Tutorial
Which C++ libraries are useful for competitive programming?
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

Difference between while(1) and while(0) in C language

Prerequisite: [while loop in C/C++](#)

In most computer programming languages, a while loop is a control flow statement that allows code to be executed repeatedly based on a given boolean condition. The boolean condition is either true or false

while(1)

It is an infinite loop which will run till a break statement is issued explicitly. Interestingly not while(1) but any integer which is non-zero will give the similar effect as while(1). Therefore, while(1), while(2) or while(-255), all will give infinite loop only.

```
while(1) or while(any non-zero integer)
{
    // loop runs infinitely
}
```

A simple usage of while(1) can be in the Client-Server program. In the program, the server runs in an infinite while loop to receive the packets sent from the clients.

But practically, it is not advisable to use while(1) in real-world because it increases the CPU usage and also blocks the code i.e one cannot come out from the while(1) until the program is closed manually. while(1) can be used at a place where condition needs to be true always.

C

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// C program to illustrate while(1)
int main()
{
    int i = 0;
    while ( 1 )
    {
        printf( "%d\n", ++i );
        if (i == 5)
            break; // Used to come
                    // out of loop
    }
    return 0;
}
chevron_right
```

filter_none

```
filter_none
edit
close
play_arrow
link
brightness_4
code

#include <iostream>
```

C++

```
using namespace std;
```

```
int main() {
    int i = 0;
    while ( 1 )
    {
        cout << ++i << "\n";
        if (i == 5)
            // Used to come
            // out of loop
            break;
    }
    return 0;
}
```

chevron_right

filter_none

Output:

```
1  
2  
3  
4  
5
```

while(0)

It is opposite of while(1). It means condition will always be false and thus code in while will never get executed.

```
while(0)
{
    // loop does not run
}
```

C

```
filter_none
edit
close

play_arrow
link
brightness_4
code

// C program to illustrate while(0)
int main()
{
    int i = 0, flag=0;
    while ( 0 )
    {
        // This line will never get executed
        printf( "%d\n", ++i );
        flag++;
        if (i == 5)
            break;
    }
    if (flag==0)
        printf ("Didn't execute the loop!");
    return 0;
}
```

chevron_right

filter_none

C++

```
filter_none
edit
close

play_arrow
link
brightness_4
code

#include <iostream>
using namespace std;

int main() {
    int i = 0, flag=0;
    while ( 0 )
    {
        // This line will never get executed
        cout << ++i << "\n";
        flag++;
        if (i == 5)
            break;
    }
    if (flag==0)
        cout << "Didn't execute the loop!";
    return 0;
}
```

chevron_right

filter_none

Output:

```
Didn't execute the loop!
```

This article is contributed by **Anshika Goyal**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes [arrow_drop_up](#)

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- Difference Between Machine Language and Assembly Language
- Difference Between Assembly Language And Machine Language
- Difference between Java and C language
- Difference between Compiled and Interpreted Language
- Difference Between Go and Python Programming Language
- Difference between %d and %i format specifier in C language
- What is the difference between a language construct and a “built-in” function in PHP ?
- Difference between Structured Query Language (SQL) and Transact-SQL (T-SQL)
- Difference between Procedural and Non-Procedural language
- Stopwatch using C language
- fgets() and gets() in C language
- Signals in C language
- C Language Introduction
- kbhit in C language
- isupper() function in C Language

Improved By : [PrateekshaPriyadarshini](#)

Article Tags :

C
Difference Between
C-Loops & Control Statements

Practice Tags :

C

thumb_up

7

To-do Done
1.3

Based on 22 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) Difference between highest and least frequencies in an array

Next

[last_page](#) Difference between set, multiset, unordered_set, unordered_multiset

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT **Geeks Classes** Wednesday **Geeks Classes** Wednesday

JAVA APP DEVELOPMENT COMPETITIVE PROGRAMMING LIVE Java Backend Development

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS



Most popular in C
Print all possible combinations of the string by replacing '*' with any other digit from the string
Program to Merge n C with Examples
C program to print odd line contents of a File followed by even line content
Introduction to the C99 Programming Language : Part II
C program to find square root of a given number

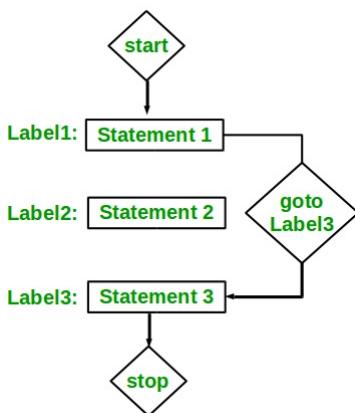
Most visited in Difference Between
Difference between PostgreSQL and MongoDB
Difference between DELETE and TRUNCATE
Difference Between SMO and SEO
Difference between Prim's and Kruskal's algorithm for MST
Monolithic vs Microservices architecture

The goto statement is a jump statement which is sometimes also referred to as unconditional jump statement. The goto statement can be used to jump from anywhere to anywhere within a function.

Syntax:

Syntax1		Syntax2
goto label;		label:
.		.
.		.
.		.
label:		goto label;

In the above syntax, the first line tells the compiler to go to or jump to the statement marked as a label. Here label is a user-defined identifier which indicates the target statement. The statement immediately followed after 'label:' is the destination statement. The 'label:' can also appear before the 'goto label;' statement in the above syntax.



Below are some examples on how to use goto statement:

Examples:

- **Type 1:** In this case, we will see a situation similar to as shown in Syntax1 above. Suppose we need to write a program where we need to check if a number is even or not and print accordingly using the goto statement. Below program explains how to do this:

C

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// C program to check if a number is
// even or not using goto statement
#include <stdio.h>

// function to check even or not
void checkEvenOrNot(int num)
{
    if (num % 2 == 0)
        // jump to even
        goto even;
    else
        // jump to odd
        goto odd;

even:
    printf("%d is even", num);
    // return if even
    return;
odd:
    printf("%d is odd", num);
}

int main() {
    int num = 26;
    checkEvenOrNot(num);
    return 0;
}
```

C++

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// C++ program to check if a number is
// even or not using goto statement
#include <iostream>
using namespace std;
```

```

// function to check even or not
void checkEvenOrNot(int num)
{
    if (num % 2 == 0)
        // jump to even
        goto even;
    else
        // jump to odd
        goto odd;

even:
    cout << num << " is even";
    // return if even
    return;
odd:
    cout << num << " is odd";
}

```

// Driver program to test above function

```

int main()
{
    int num = 26;
    checkEvenOrNot(num);
    return 0;
}

```

chevron_right

filter_none

Output:

26 is even

- **Type 2:** In this case, we will see a situation similar to as shown in Syntax1 above. Suppose we need to write a program which prints numbers from 1 to 10 using the goto statement. Below program explains how to do this.

C

```

filter_none
edit
close
play_arrow
link
brightness_4
code

// C program to print numbers
// from 1 to 10 using goto statement
#include <stdio.h>

// function to print numbers from 1 to 10
void printNumbers()
{
    int n = 1;
label:
    printf("%d ",n);
    n++;
    if (n <= 10)
        goto label;
}

// Driver program to test above function
int main() {
    printNumbers();
    return 0;
}

```

chevron_right

filter_none

C++

```

filter_none
edit
close
play_arrow
link
brightness_4
code

// C++ program to print numbers
// from 1 to 10 using goto statement
#include <iostream>
using namespace std;

// function to print numbers from 1 to 10
void printNumbers()
{
    int n = 1;
label:
    cout << n << " ";
    n++;
    if (n <= 10)
        goto label;
}
```

```
}

// Driver program to test above function
int main()
{
    printNumbers();
    return 0;
}
chevron_right
filter_none
Output:
```

1 2 3 4 5 6 7 8 9 10

Disadvantages of using goto statement:

- The use of goto statement is highly discouraged as it makes the program logic very complex.
- use of goto makes the task of analyzing and verifying the correctness of programs (particularly those involving loops) very difficult.
- Use of goto can be simply avoided using `break` and `continue` statements.

This article is contributed by **Harsh Agarwal**. If you like GeeksforGeeks and would like to contribute, you can also write an article using [contribute.geeksforgeeks.org](#) or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- Print a character n times without using loop, recursion or goto in C++
- C/C++ if else statement with Examples
- C/C++ if statement with Examples
- Continue Statement in C/C++
- Switch Statement in C/C++
- Break Statement in C/C++
- return statement in C/C++ with Examples
- Print "Even" or "Odd" without using conditional statement
- Nested switch statement in C++
- return statement vs exit() in main()
- Interesting facts about switch statement in C
- Programming puzzle (Assign value without any control statement)
- Implementing ternary operator without any conditional statement
- Data type of case labels of switch statement in C++?

Improved By : [SonalSrivastava02](#)

Article Tags :

C
C++

Practice Tags :

C
CPP

thumb_up
14

To-do Done
1.5

Based on 10 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) [std::basic_string::at in C++](#)

Next

[last_page](#) [std::string::resize\(\) in C++](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
Format specifiers in different Programming Languages
Predefined Macros in C with Examples
C program to print odd line contents of a File followed by even line content
C program to find square root of a given number

Most visited in C++
Vector of Vectors in C++ STL with Examples
C++ STL
Which C++ libraries are useful for competitive programming?
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

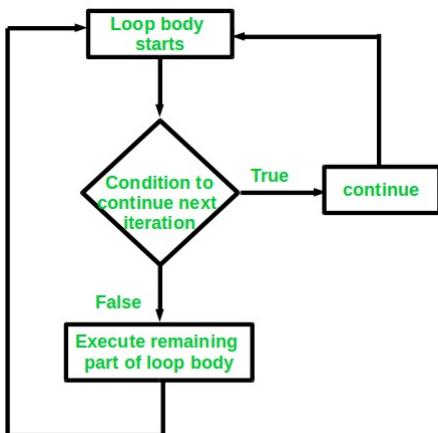
Continue Statement in C/C++

Continue is also a loop control statement just like the `break` statement. `continue` statement is opposite to that of `break statement`, instead of terminating the loop, it forces to execute the next iteration of the loop.

As the name suggest the `continue` statement forces the loop to continue or execute the next iteration. When the `continue` statement is executed in the loop, the code inside the loop following the `continue` statement will be skipped and next iteration of the loop will begin.

Syntax:

```
continue;
```



Example:

Consider the situation when you need to write a program which prints number from 1 to 10 and but not 6. It is specified that you have to do this using loop and only one loop is allowed to use. Here comes the usage of `continue` statement. What we can do here is we can run a loop from 1 to 10 and every time we have to compare the value of iterator with 6. If it is equal to 6 we will use the `continue` statement to continue to next iteration without printing anything otherwise we will print the value.

Below is the implementation of the above idea:

C

```

filter_none
edit
close
play_arrow
link
brightness_4
code

// C program to explain the use
// of continue statement
#include <stdio.h>

int main() {
    // loop from 1 to 10
    for (int i = 1; i <= 10; i++) {

        // If i is equals to 6,
        // continue to next iteration
        // without printing
        if (i == 6)
            continue;

        else
            // otherwise print the value of i
    }
}
  
```

```

        printf("%d ", i);
    }

    return 0;
}
chevron_right
filter_none

```

C++

```

filter_none
edit
close
play_arrow
link
brightness_4
code
// C++ program to explain the use
// of continue statement

#include <iostream>
using namespace std;

int main()
{
    // loop from 1 to 10
    for (int i = 1; i <= 10; i++) {

        // If i is equals to 6,
        // continue to next iteration
        // without printing
        if (i == 6)
            continue;

        else
            // otherwise print the value of i
            cout << i << " ";
    }

    return 0;
}
chevron_right
filter_none

```

Output:

```
1 2 3 4 5 7 8 9 10
```

The `continue` statement can be used with any other loop also like while or do while in a similar way as it is used with for loop above.

Exercise Problem:

Given a number n, print triangular pattern. We are allowed to use only one loop.

```
Input: 7
Output:
*
*
**
*
***
*
*****
*
```

Solution : [Print the pattern by using one loop | Set 2 \(Using Continue Statement\)](#)

This article is contributed by [Harsh Agarwal](#). If you like GeeksforGeeks and would like to contribute, you can also write an article using [contribute.geeksforgeeks.org](#) or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- Difference between continue and break statements in C++
- goto statement in C/C++
- Switch Statement in C/C++
- Break Statement in C/C++
- C/C++ if else statement with Examples
- C/C++ if statement with Examples
- return statement in C/C++ with Examples
- Nested switch statement in C++
- Print "Even" or "Odd" without using conditional statement
- return statement vs exit() in main()
- Interesting facts about switch statement in C
- Implementing ternary operator without any conditional statement
- Programming puzzle (Assign value without any control statement)
- Data type of case labels of switch statement in C++?

Article Tags :

C
C++
C Basics
CPP-Basics
Practice Tags :
C
CPP

thumb_up
7

To-do Done
1.7

Based on 7 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

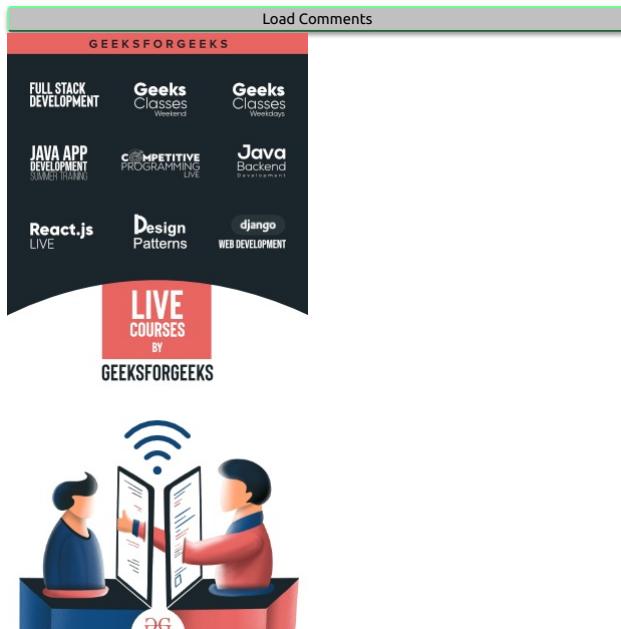
Previous

first_page Break Statement in C/C++

Next

last_page std::string::erase in C++

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
Predefined Macros in C with Examples
Format specifiers in different Programming Languages
C program to print odd line contents of a File followed by even line content
Introduction to the C99 Programming Language : Part II

Most visited in C++
Vector of Vectors in C++ STL with Examples
C++ Tutorial
What are the libraries are useful for competitive programming?
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

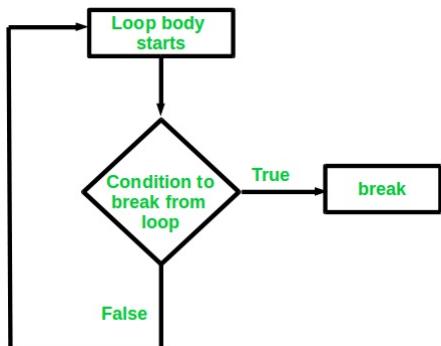
Break Statement in C/C++

The break in C or C++ is a loop control statement which is used to terminate the loop. As soon as the break statement is encountered from within a loop, the loop iterations stops there and control returns from the loop immediately to the first statement after the loop.

Syntax:

`break;`

Basically break statements are used in the situations when we are not sure about the actual number of iterations for the loop or we want to terminate the loop based on some condition.



We will see here the usage of break statement with three different types of loops:

1. Simple loops
2. Nested loops
3. Infinite loops

Let us now look at the examples for each of the above three types of loops using break statement.

1. **Simple loops:** Consider the situation where we want to search an element in an array. To do this, use a loop to traverse the array starting from the first index and compare the array elements with the given key.

Below is the implementation of this idea:

C

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// C program to illustrate
// Linear Search

#include <stdio.h>

void findElement(int arr[], int size, int key)
{
    // loop to traverse array and search for key
    for (int i = 0; i < size; i++) {
        if (arr[i] == key) {
            printf("Element found at position: %d", (i + 1));
        }
    }
}

int main() {
    int arr[] = { 1, 2, 3, 4, 5, 6 };

    // no of elements
    int n = 6;

    // key to be searched
    int key = 3;

    // Calling function to find the key
    findElement(arr, n, key);

    return 0;
}
chevron_right
```

C++

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// CPP program to illustrate
// Linear Search
#include <iostream>
using namespace std;

void findElement(int arr[], int size, int key)
{
    // loop to traverse array and search for key
    for (int i = 0; i < size; i++) {
        if (arr[i] == key) {
            cout << "Element found at position: " << (i + 1);
        }
    }
}

// Driver program to test above function
int main()
{
    int arr[] = { 1, 2, 3, 4, 5, 6 };
    int n = 6; // no of elements
    int key = 3; // key to be searched

    // Calling function to find the key
    findElement(arr, n, key);

    return 0;
}
```

chevron_right

```
filter_none
```

Output:

```
Element found at index: 3
```

The above code runs fine with no errors. But the above code is not efficient. The above code completes all the iterations even after the element is found. Suppose there are **1000** elements in the array and the key to be searched is present at 1st position so the above approach will execute **999** iterations which are of no purpose and are useless.

To avoid these useless iterations, we can use the break statement in our program. Once the break statement is encountered the control from the loop will return immediately after the condition gets satisfied. So will use the break statement with the if condition which compares the key with array elements as shown below:

C

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// C program to illustrate
// using break statement
// in Linear Search
#include <stdio.h>

void findElement(int arr[], int size, int key)
{
    // loop to traverse array and search for key
    for(int i = 0; i < size; i++) {
        if (arr[i] == key) {
            printf("Element found at position: %d", (i + 1));

            // using break to terminate loop execution
            break;
        }
    }
}

int main() {
    int arr[] = { 1, 2, 3, 4, 5, 6 };

    // no of elements
    int n = 6;

    // key to be searched
    int key = 3;

    // Calling function to find the key
    findElement(arr, n, key);

    return 0;
}
chevron_right
filter_none
```

C++

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// CPP program to illustrate
// using break statement
// in Linear Search
#include <iostream>
using namespace std;

void findElement(int arr[], int size, int key)
{
    // loop to traverse array and search for key
    for(int i = 0; i < size; i++) {
        if (arr[i] == key) {
            cout << "Element found at position: " << (i + 1);

            // using break to terminate loop execution
            break;
        }
    }
}

// Driver program to test above function
int main()
{
    int arr[] = { 1, 2, 3, 4, 5, 6 };
    int n = 6; // no of elements
    int key = 3; // key to be searched

    // Calling function to find the key
    findElement(arr, n, key);

    return 0;
}
chevron_right
filter_none
```

Output:

Element found at position: 3

2. **Nested Loops:** We can also use break statement while working with nested loops. If the break statement is used in the innermost loop, The control will come out only from the innermost loop. Below is the example of using break with nested loops:

C

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// C program to illustrate
// using break statement
// in Nested loops
#include <stdio.h>

int main() {
    // nested for loops with break statement
    // at inner loop
    for (int i = 0; i < 5; i++) {
        for (int j = 1; j <= 10; j++) {
            if (j > 3)
                break;
            else
                printf("**");
        }
        printf("\n");
    }

    return 0;
}
chevron_right
filter_none
```

C++

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// CPP program to illustrate
// using break statement
// in Nested loops
#include <iostream>
using namespace std;

int main()
{
    // nested for loops with break statement
    // at inner loop
    for (int i = 0; i < 5; i++) {
        for (int j = 1; j <= 10; j++) {
            if (j > 3)
                break;
            else
                cout << "**";
        }
        cout << endl;
    }

    return 0;
}
chevron_right
filter_none
```

Output:

```
***  
***  
***  
***  
***
```

In the above code we can clearly see that the inner loop is programmed to execute for 10 iterations. But as soon as the value of **j** becomes greater than 3 the inner loop stops executing which restricts the number of iteration of the inner loop to 3 iterations only. However the iteration of outer loop remains unaffected.

Therefore, break applies to only the loop within which it is present.

3. **Infinite Loops:** break statement can be included in an infinite loop with a condition in order to terminate the execution of the infinite loop.
Consider the below infinite loop:

C

```
filter_none
edit
```

close
play_arrow
link
brightness_4
code

```
// C program to illustrate
// using break statement
// in Infinite loops
#include <stdio.h>

int main() {
    // loop initialization expression
    int i = 0;

    // infinite while loop
    while (1) {
        printf("%d ", i);
        i++;
    }

    return 0;
}
```

chevron_right

filter_none

C++

filter_none
edit
close
play_arrow
link
brightness_4
code

```
// CPP program to illustrate
// using break statement
// in Infinite loops
#include <iostream>
using namespace std;

int main()
{
    // loop initialization expression
    int i = 0;

    // infinite while loop
    while (1) {
        cout << i << " ";
        i++;
    }

    return 0;
}
```

chevron_right

filter_none

Note: Please do not run the above program in your compiler as it is an infinite loop so you may have to forcefully exit the compiler to terminate the program.

In the above program, the loop condition based on which the loop terminates is always true. So, the loop executes infinite number of times. We can correct this by using the break statement as shown below:

C

filter_none
edit
close
play_arrow
link
brightness_4
code

```
// C program to illustrate
// using break statement
// in Infinite loops
#include <stdio.h>

int main() {
    // loop initialization expression
    int i = 1;

    // infinite while loop
    while (1) {
        if (i > 10)
            break;

        printf("%d ", i);
        i++;
    }
}
```

```
    return 0;
}
chevron_right
```

```
filter_none
```

C++

```
filter_none
edit
close

play_arrow
link
brightness_4
code

// CPP program to illustrate
// using break statement
// in Infinite loops
#include <iostream>
using namespace std;

int main()
{
    // loop initialization expression
    int i = 1;

    // infinite while loop
    while (1) {
        if (i > 10)
            break;

        cout << i << " ";
        i++;
    }

    return 0;
}
```

```
chevron_right
```

```
filter_none
```

Output:

```
1 2 3 4 5 6 7 8 9 10
```

The above code restricts the number of loop iterations to 10.

Apart from this, break can be used in Switch case statements too.

This article is contributed by **Harsh Agarwal**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up

```
Add your personal notes  
here! (max 5000 chars)
```

[Save](#)

Recommended Posts:

- [Difference between continue and break statements in C++](#)
- [Continue Statement in C/C++](#)
- [goto statement in C/C++](#)
- [C/C++ if statement with Examples](#)
- [C/C++ if else statement with Examples](#)
- [Switch Statement in C/C++](#)
- [return statement in C/C++ with Examples](#)
- [Print "Even" or "Odd" without using conditional statement](#)
- [Nested switch statement in C++](#)
- [return statement vs exit\(\) in main\(\)](#)
- [Interesting facts about switch statement in C](#)
- [Programming puzzle \(Assign value without any control statement\)](#)
- [Implementing ternary operator without any conditional statement](#)
- [Data type of case labels of switch statement in C++?](#)

Article Tags :

C
C++
C Basics
CPP-Basics
Practice Tags :
C
CPP



11

To-do Done

1.4

Based on 9 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

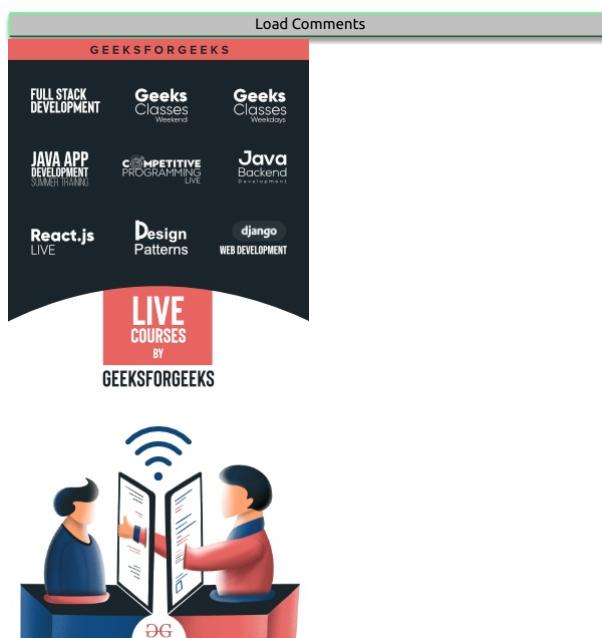
Previous

[first_page](#) Using range in switch case in C/C++

Next

[last_page](#) Continue Statement in C/C++

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C

- Print all possible combinations of the string by replacing '\$' with any other digit from the string
- Predefined Macros in C with Examples
- C program to print odd line contents of a File followed by even line content
- Introduction to the C99 Programming Language : Part II
- C program to find square root of a given number

Most visited in C++

- Vector of Vectors in C++ STL with Examples
- C++ Tutorial
- Which C++ libraries are useful for competitive programming?
- Array of Vectors in C++ STL
- Map of Vectors in C++ STL with Examples

GeeksforGeeks

A computer science portal for geeks

- 5th Floor, A-118,
- Sector-136, Noida, Uttar Pradesh - 201305
- feedback@geeksforgeeks.org
- **COMPANY**
 - About Us
 - Careers
 - Privacy Policy
 - Contact Us
- **LEARN**
 - Algorithms
 - Data Structures
 - Languages
 - CS Subjects
 - Video Tutorials
- **PRACTICE**
 - Courses
 - Company-wise
 - Topic-wise
 - How to begin?
- **CONTRIBUTE**
 - Write an Article
 - Write Interview Experience
 - Internships
 - Videos

@geeksforgeeks,

Using range in switch case in C/C++

You all are familiar with `switch` case in C/C++, but did you know **you can use range of numbers** instead of a single number or character in case statement.

- That is the case range extension of the GNU C compiler and not standard C or C++
- You can specify a range of consecutive values in a single case label, like this:
`case low ... high:`
- It can be used for ranges of ASCII character codes like this:
`case 'A' ... 'Z':`
- You need to Write spaces around the ellipses For example, write this:

```
// Correct - case 1 ... 5;
// Wrong - case 1...5;
```

```

filter_none
edit
close
play_arrow
link
brightness_4
code

// C program to illustrate
// using range in switch case
#include <stdio.h>
int main()
{
    int arr[] = { 1, 5, 15, 20 };

    for (int i = 0; i < 4; i++)
    {
        switch (arr[i])
        {
            case 1 ... 6:
                printf("%d in range 1 to 6\n", arr[i]);
                break;
            case 19 ... 20:
                printf("%d in range 19 to 20\n", arr[i]);
                break;
            default:
                printf("%d not in range\n", arr[i]);
                break;
        }
    }
    return 0;
}
chevron_right

```

[filter_none](#)

Output:

```

1 in range 1 to 6
5 in range 1 to 6
15 not in range
20 in range 19 to 20

```

Exercise : You can try above program for char array by modifying char array and case statement.

Error conditions:

1. **low > high** : The compiler gives with an error message.
2. **Overlapping case values** : If the value of a case label is within a case range that has already been used in the switch statement, the compiler gives an error message.

This article is contributed by [Mandeep Singh](#). If you like GeeksforGeeks and would like to contribute, you can also write an article using [contribute.geeksforgeeks.org](#) or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes [arrow_drop_up](#)

Add your personal notes
here! (max 5000 chars)

[Save](#)

Recommended Posts:

- [Nested switch case](#)
- [Output of C programs | Set 30 \(Switch Case\)](#)
- [Menu-Driven program using Switch-case in C](#)
- [Data type of case labels of switch statement in C++?](#)
- [Switch Statement in C/C++](#)
- [Nested switch statement in C++](#)
- [C++17 new feature : If Else and Switch Statements with initializers](#)
- [Interesting facts about switch statement in C](#)
- [Print individual digits as words without using if or switch](#)
- [Queries for elements having values within the range A to B in the given index range using Segment Tree](#)
- [C Program for Lower Case to Uppercase and vice-versa in a file](#)
- [Range-based for loop in C++](#)
- [How to delete a range of values from the Set using Iterator](#)
- [Generating random number in a range in C](#)
- [Calculate range of data types using C++](#)

Article Tags :

C
C++
cpp-switch
Practice Tags :
C
CPP

[thumb_up](#)
7

To-do Done
1.6

Based on 6 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

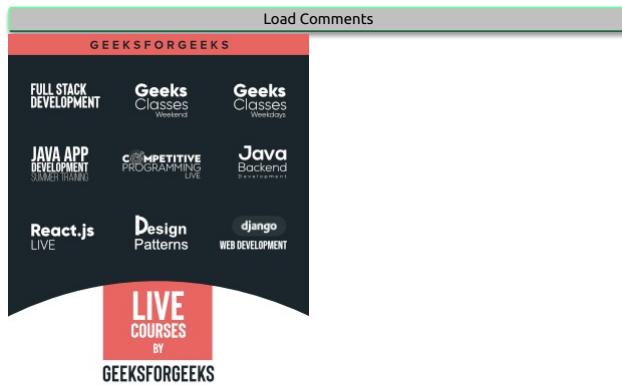
[Previous](#)

first_page pipe() System call

Next

last_page Break Statement in C/C++

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.



Most popular in C
 Print all possible combinations of the string by replacing '\$' with any other digit from the string
 Predefined Macros in C with Examples
 Format Specifiers in different Programming Languages
 How to call function within function in C or C++
 C program to print odd line contents of a File followed by even line content

Most visited in C++
 Vector of Vectors in C++ STL with Examples
 C++ Tutorial
 Which C++ libraries are useful for competitive programming?
 Array of Vectors in C++ STL
 Map of Vectors in C++ STL with Examples

Functions in C/C++

A function is a set of statements that take inputs, do some specific computation and produces output.

The idea is to put some commonly or repeatedly done task together and make a function so that instead of writing the same code again and again for different inputs, we can call the function.

Example:

Below is a simple C/C++ program to demonstrate functions.

C

```
filter_none
edit
close

play_arrow
link
brightness_4
code

#include <stdio.h>

// An example function that takes two parameters 'x' and 'y'
// as input and returns max of two input numbers
int max(int x, int y)
{
    if (x > y)
        return x;
    else
        return y;
}

// main function that doesn't receive any parameter and
// returns integer.
int main(void)
{
    int a = 10, b = 20;

    // Calling above function to find max of 'a' and 'b'
    int m = max(a, b);

    printf("m is %d", m);
    return 0;
}

chevron_right
filter_none
```

C++

filter_none

```
edit  
close  
play_arrow  
link  
brightness_4  
code  
  
#include <iostream>  
using namespace std;  
  
int max(int x, int y)  
{  
    if (x > y)  
        return x;  
    else  
        return y;  
}  
  
int main() {  
    int a = 10, b = 20;  
  
    // Calling above function to find max of 'a' and 'b'  
    int m = max(a, b);  
  
    cout << "m is " << m;  
    return 0;  
}
```

chevron_right
filter_none

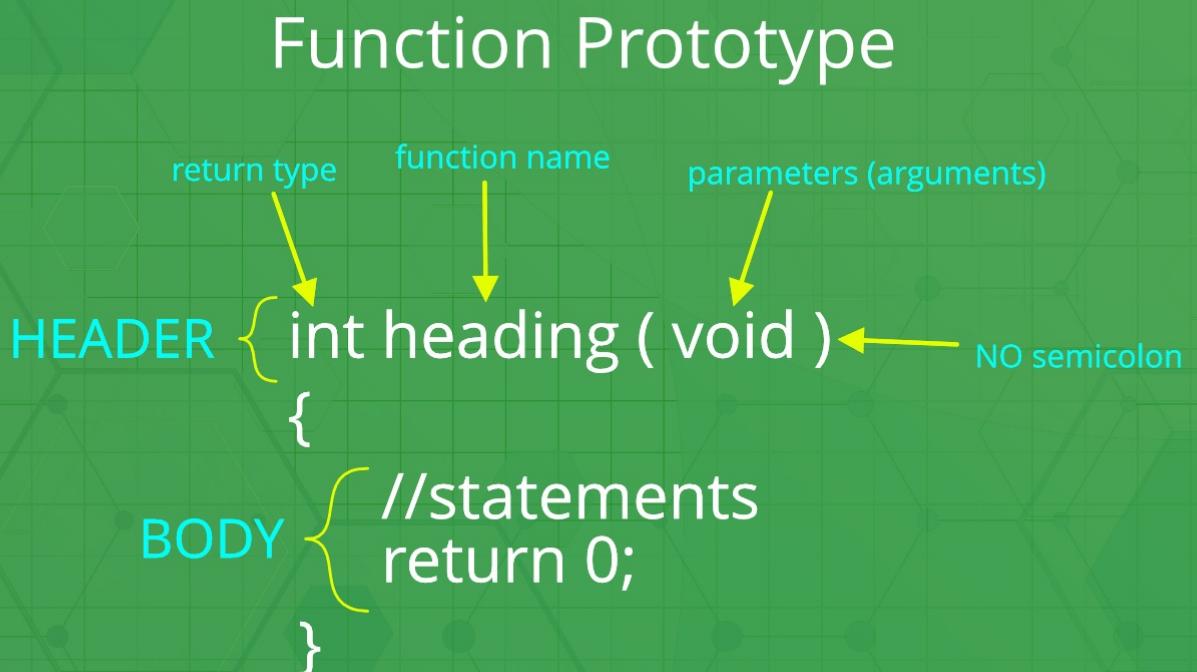
Output:
m is 20

Why do we need functions?

- Functions help us in reducing code redundancy. If functionality is performed at multiple places in software, then rather than writing the same code, again and again, we create a function and call it everywhere. This also helps in maintenance as we have to change at one place if we make future changes to the functionality.
- Functions make code modular. Consider a big file having many lines of codes. It becomes really simple to read and use the code if the code is divided into functions.
- Functions provide abstraction. For example, we can use library functions without worrying about their internal working.

Function Declaration

A function declaration tells the compiler about the number of parameters function takes, data-types of parameters and return type of function. Putting parameter names in function declaration is optional in the function declaration, but it is necessary to put them in the definition. Below are an example of function declarations. (parameter names are not there in below declarations)



```
filter_none  
edit  
close  
play_arrow  
link  
brightness_4  
code  
  
// A function that takes two integers as parameters  
// and returns an integer  
int max(int, int);  
  
// A function that takes a int pointer and an int variable as parameters
```

```
// and returns a pointer of type int
int *swap(int*,int);

// A function that takes a char as parameters
// and returns an reference variable
char *call(char b);

// A function that takes a char and an int as parameters
// and returns an integer
int fun(char, int);
chevron_right
filter_none
```

It is always recommended to declare a function before it is used (See [this](#), [this](#) and [this](#) for details)

In C, we can do both declaration and definition at the same place, like done in the above example program.

C also allows to declare and define functions separately, this is especially needed in case of library functions. The library functions are declared in header files and defined in library files. Below is an example declaration.

Parameter Passing to functions

The parameters passed to function are called **actual parameters**. For example, in the above program 10 and 20 are actual parameters. The parameters received by function are called **formal parameters**. For example, in the above program x and y are formal parameters.

There are two most popular ways to pass parameters.

Pass by Value: In this parameter passing method, values of actual parameters are copied to function's formal parameters and the two types of parameters are stored in different memory locations. So any changes made inside functions are not reflected in actual parameters of caller.

Pass by Reference Both actual and formal parameters refer to same locations, so any changes made inside the function are actually reflected in actual parameters of caller.

Parameters are always passed by value in C. For example, in the below code, value of x is not modified using the function fun().

C

```
filter_none
edit
close
play_arrow
link
brightness_4
code

#include <stdio.h>
void fun(int x)
{
    x = 30;
}

int main(void)
{
    int x = 20;
    fun(x);
    printf("x = %d", x);
    return 0;
}
chevron_right
filter_none
```

C++

```
filter_none
edit
close
play_arrow
link
brightness_4
code

#include <iostream>
using namespace std;

void fun(int x) {
    x = 30;
}

int main() {
    int x = 20;
    fun(x);
    cout << "x = " << x;
    return 0;
}
chevron_right
filter_none
```

Output:

```
(x = 20)
```

However, in C, we can use pointers to get the effect of pass by reference. For example, consider the below program. The function fun() expects a pointer ptr to an integer (or an address of an integer). It modifies the value at the address ptr. The dereference operator * is used to access the value at an address. In the statement *ptr = 30, value at address ptr is changed to 30. The address operator & is used to get the address of a variable of any data type. In the function call statement 'fun(&x)', the address of x is passed so that x can be modified using its address.

C

```
filter_none
edit
close
play_arrow
link
brightness_4
code

# include <stdio.h>
void fun(int *ptr)
{
    *ptr = 30;
}

int main()
{
    int x = 20;
    fun(&x);
    printf("x = %d", x);

    return 0;
}
chevron_right
filter_none
```

C++

```
filter_none
edit
close
play_arrow
link
brightness_4
code

#include <iostream>
using namespace std;

void fun(int *ptr)
{
    *ptr = 30;
}

int main() {
    int x = 20;
    fun(&x);
    cout << "x = " << x;

    return 0;
}
chevron_right
filter_none
```

Output:

```
x = 30
```

Following are some important points about functions in C.

1) Every C program has a function called main() that is called by operating system when a user runs the program.

2) Every function has a return type. If a function doesn't return any value, then void is used as return type. Moreover, if the return type of the function is void, we still can use return statement in the body of function definition by not specifying any constant, variable, etc. with it, by only mentioning the 'return;' statement which would symbolise the termination of the function as shown below:

```
filter_none
edit
close
play_arrow
link
brightness_4
code

void function name(int a)
{
    ..... //Function Body
    return; //Function execution would get terminated
}
chevron_right
filter_none
```

3) In C, functions can return any type except arrays and functions. We can get around this limitation by returning pointer to array or pointer to function.

4) Empty parameter list in C mean that the parameter list is not specified and function can be called with any parameters. In C, it is not a good idea to declare a function like fun(). To declare a function that can only be called without any parameter, we should use "void fun(void)".

As a side note, in C++, empty list means function can only be called without any parameter. In C++, both void fun() and void fun(void) are same.

5) If in a C program, a function is called before its declaration then the C compiler automatically assumes the declaration of that function in the following way:

```
int function name();
```

And in that case if the return type of that function is different than INT ,compiler would show an error.

More on Functions in C/C++:

- Quiz on function in C
- Importance of function prototype in C
- Functions that are executed before and after main() in C
- return statement vs exit() in main()
- How to Count Variable Numbers of Arguments in C?,
- What is evaluation order of function parameters in C?

- Does C support function overloading?
- How can we return multiple values from a function?
- What is the purpose of a function prototype?
- Static functions in C
- exit(), abort() and assert()
- Implicit return type int in C
- What happens when a function is called before its declaration in C?

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes 
Add your personal notes here! (max 5000 chars)

Recommended Posts:

- Mathematical Functions in Python | Set 2 (Logarithmic and Power Functions)
- Mathematical Functions in Python | Set 1 (Numeric Functions)
- PHP | Functions
- C | Functions | Question 11
- C | Functions | Question 5
- C | Functions | Question 6
- C | Functions | Question 9
- C | Functions | Question 10
- C | Functions | Question 7
- C | Functions | Question 8
- PHP | String Functions
- C++ Mathematical Functions
- Thread functions in C/C++
- Nested functions in C
- C | Functions | Question 4

Improved By : [sameer2209](#), [ShivankKapoor](#), [theWINTERSOLDIER](#), [akshaynaile115](#), [dmontigny27](#)

Article Tags :

C
School Programming
C-Functions
CBSE - Class 11
CPP-Basics
CPP-Functions
Functions
school-programming

Practice Tags :
C
Functions

 71

To-do Done
2

Based on 109 vote(s)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) [fork\(\) in C](#)

Next

[last_page](#) [Variable Length Arrays in C and C++](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes LIVE Geeks Classes ONDEMAND

JAVA APP DEVELOPMENT SUMMER TRAINING COMPETITIVE PROGRAMMING LIVE Java Backend Development

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS



Most popular in C

Importance of function prototype in C

Function prototype tells compiler about number of parameters function takes, data-types of parameters and return type of function. By using this information, compiler cross checks function parameters and their data-type with function definition and function call. If we ignore function prototype, program may compile with warning, and may work properly. But some times, it will give strange output and it is very hard to find such programming mistakes. Let us see with examples

```
filter_none
edit
close
play_arrow
link
brightness_4
code

#include <errno.h>
#include <stdio.h>

int main(int argc, char *argv[])
{
    FILE *fp;

    fp = fopen(argv[1], "r");
    if (fp == NULL) {
        fprintf(stderr, "%s\n", strerror(errno));
        return errno;
    }

    printf("file exist\n");

    fclose(fp);

    return 0;
}
chevron_right
filter_none
```

Above program checks existence of file, provided from command line, if given file is exist, then the program prints "file exist", otherwise it prints appropriate error message. Let us provide a filename, which does not exist in file system, and check the output of program on x86_64 architecture.

```
[narendra@media/partition/GFG]$ ./file_existence hello.c
Segmentation fault (core dumped)
```

Why this program crashed, instead it should show appropriate error message. This program will work fine on x86 architecture, but will crash on x86_64 architecture. Let us see what was wrong with code. Carefully go through the program, deliberately I haven't included prototype of "strerror()" function. This function returns "pointer to character", which will print error message which depends on errno passed to this function. Note that x86 architecture is ILP-32 model, means integer, pointers and long are 32-bit wide, that's why program will work correctly on this architecture. But x86_64 is LP-64 model, means long and pointers are 64 bit wide. In C language, when we don't provide prototype of function, the compiler assumes that function returns an integer. In our example, we haven't included "string.h" header file (strerror's prototype is declared in this file), that's why compiler assumed that function returns integer. But its return type is pointer to character. In x86_64, pointers are 64-bit wide and integers are 32-bits wide, that's why while returning from function, the returned address gets truncated (i.e. 32-bit wide address, which is size of integer on x86_64) which is invalid and when we try to dereference this address, the result is segmentation fault.

Now include the "string.h" header file and check the output, the program will work correctly.

```
[narendra@media/partition/GFG]$ ./file_existence hello.c
No such file or directory
```

Consider one more example.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

#include <stdio.h>

int main(void)
{
    int *p = malloc(sizeof(int));

    if (p == NULL) {
        perror("malloc()");
        return -1;
    }

    *p = 10;
    free(p);

    return 0;
}
chevron_right
filter_none
```

Above code will work fine on IA-32 model, but will fail on IA-64 model. Reason for failure of this code is we haven't included prototype of malloc() function and returned value is truncated in IA-64 model.

This article is compiled by **Narendra Kangalkar**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes

Add your personal notes here! (max 5000 chars)

Recommended Posts:

- What is the purpose of a function prototype?
- How to call function within function in C or C++
- arc function in C
- putchar() function in C
- Inline function in C
- strcspn() function in C/C++
- iswxdigit() function in C/C++
- strrev() function in C
- strrchr() function in C/C++
- atexit() function in C/C++
- c16rtomb() function in C/C++
- c32rtomb() function in C/C++
- wcsrchr() function in C/C++
- mbrtowc() function in C/C++
- wcsprblk() function in C/C++

Article Tags :

[C](#)

[C-Functions](#)

Practice Tags :

[C](#)



14

To-do Done

3

Based on 72 vote(s)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) Multiline macros in C

Next

[last_page](#) Sequence Points in C | Set 1

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Weekend Geeks Classes Weekdays

JAVA APP DEVELOPMENT SUMMER TRAINING COMPETITIVE PROGRAMMING LIVE Java Backend Development

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS



Most popular in C
[getchar\(\)](#) function in C with Examples
Print all possible combinations of the string by replacing '\$' with any other digit from the string
[ctype.h <cctype.h>](#) library in C/C++ with Examples
How to use make utility to build C projects?
C programs display month by month calendar for a given year

More related articles in C
[Introduction to the C99 Programming Language : Part I](#)
[Hello World Program : First program while learning Programming](#)
[Predefined Macros in C with Examples](#)
[Implicit Type Conversion in C with Examples](#)
[Format specifiers in different Programming Languages](#)

Functions that are executed before and after main() in C

With GCC family of C compilers, we can mark some functions to execute before and after main(). So some startup code can be executed before main() starts, and some cleanup code can be executed after main() ends. For example, in the following program, myStartupFun() is called before main() and myCleanupFun() is called after main().

[filter_none](#)

[edit](#)

[close](#)

[play_arrow](#)

[link](#)

[brightness_4](#)

[code](#)

```

#include<stdio.h>

/* Apply the constructor attribute to myStartupFun() so that it
   is executed before main() */
void myStartupFun (void) __attribute__ ((constructor));

/* Apply the destructor attribute to myCleanupFun() so that it
   is executed after main() */
void myCleanupFun (void) __attribute__ ((destructor));

/* implementation of myStartupFun */
void myStartupFun (void)
{
    printf ("startup code before main()\n");
}

/* implementation of myCleanupFun */
void myCleanupFun (void)
{
    printf ("cleanup code after main()\n");
}

int main (void)
{
    printf ("hello\n");
    return 0;
}

```

chevron_right
filter_none

Output:

```

startup code before main()
hello
cleanup code after main()

```

Like the above feature, GCC has added many other interesting features to standard C language. See [this](#) for more details.

Related Article :

[Executing main\(\) in C - behind the scene](#)

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

[My Personal Notes](#) *arrow_drop_up*

Add your personal notes
here! (max 5000 chars)

[Save](#)

Recommended Posts:

- Is it fine to write "void main()" or "main()" in C/C++?
- C/C++ program for calling main() in main()
- Difference between "int main()" and "int main(void)" in C/C++?
- Anything written in sizeof() is never executed in C
- Can main() be overloaded in C++?
- What does main() return in C and C++?
- Executing main() in C/C++ - behind the scene
- return statement vs exit() in main()
- How to write a running C code without main()?
- How to change the output of printf() in main()?
- How to print "GeeksforGeeks" with empty main() in C, C++ and Java?
- Functions in C/C++
- Macros vs Functions
- Functions that cannot be overloaded in C++
- Thread functions in C/C++

Article Tags :

C
C-Functions
Practice Tags :

thumb_up
14

To-do Done
2.8

Based on 43 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) [Templates and Default Arguments](#)

Next

[last_page](#) [Const Qualifier in C](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

[Load Comments](#)

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Weekend Geeks Classes Weekdays

JAVA APP DEVELOPMENT COMPETITIVE PROGRAMMING LITE Java Backend Development

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS

Most popular in C
getch() function in C with Examples
Print all possible combinations of the string by replacing '\$' with any other digit from the string
ctype.h(<cctype>) library in C/C++ with Examples
How to use make utility to build C projects?
C program to display month by month calendar for a given year

More related articles in C
Predefined Macros in C with Examples
Hello World Program - First Program while learning Programming
Implicit Type Conversion in C with Examples
Introduction to the C99 Programming Language : Part I
How to call function within function in C or C++

return statement vs exit() in main()

In C++, what is the difference between `exit(0)` and `return 0`?

When `exit(0)` is used to exit from program, destructors for locally scoped non-static objects are not called. But destructors are called if `return 0` is used.

Program 1 -- uses exit(0) to exit

```
filter_none
edit
close
play_arrow
link
brightness_4
code

#include<iostream>
#include<stdio.h>
#include<stdlib.h>

using namespace std;

class Test {
public:
    Test() {
        printf("Inside Test's Constructor\n");
    }

    ~Test(){
        printf("Inside Test's Destructor");
        getchar();
    }
};

int main() {
    Test t1;

    // using exit(0) to exit from main
    exit(0);
}
```

Output:
Inside Test's Constructor

Program 2 - uses return 0 to exit

```
filter_none
edit
close
play_arrow
link
brightness_4
code

#include<iostream>
#include<stdio.h>
#include<stdlib.h>
```

```

using namespace std;

class Test {
public:
    Test() {
        printf("Inside Test's Constructor\n");
    }

    ~Test(){
        printf("Inside Test's Destructor");
    }
};

int main() {
    Test t1;

    // using return 0 to exit from main
    return 0;
}

```

chevron_right

filter_none

Output:

Inside Test's Constructor
Inside Test's Destructor

Calling destructors is sometimes important, for example, if destructor has code to release resources like closing files.

Note that static objects will be cleaned up even if we call exit(). For example, see following program.

```

filter_none
edit
close

play_arrow
link
brightness_4
code

#include<iostream>
#include<stdio.h>
#include<stdlib.h>
```

using namespace std;

```

class Test {
public:
    Test() {
        printf("Inside Test's Constructor\n");
    }

    ~Test(){
        printf("Inside Test's Destructor");
        getchar();
    }
};
```

```

int main() {
    static Test t1; // Note that t1 is static

    exit(0);
}
```

chevron_right

filter_none

Output:

Inside Test's Constructor
Inside Test's Destructor

Contributed by **indiarox**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes *arrow_drop_up*

Add your personal notes
here! (max 5000 chars)

Save

Recommended Posts:

- [What does main\(\) return in C and C++?](#)
- [return statement in C/C++ with Examples](#)
- [exit\(0\) vs exit\(1\) in C/C++ with Examples](#)
- [Is it fine to write "void main\(\)" or "main\(\)" in C/C++?](#)
- [C/C++ program for calling main\(\) in main\(\)](#)
- [Difference between "int main\(\)" and "int main\(void\)" in C/C++?](#)
- [exit\(\) vs _Exit\(\) in C and C++](#)
- [exit\(\), abort\(\) and assert\(\)](#)
- [Can main\(\) be overloaded in C++?](#)
- [Executing main\(\) in C/C++ - behind the scene](#)
- [Functions that are executed before and after main\(\) in C](#)
- [How to change the output of printf\(\) in main\(\)?](#)
- [How to write a running C code without main\(\)](#)
- [How to overload and override main method in Java](#)
- [How to print "GeeksforGeeks" with empty main\(\) in C, C++ and Java?](#)

Article Tags :

C
C-Functions
exit
main
return
Practice Tags :

C

thumb_up
10

To-do Done

3

Based on 46 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

first_page CRASH() macro - interpretation

Next

last_page Functions that cannot be overloaded in C++

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes WEEKEND

JAVA APP DEVELOPMENT SUMMER TRAINING COMPETITIVE PROGRAMMING LIVE Java Backend ENVIRONMENT

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS

Most popular in C

Print all possible combinations of the string by replacing 'S' with any other digit from the string
getchar() function in C with Examples
How to use make utility to build C projects?
ctype.h<ctype> library in C/C++ with Examples
C program to display month by month calendar for a given year

More related articles in C

Introduction to the C99 Programming Language : Part I
Hello World Program : First program while learning Programming
Predefined Macros in C with Examples
Implicit Type Conversion in C with Examples
Format specifiers in different Programming Languages

How to Count Variable Numbers of Arguments in C?

C supports variable numbers of arguments. But there is no language provided way for finding out total number of arguments passed. User has to handle this in one of the following ways:

- 1) By passing first argument as count of arguments.
- 2) By passing last argument as NULL (or 0).
- 3) Using some printf (or scanf) like mechanism where first argument has placeholders for rest of the arguments.

Following is an example that uses first argument `arg_count` to hold count of other arguments.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

#include <stdarg.h>
#include <stdio.h>

// this function returns minimum of integer numbers passed. First
// argument is count of numbers.
int min(int arg_count, ...)
{
    int i;
    int min, a;

    // va_list is a type to hold information about variable arguments
    va_list ap;

    // va_start must be called before accessing variable argument list
    va_start(ap, arg_count);

    // Now arguments can be accessed one by one using va_arg macro
    // Initialize min as first argument in list
```

```

min = va_arg(ap, int);

// traverse rest of the arguments to find out minimum
for(i = 2; i <= arg_count; i++) {
    if((a = va_arg(ap, int)) < min)
        min = a;
}

//va_end should be executed before the function returns whenever
// va_start has been previously used in that function
va_end(ap);

return min;
}

int main()
{
    int count = 5;

    // Find minimum of 5 numbers: (12, 67, 6, 7, 100)
    printf("Minimum value is %d", min(count, 12, 67, 6, 7, 100));
    getchar();
    return 0;
}

```

chevron_right

filter_none

Output:
Minimum value is 6

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes *arrow_drop_up*

Add your personal notes
here! (max 5000 chars)

Recommended Posts:

- Variable length arguments for Macros
- Why variable name does not start with numbers in C ?
- Can we access global variable if there is a local variable with same name?
- Internal static variable vs. External static variable with Examples in C
- Command line arguments in C/C++
- Command line arguments example in C
- Templates and Default Arguments
- When do we pass arguments by reference or pointer?
- Default arguments and virtual function
- Some Interesting facts about default arguments in C++
- C++ | Function Overloading and Default Arguments | Question 5
- C++ | Function Overloading and Default Arguments | Question 4
- C++ | Function Overloading and Default Arguments | Question 2
- C++ | Function Overloading and Default Arguments | Question 3
- C++ | Function Overloading and Default Arguments | Question 5

Article Tags :

C
C-Functions
Practice Tags :

C

thumb_up
12

To-do Done
3.6

Based on 66 vote(s)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

first_page The OFFSETOF() macro

Next

last_page Use of realloc()

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C
C program to display month by month calendar for a given year
Print all possible combinations of the string by replacing '\$' with any other digit from the string
getch() function in C with Examples
ctype.h <ctype> library in C/C++ with Examples
How to use make utility to build C projects?

More related articles in C
Implicit Type Conversion in C with Examples
Introduction to the C Programming Language - Part I
Format specifiers in different Programming Languages
Hello World Program : First program while learning Programming
C program to find square root of a given number

What is evaluation order of function parameters in C?

It is compiler dependent in C. It is never safe to depend on the order of evaluation of side effects. For example, a function call like below may very well behave differently from one compiler to another:

```
filter_none
edit
close
play_arrow
link
brightness_4
code
void func (int, int);

int i = 2;
func (i++, i++);
chevron_right
filter_none
```

There is no guarantee (in either the C or the C++ standard language definitions) that the increments will be evaluated in any particular order. Either increment might happen first. func might get the arguments `2, 3', or it might get `3, 2', or even `2, 2'.

Source: http://gcc.gnu.org/onlinedocs/gcc/Non_002dbugs.html

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes

Recommended Posts:

- Evaluation order of operands
- Do not use sizeof for array parameters
- Why C treats array parameters as pointers?
- Order of operands for logical operators
- Sorting 2D Vector in C++ | Set 2 (In descending order by row and column)
- Program to copy the contents of one array into another in the reverse order
- Sorting Vector of Pairs in C++ | Set 2 (Sort in descending order by first and second)
- How to call function within function in C or C++
- arc function in C
- time() function in C
- wctrans() function in C/C++
- wmemcpy() function in C/C++
- strnset() function in C
- strlwr() function in C
- clock() function in C/C++

Article Tags :

C
GFACTS
C-Functions

Practice Tags :

7

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page Data type of character constants in C and C++](#)

Next

[last_page Can we use function on left side of an expression in C and C++?](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS



FULL STACK
DEVELOPMENT



Geeks
Classes
Weekend



Geeks
Classes
Weekdays



Java
App
Development



COMPETITIVE
PROGRAMMING



Java
Backend



React.js
LIVE



Design
Patterns



django
WEB DEVELOPMENT



LIVE
COURSES
BY
GEEKSFORGEEKS



Most popular in C
 C program to display month by month calendar for a given year
 Print all possible combinations of the string by replacing '\$' with any other digit from the string
 getch() function in C with Examples
 ctype.h(<cctype>) library in C/C++ with Examples
 How to use make utility to build C projects?

Most visited in GFacts
 Interesting and Cool Tricks in Java
 Interesting Facts About Perl
 How to Install and Configure MongoDB in Ubuntu?

Does C support function overloading?

First of all, what is function overloading? Function overloading is a feature of a programming language that allows one to have many functions with same name but with different signatures.

This feature is present in most of the Object Oriented Languages such as C++ and Java. But C (not Object Oriented Language) doesn't support this feature. However, one can achieve the similar functionality in C indirectly. One of the approach is as follows.

Have a void * type of pointer as an argument to the function. And another argument telling the actual data type of the first argument that is being passed.

```
int foo(void * arg1, int arg2);
```

Suppose, arg2 can be interpreted as follows. 0 = Struct1 type variable, 1 = Struct2 type variable etc. Here Struct1 and Struct2 are user defined struct types.

While calling the function foo at different places...

```
foo(arg1, 0); /*Here, arg1 is pointer to struct type Struct1 variable*/  
foo(arg1, 1); /*Here, arg1 is pointer to struct type Struct2 variable*/
```

Since the second argument of the foo keeps track the data type of the first type, inside the function foo, one can get the actual data type of the first argument by typecast accordingly. i.e. inside the foo function

```
filter_none  
edit  
close  
  
play_arrow  
  
link  
brightness_4  
code  
  
if(arg2 == 0)  
{  
    struct1PtrVar = (Struct1 *)arg1;  
}  
else if(arg2 == 1)  
{  
    struct2PtrVar = (Struct2 *)arg1;  
}  
else  
{  
    /*Error Handling*/  
}  
chevron_right  
  
filter_none
```

There can be several other ways of implementing function overloading in C. But all of them will have to use pointers - the most powerful feature of C.

In fact, it is said that without using the pointers, one can't use C efficiently & effectively in a real world program!

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes 

Add your personal notes
here! (max 5000 chars)

Recommended Posts:

- Function Overloading and float in C++
- Function overloading and const keyword
- C++ | Function Overloading and Default Arguments | Question 3
- C++ | Function Overloading and Default Arguments | Question 5
- C++ | Function Overloading and Default Arguments | Question 5
- C++ | Function Overloading and Default Arguments | Question 4
- C++ | Function Overloading and Default Arguments | Question 2
- Does overloading work with Inheritance?
- C++ | Operator Overloading | Question 10
- C++ | Operator Overloading | Question 3
- C++ | Operator Overloading | Question 4
- C++ | Operator Overloading | Question 5
- C++ | Operator Overloading | Question 10
- C++ | Operator Overloading | Question 6
- C++ | Operator Overloading | Question 7

Article Tags :

C
C-Functions
c-puzzle

Practice Tags :

C

thumb_up
8

To-do Done
2.8

Based on 41 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

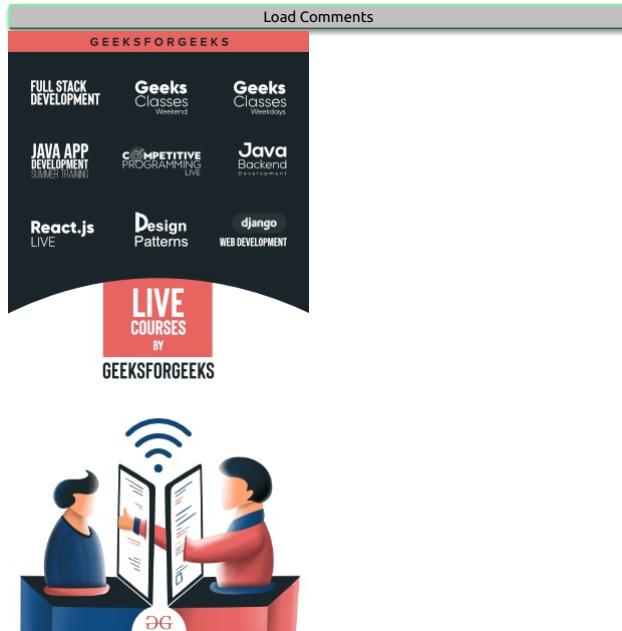
Previous

[first_page](#) Understanding "extern" keyword in C

Next

[last_page](#) How can I return multiple values from a function?

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



How can I return multiple values from a function?

We all know that a function in C can return only one value. So how do we achieve the purpose of returning multiple values.

Well, first take a look at the declaration of a function.

filter_none
edit
close

play_arrow

link
brightness_4
code

int foo(int arg1, int arg2);
chevron_right

filter_none

So we can notice here that our interface to the function is through arguments and return value only. (Unless we talk about modifying the globals inside the function)

Let us take a deeper look...Even though a function can return only one value but that value can be of pointer type. That's correct, now you're speculating right!

We can declare the function such that, it returns a structure type user defined variable or a pointer to it . And by the property of a structure, we know that a structure in C can hold multiple values of asymmetrical types (i.e. one int variable, four char variables, two float variables and so on...)

If we want the function to return multiple values of same data types, we could return the pointer to array of that data types.

We can also make the function return multiple values by using the arguments of the function. How? By providing the pointers as arguments.

Usually, when a function needs to return several values, we use one pointer in return instead of several pointers as arguments.

Please see [How to return multiple values from a function in C or C++?](#) for more details.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes [arrow_drop_up](#)

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- [How to return multiple values from a function in C or C++?](#)
- [C function argument and return values](#)
- [Return values of printf\(\) and scanf\(\) in C/C++](#)
- [std::tuple, std::pair | Returning multiple values from a function using Tuple and Pair in C++](#)
- [What does main\(\) return in C and C++?](#)
- [Implicit return type int in C](#)
- [return statement in C/C++ with Examples](#)
- [Return from void functions in C++](#)
- [return statement vs exit\(\) in main\(\)](#)
- [What is return type of getchar\(\), fgetc\(\) and getc\(\) ?](#)
- [Multiple Inheritance in C++](#)
- [Assigning multiple characters in an int in C language](#)
- [Structure Sorting \(By Multiple Rules\) in C++](#)
- [How Linkers Resolve Global Symbols Defined at Multiple Places?](#)
- [Socket Programming in C/C++: Handling multiple clients on server without multi threading](#)

Article Tags :

C

C-Functions

c-puzzle

Practice Tags :

C

thumb_up

7

To-do Done
2.3

Based on 48 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) Does C support function overloading?

Next

[last_page](#) What is the purpose of a function prototype?

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Webinars Geeks Classes Workshops

JAVA APP DEVELOPMENT SUMMER TRAINING COMPETITIVE PROGRAMMING LIVE Java Backend Development

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS

Most popular in C
getchar() function in C with Examples
Print all possible combinations of the string by replacing 's' with any other digit from the string
ctype.h<ctype.h> library in C/C++ with Examples
How to use make utility to build C projects?
C program to display month by month calendar for a given year

More related articles in C
Predefined Macros in C with Examples
Hello World Program : First program while learning Programming
Implicit Type Conversion in C with Examples
Introduction to the C99 Programming Language : Part I
How to call function within function in C or C++

What is the purpose of a function prototype?

The Function prototype serves the following purposes -

- 1) It tells the return type of the data that the function will return.
- 2) It tells the number of arguments passed to the function.
- 3) It tells the data types of each of the passed arguments.
- 4) Also it tells the order in which the arguments are passed to the function.

Therefore essentially, function prototype specifies the input/output interface to the function i.e. what to give to the function and what to expect from the function.

Prototype of a function is also called signature of the function.

What if one doesn't specify the function prototype?

Output of below kind of programs is generally asked at many places.

```
filter_none
edit
close

play_arrow
link
brightness_4
code

int main()
{
    foo();
    getchar();
    return 0;
}

void foo()
{
    printf("foo called");
}

chevron_right
filter_none
```

If one doesn't specify the function prototype, the behavior is specific to C standard (either C90 or C99) that the compilers implement. Up to C90 standard, C compilers assumed the return type of the omitted function prototype as int. And this assumption at compiler side may lead to unspecified program behavior.

Later C99 standard specified that compilers can no longer assume return type as int. Therefore, C99 became more restrictive in type checking of function prototype. But to make C99 standard backward compatible, in practice, compilers throw the warning saying that the return type is assumed as int. But they go ahead with compilation. Thus, it becomes the responsibility of programmers to make sure that the assumed function prototype and the actual function type matches.

To avoid all this implementation specifics of C standards, it is best to have function prototype.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes 

Add your personal notes here! (max 5000 chars)



Recommended Posts:

- Importance of function prototype in C
- How to call function within function in C or C++
- arc function in C
- Inline function in C
- putchar() function in C
- strcspn() function in C/C++
- strrev() function in C/C++
- iswxdigit() function in C/C++
- strrchr() function in C/C++
- atexit() function in C/C++
- c16rtomb() function in C/C++
- mbrtowc() function in C/C++
- wcschr() function in C/C++
- wcspbrk() function in C/C++
- c32rtomb() function in C/C++

Article Tags :

C
C-Functions
c-puzzle
CBSE - Class 11
school-programming
Practice Tags :

C


9

To-do Done
1.7

Based on 37 vote(s)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) How can I return multiple values from a function?

Next

[last_page](#) Write a C macro PRINT(x) which prints x

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.





Most popular in C
C program to display month by month calendar for a given year
Print all possible combinations of the string by replacing '\$' with any other digit from the string
getch() function in C with Examples
ctype.h <ctype.h> library in C/C++ with Examples
How to use make utility to build C projects?

More related articles in C
Implicit Type Conversion in C with Examples
Introduction to the C Programming Language - Part I
Format specifier in different Programming Languages
Hello World Program : First program while learning Programming
C program to find square root of a given number

Static functions in C

Prerequisite : [Static variables in C](#)

In C, functions are global by default. The "static" keyword before a function name makes it static. For example, below function `fun()` is static.

```
filter_none
edit
close

play_arrow
link
brightness_4
code

static int fun(void)
{
    printf("I am a static function ");
}
chevron_right
filter_none
```

Unlike global functions in C, access to static functions is restricted to the file where they are declared. Therefore, when we want to restrict access to functions, we make them static. Another reason for making functions static can be reuse of the same function name in other files.

For example, if we store following program in one file `file1.c`.

```
filter_none
edit
close

play_arrow
link
brightness_4
code

/* Inside file1.c */
static void fun1(void)
{
    puts("fun1 called");
}
chevron_right
filter_none
```

And store following program in another file `file2.c`

```
filter_none
edit
close

play_arrow
link
brightness_4
code

/* Inside file2.c */
int main(void)
{
    fun1();
    getchar();
    return 0;
}
chevron_right
filter_none
```

Now, if we compile the above code with command "gcc file2.c file1.c", we get the error "undefined reference to `fun1'" . This is because fun1() is declared static in file1.c and cannot be used in file2.c.

Please write comments if you find anything incorrect in the above article, or want to share more information about static functions in C.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes

Add your personal notes here! (max 5000 chars)

Recommended Posts:

- [Can static functions be virtual in C++?](#)
- [Internal static variable vs. External static variable with Examples in C](#)
- [Static Variables in C](#)
- [C++ | Static Keyword | Question 6](#)
- [C++ | Static Keyword | Question 4](#)
- [C++ | Static Keyword | Question 1](#)
- [C++ | Static Keyword | Question 3](#)
- [C++ | Static Keyword | Question 2](#)
- [C++ | Static Keyword | Question 5](#)
- [Initialization of static variables in C](#)
- [Static data members in C++](#)
- [When are static objects destroyed?](#)
- [Difference between Static and Shared libraries](#)
- ["static const" vs "#define" vs "enum"](#)
- [What are the default values of static variables in C?](#)

Article Tags :

[C](#)
[C-Functions](#)

Practice Tags :

[C](#)

 23

To-do Done
1.8

Based on 71 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) What are the operators that can be and cannot be overloaded in C++?

Next

[last_page](#) How are variables scoped in C – Static or Dynamic?

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes INTERVIEW Geeks Classes INTERVIEW

JAVA APP DEVELOPMENT SUMMER TRAINING COMPETITIVE PROGRAMMING LIVE Java Backend Development

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS



Most popular in C
[getchar\(\) function in C with Examples](#)
[Print all possible combinations of the string by replacing '\\$' with any other digit from the string](#)
[How to use make utility to build C projects?](#)
[ctype.h <ctype.h> library in C/C++ with Examples](#)
[C program to display month by month calendar for a given year](#)

exit(), abort() and assert()

[exit\(\)](#)

[filter_none](#)

[edit](#)

[close](#)

play_arrow

link
brightness_4
code

```
void exit( int status );  
chevron_right
```

filter_none

exit() terminates the process normally.

status: Status value returned to the parent process. Generally, a status value of 0 or EXIT_SUCCESS indicates success, and any other value or the constant EXIT_FAILURE is used to indicate an error. exit() performs following operations.

- * Flushes unwritten buffered data.
- * Closes all open files.
- * Removes temporary files.
- * Returns an integer exit status to the operating system.

The C standard `atexit()` function can be used to customize `exit()` to perform additional actions at program termination.

Example use of `exit`.

filter_none
edit
close

play_arrow

link
brightness_4
code

```
/* exit example */  
#include <stdio.h>  
#include <stdlib.h>
```

```
int main ()
```

```
{
```

```
FILE * pFile;  
pFile = fopen ("myfile.txt", "r");  
if (pFile == NULL)  
{  
    printf ("Error opening file");  
    exit (1);  
}  
else  
{  
    /* file operations here */  
}  
return 0;
```

```
}
```

chevron_right

filter_none

When `exit()` is called, any open file descriptors belonging to the process are closed and any children of the process are inherited by process 1, `init`, and the process parent is sent a `SIGCHLD` signal.

The mystery behind `exit()` is that it takes only integer args in the range 0 – 255 . Out of range exit values can result in unexpected exit codes. An exit value greater than 255 returns an exit code modulo 256. For example, `exit 9999` gives an exit code of 15 i.e. $(9999 \% 256 = 15)$.

Below is the C implementation to illustrate the above fact:

filter_none
edit
close

play_arrow

link
brightness_4
code

```
#include <stdio.h>  
#include <stdlib.h>  
#include <unistd.h>  
#include <sys/types.h>  
#include <sys/wait.h>
```

```
int main(void)
```

```
{
```

```
    pid_t pid = fork();  
  
    if ( pid == 0 )  
    {  
        exit(9999); //passing value more than 255  
    }
```

```
    int status;  
    waitpid(pid, &status, 0);  
  
    if ( WIFEXITED(status) )  
    {  
        int exit_status = WEXITSTATUS(status);  
  
        printf("Exit code: %d\n", exit_status);  
    }
```

```
return 0;
```

chevron_right

filter_none

Output:

Exit code: 15

Note that the above code may not work with online compiler as fork() is disabled.

Explanation: It is effect of 8-bit integer overflow. After 255 (all 8 bits set) comes 0.

So the output is "exit code modulo 256". The output above is actually the modulo of the value 9999 and 256 i.e. 15.

abort()

filter_none

edit

close

play_arrow

link

brightness_4

code

void abort(void);

chevron_right

filter_none

Unlike exit() function, abort() may not close files that are open. It may also not delete temporary files and may not flush stream buffer. Also, it does not call functions registered with atexit().

This function actually terminates the process by raising a SIGABRT signal, and your program can include a handler to intercept this signal (see this).

So programs like below might not write "Geeks for Geeks" to "tempfile.txt"

filter_none

edit

close

play_arrow

link

brightness_4

code

#include<stdio.h>

#include<stdlib.h>

int main()

{

FILE *fp = fopen("C:\\myfile.txt", "w");

if(fp == NULL)

{

printf("\n could not open file ");

getchar();

exit(1);

}

fprintf(fp, "%s", "Geeks for Geeks");

/* */

/* */

/* Something went wrong so terminate here */

abort();

getchar();

return 0;

}

chevron_right

filter_none

If we want to make sure that data is written to files and/or buffers are flushed then we should either use exit() or include a signal handler for SIGABRT.

assert()

filter_none

edit

close

play_arrow

link

brightness_4

code

void assert(int expression);

chevron_right

filter_none

If expression evaluates to 0 (false), then the expression, sourcecode filename, and line number are sent to the standard error, and then abort() function is called. If the identifier NDEBUG ("no debug") is defined with #define NDEBUG then the macro assert does nothing.

Common error outputting is in the form:

Assertion failed: expression, file filename, line line-number

filter_none

edit

close

play_arrow

link

brightness_4

code

#include<assert.h>

void open_record(char *record_name)

{

```
assert(record_name != NULL);
/* Rest of code */
}
```

```
int main(void)
{
    open_record(NULL);
}
```

chevron_right

filter_none

Related Article :

[exit\(\) vs _Exit\(\) in C and C++](#)

This article is contributed by **Rahul Gupta**. Please write comments if you find anything incorrect in the above article or you want to share more information about the topic discussed above.

References:

exit codes

<http://www.cplusplus.com/reference/clibrary/cstdlib/abort/>

<http://www.cplusplus.com/reference/clibrary/cassert/assert/>

http://www.acm.uiuc.edu/webmonkeys/book/c_guide/2.1.html

<https://www.securecoding.cert.org/confluence/display/seccode/ERR06-C.+Understand+the+termination+behavior+of+assert%28%29+and+abort%28%29>

<https://www.securecoding.cert.org/confluence/display/seccode/ERR04-C.+Choose+an+appropriate+termination+strategy>

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up

Add your personal notes
here! (max 5000 chars)

Save

Recommended Posts:

- [exit\(0\) vs exit\(1\) in C/C++ with Examples](#)
- [exit\(\) vs _Exit\(\) in C and C++](#)
- [return statement vs exit\(\) in main\(\)](#)
- [Format specifiers in different Programming Languages](#)
- [C program to find square root of a given number](#)
- [C program to print odd line contents of a File followed by even line content](#)
- [Getting System and Process Information Using C Programming and Shell in Linux](#)
- [Program to calculate Electricity Bill](#)
- [Program to print half Diamond star pattern](#)
- [C/C++ program for calling main\(\) in main\(\)](#)
- [Print all possible combinations of the string by replacing '\\$' with any other digit from the string](#)
- [How to use make utility to build C projects?](#)
- [How to call function within function in C or C++](#)
- [Role of Semicolon in various Programming Languages](#)

Improved By : [mazhar_mik](#)

Article Tags :

C

C-Functions

secure-coding

Practice Tags :

C

thumb_up

4

To-do Done
3.1

Based on 38 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) [Can we use function on left side of an expression in C and C++?](#)

Next

[last_page](#) [When should we write our own copy constructor?](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments



Most popular in C
getch() function in C with Examples
ctype.h(<cctype>) library in C/C++ with Examples
How to use make utility to build C projects?
Print all possible combinations of the string by replacing 'S' with any other digit from the string
C program to display month by month calendar for a given year

More related articles in C
Predefined Macros in C with Examples
Hello World Program: First Program while learning Programming
Implicit Type Conversion in C with Examples
How to call function within function in C or C++
C program to print odd line contents of a File followed by even line content

Implicit return type int in C

Predict the output of following C program.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

#include <stdio.h>
fun(int x)
{
    return x*x;
}
int main(void)
{
    printf("%d", fun(10));
    return 0;
}
chevron_right
```

filter_none

Output: 100

The important thing to note is, there is no return type for fun(), the program still compiles and runs fine in most of the C compilers. In C, if we do not specify a return type, compiler assumes an implicit return type as int. However, C99 standard doesn't allow return type to be omitted even if return type is int. This was allowed in older C standard C89.

In C++, the above program is not valid except few old C++ compilers like Turbo C++. Every function should specify the return type in C++.

This article is contributed by **Pravasi Meet**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up

Save

Recommended Posts:

- [Implicit Type Conversion in C with Examples](#)
- [What is return type of getchar\(\), fgetc\(\) and getc\(\) ?](#)
- [Implicit initialization of variables with 0 or 1 in C](#)
- [Difference between Type Casting and Type Conversion](#)
- [What does main\(\) return in C and C++?](#)
- [return statement in C/C++ with Examples](#)
- [Return from void functions in C++](#)
- [C function argument and return values](#)
- [How to return multiple values from a function in C or C++?](#)
- [return statement vs exit\(\) in main\(\)](#)
- [Return values of printf\(\) and scanf\(\) in C/C++](#)
- [How can I return multiple values from a function?](#)
- [Type Conversion in C](#)
- [What is the size_t data type in C?](#)
- [Is there any need of "long" data type in C and C++?](#)

Article Tags :

C

C-Functions

Practice Tags :

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

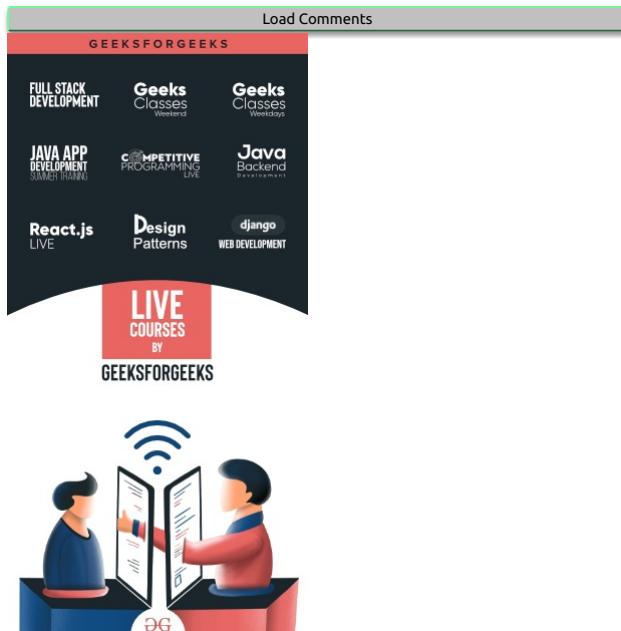
Previous

[first_page](#) C | Functions | Question 11

Next

[last_page](#) C | Advanced Pointer | Question 8

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C
 C program to display month by month calendar for a given year
 Print all possible combinations of the string by replacing '\$' with any other digit from the string
 getch() function in C with Examples
 ctype.h <ctype> library in C/C++ with Examples
 How to use make utility to build C projects?

More related articles in C
 Implicit Type Conversion in C with Examples
 Introduction to the C Programming Language - Part I
 Format specifiers in different Programming Languages
 Hello World Program : First program while learning Programming
 C program to find square root of a given number

What happens when a function is called before its declaration in C?

In C, if a function is called before its declaration, the **compiler assumes return type of the function as int**.

For example, the following program fails in compilation.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

#include <stdio.h>
int main(void)
{
    // Note that fun() is not declared
    printf("%d\n", fun());
    return 0;
}

char fun()
{
    return 'G';
}
chevron_right
```

if the function **char fun()** in above code is defined before main() then it will compile and run perfectly.
 for example, the following program will run properly.

```
filter_none
edit
close
play_arrow
link
brightness_4
code
```

```
#include <stdio.h>

char fun()
{
    return 'G';
}

int main(void)
{
    // Note that fun() is not declared
    printf("%d\n", fun());
    return 0;
}
```

[chevron_right](#)

[filter_none](#)

The following program compiles and runs fine because function is defined before main().

[filter_none](#)

[edit](#)

[close](#)

[play_arrow](#)

[link](#)

[brightness_4](#)

[code](#)

```
#include <stdio.h>
```

```
int fun()
{
    return 10;
}
```

```
int main(void)
```

```
{
    printf("%d\n", fun());
    return 0;
}
```

[chevron_right](#)

[filter_none](#)

What about parameters? compiler assumes nothing about parameters. Therefore, the compiler will not be able to perform compile-time checking of argument types and arity when the function is applied to some arguments. This can cause problems. For example, the following program compiled fine in GCC and produced garbage value as output.

[filter_none](#)

[edit](#)

[close](#)

[play_arrow](#)

[link](#)

[brightness_4](#)

[code](#)

```
#include <stdio.h>
```

```
int main (void)
{
    printf("%d", sum(10, 5));
    return 0;
}

int sum (int b, int c, int a)
{
    return (a+b+c);
}
```

[chevron_right](#)

[filter_none](#)

There is this misconception that the compiler assumes input parameters also int. Had compiler assumed input parameters int, the above program would have failed in compilation.

It is always recommended to declare a function before its use so that we don't see any surprises when the program is run (See [this](#) for more details).

Source:

http://en.wikipedia.org/wiki/Function_prototype#Uses

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes [arrow_drop_up](#)

Add your personal notes here! (max 5000 chars)

[Save](#)

Recommended Posts:

- [When are Constructors Called?](#)
- [When is copy constructor called?](#)
- [C | Variable Declaration and Scope | Question 4](#)
- [C | Variable Declaration and Scope | Question 2](#)
- [C | Variable Declaration and Scope | Question 8](#)
- [C | Variable Declaration and Scope | Question 7](#)
- [C | Variable Declaration and Scope | Question 3](#)
- [C | Variable Declaration and Scope | Question 1](#)
- [C | Variable Declaration and Scope | Question 6](#)
- [What is the difference between single quoted and double quoted declaration of char array?](#)
- [How to call function within function in C or C++](#)

- [arc function in C](#)
- [Inline function in C](#)
- [putchar\(\) function in C](#)
- [strcspn\(\) function in C/C++](#)

Improved By : [aanuj0110](#)

Article Tags :

C
C-Functions
cpp-puzzle
Practice Tags :

C

 15

To-do Done
2.1

Based on **68** vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

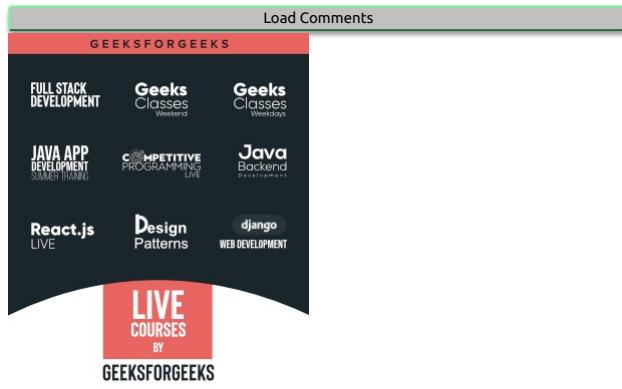
Previous

[first_page](#) Private Destructor

Next

[last_page](#) What happens when more restrictive access is given to a derived class method in C++?

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C
C program to display month by month calendar for a given year
Print all possible combinations of the string by replacing '\$' with any other digit from the string
getch() function in C with Examples
ctype.h<<ctype>> library in C/C++ with Examples
How to make utility to build C projects?

More related articles in C
Input Output Functions in C with Examples
Introduction to the C99 Programming Language : Part I
Format specifiers in different Programming Languages
Hello World Program : First program while learning Programming
C program to find square root of a given number

Noreturn function specifier in C

After the removal of "noreturn" keyword, C11 standard (known as final draft) of C programming language introduce a new "_Noreturn" function specifier that specify that the function does not return to the function that it was called from. If the programmer try to return any value from that function which is declared as _Noreturn type, then the compiler automatically generates a compile time error.

[filter_none](#)
[edit](#)
[close](#)
[play_arrow](#)
[link](#)
[brightness_4](#)
[code](#)

// C program to show how _Noreturn type
// function behave if it has return statement.
#include <stdio.h>
#include <stdlib.h>

// With return value
_Noreturn void view()
{
 return 10;
}
int main(void)

```
printf("NOT over till now\n");
return 0;
}
chevron_right
filter_none
```

Output:

```
Ready to begin...
After that abnormal termination of program.
compiler error:[Warning] function declared 'noreturn' has a 'return' statement
```

```
filter_none
edit
close

play_arrow

link
brightness_4
code

// C program to illustrate the working
// of _Noreturn type function.

#include <stdio.h>
#include <stdlib.h>
```

```
// Nothing to return
_Noreturn void show()
{
    printf("BYE BYE");
}

int main(void)
{
    printf("Ready to begin...\n");
    show();

    printf("NOT over till now\n");
    return 0;
}
```

```
chevron_right
filter_none
```

Output:

```
Ready to begin...
BYE BYE
```

Reference: http://en.cppreference.com/w/c/language/_Noreturn

This article is contributed by **Bishal Kumar Dubey**. If you like GeeksforGeeks and would like to contribute, you can also write an article using [contribute.geeksforgeeks.org](#) or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes 

Add your personal notes
here! (max 5000 chars)

Recommended Posts:

- C++ final specifier
- Understanding constexpr specifier in C++
- Difference between %d and %i format specifier in C language
- How to call function within function in C or C++
- arc function in C
- Inline function in C
- putchar() function in C
- atexit() function in C/C++
- strcspn() function in C/C++
- strrchr() function in C/C++
- gmtime() Function in C/C++
- iswpunct() function in C/C++
- towupper() function in C/C++
- iswblank() function in C/C++
- iswalnum() function in C/C++ STL

Improved By : [AbhinavMadhesiya1](#)

Article Tags :

C

Misc

C-Library

Practice Tags :

Misc

Misc

C



5

To-do Done
3.5

Based on 8 vote(s)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

[Previous](#)

[first_page tmpnam\(\) function in C](#)

[Next](#)

[last_page pthread_equal\(\) in C with example](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Webinars Geeks Classes Workshops

JAVA APP DEVELOPMENT SUMMER TRAINING COMPETITIVE PROGRAMMING LIVE Java Backend Development

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS

Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
Program to Print all Merges in C with Examples
Format specifiers in different Programming Languages
C program to print odd line contents of a File followed by even line content
Introduction to the C99 Programming Language : Part II

Most visited in Misc
Eggs dropping puzzle | Set 2
Introduction of JIRA
Components of Image Processing System
Types of Models in Object Oriented Modeling and Design
Features of Blockchain

exit() vs _Exit() in C and C++

In C, `exit()` terminates the calling process without executing the rest code which is after the `exit()` function.

Example:-

```
filter_none
edit
close

play_arrow

link
brightness_4
code

// C program to illustrate exit() function.
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    printf("START");

    exit(0); // The program is terminated here

    // This line is not printed
    printf("End of program");
}
```

Output:

START

Now the question is that if we have `exit()` function then why C11 standard introduced `_Exit()`? Actually `exit()` function performs some cleaning before termination of the program like connection termination, buffer flushes etc. The `_Exit()` function in C/C++ gives normal termination of a program without performing any cleanup tasks. For example it does not execute functions registered with `atexit`.

Syntax:

```
// Here the exit_code represent the exit status
// of the program which can be 0 or non-zero.
// The _Exit() function returns nothing.
void _Exit(int exit_code);
```

```
filter_none
edit
close

play_arrow

link
brightness_4
code
```

```
// C++ program to demonstrate use of _Exit()
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    int exit_code = 10;
    printf("Termination using _Exit");
    _Exit(exit_code);
}
```

[chevron_right](#)

[filter_none](#)

Output:

Showing difference through programs:

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// A C++ program to show difference
// between exit() and _Exit()
#include<bits/stdc++.h>
using namespace std;

void fun(void)
{
    cout << "Exiting";
}

int main()
{
    atexit(fun);
    exit(10);
}
```

[chevron_right](#)

[filter_none](#)

Output

[Exiting](#)

If we replace exit with _Exit(), then nothing is printed.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// A C++ program to show difference
// between exit() and _Exit()
#include<bits/stdc++.h>
using namespace std;

void fun(void)
{
    cout << "Exiting";
}

int main()
{
    atexit(fun);
    _Exit(10);
}
```

[chevron_right](#)

[filter_none](#)

Output

This article is contributed by **Bishal Kumar Dubey**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes [arrow_drop_up](#)

[Save](#)

Recommended Posts:

- [exit\(0\) vs exit\(1\) in C/C++ with Examples](#)
- [exit\(\), abort\(\) and assert\(\)](#)

- return statement vs exit() in main()
- Exit status of a child process in Linux
- std::basic_istream::ignore in C++ with Examples
- std::basic_istream::getline in C++ with Examples
- Format specifiers in different Programming Languages
- C program to find square root of a given number
- C program to print odd line contents of a File followed by even line content
- std::is_heap() in C++ with Examples
- std::basic_istream::gcount() in C++ with Examples
- new vs malloc() and free() vs delete in C++
- std::string::rfind in C++ with Examples
- Sum of factorials of Prime numbers in a Linked list

Improved By : Vishal Vishwakarma

Article Tags :

C
C++
CPP-Library
system-programming

Practice Tags :

C
CPP



6

To-do Done
2.5

Based on 12 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) Octal numbers in c

Next

[last_page](#) Generic keyword in C

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Weekend Geeks Classes Weekdays

JAVA APP DEVELOPMENT COMPETITIVE PROGRAMMING LIFE Java Backend DEVELOPMENT

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS

Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
Predefined Macros in C with Examples
C program to print odd line contents of a File followed by even line content
Introduction to the C99 Programming Language : Part II
C program to find square root of a given number

Most visited in C++
Vector of Vectors in C++ STL with Examples
C++ Tutorial
Which C++ libraries are useful for competitive programming?
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

Predefined Identifier `__func__` in C

Before we start discussing about `__func__`, let us write some code snippet and anticipate the output:

```
filter_none
edit
close
play_arrow
link
brightness_4
code
#include <stdio.h>

int main()
{
    printf("%s",__func__);
    return 0;
}
```

chevron_right

filter_none

Will it compile error due to not defining variable `_func_`? Well, as you would have guessed so far, it won't give any compile error and it'd print `main!`

C language standard (i.e. C99 and C11) defines a predefined identifier as follows in clause 6.4.2.2:

The identifier `_func_` shall be implicitly declared by the translator as if, immediately following the opening brace of each function definition, the declaration
`static const char _func_[] = "function-name";`

appeared, where `function-name` is the name of the lexically-enclosing function."

It means that C compiler implicitly adds `_func_` in every function so that it can be used in that function to get the function name. To understand it better, let us write this code:

filter_none
edit
close

play_arrow

link
brightness_4
code

```
#include <stdio.h>
void foo(void)
{
    printf("%s", __func__);
}
void bar(void)
{
    printf("%s", __func__);
}
```

int main()

{
 foo();
 bar();
 return 0;
}

chevron_right

filter_none

And it'll give output as `foobar`. A use case of this predefined identifier could be logging the output of a big program where a programmer can use `_func_` to get the current function instead of mentioning the complete function name explicitly. Now what happens if we define one more variable of name `_func_`

filter_none
edit
close

play_arrow

link
brightness_4
code

```
#include <stdio.h>
```

```
int __func__ = 10;
int main()
{
```

`printf("%d", __func__);`

`return 0;`

}

chevron_right

filter_none

Since C standard says compiler implicitly defines `_func_` for each function as the function-name, we should not define `_func_` at the first place. You might get error but C standard says "undefined behaviour" if someone explicitly defines `_func_`.

Just to finish the discussion on Predefined Identifier `_func_`, let us mention Predefined Macros as well (such as `_FILE_` and `_LINE_` etc.) Basically, C standard clause 6.10.8 mentions several predefined macros out of which `_FILE_` and `_LINE_` are of relevance here.

It's worthwhile to see the output of the following code snippet:

filter_none
edit
close

play_arrow

link
brightness_4
code

```
#include <stdio.h>
```

int main()

{
 printf("In file:%s, function:%s() and line:%d", __FILE__, __func__, __LINE__);

`return 0;`

}

chevron_right

filter_none

Instead of explaining the output, we will leave this to you to guess and understand the role of `_FILE_` and `_LINE_`!

Please do Like/Tweet/G+1 if you find the above useful. Also, please do leave us comment for further clarification or info. We would love to help and learn

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- Predefined Macros in C with Examples
- Format specifiers in different Programming Languages
- C program to find square root of a given number
- C program to print odd line contents of a File followed by even line content
- std::string::rfind in C++ with Examples
- Generate an array of given size with equal count and sum of odd and even numbers
- Getting System and Process Information Using C Programming and Shell in Linux
- Program to calculate Electricity Bill
- Program to print half Diamond star pattern
- C/C++ program for calling main() in main()
- Print all possible combinations of the string by replacing '\$' with any other digit from the string
- Dropbox - An Introduction
- How to use make utility to build C projects?
- How to call function within function in C or C++

Improved By : [avsadityavardhan](#)

Article Tags :

[Articles](#)

[C](#)

[C-Library](#)

Practice Tags :

[C](#)



10

To-do Done
2.6

Based on 8 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) [C Quiz - 110](#) | Question 5

Next

[last_page](#) [C Quiz - 111](#) | Question 1

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Weekend Geeks Classes Weekdays

JAVA APP DEVELOPMENT COMPETITIVE PROGRAMMING LIVE Java Backend

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS

Most popular articles

- Must Do Coding Questions for Companies like Amazon, Microsoft, Adobe, ...
- Must Do Coding Questions Company-wise
- 10 Projects That Every Developer Should Lay Their Hands-On
- C++ Tutorial
- Map of Vectors in C++ STL with Examples

Most visited in C

- Print in C++ with Examples
- Special Characters executions with help of Pragma in C/C++
- Modulo Operator (%) in C/C++ with Examples
- Program to calculate Electricity Bill
- Role of SemiColon in various Programming Languages

Callbacks in C

A callback is any executable code that is passed as an argument to other code, which is expected to call back (execute) the argument at a given time [Source : [Wiki](#)]. In simple language, If a reference of a function is passed to another function as an argument to call it, then it will be called as a Callback function.

In C, a callback function is a function that is called through a [function pointer](#).

Below is a simple example in C to illustrate the above definition to make it more clear:

[filter_none](#)
[edit](#)
[close](#)

```

play_arrow
link
brightness_4
code

// A simple C program to demonstrate callback
#include<stdio.h>

void A()
{
    printf("I am function A\n");
}

// callback function
void B(void (*ptr)())
{
    (*ptr)(); // callback to A
}

int main()
{
    void (*ptr)() = &A;

    // calling function B and passing
    // address of the function A as argument
    B(ptr);

    return 0;
}

```

chevron_right

filter_none

I am **function A**

In C++ STL, **functors** are also used for this purpose.

This article is contributed by **Ranju Kumari**. If you like GeeksforGeeks and would like to contribute, you can also write an article using [contribute.geeksforgeeks.org](#) or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes **arrow_drop_up**

Save

Recommended Posts:

- [std::basic_istream::ignore in C++ with Examples](#)
- [std::basic_istream::getline in C++ with Examples](#)
- [Format specifiers in different Programming Languages](#)
- [C program to find square root of a given number](#)
- [C program to print odd line contents of a File followed by even line content](#)
- [std::is_heap\(\) in C++ with Examples](#)
- [std::basic_istream::gcount\(\) in C++ with Examples](#)
- [Getting System and Process Information Using C Programming and Shell in Linux](#)
- [Program to calculate Electricity Bill](#)
- [Program to print half Diamond star pattern](#)
- [C/C++ program for calling main\(\) in main\(\)](#)
- [log2\(\) function in C++ with Examples](#)
- [Print all possible combinations of the string by replacing '\\$' with any other digit from the string](#)
- [std::bit_or in C++ with Examples](#)

Improved By : Subhajit Mandal

Article Tags :

C

CPP-Functions

Practice Tags :

C

thumb_up

10

To-do Done
2.8

Based on 17 vote(s)

Basic **Easy** **Medium** **Hard** **Expert**

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) [difftime\(\) C library function](#)

Next

[last_page](#) [kbhit in C language](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
`ctype.h <cctype>` library in C/C++ with Examples
Predefined Macros in C with Examples
Implicit Type Conversion in C with Examples
How to call function within function in C or C++
More related articles in C
C program to print odd line contents of a File followed by even line content
Introduction to C/C++ Programming Languages : Part II
C program to find square root of a given number
Format specifiers in different Programming Languages
Problem in comparing Floating point numbers and how to compare them correctly?

Nested functions in C

Some programmer thinks that defining a function inside an another function is known as "nested function". But the reality is that it is not a nested function, it is treated as lexical scoping. Lexical scoping is not valid in C because the compiler can't reach/find the correct memory location of the inner function.

Nested function **is not supported** by C because we cannot define a function within another function in C. We can declare a function inside a function, but it's not a nested function.
Because nested functions definitions can not access local variables of the surrounding blocks, they can access only global variables of the containing module. This is done so that lookup of global variables doesn't have to go through the directory. As in C, there are two nested scopes: local and global (and beyond this, built-ins). Therefore, nested functions have only a limited use. If we try to approach nested function in C, then we will get compile time error.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// C program to illustrate the
// concept of Nested function.
#include <stdio.h>
int main(void)
{
    printf("Main");
    int fun()
    {
        printf("fun");

        // defining view() function inside fun() function.
        int view()
        {
            printf("view");
        }
        return 1;
    }
    view();
}
chevron_right
```

Output:

Compile time error: undefined reference to `view'

An extension of the GNU C Compiler allows the declarations of nested functions. The declarations of nested functions under GCC's extension need to be prefix/start with the `auto` keyword.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// C program of nested function
// with the help of gcc extension
#include <stdio.h>
int main(void)
{
    auto int view(); // declare function with auto keyword
    view(); // calling function
```

```
printf("Main\n");
```

```
int view()
{
    printf("View\n");
    return 1;
}

printf("GEEKS");
return 0;
}
```

chevron_right

filter_none

Output:

```
View
Main
GEEKS
```

This article is contributed by **Bishal Kumar Dubey**. If you like GeeksforGeeks and would like to contribute, you can also write an article using [contribute.geeksforgeeks.org](#) or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes *arrow_drop_up*

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- Nested Loops in C with Examples
- Nested switch case
- A nested loop puzzle
- C++ | Nested Ternary Operator
- Functions in C/C++
- Functions that cannot be overloaded in C++
- Pure Functions
- Static functions in C
- Thread functions in C/C++
- Macros vs Functions
- C | Functions | Question 4
- C | Functions | Question 6
- C | Functions | Question 7
- C | Functions | Question 8
- C | Functions | Question 9

Article Tags :

C
C-Functions

Practice Tags :

C

thumb_up

7

To-do Done
2.1

Based on 7 vote(s)

Basic **Easy** **Medium** **Hard** **Expert**

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

first_page Interesting facts about data-types and modifiers in C/C++

Next

last_page Differentiate printable and control character in C ?

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
`ctype.h <cctype>` library in C/C++ with Examples
Predefined Macros in C with Examples
Implicit Type Conversion in C with Examples
How to call function within function in C or C++
More related articles in C
C program to print odd line contents of a File followed by even line content
Introduction to C/C++ Programs | Lecture 2 : Part II
C program to find square root of a given number
Format specifiers in different Programming Languages
Problem in comparing Floating point numbers and how to compare them correctly?

Parameter Passing Techniques in C/C++

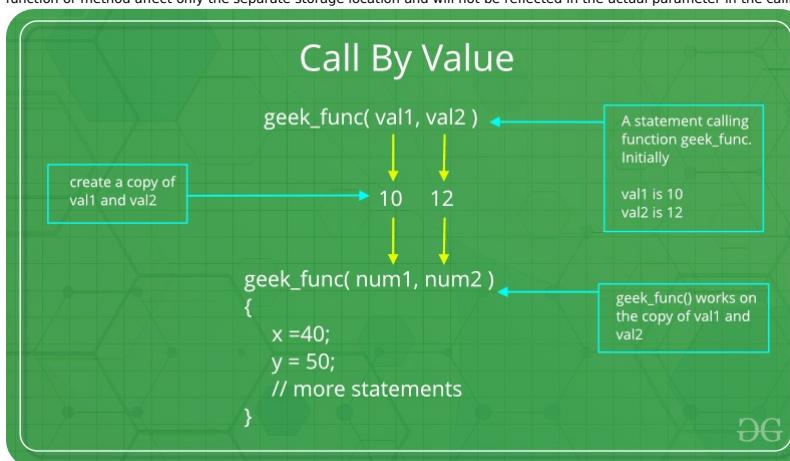
There are different ways in which parameter data can be passed into and out of methods and functions. Let us assume that a function *B()* is called from another function *A()*. In this case *A* is called the "**caller function**" and *B* is called the "**called function or callee function**". Also, the arguments which *A* sends to *B* are called *actual arguments* and the parameters of *B* are called *formal arguments*.

Terminology

- **Formal Parameter :** A variable and its type as they appear in the prototype of the function or method.
- **Actual Parameter :** The variable or expression corresponding to a formal parameter that appears in the function or method call in the calling environment.
- **Modes:**
 - **IN:** Passes info from caller to callee.
 - **OUT:** Callee writes values in caller.
 - **IN/OUT:** Caller tells callee value of variable, which may be updated by callee.

Important methods of Parameter Passing

1. **Pass By Value :** This method uses *in-mode* semantics. Changes made to formal parameter do not get transmitted back to the caller. Any modifications to the formal parameter variable inside the called function or method affect only the separate storage location and will not be reflected in the actual parameter in the calling environment. This method is also called as *call by value*.



```

filter_none
edit
close
play_arrow
link
brightness_4
code

// C program to illustrate
// call by value
#include <stdio.h>

void func(int a, int b)
{
    a += b;
    printf("In func, a = %d b = %d\n", a, b);
}
int main(void)
{
    int x = 5, y = 7;

    // Passing parameters
  
```

```

func(x, y);
printf("In main, x = %d y = %d\n", x, y);
return 0;
}
chevron_right

```

filter_none

Output:

```

In func, a = 12 b = 7
In main, x = 5 y = 7

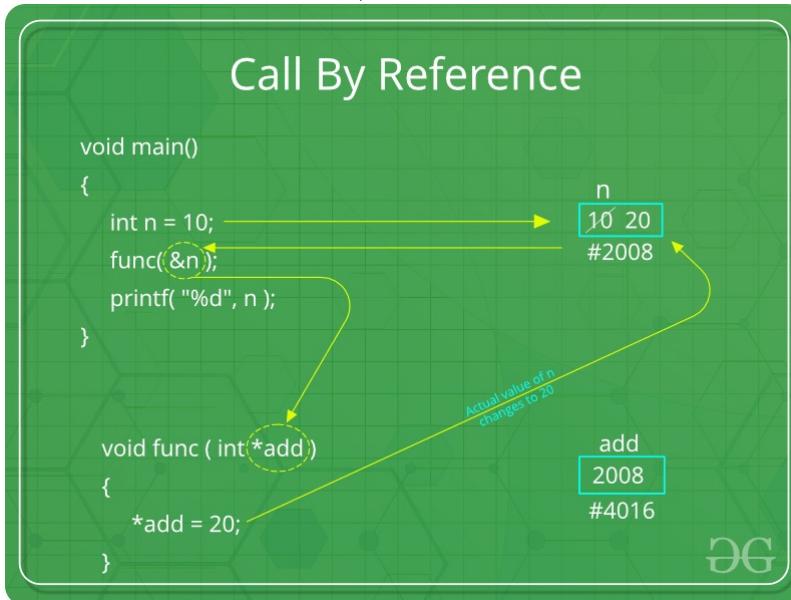
```

Languages like C, C++, Java support this type of parameter passing. **Java in fact is strictly call by value.**

Shortcomings:

- Inefficiency in storage allocation
- For objects and arrays, the copy semantics are costly

2. **Pass by reference(aliasing)** : This technique uses *in/out-mode* semantics. Changes made to formal parameter do get transmitted back to the caller through parameter passing. Any changes to the formal parameter are reflected in the actual parameter in the calling environment as formal parameter receives a reference (or pointer) to the actual data. This method is also called as **call by reference**. This method is efficient in both time and space.



filter_none

edit

close

play_arrow

link

brightness_4

code

```
// C program to illustrate
// call by reference
#include <stdio.h>
```

```
void swapnum(int* i, int* j)
{
    int temp = *i;
    *i = *j;
    *j = temp;
}
```

```
int main(void)
{
    int a = 10, b = 20;

    // passing parameters
    swapnum(&a, &b);

    printf("a is %d and b is %d\n", a, b);
    return 0;
}
```

chevron_right

filter_none

Output:

```
a is 20 and b is 10
```

C and C++ both support call by value as well as call by reference whereas Java doesn't support call by reference.

Shortcomings:

- Many potential scenarios can occur
- Programs are difficult to understand sometimes

Other methods of Parameter Passing

These techniques are older and were used in earlier programming languages like Pascal, Algol and Fortran. These techniques are not applicable in high level languages.

1. **Pass by Result** : This method uses *out-mode* semantics. Just before control is transferred back to the caller, the value of the formal parameter is transmitted back to the actual parameter. This method is sometimes called *call by result*. In general, pass by result technique is implemented by copy.
2. **Pass by Value-Result** : This method uses *in/out-mode* semantics. It is a combination of Pass-by-Value and Pass-by-result. Just before the control is transferred back to the caller, the value of the formal parameter is transmitted back to the actual parameter. This method is sometimes called as *call by value-result*
3. **Pass by name** : This technique is used in programming language such as **Algol**. In this technique, symbolic "name" of a variable is passed, which allows it both to be accessed and updated.

Example:

To double the value of *C[i]*, you can pass its name (not its value) into the following procedure.

```
procedure double(x);
  real x;
begin
  x:=x*2
end;
```

In general, the effect of pass-by-name is to textually substitute the argument in a procedure call for the corresponding parameter in the body of the procedure.

Implications of Pass-by-Name mechanism:

- The argument expression is re-evaluated each time the formal parameter is passed.
- The procedure can change the values of variables used in the argument expression and hence change the expression's value.

This article is contributed by **Krishna Bhatia**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes [arrow_drop_up](#)

Add your personal notes
here! (max 5000 chars)

Save

Recommended Posts:

- Passing by pointer Vs Passing by Reference in C++
- How to pass a 2D array as a parameter in C?
- Difference between Argument and Parameter in C/C++ with Examples
- How to print size of array parameter in C++?
- Optimization Techniques | Set 2 (swapping)
- Passing vector to a function in C++
- How to delete an element from the Set by passing its value in C++
- Passing and Returning Objects in C++
- Passing a vector to constructor in C++
- Passing NULL to printf in C
- Passing Reference to a Pointer in C++
- Equation of straight line passing through a given point which bisects it into two equal line segments
- std::basic_istream::ignore in C++ with Examples
- std::basic_istream::getline in C++ with Examples

Article Tags :

C

C++

Practice Tags :

C

CPP

thumb_up

11

To-do Done
2.1

Based on 16 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) std::basic_string::at vs std::basic_string::operator[]

Next

[last_page](#) Difference between Relational operator(==) and std::string::compare() in C++

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Weekend Geeks Classes Weekdays

JAVA APP DEVELOPMENT SUMMER TRAINING COMPETITIVE PROGRAMMING LIVE Java Backend Backend

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
[ctype.h](#) [ccattyarray](#) in C++ with Examples
Prefetching Macros in C with Examples
Implicit Type Conversion in C with Examples
Format specifiers in different Programming Languages

Power Function in C/C++

Given two numbers base and exponent, pow() function finds x raised to the power of y i.e. x^y . Basically in C exponent value is calculated using the pow() function.

Example:

```
Input: 2.0, 5.0
Output: 32
Explanation:
pow(2.0, 5.0) executes 2.0 raised to
the power 5.0, which equals 32

Input: 5.0, 2.0
Output: 25
Explanation:
pow(5.0, 2.0) executes 5.0 raised to
the power 2.0, which equals 25
```

Syntax:

```
double pow(double x, double y);
```

Parameters: The method takes two arguments:

- **x** : floating point base value
- **y** : floating point power value

Program:

C

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// C program to illustrate
// power function
#include <math.h>
#include <stdio.h>

int main()
{
    double x = 6.1, y = 4.8;

    // Storing the answer in result.
    double result = pow(x, y);
    printf("%.2lf", result);

    return 0;
}
chevron_right
filter_none
```

C++

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// CPP program to illustrate
// power function
#include <bits/stdc++.h>
using namespace std;

int main()
{
    double x = 6.1, y = 4.8;

    // Storing the answer in result.
    double result = pow(x, y);

    // printing the result upto 2
    // decimal place
    cout << fixed << setprecision(2) << result << endl;

    return 0;
}
chevron_right
filter_none
```

Output:

```
5882.79
```

The `pow()` function takes 'double' as the arguments and returns a 'double' value. This function does not always work for integers. One such example is `pow(5, 2)`. When assigned to an integer, it outputs 24 on some compilers and works fine for some other compilers. But `pow(5, 2)` without any assignment to an integer outputs 25.

- This is because 5^2 (i.e. 25) might be stored as 24.999999 or 25.0000000001 because the return type is double. When assigned to int, 25.0000000001 becomes 25 but 24.999999 will give output 24.
- To overcome this and output the accurate answer in integer format, we can add 0.5 to the result and typecast it to `int` e.g `(int)(pow(5, 2)+0.5)` will give the correct answer(25, in above example), irrespective of the compiler.

C

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// C program to illustrate
// working with integers in
// power function
#include <math.h>
#include <stdio.h>

int main()
{
    int a;

    // Using typecasting for
    // integer result
    a = (int)(pow(5, 2) + 0.5);
    printf("%d", a);

    return 0;
}
```

C++

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// CPP program to illustrate
// working with integers in
// power function
#include <bits/stdc++.h>
using namespace std;
int main()
{
    int a;

    // Using typecasting for
    // integer result
    a = (int)(pow(5, 2) + 0.5);
    cout << a;

    return 0;
}
```

This article is contributed by **Arushi Dhamija** and **Jatin Goyal**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes 

Recommended Posts:

- [Function Overloading vs Function Overriding in C++](#)
- [How to call function within function in C or C++](#)
- [Difference between Virtual function and Pure virtual function in C++](#)
- [What happens when a virtual function is called inside a non-virtual function in C++](#)
- [fma\(\) function in C++](#)
- [div\(\) function in C++](#)
- [arc function in C](#)
- [exp\(\) function C++](#)
- [log\(\) function in C++](#)
- [imag\(\) function in C++](#)

- `real()` function in C++
- `fegetenv()` function in C/C++
- `wmemcp()` function in C/C++
- `wctrans()` function in C/C++
- `fegetexceptflag()` function in C/C++

Article Tags :

C
C++
C-Library
CPP-Library
Practice Tags :
C
CPP

thumb_up
21

To-do Done
2.2

Based on 17 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

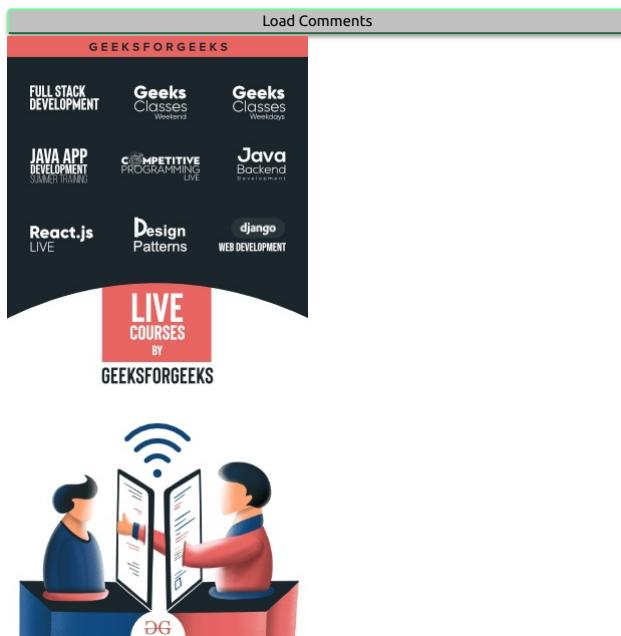
Previous

first_page `std::string::data()` in C++

Next

last_page Non-blocking I/O with pipes in C

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
`ctype.h <ctype.h>` library in C/C++ with Examples
Implicit Type Conversion in C with Examples
How to call function within function in C or C++
Format specifiers in different Programming Languages

Most visited in C++
Vector of Vectors in C++ STL with Examples
`C++11`
Which C++ libraries are useful for competitive programming?
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

tolower() function in C

The `tolower()` function is defined in the `ctype.h` header file. If the character passed is a uppercase alphabet then the `tolower()` function converts a uppercase alphabet to an lowercase alphabet.

Syntax:

```
int tolower(int ch);
```

Parameter: This method takes a mandatory parameter `ch` which is the character to be converted to lowercase.

Return Value: This function returns the **lowercase character** corresponding to the `ch`.

Below programs illustrate the `tolower()` function in C:

Example 1:-

```
filter_none
edit
close

play_arrow

link
brightness_4
code

// C program to demonstrate
// example of tolower() function.

#include <ctype.h>
#include <stdio.h>
```

```
int main()
{
    // Character to be converted to lowercase
    char ch = 'G';

    // convert ch to lowercase using toLower()
    printf("%c in lowercase is represented as = %c", ch, tolower(ch));

    return 0;
}
```

chevron_right

filter_none

Output:

© in lowercase is represented as = g

Example 2:-

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// C program to demonstrate
// example of tolower() function.
```

```
#include <ctype.h>
#include <stdio.h>

int main()
{
    int j = 0;
    char str[] = "GEEKSFORGEEKS\n";

    // Character to be converted to lowercase
    char ch = 'G';

    // convert ch to lowercase using toLower()
    char ch;

    while (str[j]) {
        ch = str[j];

        // convert ch to lowercase using toLower()
        putchar(tolower(ch));

        j++;
    }

    return 0;
}
```

chevron_right

filter_none

Output:

geeksforgeeks

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes **arrow_drop_up**

Add your personal notes
here! (max 5000 chars)

Save

Recommended Posts:

- [How to call function within function in C or C++](#)
- [arc function in C](#)
- [mbsrtowcs\(\) function in C/C++](#)
- [strcmpi\(\) function in C](#)
- [vswprintf\(\) function in C/C++](#)
- [wcrtomb\(\) function in C/C++](#)
- [wcstoul\(\) function in C/C++](#)
- [wmemcmp\(\) function in C/C++](#)
- [wmemchr\(\) function in C/C++](#)
- [fgetenv\(\) function in C/C++](#)
- [wctrans\(\) function in C/C++](#)
- [fegetexceptflag\(\) function in C/C++](#)
- [wmemcpy\(\) function in C/C++](#)
- [wcstoi\(\) function in C/C++](#)
- [wcstod\(\) function in C/C++](#)



bansal_rtk_

Check out this Author's contributed articles.

If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please Improve this article if you find anything incorrect by clicking on the "Improve Article" button below.

Article Tags :

C
C-Functions
C-Library
C-String
Practice Tags :

C



3

To-do Done
1

Based on 1 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page toupper\(\) function in C](#)

Next

[last_page strcmpi\(\) function in C](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes (Weekend) Geeks Classes (Weekdays)

JAVA APP DEVELOPMENT SUMMER TRAINING COMPETITIVE PROGRAMMING LIVE Java Backend

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS

Most popular in C
Print all possible combinations of the string by replacing 'S' with any other digit from the string
Program to print even line content of a file
Introduction to the C99 Programming Language : Part II
C program to find square root of a given number

More related articles in C
Format specifiers in different Programming Languages
Problem in comparing Floating point numbers and how to compare them correctly
Features of C Programming Language
Pointer Expressions in C with Examples
Introduction to the C99 Programming Language : Part III

time() function in C

The time() function is defined in time.h (ctime in C++) header file. This function returns the time since 00:00:00 UTC, January 1, 1970 (Unix timestamp) in seconds. If second is not a null pointer, the returned value is also stored in the object pointed to by second.

Syntax:

```
time_t time( time_t *second )
```

Parameter: This function accepts single parameter second. This parameter is used to set the time_t object which store the time.

Return Value: This function returns current calendar time as a object of type time_t.

Program 1:

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// C program to demonstrate
// example of time() function.
#include <stdio.h>
#include <time.h>

int main ()
{
    time_t seconds;

    seconds = time(NULL);
    printf("Seconds since January 1, 1970 = %ld\n", seconds);

    return(0);
}
```

chevron_right

filter_none

Output:

```
Seconds since January 1, 1970 = 1538123990
```

Example 2:

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// C program to demonstrate
// example of time() function.

#include <stdio.h>
#include <time.h>

int main()
{
    time_t seconds;

    // Stores time seconds
    time(&seconds);
    printf("Seconds since January 1, 1970 = %ld\n", seconds);

    return 0;
}
```

chevron_right

filter_none

Output:

```
Seconds since January 1, 1970 = 1538123990
```

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up

Add your personal notes
here! (max 5000 chars)

Save

Recommended Posts:

- [How to measure time taken by a function in C?](#)
- [time.h localtime\(\) function in C with Examples](#)
- [Time delay in C](#)
- [time.h header file in C with Examples](#)
- [Measure execution time with high precision in C/C++](#)
- [C program to print digital clock with current time](#)
- [C event Interview Experience \(On campus for Internship and Full Time\)](#)
- [Sorting integer data from file and calculate execution time](#)
- [C program for Time Complexity plot of Bubble, Insertion and Selection Sort using Gnuplot](#)
- [How to call function within function in C or C++](#)
- [arc function in C](#)
- [clock\(\) function in C/C++](#)
- [tolower\(\) function in C](#)
- [toupper\(\) function in C](#)
- [strlwr\(\) function in C](#)



bansal_rtk_

Check out this Author's contributed articles.

If you like GeeksforGeeks and would like to contribute, you can also write an article using [contribute.geeksforgeeks.org](#) or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please Improve this article if you find anything incorrect by clicking on the "Improve Article" button below.

Article Tags :

C
C Programs
C-Functions
C-Library

Practice Tags :

C



3

To-do Done
1.5

Based on 2 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) [strnset\(\) function in C](#)

Next

[last_page](#) [strupr\(\) function in C](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Webinars Geeks Classes Webinars

JAVA APP DEVELOPMENT SUMMER TRAINING COMPETITIVE PROGRAMMING LIVE Java Backend Development

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS

Most popular in C
Print all possible combinations of the string by replacing 'S' with any other digit from the string
Format Specifiers in different programming Languages
C program to find square root of a given number
Predefined Macros in C with Examples
How to call function within function in C or C++

Most visited in C Programs
C program to sort an array in ascending order
Check whether the given string is Palindrome using Stack
C/C++ program to print Hello World without using main() and semicolon
Value of PI() up to 50 decimal places
Program to find absolute value of a given number

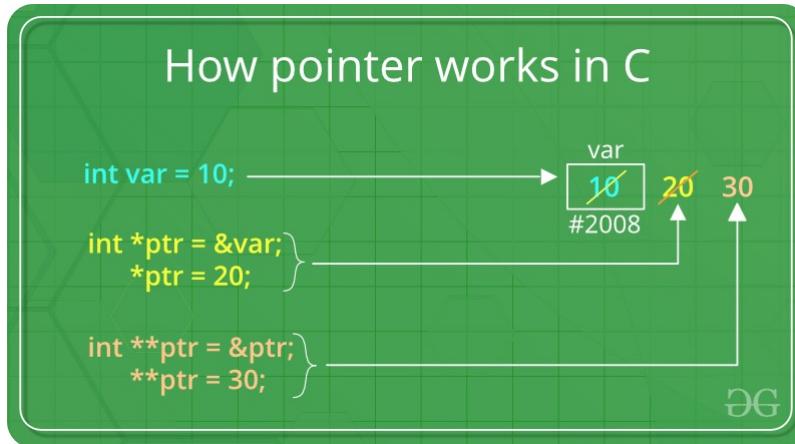
Pointers in C and C++ | Set 1 (Introduction, Arithmetic and Array)

Pointers store address of variables or a memory location.

```
// General syntax
datatype *var_name;
```

```
// An example pointer "ptr" that holds
// address of an integer variable or holds
// address of a memory whose value(s) can
// be accessed as integer values through "ptr"
int *ptr;
```

Using a Pointer:



To use pointers in C, we must understand below two operators.

- To access address of a variable to a pointer, we use the unary operator & (ampersand) that returns the address of that variable. For example &x gives us address of variable x.

filter_none

edit

close

play_arrow

link

brightness_4

code

```
// The output of this program can be different
// in different runs. Note that the program
// prints address of a variable and a variable
// can be assigned different address in different
// runs.
```

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int x;
```

```
    // Prints address of x
```

```
    printf("%p", &x);
```

```
    return 0;
```

chevron_right

filter_none

- One more operator is **unary *** (Asterisk) which is used for two things :

- To declare a pointer variable: When a pointer variable is declared in C/C++, there must be a * before its name.

filter_none

edit

close

play_arrow

link

brightness_4

code

```
// C program to demonstrate declaration of
// pointer variables.
```

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int x = 10;
```

```
// 1) Since there is * in declaration, ptr
// becomes a pointer variable (a variable
// that stores address of another variable)
// 2) Since there is int before *, ptr is
// pointer to an integer type variable
int *ptr;
```

```
// & operator before x is used to get address
// of x. The address of x is assigned to ptr.
ptr = &x;
```

```
return 0;
```

}

chevron_right

filter_none

- To access the value stored in the address we use the unary operator (*) that returns the value of the variable located at the address specified by its operand. This is also called **Dereferencing**.

C++

```

filter_none
edit
close
play_arrow
link
brightness_4
code

// C++ program to demonstrate use of * for pointers in C++
#include <iostream>
using namespace std;

int main()
{
    // A normal integer variable
    int Var = 10;

    // A pointer variable that holds address of var.
    int *ptr = &Var;

    // This line prints value at address stored in ptr.
    // Value stored is value of variable "var"
    cout << "Value of Var = " << *ptr << endl;

    // The output of this line may be different in different
    // runs even on same machine.
    cout << "Address of Var = " << ptr << endl;

    // We can also use ptr as lvalue (Left hand
    // side of assignment)
    *ptr = 20; // Value at address is now 20

    // This prints 20
    cout << "After doing *ptr = 20, *ptr is " << *ptr << endl;

    return 0;
}

// This code is contributed by
// shubhamsingh10
chevron_right
filter_none

```

C

```

filter_none
edit
close
play_arrow
link
brightness_4
code

// C program to demonstrate use of * for pointers in C
#include <stdio.h>

int main()
{
    // A normal integer variable
    int Var = 10;

    // A pointer variable that holds address of var.
    int *ptr = &Var;

    // This line prints value at address stored in ptr.
    // Value stored is value of variable "var"
    printf("Value of Var = %d\n", *ptr);

    // The output of this line may be different in different
    // runs even on same machine.
    printf("Address of Var = %p\n", ptr);

    // We can also use ptr as lvalue (Left hand
    // side of assignment)
    *ptr = 20; // Value at address is now 20

    // This prints 20
    printf("After doing *ptr = 20, *ptr is %d\n", *ptr);

    return 0;
}

chevron_right
filter_none

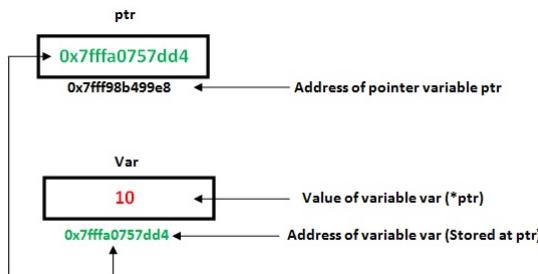
```

Output :

Value of Var = 10

```
Address of Var = 0x7ffffa057dd4  
After doing *ptr = 20, *ptr is 20
```

Below is pictorial representation of above program:



Pointer Expressions and Pointer Arithmetic

A limited set of arithmetic operations can be performed on pointers. A pointer may be:

- incremented (`++`)
- decremented (`--`)
- an integer may be added to a pointer (`+` or `+=`)
- an integer may be subtracted from a pointer (`-` or `-=`)

Pointer arithmetic is meaningless unless performed on an array.

Note : Pointers contain addresses. Adding two addresses makes no sense, because there is no idea what it would point to. Subtracting two addresses lets you compute the offset between these two addresses.

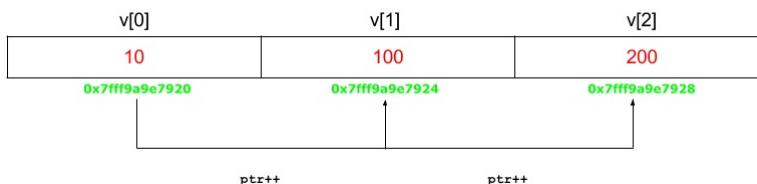
```
filter_none  
edit  
close  
  
play_arrow  
  
link  
brightness_4  
code  
  
// C++ program to illustrate Pointer Arithmetic  
// in C/C++  
#include <bits/stdc++.h>
```

```
// Driver program  
int main()  
{  
    // Declare an array  
    int v[3] = {10, 100, 200};  
  
    // Declare pointer variable  
    int *ptr;  
  
    // Assign the address of v[0] to ptr  
    ptr = v;  
  
    for (int i = 0; i < 3; i++)  
    {  
        printf("Value of *ptr = %d\n", *ptr);  
        printf("Value of ptr = %p\n\n", ptr);  
  
        // Increment pointer ptr by 1  
        ptr++;  
    }  
}
```

chevron_right

filter_none

```
Output: Value of *ptr = 10  
Value of ptr = 0x7ffcae30c710  
  
Value of *ptr = 100  
Value of ptr = 0x7ffcae30c714  
  
Value of *ptr = 200  
Value of ptr = 0x7ffcae30c718
```



Array Name as Pointers

An array name acts like a pointer constant. The value of this pointer constant is the address of the first element. For example, if we have an array named `val` then `val` and `&val[0]` can be used interchangeably.

```
filter_none  
edit  
close  
  
play_arrow  
  
link  
brightness_4  
code
```

```
// C++ program to illustrate Array Name as Pointers in C++
```

```
#include <bits/stdc++.h>
```

```
using namespace std;
```

```
void geeks()
```

```
{
```

```
    // Declare an array  
    int val[3] = { 5, 10, 15};
```

```
    // Declare pointer variable  
    int *ptr;
```

```
    // Assign address of val[0] to ptr.  
    // We can use ptr=&val[0];(both are same)  
    ptr = val ;  
    cout << "Elements of the array are: ";  
    cout << ptr[0] << " " << ptr[1] << " " << ptr[2];
```

```
    return;
```

```
}
```

```
// Driver program
```

```
int main()
```

```
{
```

```
    geeks();
```

```
    return 0;
```

```
}
```

```
chevron_right
```

```
filter_none
```

```
Output:
```

```
Elements of the array are: 5 10 15
```

Val[0]	Val[1]	Val[2]
5	10	15
ptr[0]	ptr[1]	ptr[2]

Now if this ptr is sent to a function then the array val can be accessed in a similar fashion.

Pointers and Multidimensional Arrays

Consider pointer notation for the two-dimensional numeric arrays. consider the following declaration

```
int nums[2][3] = { {16, 18, 20}, {25, 26, 27} };
```

In general, `nums[i][j]` is equivalent to `*(*(nums+i)+j)`

POINTER NOTATION	ARRAY NOTATION	VALUE
<code>*(*nums + 1)</code>	<code>nums[0][1]</code>	18
<code>*(*nums + 2)</code>	<code>nums[0][2]</code>	20
<code>*(*nums + 1) + 0</code>	<code>nums[1][0]</code>	25
<code>*(*nums + 1) + 1</code>	<code>nums[1][1]</code>	26
<code>*(*nums + 1) + 2</code>	<code>nums[1][2]</code>	27

Related Articles:

- [How to pass a 2D array as a parameter in C?](#)
- [Multidimensional Pointer Arithmetic in C/C++](#)
- [Pointer vs Array](#)
- [Why C treats array parameters as pointers?](#)
- [What's difference between "array" and "&array" for "int array\[5\]"?](#)

Applications of pointers in C/C++.

[Quizzes](#) - Quiz on Pointer Basics , Quiz on Advanced Pointer

Reference:

https://www.ntu.edu.sg/home/ehchua/programming/cpp/cp4_PointerReference.html

This article is contributed by **Ashirav Kariya**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes 

Add your personal notes
here! (max 5000 chars)

Recommended Posts:

- [Introduction of Smart Pointers in C++ and It's Types](#)
- [Difference between pointer to an array and array of pointers](#)
- [C program to sort an array using pointers](#)
- [Program to reverse an array using pointers](#)
- [Why C treats array parameters as pointers?](#)
- [Declare a C/C++ function returning pointer to array of integer pointers](#)
- [Sum of array using pointer arithmetic](#)
- [Array in Python | Set 1 \(Introduction and Functions\)](#)
- [Applications of Pointers in C/C++](#)
- [Features and Use of Pointers in C/C++](#)
- [Pointers in C/C++ with Examples](#)

- What are near, far and huge pointers?
- Pointers vs References in C++
- Pointers and References in C++
- The length of a string using pointers

Improved By : SagarUdasi, Prasant Nair Cricket Research, SHUBHAMSINGH10, rutujak

Article Tags :

C
C++
School Programming
C-Pointers
CPP-Basics
cpp-pointer
Pointers
Practice Tags :
Pointers
C
CPP

113

To-do Done
2.3

Based on 159 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

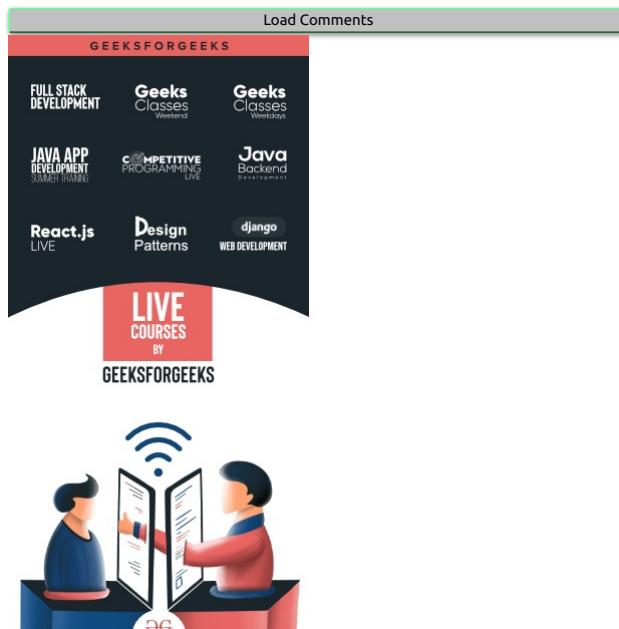
Previous

[first_page](#) Types of Exception in Java with Examples

Next

[last_page](#) Print matrix in diagonal pattern

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
ctype.h <ctype.h> library in C/C++ with Examples
Predefined Macros in C with Examples
Implicit Type Conversion in C with Examples
How to call function within function in C or C++

Most visited in C++
Vector of Vectors in C++ STL with Examples
C++ Tutorial
Which C++ libraries are useful for competitive programming?
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

Double Pointer (Pointer to Pointer) in C

Prerequisite : [Pointers in C and C++](#)

We already know that a pointer points to a location in memory and thus used to store the address of variables. So, when we define a pointer to pointer. The first pointer is used to store the address of the variable. And the second pointer is used to store the address of the first pointer. That is why they are also known as double pointers.

How to declare a pointer to pointer in C?

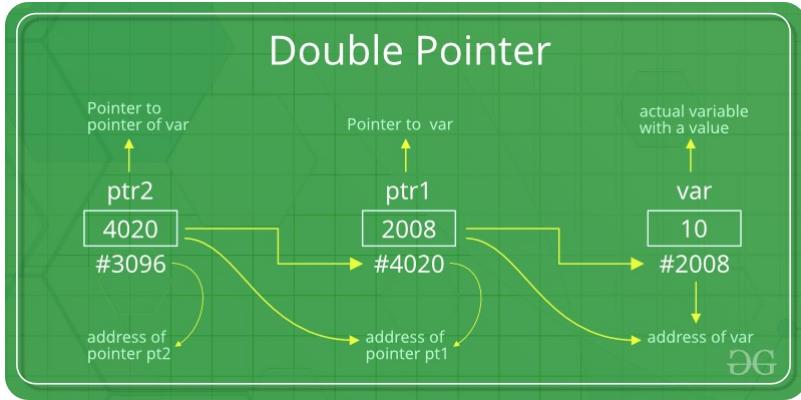
Declaring Pointer to Pointer is similar to declaring pointer in C. The difference is we have to place an additional '*' before the name of pointer.

Syntax:

```
int **ptr; // declaring double pointers
```

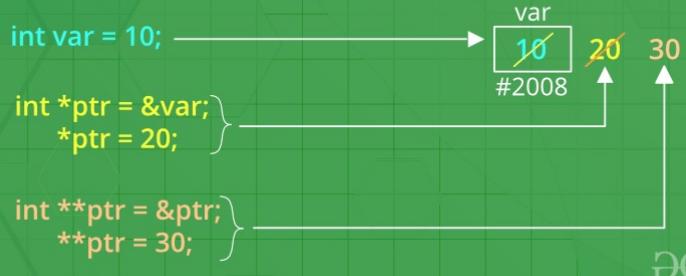
Below diagram explains the concept of Double Pointers:

Double Pointer



The above diagram shows the memory representation of a pointer to pointer. The first pointer **ptr1** stores the address of the variable and the second pointer **ptr2** stores the address of the first pointer.

How pointer works in C



Let us understand this more clearly with the help of the below program:

```
filter_none
edit
close
play_arrow
link
brightness_4
code
#include <stdio.h>

// C program to demonstrate pointer to pointer
int main()
{
    int var = 789;

    // pointer for var
    int *ptr2;

    // double pointer for ptr2
    int **ptr1;

    // storing address of var in ptr2
    ptr2 = &var;

    // Storing address of ptr2 in ptr1
    ptr1 = &ptr2;

    // Displaying value of var using
    // both single and double pointers
    printf("Value of var = %d\n", var );
    printf("Value of var using single pointer = %d\n", *ptr2 );
    printf("Value of var using double pointer = %d\n", **ptr1);

    return 0;
}
```

chevron_right

filter_none

Output:

```
Value of var = 789
Value of var using single pointer = 789
Value of var using double pointer = 789
```

Related Post :

Function Pointer in C

This article is contributed by **Harsh Agarwal**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- What is a Pointer to a Null pointer
- Delete multiple occurrences of key in Linked list using double pointer
- 'this' pointer in C++
- C++ | this pointer | Question 1
- A C/C++ Pointer Puzzle
- Pointer to an Array | Array Pointer
- C++ | this pointer | Question 5
- C++ | this pointer | Question 4
- C++ | this pointer | Question 3
- NULL pointer in C
- Opaque Pointer
- void pointer in C / C++
- C++ | this pointer | Question 2
- Pointer vs Array in C
- Function Pointer in C

Improved By : [cup_to_kohli_hi_laega](#), [tony99019](#)

Article Tags :

C
C-Pointers
cpp-double-pointer
cpp-pointer
Practice Tags :

C

45

To-do Done
1.7

Based on 79 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

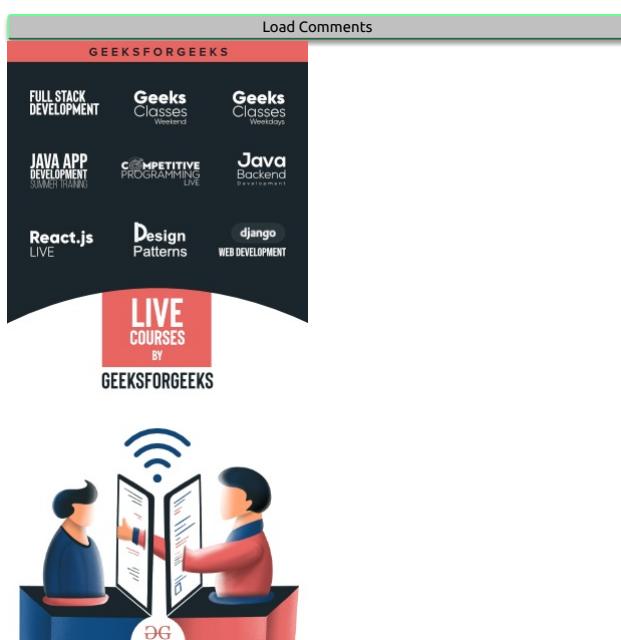
Previous

[first_page](#) Hygienic Macros : An Introduction

Next

[last_page](#) Executing main() in C/C++ - behind the scene

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C
Print all possible combinations of the string by replacing 'S' with any other digit from the string
Format specifiers in C with Examples
Predefined Macros in C with Examples
C program to print odd line contents of a File followed by even line content
C program to find square root of a given number

More related articles in C
Introduction to the C Programming Language : Part II
Program to print floating point numbers and how to compare them correctly?
Features of C Programming Language
<cfloat> float.h in C/C++ with Examples
Getting System and Process Information Using C Programming and Shell in Linux

Why C treats array parameters as pointers?

In C, array parameters are treated as pointers. The following two definitions of foo() look different, but to the compiler they mean exactly the same thing. It's preferable to use whichever syntax is more accurate for readability. If the pointer coming in really is the base address of a whole array, then we should use [].

[filter_none](#)
[edit](#)
[close](#)
[play_arrow](#)
[link](#)

```
brightness_4
code

void foo(int arr_param[])
{
    /* Silly but valid. Just changes the local pointer */
    arr_param = NULL;
}

void foo(int *arr_param)
{
    /* ditto */
    arr_param = NULL;
}
```

chevron_right
filter_none

Array parameters treated as pointers because of efficiency. It is inefficient to copy the array data in terms of both memory and time; and most of the times, when we pass an array our intention is to just tell the array we interested in, not to create a copy of the array.

Asked by Shobhit

References:
<http://cslibrary.stanford.edu/101/EssentialC.pdf>

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes [arrow_drop_up](#)

Add your personal notes here! (max 5000 chars)

[Save](#)

Recommended Posts:

- [Do not use sizeof for array parameters](#)
- [Difference between pointer to an array and array of pointers](#)
- [C program to sort an array using pointers](#)
- [Pointers in C and C++ | Set 1 \(Introduction, Arithmetic and Array\)](#)
- [Program to reverse an array using pointers](#)
- [Declare a C/C++ function returning pointer to array of integer pointers](#)
- [What is evaluation order of function parameters in C?](#)
- [Applications of Pointers in C/C++](#)
- [Pointers in C/C++ with Examples](#)
- [What are near, far and huge pointers?](#)
- [Pointers vs References in C++](#)
- [Features and Use of Pointers in C/C++](#)
- [The length of a string using pointers](#)
- [Output of C programs | Set 64 \(Pointers\)](#)
- [What are Wild Pointers? How can we avoid?](#)

Article Tags :

[C](#)
[C-Pointers](#)
[pointer](#)

Practice Tags :
[C](#)

[thumb_up](#)
25

To-do Done
2.1

Based on 78 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) [For Versus While](#)

Next

[last_page](#) [What all is inherited from parent class in C++?](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

[Load Comments](#)



Most popular in C
 Print all possible combinations of the string by replacing '\$' with any other digit from the string
 Format specifiers in different Programming Languages
 C program to find square root of a given number
 Predefined Macros in C with Examples
 C program to print odd line contents of a File followed by even line content

More related articles in C
 Introduction to the C99 Programming Language : Part II
 Features of Programming language
 Problem in comparing Floating point numbers and how to compare them correctly?
 <cfloat> float.h in C/C++ with Examples
 Getting System and Process Information Using C Programming and Shell in Linux

Output of the program | Dereference, Reference, Dereference, Reference....

Predict the output of below program

```
filter_none
edit
close
play_arrow
link
brightness_4
code

#include<stdio.h>
int main()
{
    char *ptr = "geeksforgeeks";
    printf("%c\n", *&ptr);

    getchar();
    return 0;
}
chevron_right
filter_none
Output: g
```

Explanation: The operator * is used for dereferencing and the operator & is used to get the address. These operators cancel effect of each other when used one after another. We can apply them alternatively any no. of times. For example *ptr gives us g, &ptr gives address of g, *&ptr again g, &*&ptr address of g, and finally *&*&ptr gives 'g'

Now try below

```
filter_none
edit
close
play_arrow
link
brightness_4
code

#include<stdio.h>
int main()
{
    char *ptr = "geeksforgeeks";
    printf("%s\n", *&ptr);

    getchar();
    return 0;
}
chevron_right
filter_none
```

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes

Add your personal notes here! (max 5000 chars)

Recommended Posts:

- Output of C++ Program | Set 19
- Output of C++ Program | Set 20
- Output of C Program | Set 29
- Output of C++ Program | Set 9
- Output of C++ Program | Set 13
- Output of C++ Program | Set 14
- Output of C++ Program | Set 15
- Output of C++ Program | Set 16
- Output of C++ Program | Set 17
- Output of C++ Program | Set 18
- Output of C Program | Set 27
- Output of C++ Program | Set 12
- Output of C++ Program | Set 8
- Output of C Program | Set 22
- Output of C++ Program | Set 10

Improved By : MOHAKGOEL

Article Tags :

C
Program Output
C-Output
C-Pointers
Practice Tags :
C

thumb_up
20

To-do Done
2

Based on 61 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

first_page Program for Sum of the digits of a given number

Next

last_page Output of the Program | Use Macros Carefully!

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

The image shows the top portion of the GeeksforGeeks website. At the top, there's a green bar with the text "Load Comments". Below it is a red header bar with the "GEEKSFORGEEKS" logo. The main content area has a dark background with various course and program categories listed in white boxes:

- FULL STACK DEVELOPMENT
- Geeks Classes Weekend
- Geeks Classes Weekdays
- JAVA APP DEVELOPMENT SUMMER TRAINING
- COMPETITIVE PROGRAMMING LIVE
- Java Backend Development
- React.js LIVE
- Design Patterns
- django WEB DEVELOPMENT

Below the categories, there's a red banner with the text "LIVE COURSES BY GEEKSFORGEEKS". Underneath the banner, there's an illustration of two people looking at a screen. At the bottom left, there's some small text about popular programs and visited pages.

Most popular in C
Print all possible combinations of the string by replacing 'S' with any other digit from the string
ctype.h Header in C/C++ with Examples
C programs to display month by month calendar for a given year
Predefined Macros in C with Examples
Implicit Type Conversion in C with Examples

Most visited in Program Output
Random Numbers Ecosystem in Julia - The Pseudo Side

Dangling, Void , Null and Wild Pointers

Dangling pointer

A pointer pointing to a memory location that has been deleted (or freed) is called dangling pointer. There are **three** different ways where Pointer acts as dangling pointer

1. De-allocation of memory

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// Deallocating a memory pointed by ptr causes
// dangling pointer
```

```

#include <stdlib.h>
#include <stdio.h>
int main()
{
    int *ptr = (int *)malloc(sizeof(int));

    // After below free call, ptr becomes a
    // dangling pointer
    free(ptr);

    // No more a dangling pointer
    ptr = NULL;
}
chevron_right
filter_none

```

2. Function Call

```

filter_none
edit
close
play_arrow
link
brightness_4
code

// The pointer pointing to local variable becomes
// dangling when local variable is not static.
#include<stdio.h>

int *fun()
{
    // x is local variable and goes out of
    // scope after an execution of fun() is
    // over.
    int x = 5;

    return &x;
}

// Driver Code
int main()
{
    int *p = fun();
    fflush(stdin);

    // p points to something which is not
    // valid anymore
    printf("%d", *p);
    return 0;
}
chevron_right

```

A garbage Address

The above problem doesn't appear (or p doesn't become dangling) if x is a static variable.

```

filter_none
edit
close
play_arrow
link
brightness_4
code

// The pointer pointing to local variable doesn't
// become dangling when local variable is static.
#include<stdio.h>

int *fun()
{
    // x now has scope throughout the program
    static int x = 5;

    return &x;
}

int main()
{
    int *p = fun();
    fflush(stdin);

    // Not a dangling pointer as it points
    // to static variable.
    printf("%d", *p);
}
chevron_right

```

Output:

3. Variable goes out of scope

```
void main()
{
    int *ptr;
    ....
    ....
    {
        int ch;
        ptr = &ch;
    }
    ....
    // Here ptr is dangling pointer
}
```

Void pointer

Void pointer is a specific pointer type - void * - a pointer that points to some data location in storage, which doesn't have any specific type. Void refers to the type. Basically the type of data that it points to is can be any. If we assign address of char data type to void pointer it will become char Pointer, if int data type then int pointer and so on. Any pointer type is convertible to a void pointer hence it can point to any value.

Important Points

1. void pointers **cannot be dereferenced**. It can however be done using typecasting the void pointer
2. Pointer arithmetic is not possible on pointers of void due to lack of concrete value and thus size.

Example:

```
filter_none
edit
close

play_arrow
link
brightness_4
code

#include<stdlib.h>

int main()
{
    int x = 4;
    float y = 5.5;

    //A void pointer
    void *ptr;
    ptr = &x;

    // (int*)ptr - does type casting of void
    // *((int*)ptr) dereferences the typecasted
    // void pointer variable.
    printf("Integer variable is = %d", *( (int*) ptr ) );

    // void pointer is now float
    ptr = &y;
    printf("\nFloat variable is= %f", *( (float*) ptr ) );

    return 0;
}
chevron_right
filter_none
Output:
Integer variable is = 4
Float variable is= 5.500000
```

Refer [void pointer article](#) for details.

NULL Pointer

NULL Pointer is a pointer which is pointing to nothing. In case, if we don't have address to be assigned to a pointer, then we can simply use NULL.

```
filter_none
edit
close

play_arrow
link
brightness_4
code

#include <stdio.h>
int main()
{
    // Null Pointer
    int *ptr = NULL;

    printf("The value of ptr is %p", ptr);
    return 0;
}
chevron_right
filter_none
Output:
```

Important Points

1. **NULL vs Uninitialized pointer** - An uninitialized pointer stores an undefined value. A null pointer stores a defined value, but one that is defined by the environment to not be a valid address for any member or object.
2. **NULL vs Void Pointer** - Null pointer is a value, while void pointer is a type

Wild pointer

A pointer which has not been initialized to anything (not even NULL) is known as wild pointer. The pointer may be initialized to a non-NUL garbage value that may not be a valid address.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

int main()
{
    int *p; /* wild pointer */

    int x = 10;

    // p is not a wild pointer now
    p = &x;

    return 0;
}
chevron_right
filter_none
```

This article is contributed by **Spoorthi Arun**. If you like GeeksforGeeks and would like to contribute, you can also write an article using [contribute.geeksforgeeks.org](#) or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- [What are Wild Pointers? How can we avoid?](#)
- [void pointer in C / C++](#)
- [How does "void **" differ in C and C++?](#)
- [Return from void functions in C++](#)
- [NULL pointer in C](#)
- [Passing NULL to printf in C](#)
- [Understanding "static" in "public static void main" in Java](#)
- [Is it fine to write "void main\(\)" or "main\(\)" in C/C++?](#)
- [Difference between "int main\(\)" and "int main\(void\)" in C/C++?](#)
- [Applications of Pointers in C/C++](#)
- [Pointers and References in C++](#)
- [What are near, far and huge pointers?](#)
- [Features and Use of Pointers in C/C++](#)
- [Pointers in C/C++ with Examples](#)
- [Pointers vs References in C++](#)

Improved By : [ashoklb](#), [sharathmaidargi](#)

Article Tags :

C
C++
School Programming
C-Pointers
cpp-pointer

Practice Tags :

C
CPP

thumb_up
72

To-do Done
2.2

Based on 103 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) Execution of printf with ++ operators

Next

[last_page](#) Default array values in Java

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

[Load Comments](#)



Most popular in C
C program to display month by month calendar for a given year
Print all possible combinations of the string by replacing '\$' with any other digit from the string
ctype.h(<cctype>) library in C/C++ with Examples
Implicit Type Conversion in C with Examples
Format specifiers in different Programming Languages

Most visited in C++
Vector of Vectors in C++ STL with Examples
C++ STL with Examples
Which C++ libraries are useful for competitive programming?
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

An Uncommon representation of array elements

Consider the below program.

```
filter_none
edit
close
play_arrow
link
brightness_4
code
int main( )
{
    int arr[2] = {0,1};
    printf("First Element = %d\n",arr[0]);
    getchar();
    return 0;
}
chevron_right
filter_none
```

Pretty Simple program.. huh... Output will be 0.

Now if you replace `arr[0]` with `0[arr]`, the output would be same. Because compiler converts the array operation in pointers before accessing the array elements.

e.g. `arr[0]` would be `*(arr + 0)` and therefore `0[arr]` would be `*(0 + arr)` and you know that both `*(arr + 0)` and `*(0 + arr)` are same.

Please write comments if you find anything incorrect in the above article.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes [arrow_drop_up](#)

Add your personal notes here! (max 5000 chars)

[Save](#)

Recommended Posts:

- Sum of array Elements without using loops and recursion
- How will you show memory representation of C variables?
- Difference between pointer to an array and array of pointers
- Jagged Array or Array of Arrays in C with Examples
- What's difference between "array" and "&array" for "int array[5]" ?
- std::transform() in C++ STL (Perform an operation on all elements)
- Sum of an array using MPI
- How to pass an array by value in C ?
- Pointer to an Array | Array Pointer
- Array class in C++
- 4 Dimensional Array in C/C++
- Pointer vs Array in C
- How to dynamically allocate a 2D array in C?
- Difference between Structure and Array in C
- Difference between pointer and array in C?

Article Tags :

C
C-Pointers

Practice Tags :

C

[thumb_up](#)

20

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page Storage for Strings in C](#)

Next

[last_page C function to Swap strings](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

The image shows the top navigation bar of the GeeksforGeeks website. It features a red header with the 'GEEKSFORGEEKS' logo. Below the header are several navigation links: 'FULL STACK DEVELOPMENT', 'Geeks Classes Webinar', 'Geeks Classes Webinars', 'JAVA APP DEVELOPMENT SUMMER TRAINING', 'COMPETITIVE PROGRAMMING LIVE', 'Java Backend Development', 'React.js LIVE', 'Design Patterns', and 'django WEB DEVELOPMENT'. A large red banner in the center says 'LIVE COURSES BY GEEKSFORGEEKS'. Below the banner is an illustration of two people looking at a screen with a Wi-Fi signal icon above it. At the bottom left, there's a 'Most popular in C' section with links to various C-related articles.

Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
ctype.h->ctype.h library in C/C++ with Examples
C program to display month by month calendar for a given year
Predefined Macros in C with Examples
Implicit Type Conversion in C with Examples

More related articles in C
Format specifiers in different Programming Languages
How to call function within function in C or C++
C program to print odd line contents of a file followed by even line content
Nested Loops in C with Examples
C program to find square root of a given number

How to declare a pointer to a function?

Well, we assume that you know what does it mean by pointer in C. So how do we create a pointer to an integer in C?

Huh..it is pretty simple..

```
int * ptrInteger; /*We have put a * operator between int
and ptrInteger to create a pointer.*/
```

Here ptrInteger is a pointer to integer. If you understand this, then logically we should not have any problem in declaring a pointer to a function

So let us first see ..how do we declare a function? For example,

```
int foo(int);
```

Here foo is a function that returns int and takes one argument of int type. So as a logical guy will think, by putting a * operator between int and foo(int) should create a pointer to a function i.e.

```
int * foo(int);
```

But Oops..C operator precedence also plays role here ..so in this case, operator () will take priority over operator *. And the above declaration will mean – a function foo with one argument of int type and return value of int * i.e. integer pointer. So it did something that we didn't want to do.

So as a next logical step, we have to bind operator * with foo somehow. And for this, we would change the default precedence of C operators using () operator.

```
int (*foo)(int);
```

That's it. Here * operator is with foo which is a function name. And it did the same that we wanted to do.

So that wasn't as difficult as we thought earlier!

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes [arrow_drop_up](#)

Add your personal notes
here! (max 5000 chars)

[Save](#)

Recommended Posts:

- Declare a C/C++ function returning pointer to array of integer pointers
- Function Pointer in C
- Different ways to declare variable as constant in C and C++
- Double Pointer (Pointer to Pointer) in C
- What is a Pointer to a Null pointer
- 'this' pointer in C++
- Opaque Pointer
- void pointer in C / C++
- A C/C++ Pointer Puzzle
- NULL pointer in C
- Pointer vs Array in C
- Type of 'this' pointer in C++

- C++ | this pointer | Question 5
- C++ | this pointer | Question 4
- C++ | this pointer | Question 3

Article Tags :

C

C-Pointers

pointer

Practice Tags :

C



46

To-do Done
2.2

Based on 101 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

first_page How to print % using printf()?

Next

last_page For Versus While

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT

Geeks Classes Weekend

Geeks Classes Weekdays

JAVA APP DEVELOPMENT SUMMER TRAINING

COMPETITIVE PROGRAMMING LIVE

Java Backend Development

React.js LIVE

Design Patterns

django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS



Most popular in C
cctype.h(<cctype.h>) library in C/C++ with Examples
 Print all possible combinations of the string by replacing 'S' with any other digit from the string
 C program to display month by month calendar for a given year
 Predefined Macros in C with Examples
 Implicit Type Conversion in C with Examples

More related articles in C
 How to call function within function in C or C++
 C program to print odd line contents of a file followed by even line content
 Nested Loops in C with Examples
 Introduction to the C99 Programming Language : Part II
 How to create GUI in C programming using GTK Toolkit

Pointer vs Array in C

Most of the time, pointer and array accesses can be treated as acting the same, the major exceptions being:

- 1) the sizeof operator
 - o sizeof(array) returns the amount of memory used by all elements in array
 - o sizeof(pointer) only returns the amount of memory used by the pointer variable itself
- 2) the & operator
 - o &array is an alias for &array[0] and returns the address of the first element in array
 - o &pointer returns the address of pointer

3) a string literal initialization of a character array

- o char array[] = "abc" sets the first four elements in array to 'a', 'b', 'c', and '\0'
- o char *pointer = "abc" sets pointer to the address of the "abc" string (which may be stored in read-only memory and thus unchangeable)

4) Pointer variable can be assigned a value whereas array variable cannot be.

```
int a[10];
int *p;
p=a; /*Legal*/
a=p; /*Illegal*/
```

5) Arithmetic on pointer variable is allowed.

```
+++; /*Legal*/
a++; /*Illegal*/
```

Please refer [Difference between pointer and array in C?](#) for more details.

References: http://icecube.wisc.edu/~dglo/c_class/array_ptr.html

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes

Add your personal notes here! (max 5000 chars)

Recommended Posts:

- Difference between pointer to an array and array of pointers
- Pointer to an Array | Array Pointer
- Sum of array using pointer arithmetic
- Difference between pointer and array in C?
- Double Pointer (Pointer to Pointer) in C
- What is a Pointer to a Null pointer
- 'this' pointer in C++
- Opaque Pointer
- A C/C++ Pointer Puzzle
- C++ | this pointer | Question 5
- C++ | this pointer | Question 4
- C++ | this pointer | Question 3
- C++ | this pointer | Question 2
- C++ | this pointer | Question 1
- Function Pointer in C

Article Tags :

C
C-Pointers
cpp-array
cpp-pointer
Practice Tags :
C

26

To-do Done
2.3

Based on 71 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

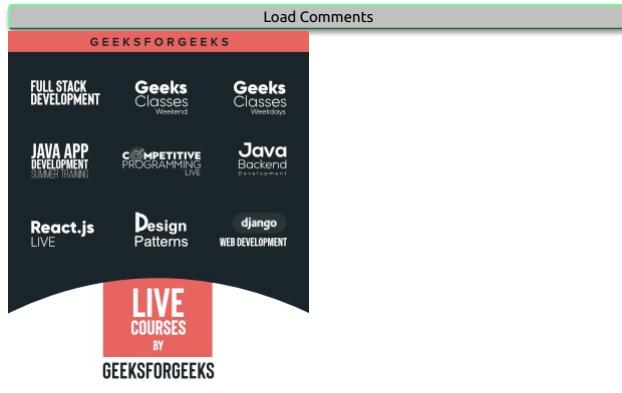
Previous

[first_page](#) What all is inherited from parent class in C++?

Next

[last_page](#) What is Memory Leak? How can we avoid?

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C
Print all possible combinations of the string by replacing 'S' with any other digit from the string
ctype.h & cctype.h library in C/C++ with Examples
C program to display month by month calendar for a given year
Predefined Macros in C with Examples
Implicit Type Conversion in C with Examples

More related articles in C
How to call function within function in C or C++
C program to print odd line contents of a file followed by even line content
Nested Loops in C with Examples
Introduction to the C99 Programming Language : Part II
How to create GUI in C programming using GTK Toolkit

void pointer in C / C++

A void pointer is a pointer that has no associated data type with it. A void pointer can hold address of any type and can be typcasted to any type.

int a = 10;
char b = 'x';

```
void *p = &a; // void pointer holds address of int 'a'  
p = &b; // void pointer holds address of char 'b'  
chevron_right
```

[filter_none](#)

Advantages of void pointers:

- 1) malloc() and calloc() return void * type and this allows these functions to be used to allocate memory of any data type (just because of void *)

[filter_none](#)
[edit](#)
[close](#)

[play_arrow](#)

[link](#)
[brightness_4](#)
[code](#)

```
int main(void)  
{  
    // Note that malloc() returns void * which can be  
    // typecasted to any type like int *, char *, ..  
    int *x = malloc(sizeof(int) * n);  
}
```

[chevron_right](#)

[filter_none](#)

Note that the above program compiles in C, but doesn't compile in C++. In C++, we must explicitly typecast return value of malloc to (int *).

2) void pointers in C are used to implement generic functions in C. For example [compare](#) function which is used in [qsort\(\)](#).

Some Interesting Facts:

- 1) void pointers cannot be dereferenced. For example the following program doesn't compile.

[filter_none](#)
[edit](#)
[close](#)

[play_arrow](#)

[link](#)
[brightness_4](#)
[code](#)

```
#include<stdio.h>  
int main()  
{  
    int a = 10;  
    void *ptr = &a;  
    printf("%d", *ptr);  
    return 0;  
}
```

[chevron_right](#)

[filter_none](#)

Output:

Compiler Error: 'void*' is not a pointer-to-object type

The following program compiles and runs fine.

[filter_none](#)
[edit](#)
[close](#)

[play_arrow](#)

[link](#)
[brightness_4](#)
[code](#)

```
#include<stdio.h>  
int main()  
{  
    int a = 10;  
    void *ptr = &a;  
    printf("%d", *(int *)ptr);  
    return 0;  
}
```

[chevron_right](#)

[filter_none](#)

Output:

10

2) The C standard doesn't allow pointer arithmetic with void pointers. However, in GNU C it is allowed by considering the size of void is 1. For example the following program compiles and runs fine in gcc.

[filter_none](#)
[edit](#)
[close](#)

[play_arrow](#)

[link](#)
[brightness_4](#)
[code](#)

```
#include<stdio.h>  
int main()  
{  
    int a[2] = {1, 2};  
    void *ptr = &a;  
    ptr = ptr + sizeof(int);  
    printf("%d", *(int *)ptr);  
    return 0;  
}
```

}

chevron_right

filter_none

Output:

2

Note that the above program may not work in other compilers.

References:

<http://stackoverflow.com/questions/20967868/should-the-compiler-warn-on-pointer-arithmetic-with-a-void-pointer>

<http://stackoverflow.com/questions/692564/concept-of-void-pointer-in-c-programming>

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes *arrow_drop_up*

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- How does "void **" differ in C and C++?
- Double Pointer (Pointer to Pointer) in C
- Return from void functions in C++
- Dangling, Void , Null and Wild Pointers
- What is a Pointer to a Null pointer
- Is it fine to write "void main()" or "main()" in C/C++?
- Difference between "int main()" and "int main(void)" in C/C++?
- 'this' pointer in C++
- C++ | this pointer | Question 5
- C++ | this pointer | Question 4
- C++ | this pointer | Question 3
- Opaque Pointer
- NULL pointer in C
- C++ | this pointer | Question 1
- C++ | this pointer | Question 2

Article Tags :

C

C Basics

C-Pointers

cpp-pointer

Practice Tags :

C

thumb_up

26

To-do Done
2.3

Based on 47 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

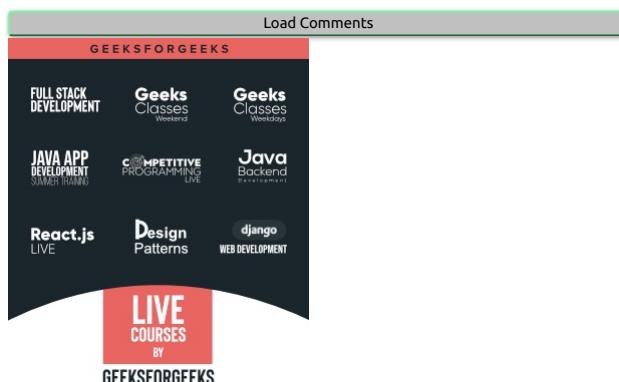
Previous

[first_page](#) Assertions in C/C++

Next

[last_page](#) C | Variable Declaration and Scope | Question 8

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
C program to display month by month calendar for a given year
`ctype.h <ctype.h>` library in C/C++ with Examples
Implicit Type Conversion in C with Examples
How to call function within function in C or C++

NULL pointer in C

At the very high level, we can think of NULL as a null pointer which is used in C for various purposes. Some of the most common use cases for NULL are

- To initialize a pointer variable when that pointer variable isn't assigned any valid memory address yet.
- To check for a null pointer before accessing any pointer variable. By doing so, we can perform error handling in pointer related code e.g. dereference pointer variable only if it's not NULL.
- To pass a null pointer to a function argument when we don't want to pass any valid memory address.

The example of a is

```
filter_none
edit
close

play_arrow

link
brightness_4
code

int * pInt = NULL;
chevron_right
```

The example of b is

```
filter_none
edit
close

play_arrow

link
brightness_4
code

if(pInt != NULL) /*We could use if(pInt) as well*/
{
    /*Some code*/
}
else
{
    /*Some code*/
}
chevron_right
```

The example of c is

```
filter_none
edit
close

play_arrow

link
brightness_4
code

int fun(int *ptr)
{
    /*Fun specific stuff is done with ptr here*/
    return 10;
}

fun(NULL);
chevron_right
```

It should be noted that NULL pointer is different from an uninitialized and dangling pointer. In a specific program context, all uninitialized or dangling or NULL pointers are invalid but NULL is a specific invalid pointer which is mentioned in C standard and has specific purposes. What we mean is that uninitialized and dangling pointers are invalid but they can point to some memory address that may be accessible through the memory access is unintended.

```
filter_none
edit
close

play_arrow

link
brightness_4
code

#include <stdio.h>
int main()
{
    int *i, *j;
    int *ii = NULL, *jj = NULL;
    if(i == j)
    {
        printf("This might get printed if both i and j are same by chance.");
    }
    if(ii == jj)
    {
        printf("This is always printed coz ii and jj are same.");
    }
    return 0;
}
chevron_right
```

By specifically mentioning NULL pointer, C standard gives mechanism using which a C programmer can use and check whether a given pointer is legitimate or not. But what exactly is NULL and how it's defined? Strictly speaking, NULL expands to an implementation-defined null pointer constant which is defined in many header files such as "stdio.h", "stddef.h", "stdlib.h" etc. Let us see what C standards say about null pointer. From C11 standard clause 6.3.2.3,

"An integer constant expression with the value 0, or such an expression cast to type void *, is called a null pointer constant. If a null pointer constant is converted to a pointer type, the resulting pointer, called a null pointer, is guaranteed to compare unequal to a pointer to any object or function."

Before we proceed further on this NULL discussion :), let's mention few lines about C standard just in case you wants to refer it for further study. Please note that ISO/IEC 9899:2011 is the C language's latest standard which was published in Dec 2011. This is also called the C11 standard. For completeness, let us mention that previous standards for C were C99, C90 (also known as ISO C) and C89 (also known as ANSI C). Though the actual C11 standard can be purchased from ISO, there's a draft document which is available in public domain for free.

Coming to our discussion, NULL macro is defined as `((void *)0)` in header files of most of the C compiler implementations. But C standard is saying that 0 is also a null pointer constant. It means that the following is also perfectly legal as per standard.

`filter_none`

`edit`

`close`

`play_arrow`

`link`

`brightness_4`

`code`

`int * ptr = 0;`

`chevron_right`

`filter_none`

Please note that 0 in the above C statement is used in pointer-context and it's different from 0 as integer. This is one of the reasons why the usage of NULL is preferred because it makes it explicit in code that programmer is using null pointer, not integer 0. Another important concept about NULL is that "*NULL expands to an implementation-defined null pointer constant*". This statement is also from C11 clause 7.19. It means that internal representation of the null pointer could be non-zero bit pattern to convey NULL pointer. That's why NULL always needn't be internally represented as all zeros bit pattern. A compiler implementation can choose to represent "null pointer constant" as a bit pattern for all 1s or anything else. But again, as a C programmer, we needn't worry much on the internal value of the null pointer unless we are involved in Compiler coding or even below the level of coding. Having said so, typically NULL is represented as all bits set to 0 only. To know this on a specific platform, one can use the following

`filter_none`

`edit`

`close`

`play_arrow`

`link`

`brightness_4`

`code`

`#include<stdio.h>`

`int main()`

`{`

`printf("%d",NULL);`

`return 0;`

`}`

`chevron_right`

`filter_none`

Most likely, it's printing 0 which is the typical internal null pointer value but again it can vary depending on the C compiler/platform. You can try few other things in above program such as `printf("%c",NULL)` or `printf("%s",NULL)` and even `printf("%f",NULL)`. The outputs of these are going to be different depending on the platform used but it'd be interesting especially usage of `%f` with NULL!

Can we use `sizeof()` operator on NULL in C? Well, usage of `sizeof(NULL)` is allowed but the exact size would depend on platform.

`filter_none`

`edit`

`close`

`play_arrow`

`link`

`brightness_4`

`code`

`#include<stdio.h>`

`int main()`

`{`

`printf("%lu",sizeof(NULL));`

`return 0;`

`}`

`chevron_right`

`filter_none`

Since NULL is defined as `((void*)0)`, we can think of NULL as a special pointer and its size would be equal to any pointer. If the pointer size of a platform is 4 bytes, the output of the above program would be 4. But if pointer size on a platform is 8 bytes, the output of the above program would be 8.

What about dereferencing of NULL? What's going to happen if we use the following C code

`filter_none`

`edit`

`close`

`play_arrow`

`link`

`brightness_4`

`code`

`#include<stdio.h>`

`int main()`

`{`

`int * ptr = NULL;`

`printf("%d",*ptr);`

`return 0;`

`}`

`chevron_right`

`filter_none`

On some machines, the above would compile successfully but crashes when the program is run through it needn't show the same behaviour across all the machines. Again it depends on a lot of factors. But the idea of mentioning the above snippet is that we should always check for NULL before accessing it.

Since NULL is typically defined as `((void*)0)`, let us discuss a little bit about void type as well. As per C11 standard clause 6.2.5, "*The void type comprises an empty set of values; it is an incomplete object type that cannot be completed*". Even C11 clause 6.5.3.4 mentions that "*The sizeof operator shall not be applied to an expression that has function type or an incomplete type, to the parenthesized name of such a type, or to an expression that designates a bit-field member*." Basically, it means that void is an incomplete type whose size doesn't make any sense in C programs but implementations (such as gcc) can choose `sizeof(void)` as 1 so that the flat memory pointed by void pointer can be viewed as untyped memory i.e. a sequence of bytes. But the output of the following needn't to same on all platforms.

`filter_none`

`edit`

close
play_arrow
link
brightness_4
code

```
#include<stdio.h>
int main()
{
    printf("%lu",sizeof(void));
    return 0;
}
```

chevron_right

filter_none

On gcc, the above would output 1. What about sizeof(void *)? Here C11 has mentioned guidelines. From clause 6.2.5, "A pointer to void shall have the same representation and alignment requirements as a pointer to a character type". That's why the output of the following would be same as any pointer size on a machine.

filter_none
edit
close

play_arrow

link
brightness_4
code

```
#include<stdio.h>
int main()
{
    printf("%lu",sizeof(void *));
    return 0;
}
```

chevron_right

filter_none

Inspite of mentioning machine dependent stuff as above, we as C programmers should always strive to make our code as portable as possible. So we can conclude on NULL as follows:

1. Always initialize pointer variables as NULL.
2. Always perform a NULL check before accessing any pointer.

Please do Like/Tweet/G+1 if you find the above useful. Also, please do leave us to comment for further clarification or info. We would love to help and learn

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- [What is a Pointer to a Null pointer](#)
- [Double Pointer \(Pointer to Pointer\) in C](#)
- [Passing NULL to printf in C](#)
- [Dangling, Void , Null and Wild Pointers](#)
- ['this' pointer in C++](#)
- [Function Pointer in C](#)
- [C++ | this pointer | Question 1](#)
- [C++ | this pointer | Question 2](#)
- [C++ | this pointer | Question 4](#)
- [C++ | this pointer | Question 5](#)
- [Opaque Pointer](#)
- [C++ | this pointer | Question 3](#)
- [void pointer in C / C++](#)
- [Pointer vs Array in C](#)
- [A C/C++ Pointer Puzzle](#)

Improved By : [nishant2raj](#)

Article Tags :

C
C-Pointers
cpp-pointer

Practice Tags :

C

thumb_up
27

To-do Done
3.6

Based on 46 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) C Quiz - 101 | Question 5

Next

[last_page](#) Add two numbers using ++ and/or --

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments



Most popular in C
Print all possible combinations of the string by replacing 'S' with any other digit from the string
C program to display month by month calendar for a given year
`cctype.h(<cctype.h>)` library in C/C++ with Examples
Implicit Type Conversion in C with Examples
How to call function within function in C or C++

More related articles in C
Format specifiers in different Programming Languages
Program to Print Maps with Examples
C program to print odd line contents of a File followed by even line content
Nested Loops in C with Examples
C program to find square root of a given number

Function Pointer in C

In C, like **normal data pointers** (`int * , char * , etc`), we can have pointers to functions. Following is a simple example that shows declaration and function call using function pointer.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

#include <stdio.h>
// A normal function with an int parameter
// and void return type
void fun(int a)
{
    printf("Value of a is %d\n", a);
}

int main()
{
    // fun_ptr is a pointer to function fun()
    void (*fun_ptr)(int) = &fun;

    /* The above line is equivalent of following two
     void (*fun_ptr)(int);
     fun_ptr = &fun;
     */

    // Invoking fun() using fun_ptr
    (*fun_ptr)(10);

    return 0;
}
chevron_right
```

Output:

`Value of a is 10`

Why do we need an extra bracket around function pointers like `fun_ptr` in above example?

If we remove bracket, then the expression "void (*fun_ptr)(int)" becomes "void *fun_ptr(int)" which is declaration of a function that returns void pointer. See following post for details.

[How to declare a pointer to a function?](#)

Following are some interesting facts about function pointers.

1) Unlike normal pointers, a function pointer points to code, not data. Typically a function pointer stores the start of executable code.

2) Unlike normal pointers, we do not allocate de-allocate memory using function pointers.

3) A function's name can also be used to get functions' address. For example, in the below program, we have removed address operator '`&`' in assignment. We have also changed function call by removing `*`, the program still works.

```
filter_none
edit
close
play_arrow
link
```

```

brightness_4
code

#include <stdio.h>
// A normal function with an int parameter
// and void return type
void fun(int a)
{
    printf("Value of a is %d\n", a);
}

int main()
{
    void (*fun_ptr)(int) = fun; // & removed

    fun_ptr(10); // * removed

    return 0;
}

```

chevron_right
filter_none

Output:

Value of a is 10

4) Like normal pointers, we can have an array of function pointers. Below example in point 5 shows syntax for array of pointers.

5) Function pointer can be used in place of switch case. For example, in below program, user is asked for a choice between 0 and 2 to do different tasks.

```

filter_none
edit
close

play_arrow

link
brightness_4
code

#include <stdio.h>
void add(int a, int b)
{
    printf("Addition is %d\n", a+b);
}

void subtract(int a, int b)
{
    printf("Subtraction is %d\n", a-b);
}

void multiply(int a, int b)
{
    printf("Multiplication is %d\n", a*b);
}

int main()
{
    // fun_ptr_arr is an array of function pointers
    void (*fun_ptr_arr[])(int, int) = {add, subtract, multiply};
    unsigned int ch, a = 15, b = 10;

    printf("Enter Choice: 0 for add, 1 for subtract and 2 "
           "for multiply\n");
    scanf("%d", &ch);

    if (ch > 2) return 0;

    (*fun_ptr_arr[ch])(a, b);

    return 0;
}

```

chevron_right
filter_none

Enter Choice: 0 for add, 1 for subtract and 2 for multiply
2
Multiplication is 150

6) Like normal data pointers, a function pointer can be passed as an argument and can also be returned from a function.

For example, consider the following C program where wrapper() receives a void fun() as parameter and calls the passed function.

```

filter_none
edit
close

play_arrow

link
brightness_4
code

// A simple C program to show function pointers as parameter
#include <stdio.h>

// Two simple functions
void fun1() { printf("Fun1\n"); }
void fun2() { printf("Fun2\n"); }

// A function that receives a simple function

```

```
// as parameter and calls the function
```

```
void wrapper(void (*fun)())
```

```
{
```

```
    fun();
```

```
}
```

```
int main()
```

```
{
```

```
    wrapper(fun1);
```

```
    wrapper(fun2);
```

```
    return 0;
```

```
}
```

```
chevron_right
```

```
filter_none
```

This point in particular is very useful in C. In C, we can use function pointers to avoid code redundancy. For example a simple `qsort()` function can be used to sort arrays in ascending order or descending or by any other order in case of array of structures. Not only this, with function pointers and void pointers, it is possible to use `qsort` for any data type.

```
filter_none
```

```
edit
```

```
close
```

```
play_arrow
```

```
link
```

```
brightness_4
```

```
code
```

```
// An example for qsort and comparator
```

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
// A sample comparator function that is used
```

```
// for sorting an integer array in ascending order.
```

```
// To sort any array for any other data type and/or
```

```
// criteria, all we need to do is write more compare
```

```
// functions. And we can use the same qsort()
```

```
int compare (const void * a, const void * b)
```

```
{
```

```
    return ( *(int*)a - *(int*)b );
```

```
}
```

```
int main ()
```

```
{
```

```
    int arr[] = {10, 5, 15, 12, 90, 80};
```

```
    int n = sizeof(arr)/sizeof(arr[0]), i;
```

```
    qsort (arr, n, sizeof(int), compare);
```

```
    for (i=0; i<n; i++)
```

```
        printf ("%d ", arr[i]);
```

```
    return 0;
```

```
}
```

```
chevron_right
```

```
filter_none
```

Output:

```
5 10 12 15 80 90
```

Similar to `qsort()`, we can write our own functions that can be used for any data type and can do different tasks without code redundancy. Below is an example search function that can be used for any data type. In fact we can use this search function to find close elements (below a threshold) by writing a customized compare function.

```
filter_none
```

```
edit
```

```
close
```

```
play_arrow
```

```
link
```

```
brightness_4
```

```
code
```

```
#include <stdio.h>
```

```
#include <stdbool.h>
```

```
// A compare function that is used for searching an integer
```

```
// array
```

```
bool compare (const void * a, const void * b)
```

```
{
```

```
    return ( *(int*)a == *(int*)b );
```

```
}
```

```
// General purpose search() function that can be used
```

```
// for searching an element *x in an array arr[] of
```

```
// arr_size. Note that void pointers are used so that
```

```
// the function can be called by passing a pointer of
```

```
// any type. ele_size is size of an array element
```

```
int search(void *arr, int arr_size, int ele_size, void *x,
```

```
          bool compare (const void * , const void *))
```

```
{
```

```
    // Since char takes one byte, we can use char pointer
```

```
    // for any type/ To get pointer arithmetic correct,
```

```
    // we need to multiply index with size of an array
```

```
    // element ele_size
```

```
    char *ptr = (char *)arr;
```

```
    int i;
```

```
    for (i=0; i<arr_size; i++)
```

```

if (compare(ptr + i*ele_size, x))
    return i;

// If element not found
return -1;
}

int main()
{
    int arr[] = {2, 5, 7, 90, 70};
    int n = sizeof(arr)/sizeof(arr[0]);
    int x = 7;
    printf ("Returned index is %d ", search(arr, n,
                                             sizeof(int), &x, compare));
    return 0;
}

```

chevron_right

filter_none

Output:

Returned index is 2

The above search function can be used for any data type by writing a separate customized compare().

7) Many object oriented features in C++ are implemented using function pointers in C. For example [virtual functions](#). Class methods are another example implemented using function pointers. Refer [this book](#) for more details.

Related Article:[Pointers in C and C++ | Set 1 \(Introduction, Arithmetic and Array\)](#)

References:

<http://www.cs.cmu.edu/~ab/15-123S11/AnnotatedNotes/Lecture14.pdf>

http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-087-practical-programming-in-c-january-iap-2010/lecture-notes/MIT6_087IAP10_lec08.pdf

<http://www.cs.cmu.edu/~guna/15-123S11/Lectures/Lecture14.pdf>

This article is contributed by **Abhay Rathi**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes *arrow_drop_up*

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- [How to declare a pointer to a function?](#)
- [Double Pointer \(Pointer to Pointer\) in C](#)
- [What is a Pointer to a Null pointer](#)
- ['this' pointer in C++](#)
- [C++ | this pointer | Question 5](#)
- [C++ | this pointer | Question 2](#)
- [C++ | this pointer | Question 4](#)
- [C++ | this pointer | Question 3](#)
- [void pointer in C / C++](#)
- [NULL pointer in C](#)
- [Opaque Pointer](#)
- [C++ | this pointer | Question 1](#)
- [Pointer to an Array | Array Pointer](#)
- [A C/C++ Pointer Puzzle](#)
- [Pointer vs Array in C](#)

Article Tags :

C
C-Pointers
CPP-Functions
cpp-pointer

Practice Tags :

C

thumb_up
65

To-do Done
3.4

Based on 117 vote(s)

Basic **Easy** **Medium** **Hard** **Expert**

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

first_page [Generic Linked List in C](#)

Next

last_page [How to write long strings in Multi-lines C/C++?](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments



Most popular in C
Print all possible combinations of the string by replacing 'S' with any other digit from the string
C program to display month by month calendar for a given year
`ctype.h(<cctype>)` library in C/C++ with Examples
Implicit Type Conversion in C with Examples
How to call function within function in C or C++
More related articles in C
Format specifiers in different Programming Languages
Program Mats Examples
C program to print odd line contents of a File followed by even line content
Nested Loops in C with Examples
C program to find square root of a given number

What are near, far and huge pointers?

These are some old concepts used in 16 bit intel architectures in the days of MS DOS, not much useful anymore.

Near pointer is used to store 16 bit addresses means within current segment on a 16 bit machine. The limitation is that we can only access 64kb of data at a time.

A far pointer is typically 32 bit that can access memory outside current segment. To use this, compiler allocates a segment register to store segment address, then another register to store offset within current segment.

Like far pointer, **huge pointer** is also typically 32 bit and can access outside segment. In case of far pointers, a segment is fixed. In far pointer, the segment part cannot be modified, but in Huge it can be

See below links for more details.

http://www.answers.com/Q/What_are_near_far_and_huge_pointers_in_C

<https://www.quora.com/What-is-the-difference-between-near-far-huge-pointers-in-C-C++>

<http://stackoverflow.com/questions/8727122/explain-the-difference-between-near-far-and-huge-pointers-in-c>

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes [arrow_drop_up](#)

[Save](#)

Recommended Posts:

- [Implement your own tail](#) (Read last n lines of a huge file)
- [Features and Use of Pointers in C/C++](#)
- [Applications of Pointers in C/C++](#)
- [Pointers vs References in C++](#)
- [Pointers in C/C++ with Examples](#)
- [Chain of Pointers in C with Examples](#)
- [Output of C programs | Set 64 \(Pointers\)](#)
- [The length of a string using pointers](#)
- [What are Wild Pointers? How can we avoid?](#)
- [Program to reverse an array using pointers](#)
- [Introduction of Smart Pointers in C++ and It's Types](#)
- [Difference between Iterators and Pointers in C/C++ with Examples](#)
- [C program to sort an array using pointers](#)
- [Why C treats array parameters as pointers?](#)
- [Pointers in C and C++ | Set 1 \(Introduction, Arithmetic and Array\)](#)

Article Tags :

C
C-Pointers
Practice Tags :
C

[thumb_up](#)
9

To-do Done
1.8

Based on 16 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

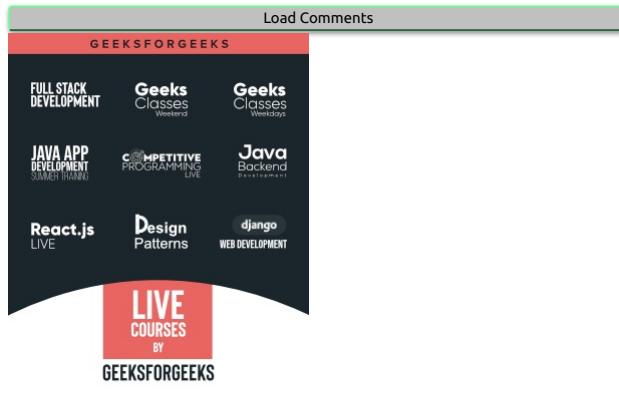
Post navigation

Previous

[first_page Operators in C | Set 1 \(Arithmetic Operators\)](#)

Next

[last_page Operators in C | Set 2 \(Relational and Logical Operators\)](#)



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
Predefined Macros in C with Examples
For Loops in C and other Programming Languages
C program to print odd line contents of a File followed by even line content
Introduction to the C99 Programming Language : Part II

More related articles in C
C program to find square root of a given number
Problem in comparing Floating point numbers and how to compare them correctly?
Features of C Programming Language
Pointer Expressions in C with Examples
Introduction to the C99 Programming Language : Part III

Generic Linked List in C

Unlike C++ and Java, C doesn't support generics. How to create a linked list in C that can be used for any data type? In C, we can use `void pointer` and function pointer to implement the same functionality. The great thing about void pointer is it can be used to point to any data type. Also, size of all types of pointers is always same, so we can always allocate a linked list node. Function pointer is needed process actual content stored at address pointed by void pointer.

Following is a sample C code to demonstrate working of generic linked list.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// C program for generic linked list
#include<stdio.h>
#include<stdlib.h>

/* A linked list node */
struct Node
{
    // Any data type can be stored in this node
    void *data;

    struct Node *next;
};

/* Function to add a node at the beginning of Linked List.
   This function expects a pointer to the data to be added
   and size of the data type */
void push(struct Node** head_ref, void *new_data, size_t data_size)
{
    // Allocate memory for node
    struct Node* new_node = (struct Node*)malloc(sizeof(struct Node));

    new_node->data = malloc(data_size);
    new_node->next = (*head_ref);

    // Copy contents of new_data to newly allocated memory.
    // Assumption: char takes 1 byte.
    int i;
    for (i=0; i<data_size; i++)
        *(char*)(new_node->data + i) = *(char*)(new_data + i);

    // Change head pointer as new node is added at the beginning
    (*head_ref) = new_node;
}

/* Function to print nodes in a given linked list. fpitr is used
   to access the function to be used for printing current node data.
   Note that different data types need different specifier in printf() */
void printList(struct Node *node, void (*fpitr)(void *))
{
```

```

while (node != NULL)
{
    (*fptr)(node->data);
    node = node->next;
}

// Function to print an integer
void printInt(void *n)
{
    printf(" %d", *(int *)n);
}

// Function to print a float
void printFloat(void *f)
{
    printf(" %f", *(float *)f);
}

/* Driver program to test above function */
int main()
{
    struct Node *start = NULL;

    // Create and print an int linked list
    unsigned int_size = sizeof(int);
    int arr[] = {10, 20, 30, 40, 50}, i;
    for (i=4; i>=0; i--)
        push(&start, &arr[i], int_size);
    printf("Created integer linked list is \n");
    printList(start, printInt);

    // Create and print a float linked list
    unsigned float_size = sizeof(float);
    start = NULL;
    float arr2[] = {10.1, 20.2, 30.3, 40.4, 50.5};
    for (i=4; i>=0; i--)
        push(&start, &arr2[i], float_size);
    printf("\n\nCreated float linked list is \n");
    printList(start, printFloat);

    return 0;
}

```

[chevron_right](#)

[filter_none](#)

Output:

```

Created integer linked list is
10 20 30 40 50

Created float linked list is
10.100000 20.200001 30.299999 40.400002 50.500000

```

This article is contributed by **Himanshu Gupta**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes [arrow_drop_up](#)

[Save](#)

Recommended Posts:

- Create new linked list from two given linked list with greater element at each node
- XOR Linked List - A Memory Efficient Doubly Linked List | Set 1
- Merge a linked list into another linked list at alternate positions
- XOR Linked List - A Memory Efficient Doubly Linked List | Set 2
- Convert singly linked list into circular linked list
- Difference between Singly linked list and Doubly linked list
- Convert Singly Linked List to XOR Linked List
- Check if a linked list is Circular Linked List
- Construct a Maximum Sum Linked List out of two Sorted Linked Lists having some Common nodes
- Create a linked list from two linked lists by choosing max element at each position
- Construct a Doubly linked linked list from 2D Matrix
- Length of longest palindrome list in a linked list using O(1) extra space
- Partitioning a linked list around a given value and If we don't care about making the elements of the list "stable"
- Rotate the sub-list of a linked list from position M to N to the right by K places
- Generic Implementation of QuickSort Algorithm in C

Article Tags :

Advanced Data Structure

C

Linked List

C-Pointers

Practice Tags :

Linked List

C

[thumb_up](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) Search an element in a Linked List (Iterative and Recursive)

Next

[last_page](#) Swap nodes in a linked list without swapping data

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Weekend Geeks Classes Weekdays

JAVA APP DEVELOPMENT COMPETITIVE PROGRAMMING LN Java Backend

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS

Most popular in Advanced Data Structure
Array Range Queries to find the Maximum Armstrong number with updates
Parathensis Theorem
Array range queries to find the number of perfect square elements with updates
Number of elements less than or equal to a number in a subarray : MO's Algorithm
Queries to find kth smallest element and point update : Ordered Set in C++

Most visited in C
P(i)n in C/C++ with Examples
Sieve of Eratosthenes execution with help of Pragma in C/C++
Modulo Operator (%) in C/C++ with Examples
Program to calculate Electricity Bill
Role of SemiColon in various Programming Languages

restrict keyword in C

In the [C programming language](#) (after 99 standard), a new keyword is introduced known as restrict.

- restrict keyword is mainly used in pointer declarations as a type qualifier for pointers.
- It doesn't add any new functionality. It is only a way for programmer to inform about an optimizations that compiler can make.
- When we use restrict with a pointer ptr, it tells the compiler that ptr is the only way to access the object pointed by it and compiler doesn't need to add any additional checks.
- If a programmer uses restrict keyword and violate the above condition, result is undefined behavior.
- restrict is not supported by C++. It is a C only keyword.

```
filter_none
edit
close

play_arrow
link
brightness_4
code

// C program to use restrict keyword.
#include <stdio.h>

// Note that the purpose of restrict is to
// show only syntax. It doesn't change anything
// in output (or logic). It is just a way for
// programmer to tell compiler about an
// optimization
void use(int* a, int* b, int* restrict c)
{
    *a += *c;

    // Since c is restrict, compiler will
    // not reload value at address c in
    // its assembly code. Therefore generated
    // assembly code is optimized
    *b += *c;
}

int main(void)
{
    int a = 50, b = 60, c = 70;
    use(&a, &b, &c);
    printf("%d %d %d", a, b, c);
    return 0;
}
chevron_right
```

filter none

Output:

120 130 70

This article is contributed by **Bishal Kumar Dubey**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes [arrow_drop_up](#)

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- How to restrict dynamic allocation of objects in C++?
- C++ mutable keyword
- Use of explicit keyword in C++
- _Generic keyword in C
- C++ | Static Keyword | Question 4
- C++ | friend keyword | Question 1
- Understanding "register" keyword in C
- C++ | Static Keyword | Question 2
- C++ | Static Keyword | Question 1
- C++ | Static Keyword | Question 3
- C++ | Static Keyword | Question 6
- C++ | const keyword | Question 5
- C++ | Static Keyword | Question 5
- Understanding "extern" keyword in C
- C++ | const keyword | Question 3

Article Tags :

C
C-Data Types
C-Pointers

Practice Tags :

C

thumb_up
12

To-do Done
2.3

Based on 8 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

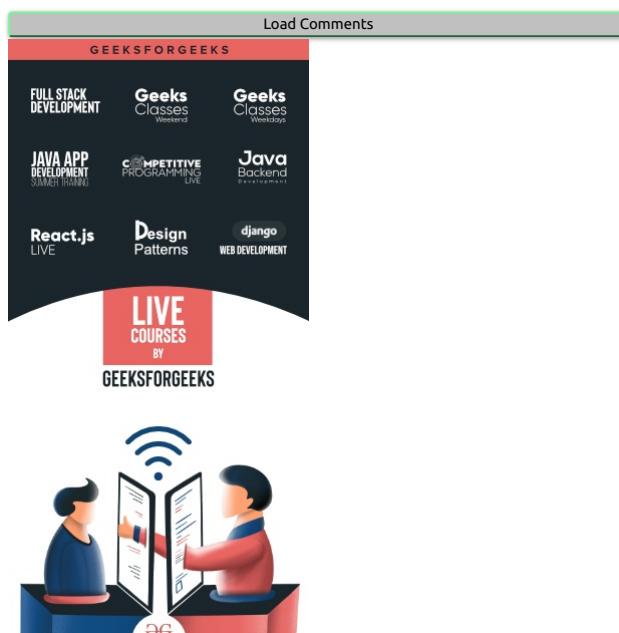
Previous

[first_page](#) Encode an ASCII string into Base-64 Format

Next

[last_page](#) isspace() in C/C++ and its application to count whitespace characters

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
Predefined Macro in C with Examples
C program to print odd line contents of a File followed by even line content
Introduction to the C99 Programming Language : Part II
C program to find square root of a given number

More related articles in C
Format specifiers in different Programming Languages
Problem of comparing Floating point numbers and how to compare them correctly?
Feature of C Programming Language
Pointer Expressions in C with Examples
Introduction to the C99 Programming Language : Part III

Difference between const char *p, char * const p and const char * const p

Prerequisite: [Pointers](#)

There is a lot of confusion when char, const, *, p are all used in different permutations and meanings change according to which is placed where. Following article focus on differentiation and usage of all of these.

The qualifier **const** can be applied to the declaration of any variable to specify that its value will not be changed. const keyword applies to whatever is immediately to its left. If there is nothing to its left, it applies to whatever is immediately to its right.

1. **const char *ptr :** This is a pointer to a constant character. **You cannot change the value pointed by ptr, but you can change the pointer itself.** "const char **" is a (non-const) pointer to a const char.
filter_none
edit
close
play_arrow
link
brightness_4
code

// C program to illustrate
// char const *p
#include<stdio.h>
#include<stdlib.h>

int main()
{
 char a ='A', b ='B';
 const char *ptr = &a;

 // *ptr = b; illegal statement (assignment of read-only location *ptr)

 // ptr can be changed
 printf("value pointed to by ptr: %c\n", *ptr);
 ptr = &b;
 printf("value pointed to by ptr: %c\n", *ptr);
}
chevron_right

filter_none

Output:

```
value pointed to by ptr:A
value pointed to by ptr:B
```

NOTE: There is no difference between **const char *p** and **char const *p** as both are pointer to a const char and position of '*'(asterisk) is also same.

2. **char *const ptr :** This is a constant pointer to non-constant character. **You cannot change the pointer p, but can change the value pointed by ptr.**

filter_none
edit
close
play_arrow
link
brightness_4
code

// C program to illustrate
// char* const p
#include<stdio.h>
#include<stdlib.h>

int main()
{
 char a ='A', b ='B';
 char *const ptr = &a;
 printf("Value pointed to by ptr: %c\n", *ptr);
 printf("Address ptr is pointing to: %d\n\n", ptr);

 // *ptr = b; illegal statement (assignment of read-only variable ptr)

 // changing the value at the address ptr is pointing to
 *ptr = b;
 printf("Value pointed to by ptr: %c\n", *ptr);
 printf("Address ptr is pointing to: %d\n", ptr);
}

chevron_right

filter_none

Output:

```
Value pointed to by ptr: A
Address ptr is pointing to: -1443150762

Value pointed to by ptr: B
Address ptr is pointing to: -1443150762
```

NOTE: Pointer always points to same address, only the value at the location is changed.

3. **const char * const ptr :** This is a constant pointer to constant character. **You can neither change the value pointed by ptr nor the pointer ptr.**

filter_none
edit
close
play_arrow
link
brightness_4
code

// C program to illustrate
// const char * const ptr

```
#include<stdio.h>
#include<stdlib.h>

int main()
{
    char a ='A', b ='B';
    const char *const ptr = &a;

    printf( "Value pointed to by ptr: %c\n", *ptr);
    printf( "Address ptr is pointing to: %d\n\n", ptr);

    // ptr = &b; illegal statement (assignment of read-only variable ptr)
    // *ptr = b; illegal statement (assignment of read-only location *ptr)
}
```

chevron_right

filter_none

Output:

```
Value pointed to by ptr: A
Address ptr is pointing to: -25509542
```

NOTE: `char const * const ptr` is same as `const char *const ptr`.

Quiz on const keyword

This article is contributed by **Yash Singla**. If you like GeeksforGeeks and would like to contribute, you can also write an article using [contribute.geeksforgeeks.org](#) or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes *arrow_drop_up*

Add your personal notes
here! (max 5000 chars)

Save

Recommended Posts:

- [Difference between const int*, const int * const, and int const *](#)
- [Const Qualifier in C](#)
- [How to modify a const variable in C?](#)
- [C++ | const keyword | Question 2](#)
- [C++ | const keyword | Question 3](#)
- [C++ | const keyword | Question 5](#)
- [C++ | const keyword | Question 1](#)
- [Difference between #define and const in C?](#)
- [Const member functions in C++](#)
- [C++ | const keyword | Question 5](#)
- [Why copy constructor argument should be const in C++?](#)
- [Different ways to use Const with Reference to a Pointer in C++](#)
- ["static const" vs "#define" vs "enum"](#)
- [Function overloading and const keyword](#)
- [What's difference between char s\[\] and char *s in C?](#)

Improved By : Deepankar Mullick

Article Tags :

C
C++
C-Pointer Basics
cpp-pointer
Practice Tags :
C
CPP

thumb_up
20

To-do Done
2.5

Based on 22 vote(s)

Basic **Easy** **Medium** **Hard** **Expert**

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) Quickly check if two STL vectors contain same elements or not

Next

[last_page](#) `getline` (string) in C++

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
Predefined Macros in C with Examples
C program to print odd line contents of a File followed by even line content
Introduction to the C99 Programming Language : Part II
C program to find square root of a given number

Most visited in C++
Vector of Vectors in C++ STL with Examples
C++ STL
Which C++ libraries are useful for competitive programming?
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

Pointer to an Array | Array Pointer

Prerequisite: [Pointers Introduction](#)

Pointer to Array

Consider the following program:

```
filter_none
edit
close

play_arrow
link
brightness_4
code

#include<stdio.h>

int main()
{
    int arr[5] = { 1, 2, 3, 4, 5 };
    int *ptr = arr;

    printf("%p\n", ptr);
    return 0;
}
chevron_right
filter_none
```

In this program, we have a pointer `ptr` that points to the 0th element of the array. Similarly, we can also declare a pointer that can point to whole array instead of only one element of the array. This pointer is useful when talking about multidimensional arrays.

Syntax:

```
data_type (*var_name)[size_of_array];
```

Example:

```
int (*ptr)[10];
```

Here `ptr` is pointer that can point to an array of 10 integers. Since subscript have higher precedence than indirection, it is necessary to enclose the indirection operator and pointer name inside parentheses. Here the type of `ptr` is 'pointer to an array of 10 integers'.

Note : The pointer that points to the 0th element of array and the pointer that points to the whole array are totally different. The following program shows this:

C++

```
filter_none
edit
close

play_arrow
link
brightness_4
code

// C++ program to understand difference between
// pointer to an integer and pointer to an
// array of integers.
#include <iostream>
using namespace std;
int main()
{
    // Pointer to an integer
    int *p;
```

```

// Pointer to an array of 5 integers
int (*ptr)[5];
int arr[5];

// Points to 0th element of the arr.
p = arr;

// Points to the whole array arr.
ptr = &arr;

cout << "p =" << p <<, ptr = "<< ptr<< endl;
p++;
ptr++;
cout << "p =" << p <<, ptr = "<< ptr<< endl;

return 0;
}

```

*// This code is contributed by SHUBHAMSINGH10
chevron_right*

filter_none

C

```

filter_none
edit
close
play_arrow
link
brightness_4
code

// C program to understand difference between
// pointer to an integer and pointer to an
// array of integers.
#include<stdio.h>

```

```

int main()
{
    // Pointer to an integer
    int *p;

    // Pointer to an array of 5 integers
    int (*ptr)[5];
    int arr[5];

    // Points to 0th element of the arr.
    p = arr;

    // Points to the whole array arr.
    ptr = &arr;

    printf("p = %p, ptr = %p\n", p, ptr);

    p++;
    ptr++;

    printf("p = %p, ptr = %p\n", p, ptr);
}

return 0;
}

```

chevron_right

filter_none

Output:

```

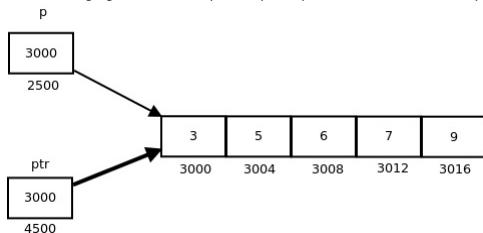
p = 0x7fff4f32fd50, ptr = 0x7fff4f32fd50
p = 0x7fff4f32fd54, ptr = 0x7fff4f32fd64

```

p: is pointer to 0th element of the array *arr*, while **ptr** is a pointer that points to the whole array *arr*.

- The base type of *p* is *int* while base type of *ptr* is 'an array of 5 integers'.
- We know that the pointer arithmetic is performed relative to the base size, so if we write *ptr++*, then the pointer *ptr* will be shifted forward by 20 bytes.

The following figure shows the pointer *p* and *ptr*. Darker arrow denotes pointer to an array.



On dereferencing a pointer expression we get a value pointed to by that pointer expression. Pointer to an array points to an array, so on dereferencing it, we should get the array, and the name of array denotes the base address. So whenever a pointer to an array is dereferenced, we get the base address of the array to which it points.

C++

filter_none
edit

```

close
play_arrow
link
brightness_4
code

// C++ program to illustrate sizes of
// pointer of array
#include <bits/stdc++.h>
using namespace std;

```

```

int main()
{
    int arr[] = { 3, 5, 6, 7, 9 };
    int *p = arr;
    int (*ptr)[5] = &arr;

    cout << "p = " << p << endl;
    cout << "*p = " << *p << endl;
    cout << "ptr = " << ptr << endl;
    cout << "arr = " << arr << endl;

    cout << "sizeof(p) = " << sizeof(p) << endl;
    cout << "sizeof(*p) = " << sizeof(*p) << endl;
    cout << "sizeof(ptr) = " << sizeof(ptr) << endl;
    cout << "sizeof(*ptr) = " << sizeof(*ptr) << endl;
    return 0;
}

```

```

// This code is contributed by shubhamsingh10
chevron_right
filter_none

```

C

```

filter_none
edit
close
play_arrow
link
brightness_4
code

// C program to illustrate sizes of
// pointer of array
#include<stdio.h>

int main()
{
    int arr[] = { 3, 5, 6, 7, 9 };
    int *p = arr;
    int (*ptr)[5] = &arr;

    printf("p = %p, ptr = %p\n", p, ptr);
    printf("*p = %d, *ptr = %p\n", *p, *ptr);

    printf("sizeof(p) = %lu, sizeof(*p) = %lu\n",
           sizeof(p), sizeof(*p));
    printf("sizeof(ptr) = %lu, sizeof(*ptr) = %lu\n",
           sizeof(ptr), sizeof(*ptr));
    return 0;
}

```

```

chevron_right
filter_none

```

Output:

```

p = 0x7ffddee5010, ptr = 0x7ffddee5010
*p = 3, *ptr = 0x7ffddee5010
sizeof(p) = 8, sizeof(*p) = 4
sizeof(ptr) = 8, sizeof(*ptr) = 20

```

Pointer to Multidimensional Arrays

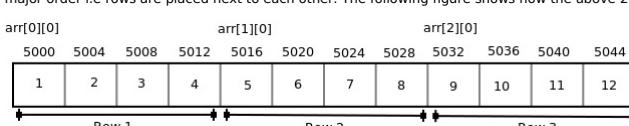
- Pointers and two dimensional Arrays: In a two dimensional array, we can access each element by using two subscripts, where first subscript represents the row number and second subscript represents the column number. The elements of 2-D array can be accessed with the help of pointer notation also. Suppose arr is a 2-D array, we can access any element $arr[i][j]$ of the array using the pointer expression $*(*arr + i) + j$. Now we'll see how this expression can be derived.

Let us take a two dimensional array $arr[3][4]$:

```
int arr[3][4] = { {1, 2, 3, 4}, {5, 6, 7, 8}, {9, 10, 11, 12} };
```

	Col 1	Col 2	Col 3	Col 4
Row 1	1	2	3	4
Row 2	5	6	7	8
Row 3	9	10	11	12

Since memory in a computer is organized linearly it is not possible to store the 2-D array in rows and columns. The concept of rows and columns is only theoretical, actually, a 2-D array is stored in row-major order i.e rows are placed next to each other. The following figure shows how the above 2-D array will be stored in memory.

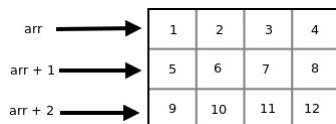


Each row can be considered as a 1-D array, so a two-dimensional array can be considered as a collection of one-dimensional arrays that are placed one after another. In other words, we can say that 2-D

dimensional arrays that are placed one after another. So here `arr` is an array of 3 elements where each element is a 1-D array of 4 integers.

We know that the name of an array is a constant pointer that points to 0th 1-D array and contains address 5000. Since `arr` is a 'pointer to an array of 4 integers', according to pointer arithmetic the expression `arr + 1` will represent the address 5016 and expression `arr + 2` will represent address 5032.

So we can say that `arr` points to the 0th 1-D array, `arr + 1` points to the 1st 1-D array and `arr + 2` points to the 2nd 1-D array.



<code>arr</code>	- Points to 0 th element of arr	- Points to 0 th 1-D array	- 5000
<code>arr + 1</code>	- Points to 1 th element of arr	- Points to 1 st 1-D array	- 5016
<code>arr + 2</code>	- Points to 2 th element of arr	- Points to 2 nd 1-D array	- 5032

In general we can write:

<code>arr + i</code>	Points to i th element of arr ->	Points to i th 1-D array
----------------------	---	-------------------------------------

- Since `arr + i` points to ith element of `arr`, on dereferencing it will get ith element of `arr` which is of course a 1-D array. Thus the expression `*(arr + i)` gives us the base address of ith 1-D array.
- We know, the pointer expression `*(arr + i)` is equivalent to the subscript expression `arr[i]`. So `*(arr + i)` which is same as `arr[i]` gives us the base address of ith 1-D array.

<code>*(arr + 0)</code>	- arr[0]	- Base address of 0 th 1-D array	- Points to 0 th element of 0 th 1-D array	- 5000
<code>*(arr + 1)</code>	- arr[1]	- Base address of 1 st 1-D array	- Points to 0 th element of 1 st 1-D array	- 5016
<code>*(arr + 2)</code>	- arr[2]	- Base address of 2 nd 1-D array	- Points to 0 th element of 2 nd 1-D array	- 5032

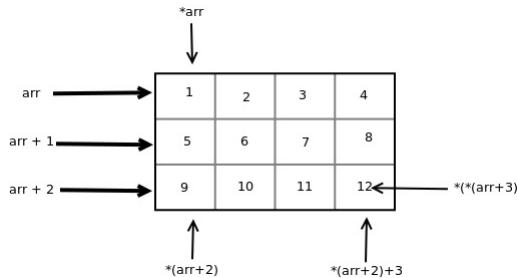
In general we can write:

<code>*(arr + i)</code>	- arr[i]	- Base address of i th 1-D array ->	Points to 0 th element of i th 1-D array
-------------------------	----------	--	--

Note: Both the expressions `(arr + i)` and `*(arr + i)` are pointers, but their base type are different. The base type of `(arr + i)` is 'an array of 4 units' while the base type of `*(arr + i)` or `arr[i]` is int.

- To access an individual element of our 2-D array, we should be able to access any jth element of ith 1-D array.
- Since the base type of `*(arr + i)` is int and it contains the address of 0th element of ith 1-D array, we can get the addresses of subsequent elements in the ith 1-D array by adding integer values to `*(arr + i)`.
- For example `*(arr + i) + 1` will represent the address of 1st element of ith 1-D array and `*(arr + i) + 2` will represent the address of 2nd element of ith 1-D array.
- Similarly `*(arr + i) + j` will represent the address of jth element of ith 1-D array. On dereferencing this expression we can get the jth element of the ith 1-D array.

<code>arr</code>	Points to 0 th 1-D array
<code>*arr</code>	Points to 0 th element of 0 th 1-D array
<code>(arr + i)</code>	Points to i th 1-D array
<code>*(arr + i)</code>	Points to 0 th element of i th 1-D array
<code>*(arr + i) + j</code>	Points to j th element of i th 1-D array
<code>*(arr + i) + j</code>	Represents the value of j th element of i th 1-D array



```
filter_none
edit
close

play_arrow

link
brightness_4
code

// C program to print the values and
// address of elements of a 2-D array
#include<stdio.h>
```

```
int main()
{
    int arr[3][4] = {
        { 10, 11, 12, 13 },
        { 20, 21, 22, 23 },
        { 30, 31, 32, 33 }
    };

    int i, j;
    for (i = 0; i < 3; i++)
    {
        printf("Address of %dth array = %p %p\n",
               i, arr[i], *(arr + i));

        for (j = 0; j < 4; j++)
            printf("%d %d ", arr[i][j], *(*(arr + i) + j));
        printf("\n");
    }

    return 0;
}
```

chevron_right

filter_none

Output:

Address of 0th array = 0x7ffe50edd580 0x7ffe50edd580
10 10 11 11 12 12 13 13

```

Address of 1th array = 0x7ffe50edd590 0x7ffe50edd590
20 21 21 22 22 23 23
Address of 2th array = 0x7ffe50edd5a0 0x7ffe50edd5a0
30 30 31 31 32 32 33 33

```

2. Pointers and Three Dimensional Arrays

In a three dimensional array we can access each element by using three subscripts. Let us take a 3-D array-

```
int arr[2][3][2] = { {5, 10}, {6, 11}, {7, 12}, {20, 30}, {21, 31}, {22, 32} };
```

We can consider a three dimensional array to be an array of 2-D array i.e each element of a 3-D array is considered to be a 2-D array. The 3-D array *arr* can be considered as an array consisting of two elements where each element is a 2-D array. The name of the array *arr* is a pointer to the 0th 2-D array.

arr	Points to 0 th 2-D array.
arr + i	Points to i th 2-D array.
*(arr + i)	Gives base address of i th 2-D array, so points to 0 th element of i th 2-D array, each element of 2-D array is a 1-D array, so it points to 0 th 1-D array of i th 2-D array.
*(arr + i) + j	Points to j th 1-D array of i th 2-D array.
*(*arr + i) + j	Gives base address of j th 1-D array of i th 2-D array so it points to 0 th element of j th 1-D array of i th 2-D array.
*(*arr + i) + j + k	Represents the value of j th element of i th 1-D array.
*(*arr + i) + j + k	Gives the value of k th element of j th 1-D array of i th 2-D array.

Thus the pointer expression ***(*(*arr + i) + j) + k** is equivalent to the subscript expression **arr[i][j][k]**.

We know the expression ***(arr + i)** is equivalent to **arr[i]** and the expression ***(*arr + i) + j** is equivalent **arr[i][j]**. So we can say that **arr[i]** represents the base address of ith 2-D array and **arr[i][j]** represents the base address of the jth 1-D array.

filter_none
edit
close

play_arrow

link

brightness_4
code

```
// C program to print the elements of 3-D
// array using pointer notation
#include<stdio.h>
int main()
{
    int arr[2][3][2] = {
        {
            {5, 10},
            {6, 11},
            {7, 12},
        },
        {
            {20, 30},
            {21, 31},
            {22, 32},
        }
    };

    int i, j, k;
    for (i = 0; i < 2; i++)
    {
        for (j = 0; j < 3; j++)
        {
            for (k = 0; k < 2; k++)
                printf("%d\t", *(*(*arr + i) + j) + k);
            printf("\n");
        }
    }

    return 0;
}
```

chevron_right

filter_none

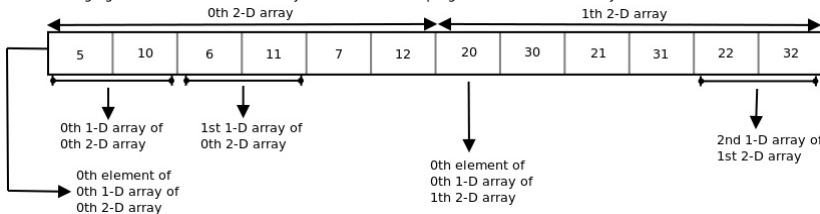
Output:

```

5 10
6 11
7 12
20 30
21 31
22 32

```

The following figure shows how the 3-D array used in the above program is stored in memory.



Subscripting Pointer to an Array

Suppose *arr* is a 2-D array with 3 rows and 4 columns and *ptr* is a pointer to an array of 4 integers, and *ptr* contains the base address of array *arr*.

```
int arr[3][4] = {{10, 11, 12, 13}, {20, 21, 22, 23}, {30, 31, 32, 33}};
int (*ptr)[4];
ptr = arr;
```

5000	→	5000
10	11	12
20	21	22
30	31	32

Since *ptr* is a pointer to an array of 4 integers, *ptr + i* will point to *i*th row. On dereferencing *ptr + i*, we get base address of *i*th row. To access the address of *j*th element of *i*th row we can add *j* to the pointer expression *(*ptr + i)*. So the pointer expression *(*ptr + i) + j* gives the address of *j*th element of *i*th row and the pointer expression **(*ptr + i) + j* gives the value of the *j*th element of *i*th row. We know that the pointer expression **(*ptr + i) + j* is equivalent to subscript expression *ptr[i][j]*. So if we have a pointer variable containing the base address of 2-D array, then we can access the elements of array by double subscripting that pointer variable.

```
filter_none
edit
close

play_arrow
link
brightness_4
code

// C program to print elements of a 2-D array
// by scripting a pointer to an array
#include<stdio.h>
```

```
int main()
{
    int arr[3][4] = {
        {10, 11, 12, 13},
        {20, 21, 22, 23},
        {30, 31, 32, 33}
    };

    int (*ptr)[4];
    ptr = arr;
    printf("%p %p\n", ptr, ptr + 1, ptr + 2);
    printf("%p %p %p\n", *ptr, *(ptr + 1), *(ptr + 2));
    printf("%d %d %d\n", **ptr, *((ptr + 1) + 2), *((ptr + 2) + 3));
    printf("%d %d %d\n", ptr[0][0], ptr[1][2], ptr[2][3]);
    return 0;
}
```

chevron_right

filter_none

Output:

```
0x7ffead967560 0x7ffead967570 0x7ffead967580
0x7ffead967560 0x7ffead967570 0x7ffead967580
10 22 33
10 22 33
```

This article is contributed by [Anuj Chauhan](#). If you like GeeksforGeeks and would like to contribute, you can also write an article using [contribute.geeksforgeeks.org](#) or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.,

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up

Add your personal notes
here! (max 5000 chars)

Save

Recommended Posts:

- Difference between pointer to an array and array of pointers
- Pointer vs Array in C
- Difference between pointer and array in C?
- Sum of array using pointer arithmetic
- Declare a C/C++ function returning pointer to array of integer pointers
- Double Pointer (Pointer to Pointer) In C
- What is a Pointer to a Null pointer
- 'this' pointer in C++
- C++ | this pointer | Question 5
- C++ | this pointer | Question 4
- C++ | this pointer | Question 3
- C++ | this pointer | Question 2
- C++ | this pointer | Question 1
- NULL pointer in C
- void pointer in C / C++

Improved By : [flandraco](#), [BabisSarantoglou](#), [nidhi_biet](#), [SHUBHAMSINGH10](#)

Article Tags :

C
C++
C-Pointers
cpp-pointer
Practice Tags :
C
CPP

thumb_up
69

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

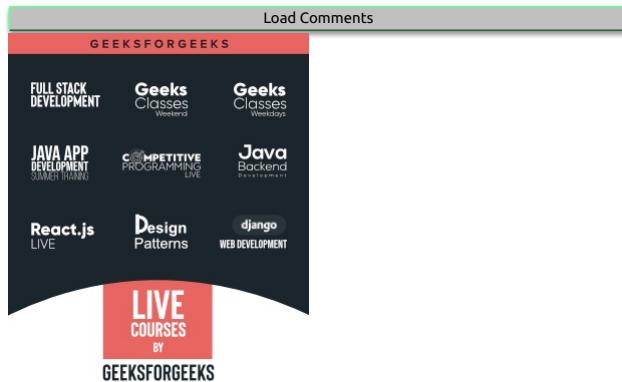
Previous

[first_page Comparison of boolean data type in C++ and Java](#)

Next

[last_page std::mismatch\(\) with examples in C++](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

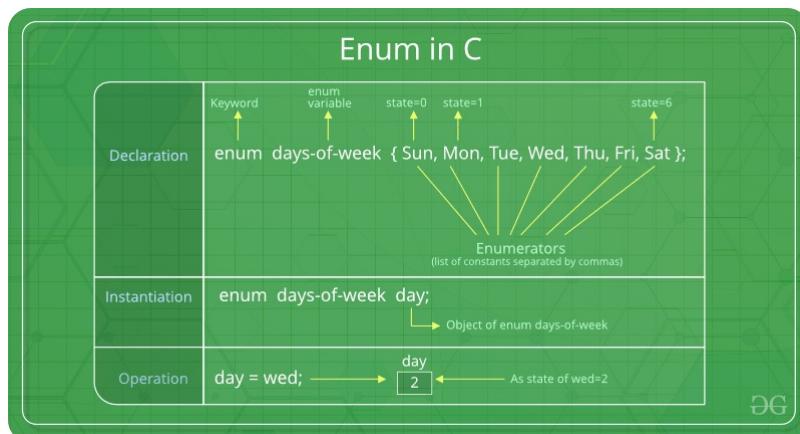


Most popular in C
Print all possible combinations of the string by replacing 's' with any other digit from the string
How to use function within function in C or C++
Format specifiers in different Programming Languages
Predefined Macros in C with Examples
C program print odd line contents of a File followed by even line content

Most visited in C++
Vector of Vectors in C++ STL with Examples
C++ Tutorial
Which C++ libraries are useful for competitive programming?
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

Enumeration (or enum) in C

Enumeration (or enum) is a user defined data type in C. It is mainly used to assign names to integral constants, the names make a program easy to read and maintain.



```
enum State {Working = 1, Failed = 0};
```

The keyword 'enum' is used to declare new enumeration types in C and C++. Following is an example of enum declaration.

```
// The name of enumeration is "flag" and the constant
// are the values of the flag. By default, the values
// of the constants are as follows:
// constant1 = 0, constant2 = 1, constant3 = 2 and
// so on.
enum flag{constant1, constant2, constant3, .....};
```

Variables of type enum can also be defined. They can be defined in two ways:

```
// In both of the below cases, "day" is
// defined as the variable of type week.

enum week{Mon, Tue, Wed};
enum week day;

// Or

enum week{Mon, Tue, Wed}day;
```

filter_none
edit
close
play_arrow
link

```
brightness_4  
code  
// An example program to demonstrate working  
// of enum in C  
#include<stdio.h>
```

```
enum week{Mon, Tue, Wed, Thur, Fri, Sat, Sun};
```

```
int main()  
{  
    enum week day;  
    day = Wed;  
    printf("%d",day);  
    return 0;  
}
```

```
chevron_right
```

```
filter_none
```

Output:

```
2
```

In the above example, we declared "day" as the variable and the value of "Wed" is allocated to day, which is 2. So as a result, 2 is printed.

Another example of enumeration is:

```
filter_none  
edit  
close  
play_arrow  
link  
brightness_4  
code  
// Another example program to demonstrate working  
// of enum in C  
#include<stdio.h>
```

```
enum year{Jan, Feb, Mar, Apr, May, Jun, Jul,  
         Aug, Sep, Oct, Nov, Dec};
```

```
int main()  
{  
    int i;  
    for (i=Jan; i<=Dec; i++)  
        printf("%d ", i);  
  
    return 0;  
}
```

```
chevron_right
```

```
filter_none
```

Output:

```
0 1 2 3 4 5 6 7 8 9 10 11
```

In this example, the for loop will run from i = 0 to i = 11, as initially the value of i is Jan which is 0 and the value of Dec is 11.

Interesting facts about initialization of enum.

- Two enum names can have same value. For example, in the following C program both 'Failed' and 'Freezed' have same value 0.

```
filter_none  
edit  
close  
play_arrow  
link  
brightness_4  
code  
#include <stdio.h>  
enum State {Working = 1, Failed = 0, Freezed = 0};
```

```
int main()  
{  
    printf("%d, %d, %d", Working, Failed, Freezed);  
    return 0;  
}
```

```
chevron_right
```

```
filter_none
```

Output:

```
1, 0, 0
```

- If we do not explicitly assign values to enum names, the compiler by default assigns values starting from 0. For example, in the following C program, sunday gets value 0, monday gets 1, and so on.

```
filter_none  
edit  
close  
play_arrow  
link  
brightness_4  
code  
#include <stdio.h>
```

```
enum day {sunday, monday, tuesday, wednesday, thursday, friday, saturday};
```

```
int main()
{
    enum day d = thursday;
    printf("The day number stored in d is %d", d);
    return 0;
}
```

[chevron_right](#)

[filter_none](#)

Output:

```
The day number stored in d is 4
```

3. We can assign values to some name in any order. All unassigned names get value as value of previous name plus one.

```
filter_none
edit
close

play_arrow
link
brightness_4
code

#include <stdio.h>
enum day {sunday = 1, monday, tuesday = 5,
           wednesday, thursday = 10, friday, saturday};
```

```
int main()
{
    printf("%d %d %d %d %d", sunday, monday, tuesday,
           wednesday, thursday, friday, saturday);
    return 0;
}
```

[chevron_right](#)

[filter_none](#)

Output:

```
1 2 5 6 10 11 12
```

4. The value assigned to enum names must be some integeral constant, i.e., the value must be in range from minimum possible integer value to maximum possible integer value.

5. All enum constants must be unique in their scope. For example, the following program fails in compilation.

```
filter_none
edit
close

play_arrow
link
brightness_4
code

enum state {working, failed};
enum result {failed, passed};

int main() { return 0; }
```

[chevron_right](#)

[filter_none](#)

Output:

```
Compile Error: 'failed' has a previous declaration as 'state failed'
```

Exercise:

Predict the output of following C programs

Program 1:

```
filter_none
edit
close

play_arrow
link
brightness_4
code

#include <stdio.h>
enum day {sunday = 1, tuesday, wednesday, thursday, friday, saturday};

int main()
{
    enum day d = thursday;
    printf("The day number stored in d is %d", d);
    return 0;
}
```

[chevron_right](#)

[filter_none](#)

Program 2:

```
filter_none
edit
close

play_arrow
```

```
link  
brightness_4  
code  
  
#include <stdio.h>  
enum State {WORKING = 0, FAILED, FREEZED};  
enum State currState = 2;  
  
enum State FindState() {  
    return currState;  
}  
  
int main() {  
    (FindState() == WORKING)? printf("WORKING"): printf("NOT WORKING");  
    return 0;  
}
```

[chevron_right](#)
[filter_none](#)

Enum vs Macro

We can also use macros to define names constants. For example we can define 'Working' and 'Failed' using following macro.

```
filter_none  
edit  
close  
  
play_arrow  
  
link  
brightness_4  
code  
  
#define Working 0  
#define Failed 1  
#define Freezed 2  
chevron_right  
  
filter_none
```

There are multiple advantages of using enum over macro when many related named constants have integral values.

- a) Enums follow scope rules.
- b) Enum variables are automatically assigned values. Following is simpler

```
filter_none  
edit  
close  
  
play_arrow  
  
link  
brightness_4  
code  
  
enum state {Working, Failed, Freezed};  
chevron_right  
  
filter_none
```

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

Introduction part of this article is contributed by **Piyush Vashistha**. If you like GeeksforGeeks and would like to contribute, you can also write an article using [contribute.geeksforgeeks.org](#) or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes [arrow_drop_up](#)

[Save](#)

Recommended Posts:

- "static const" vs "#define" vs "enum"
- Format specifiers in different Programming Languages
- C program to find square root of a given number
- C program to print odd line contents of a File followed by even line content
- Getting System and Process Information Using C Programming and Shell in Linux
- Program to calculate Electricity Bill
- Program to print half Diamond star pattern
- C/C++ program for calling main() in main()
- Print all possible combinations of the string by replacing '\$' with any other digit from the string
- typeid operator in C++ with Examples
- How to use make utility to build C projects?
- How to call function within function in C or C++
- Role of SemiColon in various Programming Languages
- C program to display month by month calendar for a given year

Improved By : KartheekMudarakola

Article Tags :

C
C Basics
C-Struct-Union-Enum
cpp-data-types

Practice Tags :

C

[thumb_up](#)
103

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) Union in C

Next

[last_page](#) C | Misc | Question 6

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Weekend Geeks Classes Weekdays

JAVA APP DEVELOPMENT SUMMER TRAINING COMPETITIVE PROGRAMMING LIVE Java Backend Development

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS

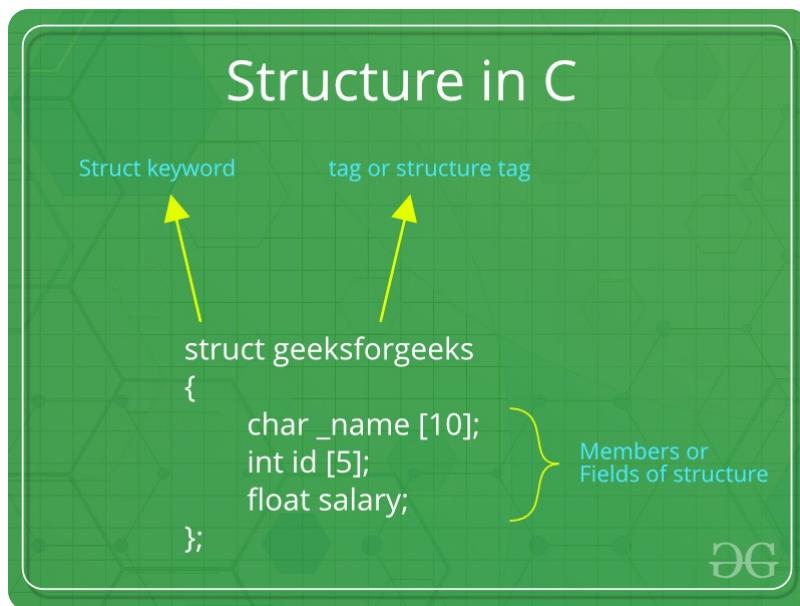
Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
ctype.h<ctype.h> library in C/C++ with Examples
Implicit Type Conversion in C with Examples
How to call function within function in C or C++
Format specifiers in different Programming Languages

More related articles in C
Predefined Macros in C with Examples
C program to print odd line content of a file followed by even line content
C program to find square root of a given number
Introduction to the C99 Programming Language : Part II
Problem in comparing Floating point numbers and how to compare them correctly?

Structures in C

What is a structure?

A structure is a user defined data type in C/C++. A structure creates a data type that can be used to group items of possibly different types into a single type.



How to create a structure?

'struct' keyword is used to create a structure. Following is an example.

```
filter_none
edit
close
play_arrow
link
brightness_4
code
struct address
{
    char name[50];
    char street[100];
}
```

```
char city[50];
char state[20];
int pin;
};

chevron_right
filter_none
```

How to declare structure variables?

A structure variable can either be declared with structure declaration or as a separate declaration like basic types.

```
filter_none
edit
close

play_arrow
link
brightness_4
code

// A variable declaration with structure declaration.
struct Point
{
    int x, y;
} p1; // The variable p1 is declared with 'Point'

// A variable declaration like basic data types
struct Point
{
    int x, y;
};

int main()
{
    struct Point p1; // The variable p1 is declared like a normal variable
}
chevron_right
filter_none
```

Note: In C++, the struct keyword is optional before in declaration of a variable. In C, it is mandatory.

How to initialize structure members?

Structure members **cannot be** initialized with declaration. For example the following C program fails in compilation.

```
filter_none
edit
close

play_arrow
link
brightness_4
code

struct Point
{
    int x = 0; // COMPILER ERROR: cannot initialize members here
    int y = 0; // COMPILER ERROR: cannot initialize members here
};

chevron_right
filter_none
```

The reason for above error is simple, when a datatype is declared, no memory is allocated for it. Memory is allocated only when variables are created.

Structure members **can be** initialized using curly braces '{}'. For example, following is a valid initialization.

```
filter_none
edit
close

play_arrow
link
brightness_4
code

struct Point
{
    int x, y;
};

int main()
{
    // A valid initialization. member x gets value 0 and y
    // gets value 1. The order of declaration is followed.
    struct Point p1 = {0, 1};
}

chevron_right
filter_none
```

How to access structure elements?

Structure members are accessed using dot(.) operator.

```
filter_none
edit
close

play_arrow
link
```

```
brightness_4
code

#include<stdio.h>

struct Point
{
    int x, y;
};

int main()
{
    struct Point p1 = {0, 1};

    // Accessing members of point p1
    p1.x = 20;
    printf ("x = %d, y = %d", p1.x, p1.y);

    return 0;
}
chevron_right
```

[filter_none](#)

Output:

```
x = 20, y = 1
```

What is designated Initialization?

Designated initialization allows structure members to be initialized in any order. This feature has been added in C99 standard.

```
filter_none
edit
close

play_arrow
link
brightness_4
code

#include<stdio.h>
```

```
struct Point
{
    int x, y, z;
};

int main()
{
    // Examples of initialization using designated initialization
    struct Point p1 = {.y = 0, .z = 1, .x = 2};
    struct Point p2 = {.x = 20};

    printf ("x = %d, y = %d, z = %d\n", p1.x, p1.y, p1.z);
    printf ("x = %d", p2.x);
    return 0;
}
```

[chevron_right](#)

[filter_none](#)

Output:

```
x = 2, y = 0, z = 1
x = 20
```

This feature is not available in C++ and works only in C.

What is an array of structures?

Like other primitive data types, we can create an array of structures.

```
filter_none
edit
close

play_arrow
link
brightness_4
code

#include<stdio.h>
```

```
struct Point
{
    int x, y;
};

int main()
{
    // Create an array of structures
    struct Point arr[10];

    // Access array members
    arr[0].x = 10;
    arr[0].y = 20;

    printf("%d %d", arr[0].x, arr[0].y);
    return 0;
}
```

[chevron_right](#)

[filter_none](#)

Output:

10 20

What is a structure pointer?

Like primitive types, we can have pointer to a structure. If we have a pointer to structure, members are accessed using arrow (->) operator.

```
filter_none
edit
close

play_arrow
link
brightness_4
code

#include<stdio.h>

struct Point
{
    int x, y;
};

int main()
{
    struct Point p1 = {1, 2};

    // p2 is a pointer to structure p1
    struct Point *p2 = &p1;

    // Accessing structure members using structure pointer
    printf("%d %d", p2->x, p2->y);
    return 0;
}
```

chevron_right

filter_none

Output:

1 2

What is structure member alignment?

See <https://www.geeksforgeeks.org/structure-member-alignment-padding-and-data-packing/>

Limitations of C Structures

In C language, Structures provide a method for packing together data of different types. A Structure is a helpful tool to handle a group of logically related data items. However, C structures have some limitations.

- The C structure does not allow the struct data type to be treated like built-in data types:
- We cannot use operators like +, - etc. on Structure variables. For example, consider the following code:

```
filter_none
edit
close

play_arrow
link
brightness_4
code

struct number
{
    float x;
};

int main()
{
    struct number n1,n2,n3;
    n1.x=4;
    n2.x=3;
    n3=n1+n2;
    return 0;
}

/*Output:

prog.c: In function 'main':
prog.c:10:7: error:
invalid operands to binary + (have 'struct number' and 'struct number')
    n3=n1+n2;
```

chevron_right

filter_none

- No Data Hiding:** C Structures do not permit data hiding. Structure members can be accessed by any function, anywhere in the scope of the Structure.
- Functions inside Structure:** C structures do not permit functions inside Structure
- Static Members:** C Structures cannot have static members inside their body
- Access Modifiers:** C Programming language do not support access modifiers. So they cannot be used in C Structures.
- Construction creation in Structures:** Structures in C cannot have constructor inside Structures.

Related Article : C Structures vs C++ Structures

We will soon be discussing union and other struct related topics in C. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up

Add your personal notes
here! (max 5000 chars)

Save

Recommended Posts:

- Difference between C structures and C++ structures
- Structures in C++
- Slack Bytes in Structures : Explained with Example
- C program to store Student records as Structures and Sort them by Name
- C/C++ program to add N distances given in inch-feet system using Structures
- Format specifiers in different Programming Languages
- C program to find square root of a given number
- C program to print odd line contents of a File followed by even line content
- C/C++ program to add N distances given in inch-feet system using Structures
- Getting System and Process Information Using C Programming and Shell in Linux
- Program to calculate Electricity Bill
- Program to print half Diamond star pattern
- C/C++ program for calling main() in main()
- Count of integral coordinates that lies inside a Square

Improved By : [RajeetGoyal](#), [mxp](#), [RishabhPrabhu](#), [SidharthSunil](#), [shubham_singh](#)

Article Tags :

C
C Basics
C-Struct-Union-Enum
CBSE - Class 11
cpp-structure
school-programming
Practice Tags :

C



78

To-do Done
2.4

Based on 58 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) [C++](#) | [Misc C++](#) | [Question 7](#)

Next

[last_page](#) [C](#) | [Structure & Union](#) | [Question 10](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Weekend Geeks Classes Weekdays

JAVA APP DEVELOPMENT SUMMER TRAINING COMPETITIVE PROGRAMMING LIVE Java Backend Development

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS

Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
ctype.h<ctype> Library in C/C++ with Examples
Program to print Infix with Examples
Implicit Type Conversion in C with Examples
How to call function within function in C or C++
More related articles in C
C program to print odd line contents of a File followed by even line content
Introduction to the C99 Programming Language : Part II
C program to find square root of a given number
Format specifiers in different Programming Languages
Problem in comparing Floating point numbers and how to compare them correctly?

Union in C

Like [Structures](#), union is a user defined data type. In union, all members share the same memory location.

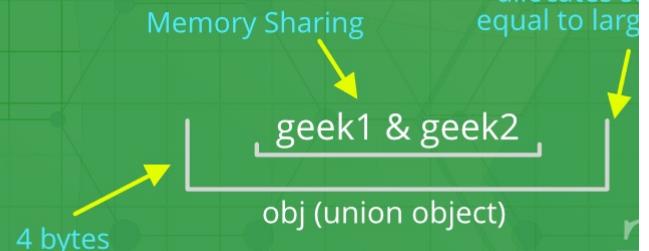
Structure

```
struct Geeksforgeeks
{
    char X;          //size 1 byte
    float Y;         //size 4 byte
} obj;
```



Unions

```
union Geeksforgeeks
{
    char X;
    float Y;
} obj;
```



For example in the following C program, both x and y share the same location. If we change x, we can see the changes being reflected in y.

```
filter_none
edit
close

play_arrow
link
brightness_4
code

#include <stdio.h>

// Declaration of union is same as structures
union test {
    int x, y;
};

int main()
{
    // A union variable t
    union test t;

    t.x = 2; // t.y also gets value 2
    printf("After making x = 2:\n x = %d, y = %d\n\n",
           t.x, t.y);

    t.y = 10; // t.x is also updated to 10
    printf("After making y = 10:\n x = %d, y = %d\n\n",
           t.x, t.y);

    return 0;
}

chevron_right
filter_none
Output:
After making x = 2:
x = 2, y = 2

After making y = 10:
x = 10, y = 10
```

How is the size of union decided by compiler?

Size of a union is taken according the size of largest member in union.

```
filter_none
edit
close

play_arrow
link
brightness_4
code

#include <stdio.h>

union test1 {
    int x;
    int y;
} Test1;
```

```

union test2 {
    int x;
    char y;
} Test2;

union test3 {
    int arr[10];
    char y;
} Test3;

int main()
{
    printf("sizeof(test1) = %lu, sizeof(test2) = %lu, "
        "sizeof(test3) = %lu",
        sizeof(Test1),
        sizeof(Test2), sizeof(Test3));
    return 0;
}

```

Chevron_right

filter_none

Output:

```
sizeof(test1) = 4, sizeof(test2) = 4, sizeof(test3) = 40
```

Pointers to unions?

Like structures, we can have pointers to unions and can access members using the arrow operator (->). The following example demonstrates the same.

```

filter_none
edit
close

play_arrow

link
brightness_4
code

#include <stdio.h>

union test {
    int x;
    char y;
};

int main()
{
    union test p1;
    p1.x = 65;

    // p2 is a pointer to union p1
    union test* p2 = &p1;

    // Accessing union members using pointer
    printf("%d %c", p2->x, p2->y);
    return 0;
}

```

Chevron_right

filter_none

Output:

```
65 A
```

What are applications of union?

Unions can be useful in many situations where we want to use the same memory for two or more members. For example, suppose we want to implement a binary tree data structure where each leaf node has a double data value, while each internal node has pointers to two children, but no data. If we declare this as:

```

filter_none
edit
close

play_arrow

link
brightness_4
code

struct NODE {
    struct NODE* left;
    struct NODE* right;
    double data;
};


```

Chevron_right

filter_none

then every node requires 16 bytes, with half the bytes wasted for each type of node. On the other hand, if we declare a node as following, then we can save space.

```

filter_none
edit
close

play_arrow

link
brightness_4
code

struct NODE {
    bool is_leaf;
    union {

```

```
struct
{
    struct NODE* left;
    struct NODE* right;
} internal;
double data;
} info;
};

chevron_right
filter_none
```

The above example is taken from [Computer Systems : A Programmer's Perspective \(English\) 2nd Edition](#) book.

References:

http://en.wikipedia.org/wiki/Union_type

[Computer Systems : A Programmer's Perspective \(English\) 2nd Edition](#)

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes [arrow_drop_up](#)

[Save](#)

Recommended Posts:

- [Difference between Structure and Union in C](#)
- [C | Structure & Union | Question 6](#)
- [Anonymous Union and Structure in C](#)
- [C | Structure & Union | Question 1](#)
- [C | Structure & Union | Question 10](#)
- [C | Structure & Union | Question 9](#)
- [C | Structure & Union | Question 8](#)
- [C | Structure & Union | Question 7](#)
- [C | Structure & Union | Question 5](#)
- [C | Structure & Union | Question 4](#)
- [C | Structure & Union | Question 10](#)
- [C | Structure & Union | Question 2](#)
- [Output of C++ programs | Set 41 \(Structure and Union\)](#)
- [Format specifiers in different Programming Languages](#)

Improved By : [RajeetGoyal](#), [RishabhPrabhu](#), [Code_r](#)

Article Tags :

C
C Basics
C-Struct-Union-Enum
cpp-structure

Practice Tags :

C

[thumb_up](#)
23

To-do Done
2.6

Based on 39 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) [C | Structure & Union | Question 10](#)

Next

[last_page](#) [Enumeration \(or enum\) in C](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

[Load Comments](#)



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
ctype.h(<cctype>) library in C/C++ with Examples

Implicit Type Conversion in C with Examples
Format specifiers in different Programming Languages
C program to find square root of a given number

More related articles in C
Predefined Macros in C with Examples
How to call a function with variable in C or C++
C program to print end line contents of a file followed by even line content
Introduction to the C99 Programming Language : Part II
Features of C Programming Language

Struct Hack

What will be the size of following structure?

```
filter_none
edit
close
play_arrow
link
brightness_4
code

struct employee
{
    int    emp_id;
    int    name_len;
    char   name[0];
};

chevron_right
filter_none
```

4 + 4 + 0 = 8 bytes.

And what about size of "name[0]". In gcc, when we create an array of zero length, it is considered as array of incomplete type that's why gcc reports its size as "0" bytes. This technique is known as "Struct Hack". When we create array of zero length inside structure, it must be (and only) last member of structure. Shortly we will see how to use it.

"Struct Hack" technique is used to create variable length member in a structure. In the above structure, string length of "name" is not fixed, so we can use "name" as variable length array.

Let us see below memory allocation.

```
struct employee *e = malloc(sizeof(*e) + sizeof(char) * 128);
```

is equivalent to

```
filter_none
edit
close
play_arrow
link
brightness_4
code

struct employee
{
    int    emp_id;
    int    name_len;
    char   name[128]; /* character array of size 128 */
};

chevron_right
filter_none
```

And below memory allocation

```
struct employee *e = malloc(sizeof(*e) + sizeof(char) * 1024);
```

is equivalent to

```
filter_none
edit
close
play_arrow
link
brightness_4
code
```

```
struct employee
{
    int emp_id;
    int name_len;
    char name[1024]; /* character array of size 1024 */
};
```

[chevron_right](#)

[filter_none](#)

Note: since name is character array, in malloc instead of "sizeof(char) * 128", we can use "128" directly. sizeof is used to avoid confusion.

Now we can use "name" same as pointer. e.g.

```
e->emp_id = 100;
e->name_len = strlen("Geeks For Geeks");
strcpy(e->name, "Geeks For Geeks", e->name_len);
```

When we allocate memory as given above, compiler will allocate memory to store "emp_id" and "name_len" plus contiguous memory to store "name". When we use this technique, gcc guarantees that, "name" will get contiguous memory.

Obviously there are other ways to solve problem, one is we can use character pointer. But there is no guarantee that character pointer will get contiguous memory, and we can take advantage of this contiguous memory. For example, by using this technique, we can allocate and deallocate memory by using single malloc and free call (because memory is contiguous). Other advantage of this is, suppose if we want to write data, we can write whole data by using single "write()" call. e.g.

```
write(fd, e, sizeof(*e) + name_len); /* write emp_id + name_len + name */
```

If we use character pointer, then we need 2 write calls to write data. e.g.

```
write(fd, e, sizeof(*e)); /* write emp_id + name_len */
write(fd, e->name, e->name_len); /* write name */
```

Note: In C99, there is feature called "flexible array members", which works same as "Struct Hack"

This article is compiled by **Narendra Kangalkar**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes [arrow_drop_up](#)

Add your personal notes here! (max 5000 chars)

[Save](#)

Recommended Posts:

- Operations on struct variables in C
- Conversion of Struct data type to Hex String and vice versa
- Is sizeof for a struct equal to the sum of sizeof of each member?
- Format specifiers in different Programming Languages
- C program to find square root of a given number
- C program to print odd line contents of a File followed by even line content
- C/C++ program to add N distances given in inch-feet system using Structures
- Getting System and Process Information Using C Programming and Shell in Linux
- Program to calculate Electricity Bill
- Program to print half Diamond star pattern
- C/C++ program for calling main() in main()
- Print all possible combinations of the string by replacing '\$' with any other digit from the string
- How to use make utility to build C projects?
- How to call function within function in C or C++

Article Tags :

C
C-Struct-Union-Enum
cpp-structure
Practice Tags :

C

[thumb_up](#)
9

To-do Done
4.1

Based on 48 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) [const_cast in C++ | Type Casting operators](#)

Next

[last_page](#) [Default arguments and virtual function](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

[Load Comments](#)



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
Predefined Macros in C with Examples
C program to print odd line contents of a File followed by even line content
Format specifiers in different Programming Languages
Introduction to the C99 Programming Language : Part II

More related articles in C
C program to find square root of a given number
Problems in Comparing Floating Point numbers and how to compare them correctly?
Features of C Programming Language
Pointer Expressions in C with Examples
Introduction to the C99 Programming Language : Part III

Structure Member Alignment, Padding and Data Packing

What do we mean by data alignment, structure packing and padding?

Predict the output of following program.

```
filter_none
```

```
edit
```

```
close
```

```
play_arrow
```

```
link
```

```
brightness_4
```

```
code
```

```
#include <stdio.h>
```

```
// Alignment requirements
// (typical 32 bit machine)
```

```
// char      1 byte
// short int  2 bytes
// int       4 bytes
// double    8 bytes
```

```
// structure A
```

```
typedef struct structa_tag
{
    char      c;
    short int s;
} structa_t;
```

```
// structure B
```

```
typedef struct structb_tag
{
    short int s;
    char      c;
    int       i;
} structb_t;
```

```
// structure C
```

```
typedef struct structc_tag
{
    char      c;
    double   d;
    int       s;
} structc_t;
```

```
// structure D
```

```
typedef struct structd_tag
{
    double   d;
    int      s;
    char      c;
} structd_t;
```

```
int main()
```

```
{  
    printf("sizeof(structa_t) = %d\n", sizeof(structa_t));  
    printf("sizeof(structb_t) = %d\n", sizeof(structb_t));  
    printf("sizeof(structc_t) = %d\n", sizeof(structc_t));  
}
```

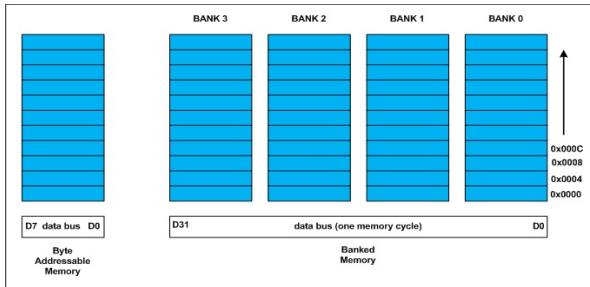
```

printf("sizeof(structd_t) = %d\n", sizeof(structd_t));
return 0;
}
chevron_right
filter_none

```

Before moving further, write down your answer on a paper, and read on. If you urge to see explanation, you may miss to understand any lacuna in your analogy. **Data Alignment:**

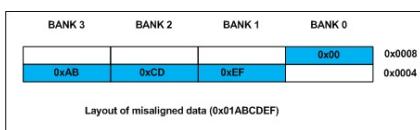
Every data type in C/C++ will have alignment requirement (infact it is mandated by processor architecture, not by language). A processor will have processing word length as that of data bus size. On a 32 bit machine, the processing word size will be 4 bytes.



Historically memory is byte addressable and arranged sequentially. If the memory is arranged as single bank of one byte width, the processor needs to issue 4 memory read cycles to fetch an integer. It is more economical to read all 4 bytes of integer in one memory cycle. To take such advantage, the memory will be arranged as group of 4 banks as shown in the above figure.

The memory addressing still be sequential. If bank 0 occupies an address X, bank 1, bank 2 and bank 3 will be at (X + 1), (X + 2) and (X + 3) addresses. If an integer of 4 bytes is allocated on X address (X is multiple of 4), the processor needs only one memory cycle to read entire integer.

Where as, if the integer is allocated at an address other than multiple of 4, it spans across two rows of the banks as shown in the below figure. Such an integer requires two memory read cycle to fetch the data.



A variable's **data alignment** deals with the way the data stored in these banks. For example, the natural alignment of **int** on 32-bit machine is 4 bytes. When a data type is naturally aligned, the CPU fetches it in minimum read cycles.

Similarly, the natural alignment of **short int** is 2 bytes. It means, a **short int** can be stored in bank 0 - bank 1 pair or bank 2 - bank 3 pair. A **double** requires 8 bytes, and occupies two rows in the memory banks. Any misalignment of **double** will force more than two read cycles to fetch **double** data.

Note that a **double** variable will be allocated on 8 byte boundary on 32 bit machine and requires two memory read cycles. On a 64 bit machine, based on number of banks, **double** variable will be allocated on 8 byte boundary and requires only one memory read cycle.

Structure Padding:

In C/C++ a structures are used as data pack. It doesn't provide any data encapsulation or data hiding features (C++ case is an exception due to its semantic similarity with classes).

Because of the alignment requirements of various data types, every member of structure should be naturally aligned. The members of structure allocated sequentially increasing order. Let us analyze each struct declared in the above program.

Output of Above Program:

For the sake of convenience, assume every structure type variable is allocated on 4 byte boundary (say 0x0000), i.e. the base address of structure is multiple of 4 (need not necessary always, see explanation of **structc_t**).

structure A

The **structa_t** first element is **char** which is one byte aligned, followed by **short int**. **short int** is 2 byte aligned. If the the **short int** element is immediately allocated after the **char** element, it will start at an odd address boundary. The compiler will insert a padding byte after the **char** to ensure **short int** will have an address multiple of 2 (i.e. 2 byte aligned). The total size of **structa_t** will be **sizeof(char)** + 1 (padding) + **sizeof(short)**, $1 + 1 + 2 = 4$ bytes.

structure B

The first member of **structb_t** is **short int** followed by **char**. Since **char** can be on any byte boundary no padding required in between **short int** and **char**, on total they occupy 3 bytes. The next member is **int**. If the **int** is allocated immediately, it will start at an odd byte boundary. We need 1 byte padding after the **char** member to make the address of next **int** member is 4 byte aligned. On total, the **structb_t** requires $2 + 1 + 1$ (padding) + 4 = 8 bytes.

structure C - Every structure will also have alignment requirements

Applying same analysis, **structc_t** needs **sizeof(char)** + 7 byte padding + **sizeof(double)** + **sizeof(int)** = $1 + 7 + 8 + 4 = 20$ bytes. However, the **sizeof(structc_t)** will be 24 bytes. It is because, along with structure members, structure type variables will also have natural alignment. Let us understand it by an example. Say, we declared an array of **structc_t** as shown below

```
structc_t structc_array[3];
```

Assume, the base address of **structc_array** is 0x0000 for easy calculations. If the **structc_t** occupies 20 (0x14) bytes as we calculated, the second **structc_t** array element (indexed at 1) will be at 0x0000 + 0x0014 = 0x0014. It is the start address of index 1 element of array. The double member of this **structc_t** will be allocated on $0x0014 + 0x1 + 0x7 = 0x001C$ (decimal 28) which is not multiple of 8 and conflicting with the alignment requirements of **double**. As we mentioned on the top, the alignment requirement of **double** is 8 bytes.

Inorder to avoid such misalignment, compiler will introduce alignment requirement to every structure. It will be as that of the largest member of the structure. In our case alignment of **structa_t** is 2, **structb_t** is 4 and **structc_t** is 8. If we need nested structures, the size of largest inner structure will be the alignment of immediate larger structure.

In **structc_t** of the above program, there will be padding of 4 bytes after **int** member to make the structure size multiple of its alignment. Thus the **sizeof (structc_t)** is 24 bytes. It guarantees correct alignment even in arrays. You can cross check.

structure D - How to Reduce Padding?

By now, it may be clear that padding is unavoidable. There is a way to minimize padding. The programmer should declare the structure members in their increasing/decreasing order of size. An example is **structd_t** given in our code, whose size is 16 bytes in lieu of 24 bytes of **structc_t**.

What is structure packing?

Some times it is mandatory to avoid padded bytes among the members of structure. For example, reading contents of ELF file header or BMP or JPEG file header. We need to define a structure similar to that of the header layout and map it. However, care should be exercised in accessing such members. Typically reading byte by byte is an option to avoid misaligned exceptions. There will be hit on performance.

Most of the compilers provide non standard extensions to switch off the default padding like pragmas or command line switches. Consult the documentation of respective compiler for more details.

Pointer Mishaps:

There is possibility of potential error while dealing with pointer arithmetic. For example, dereferencing a generic pointer (`void *`) as shown below can cause misaligned exception,

```
// Dereferencing a generic pointer (not safe)
// There is no guarantee that pGeneric is integer aligned
```

```
*(int *)pGeneric;
```

It is possible above type of code in programming. If the pointer `pGeneric` is not aligned as per the requirements of casted data type, there is possibility to get misaligned exception.

Infact few processors will not have the last two bits of address decoding, and there is no way to access *misaligned* address. The processor generates misaligned exception, if the programmer tries to access such address.

A note on malloc() returned pointer

The pointer returned by `malloc()` is `void *`. It can be converted to any data type as per the need of programmer. The implementer of `malloc()` should return a pointer that is aligned to maximum size of primitive data types (those defined by compiler). It is usually aligned to 8 byte boundary on 32 bit machines.

Object File Alignment, Section Alignment, Page Alignment

These are specific to operating system implementer, compiler writers and are beyond the scope of this article. Infact, I don't have much information.

General Questions:

1. Is alignment applied for stack?

Yes. The stack is also memory. The system programmer should load the stack pointer with a memory address that is properly aligned. Generally, the processor won't check stack alignment, it is the programmer's responsibility to ensure proper alignment of stack memory. Any misalignment will cause run time surprises.

For example, if the processor word length is 32 bit, stack pointer also should be aligned to be multiple of 4 bytes.

2. If char data is placed in a bank other bank 0, it will be placed on wrong data lines during memory read. How the processor handles char type?

Usually, the processor will recognize the data type based on instruction (e.g. LDRB on ARM processor). Depending on the bank it is stored, the processor shifts the byte onto least significant data lines.

3. When arguments passed on stack, are they subjected to alignment?

Yes. The compiler helps programmer in making proper alignment. For example, if a 16-bit value is pushed onto a 32-bit wide stack, the value is automatically padded with zeros out to 32 bits. Consider the following program.

```
filter_none
edit
close

play_arrow
link
brightness_4
code

void argument_alignment_check( char c1, char c2 )
{
    // Considering downward stack
    // (on upward stack the output will be negative)
    printf("Displacement %d\n", (int)&c2 - (int)&c1);
}

chevron_right
filter_none
```

The output will be 4 on a 32 bit machine. It is because each character occupies 4 bytes due to alignment requirements.

4. What will happen if we try to access a misaligned data?

It depends on processor architecture. If the access is misaligned, the processor automatically issues sufficient memory read cycles and packs the data properly onto the data bus. The penalty is on performance. Where as few processors will not have last two address lines, which means there is no-way to access odd byte boundary. Every data access must be aligned (4 bytes) properly. A misaligned access is critical exception on such processors. If the exception is ignored, read data will be incorrect and hence the results.

5. Is there any way to query alignment requirements of a data type.

Yes. Compilers provide non standard extensions for such needs. For example, `__alignof()` in Visual Studio helps in getting the alignment requirements of data type. Read MSDN for details.

6. When memory reading is efficient in reading 4 bytes at a time on 32 bit machine, why should a double type be aligned on 8 byte boundary?

It is important to note that most of the processors will have math co-processor, called Floating Point Unit (FPU). Any floating point operation in the code will be translated into FPU instructions. The main processor is nothing to do with floating point execution. All this will be done behind the scenes.

As per standard, double type will occupy 8 bytes. And, every floating point operation performed in FPU will be of 64 bit length. Even float types will be promoted to 64 bit prior to execution.

The 64 bit length of FPU registers forces double type to be allocated on 8 byte boundary. I am assuming (I don't have concrete information) in case of FPU operations, data fetch might be different, I mean the data bus, since it goes to FPU. Hence, the address decoding will be different for double types (which is expected to be on 8 byte boundary). It means, *the address decoding circuits of floating point unit will not have last 3 pins*.

Answers:

```
sizeof(struct_a_t) = 4
sizeof(structb_t) = 8
sizeof(structc_t) = 24
sizeof(structd_t) = 16
```

Update: 1-May-2013

It is observed that on latest processors we are getting size of `struct_c` as 16 bytes. I yet to read relevant documentation. I will update once I got proper information (written to few experts in hardware).

On older processors (AMD Athlon X2) using same set of tools (GCC 4.7) I got `struct_c` size as 24 bytes. The size depends on how memory banking organized at the hardware level.

-- by Venki. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up

Add your personal notes
here! (max 5000 chars)

Save

Recommended Posts:

- [How to avoid Structure Padding in C?](#)
- [Can we access private data members of a class without using a member or a friend function?](#)
- [Commonly Asked Data Structure Interview Questions | Set 1](#)
- [Design a Queue data structure to get minimum or maximum in O\(1\) time](#)
- [Count the number of objects using Static member function](#)
- [Is sizeof for a struct equal to the sum of sizeof of each member?](#)
- [Difference between fundamental data types and derived data types](#)
- [Anonymous Union and Structure in C](#)
- [C | Structure & Union | Question 10](#)
- [C | Structure & Union | Question 2](#)

- C | Structure & Union | Question 4
- C | Structure & Union | Question 1
- Difference between Structure and Union in C
- C | Structure & Union | Question 5
- C | Structure & Union | Question 6

Article Tags :

Articles
C
C-Struct-Union-Enum
Practice Tags :

C

37

To-do Done
4.6

Based on 98 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) Reentrant Function

Next

[last_page](#) Order of operands for logical operators

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

[Load Comments](#)

GEEKSFORGEEKS

FULL STACK DEVELOPMENT
Geeks Classes Webinars
Geeks Classes Workshops

JAVA APP DEVELOPMENT SUMMER TRAINING
COMPETITIVE PROGRAMMING LIVE
Java Backend Development

React.js LIVE
Design Patterns
django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS

Most popular in Articles
new vs delete and new[] vs delete in C++
How to build a simple music player app using Android Studio
When to use each sorting algorithms
How to change the default icon of Android App
Difference Between sum of degrees of odd and even degree nodes in an Undirected Graph

Most visited in C
P(n) in C++ with Examples
Speed up Code executions with help of Pragma in C/C++
Modulo Operator (%) In C/C++ with Examples
Program to Calculate Electricity Bill
Role of SemiColon in various Programming Languages

Operations on struct variables in C

In C, the only operation that can be applied to *struct* variables is assignment. Any other operation (e.g. equality check) is not allowed on *struct* variables.
For example, program 1 works without any error and program 2 fails in compilation.

Program 1

```
filter_none
edit
close
play_arrow
link
brightness_4
code
#include <stdio.h>

struct Point {
    int x;
    int y;
};

int main()
{
    struct Point p1 = {10, 20};
    struct Point p2 = p1; // works: contents of p1 are copied to p2
    printf(" p2.x = %d, p2.y = %d", p2.x, p2.y);
    getchar();
    return 0;
}
chevron_right
```

filter_none

Program 2

```

filter_none
edit
close
play_arrow
link
brightness_4
code

#include <stdio.h>

struct Point {
    int x;
    int y;
};

int main()
{
    struct Point p1 = {10, 20};
    struct Point p2 = p1; // works: contents of p1 are copied to p2
    if (p1 == p2) // compiler error: cannot do equality check for
                  // whole structures
    {
        printf("p1 and p2 are same ");
    }
    getchar();
    return 0;
}
Chevron_right
filter_none

```

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes *arrow_drop_up*

Add your personal notes
here! (max 5000 chars)

Save

Recommended Posts:

- Difference between Static variables and Register variables in C
- Struct Hack
- Conversion of Struct data type to Hex String and vice versa
- Static Variables in C
- Variables and Keywords in C
- Is sizeof for a struct equal to the sum of sizeof of each member?
- Implicit initialization of variables with 0 or 1 in C
- Can Global Variables be dangerous ?
- Constants vs Variables in C language
- Initialization of static variables in C
- How will you show memory representation of C variables?
- Initialization of variables sized arrays in C
- What are the default values of static variables in C?
- How are variables scoped in C - Static or Dynamic?
- C Program to print environment variables

Improved By : Kaalan

Article Tags :

C
C-Struct-Union-Enum
Practice Tags :
C

thumb_up
6

To-do Done
1.8

Based on 26 vote(s)

Basic	Easy	Medium	Hard	Expert
-------	------	--------	------	--------

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

first_page Use of realloc()

Next

last_page Structure Member Alignment, Padding and Data Packing

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Weekend Geeks Classes Weekdays

JAVA APP DEVELOPMENT SERVER TRAINING COMPETITIVE PROGRAMMING LITE Java Backend Development

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS

Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
`ctype.h <cctype.h>` library in C/C++ with Examples
Predefined Macros in C with Examples
Implicit Type Conversion in C with Examples
Format specifiers in different Programming Languages

More related articles in C
How to call function within function in C or C++
C program to print contents of file followed by even line content
Introduction to the C99 Programming Language - Part II
C program to find square root of a given number
Problem in comparing Floating point numbers and how to compare them correctly?

Bit Fields in C

In C, we can specify size (in bits) of structure and union members. The idea is to use memory efficiently when we know that the value of a field or group of fields will never exceed a limit or is within a small range.

For example, consider the following declaration of date without the use of bit fields.

```
filter_none
edit
close
play_arrow
link
brightness_4
code
#include <stdio.h>

// A simple representation of the date
struct date {
    unsigned int d;
    unsigned int m;
    unsigned int y;
};

int main()
{
    printf("Size of date is %lu bytes\n",
           sizeof(struct date));
    struct date dt = { 31, 12, 2014 };
    printf("Date is %d/%d/%d", dt.d, dt.m, dt.y);
}
chevron_right
```

Output:

```
Size of date is 12 bytes
Date is 31/12/2014
```

The above representation of 'date' takes 12 bytes on a compiler where an unsigned int takes 4 bytes. Since we know that the value of d is always from 1 to 31, the value of m is from 1 to 12, we can optimize the space using bit fields.

```
filter_none
edit
close
play_arrow
link
brightness_4
code
#include <stdio.h>

// Space optimized representation of the date
struct date {
    // d has value between 1 and 31, so 5 bits
    // are sufficient
    unsigned int d : 5;

    // m has value between 1 and 12, so 4 bits
    // are sufficient
    unsigned int m : 4;
```

```

    unsigned int y;
};

int main()
{
    printf("Size of date is %lu bytes\n", sizeof(struct date));
    struct date dt = { 31, 12, 2014 };
    printf("Date is %d/%d/%d", dt.d, dt.m, dt.y);
    return 0;
}

```

chevron_right

filter_none

Output:

```
Size of date is 8 bytes
Date is 31/12/2014
```

However if the same code is written using signed int and the value of the fields goes beyond the bits allocated to the variable and something interesting can happen. For example consider the same code but with signed integers:

```

filter_none
edit
close

play_arrow

link
brightness_4
code

#include <stdio.h>

// Space optimized representation of the date
struct date {
    // d has value between 1 and 31, so 5 bits
    // are sufficient
    int d : 5;

    // m has value between 1 and 12, so 4 bits
    // are sufficient
    int m : 4;

    int y;
};

int main()
{
    printf("Size of date is %lu bytes\n",
           sizeof(struct date));
    struct date dt = { 31, 12, 2014 };
    printf("Date is %d/%d/%d", dt.d, dt.m, dt.y);
    return 0;
}

```

chevron_right

filter_none

Output:

```
Size of date is 8 bytes
Date is -1/-4/2014
```

The output comes out to be negative. What happened behind is that the value 31 was stored in 5 bit signed integer which is equal to 11111. The MSB is a 1, so it's a negative number and you need to calculate the 2's complement of the binary number to get its actual value which is what is done internally. By calculating 2's complement you will arrive at the value 00001 which is equivalent to decimal number 1 and since it was a negative number you get a -1. A similar thing happens to 12 in which case you get 4-bit representation as 1100 which on calculating 2's complement you get the value of -4.

Following are some interesting facts about bit fields in C.

1) A special unnamed bit field of size 0 is used to force alignment on next boundary. For example consider the following program.

```

filter_none
edit
close

play_arrow

link
brightness_4
code

#include <stdio.h>

// A structure without forced alignment
struct test1 {
    unsigned int x : 5;
    unsigned int y : 8;
};

// A structure with forced alignment
struct test2 {
    unsigned int x : 5;
    unsigned int : 0;
    unsigned int y : 8;
};

int main()
{
    printf("Size of test1 is %lu bytes\n",
           sizeof(struct test1));
    printf("Size of test2 is %lu bytes\n",
           sizeof(struct test2));
    return 0;
}

```

chevron_right

filter_none

Output:

```
Size of test1 is 4 bytes
Size of test2 is 8 bytes
```

2) We cannot have pointers to bit field members as they may not start at a byte boundary.

filter_none

edit

close

play_arrow

link

brightness_4

code

```
#include <stdio.h>
struct test {
    unsigned int x : 5;
    unsigned int y : 5;
    unsigned int z;
};
int main()
{
    struct test t;

    // Uncommenting the following line will make
    // the program compile and run
    printf("Address of t.x is %p", &t.x);

    // The below line works fine as z is not a
    // bit field member
    printf("Address of t.z is %p", &t.z);
    return 0;
}
```

chevron_right

filter_none

Output:

```
prog.c: In function 'main':
prog.c:14:1: error: cannot take address of bit-field 'x'
printf("Address of t.x is %p", &t.x);
^
```

3) It is implementation defined to assign an out-of-range value to a bit field member.

filter_none

edit

close

play_arrow

link

brightness_4

code

```
#include <stdio.h>
struct test {
    unsigned int x : 2;
    unsigned int y : 2;
    unsigned int z : 2;
};
int main()
{
    struct test t;
    t.x = 5;
    printf("%d", t.x);
    return 0;
}
```

chevron_right

filter_none

Output:

Implementation-Dependent

4) In C++, we can have static members in a structure/class, but bit fields cannot be static.

filter_none

edit

close

play_arrow

link

brightness_4

code

```
// The below C++ program compiles and runs fine
struct test1 {
    static unsigned int x;
};
int main() {}
```

chevron_right

filter_none

Output:

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// But below C++ program fails in the compilation
// as bit fields cannot be static
struct test1 {
    static unsigned int x : 5;
};

int main() {}

chevron_right
```

```
filter_none
```

Output:

```
prog.cpp:5:29: error: static member 'x' cannot be a bit-field
    static unsigned int x : 5;
                           ^

```

5) Array of bit fields is not allowed. For example, the below program fails in the compilation.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

struct test {
    unsigned int x[10] : 5;
};

int main()
{}
```

```
chevron_right
filter_none
```

Output:

```
prog.c:3:1: error: bit-field 'x' has invalid type
unsigned int x[10]: 5;
^
```

Exercise:

Predict the output of following programs. Assume that unsigned int takes 4 bytes and long int takes 8 bytes.

1)

```
filter_none
edit
close
play_arrow
link
brightness_4
code

#include <stdio.h>
struct test {
    unsigned int x;
    unsigned int y : 33;
    unsigned int z;
};
int main()
{
    printf("%lu", sizeof(struct test));
    return 0;
}
```

```
chevron_right
filter_none
```

2)

```
filter_none
edit
close
play_arrow
link
brightness_4
code

#include <stdio.h>
struct test {
    unsigned int x;
    long int y : 33;
    unsigned int z;
```

```
};

int main()
{
    struct test t;
    unsigned int* ptr1 = &t.x;
    unsigned int* ptr2 = &t.z;
    printf("%d", ptr2 - ptr1);
    return 0;
}
```

chevron_right

filter_none

3)

```
filter_none
edit
close
play_arrow
link
brightness_4
code

union test {
    unsigned int x : 3;
    unsigned int y : 3;
    int z;
};
```

```
int main()
{
    union test t;
    t.x = 5;
    t.y = 4;
    t.z = 1;
    printf("t.x = %d, t.y = %d, t.z = %d",
           t.x, t.y, t.z);
    return 0;
}
```

chevron_right

filter_none

4) Use bit fields in C to figure out a way whether a machine is little-endian or big-endian.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes *arrow_drop_up*

Add your personal notes here! (max 5000 chars)

Recommended Posts:

- XOR of elements in a given range with updates using Fenwick Tree
- Query to count odd and even parity elements in subarray after XOR with K
- Maximum weighted edge in path between two nodes in an N-ary tree using binary lifting
- Split a binary string into K subsets minimizing sum of products of occurrences of 0 and 1
- Binary string with given frequencies of sums of consecutive pairs of characters
- Count of even and odd set bit Array elements after XOR with K for Q queries
- Program to convert Hexa-Decimal Number to its equivalent BCD
- Program to convert a BCD to Hexa-Decimal Number
- Convert the given BCD to its equivalent Binary form
- Count of integers up to N which represent a Binary number
- Remove all even parity nodes from a Doubly and Circular Singly Linked List
- Check if left and right shift of any string results into given string
- Count of pairs in an Array with same number of set bits
- Format specifiers in different Programming Languages

Improved By : [Khopadi](#)

Article Tags :

[Bit Magic](#)

[C](#)

[C-Struct-Union-Enum](#)

[cpp-structure](#)

Practice Tags :

[Bit Magic](#)

[C](#)

thumb_up

10

To-do Done
3.2

Based on 42 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) Cisco Interview Experience | Set 9 (For Experienced)

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Frontend Geeks Classes Backend

JAVA APP DEVELOPMENT SUMMER TRAINING COMPETITIVE PROGRAMMING LIVE Java Backend Backend

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS



Most popular in Bit Magic
Range Queries for finding the Sum of all even parity numbers
Minimum flip required to keep all 1s together in a Binary string
Find sum of xor of all unordered triplets of the array
Number of integers in a range [L, R] which are divisible by exactly K of its digits
Count of distinct XORs formed by rearranging two Binary strings

Most visited in C
P(n) in C++ with Examples
Speed up Code executions with help of Pragma in C/C++
Modulo Operator (%) in C/C++ with Examples
Program to calculate Electricity Bill
Role of SemiColon in various Programming Languages

Structure Sorting (By Multiple Rules) in C++

Prerequisite : [Structures in C](#)

Name and marks in different subjects (physics, chemistry and maths) are given for all students. The task is to compute total marks and ranks of all students. And finally display all students sorted by rank.

Rank of student is computed using below rules.

1. If total marks are different, then students with higher marks gets better rank.
2. If total marks are same, then students with higher marks in Maths gets better rank.
3. If total marks are same and marks in Maths are also same, then students with better marks in Physics gets better rank.
4. If all marks (total, Maths, Physics and Chemistry) are same, then any student can be assigned better rank.

Recommended: Please solve it on “[PRACTICE](#)” first, before moving on to the solution.

We use below structure to store details of students.

```
struct Student
{
    string name; // Given
    int math; // Marks in math (Given)
    int phy; // Marks in Physics (Given)
    int che; // Marks in Chemistry (Given)
    int total; // Total marks (To be filled)
    int rank; // Rank of student (To be filled)
};
```

We use `std::sort()` for **Structure Sorting**. In Structure sorting, all the respective properties possessed by the structure object are sorted on the basis of one (or more) property of the object.

In this example, marks of students in different subjects are provided by user. These marks in individual subjects are added to calculate the total marks of the student, which is then used to sort different students on the basis of their ranks (as explained above).

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// C++ program to demonstrate structure sorting in C++
#include <bits/stdc++.h>
using namespace std;

struct Student
{
    string name; // Given
    int math; // Marks in math (Given)
    int phy; // Marks in Physics (Given)
    int che; // Marks in Chemistry (Given)
    int total; // Total marks (To be filled)
    int rank; // Rank of student (To be filled)
};

// Function for comparing two students according
// to given rules
bool compareTwoStudents(Student a, Student b)
```

```

{
    // If total marks are not same then
    // returns true for higher total
    if (a.total != b.total )
        return a.total > b.total;

    // If marks in Maths are not same then
    // returns true for higher marks
    if (a.math != b.math)
        return a.math > b.math;

    return (a.phy > b.phy);
}

// Fills total marks and ranks of all Students
void computeRanks(Student a[], int n)
{
    // To calculate total marks for all Students
    for (int i=0; i<n; i++)
        a[i].total = a[i].math + a[i].phy + a[i].che;

    // Sort structure array using user defined
    // function compareTwoStudents()
    sort(a, a+n, compareTwoStudents);

    // Assigning ranks after sorting
    for (int i=0; i<n; i++)
        a[i].rank = i+1;
}

// Driver code
int main()
{
    int n = 5;

    // array of structure objects
    Student a[n];

    // Details of Student 1
    a[0].name = "Bryan";
    a[0].math = 80 ;
    a[0].phy = 95 ;
    a[0].che = 85 ;

    // Details of Student 2
    a[1].name= "Kevin" ;
    a[1].math= 95 ;
    a[1].phy= 85 ;
    a[1].che= 99 ;

    // Details of Student 3
    a[2].name = "Nick" ;
    a[2].math = 95 ;
    a[2].phy = 85 ;
    a[2].che = 80 ;

    // Details of Student 4
    a[3].name = "AJ" ;
    a[3].math = 80 ;
    a[3].phy = 70 ;
    a[3].che = 90 ;

    // Details of Student 5
    a[4].name = "Howie" ;
    a[4].math = 80 ;
    a[4].phy = 80 ;
    a[4].che = 80 ;

    computeRanks(a, n);

    //Column names for displaying data
    cout << "Rank" << "t" << "Name" << "t";
    cout << "Maths" << "t" << "Physics" << "t"
        << "Chemistry";
    cout << "t" << "Totaln";

    // Display details of Students
    for (int i=0; i<n; i++)
    {
        cout << a[i].rank << "t";
        cout << a[i].name << "t";
        cout << a[i].math << "t"
            << a[i].phy << "t"
            << a[i].che << "t";
        cout << a[i].total << "t";
        cout << "n";
    }

    return 0;
}

```

Output:

Rank	Name	Maths	Physics	Chemistry	Total
1	Kevin	95	85	99	279
2	Nick	95	85	80	260
3	Bryan	80	95	85	260
4	Howie	80	80	80	240
5	AJ	80	70	90	240

Related Articles:[sort\(\) in C++ STL](#)[Comparator function of qsort\(\) in C](#)[C qsort\(\) vs C++ sort\(\)](#)[Sort an array according to count of set bits](#)

This article is contributed by [Abhinav Tiwari](#). If you like GeeksforGeeks and would like to contribute, you can also write an article using [contribute.geeksforgeeks.org](#) or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes 

Add your personal notes here! (max 5000 chars)

[Save](#)

Recommended Posts:

- [Know Your Sorting Algorithm | Set 1 \(Sorting Weapons used by Programming Languages\)](#)
- [Know Your Sorting Algorithm | Set 2 \(Introsort- C++'s Sorting Weapon\)](#)
- [Sorting objects using In-Place sorting algorithm](#)
- [Scope rules in C](#)
- [Rules for operator overloading](#)
- [Generate an array of size N according to the given rules](#)
- [Static Data Structure vs Dynamic Data Structure](#)
- [sorting in fork\(\)](#)
- [Sorting in Java](#)
- [Sorting a vector in C++](#)
- [Alternative Sorting](#)
- [External Sorting](#)
- [When to use each sorting algorithms](#)
- [Sorting Terminology](#)
- [Sorting Big Integers](#)

Article Tags :

[Arrays](#)
[C](#)
[C++](#)
[Searching](#)
[Sorting](#)
[C-Struct-Union-Enum](#)
[cpp-structure](#)

Practice Tags :
[Arrays](#)
[Searching](#)
[Sorting](#)
[C](#)
[CPP](#)

 7

To-do Done
2.1

Based on 15 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) [C](#) | Macro & Preprocessor | Question 15

Next

[last_page](#) Find minimum elements after considering all possible transformations

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

[Load Comments](#)



Most popular in Arrays
Count of subarrays which start and end with the same element
Minimize the maximum difference of adjacent elements after at most K insertions
Longest Mountain Subarray
Search an element in a sorted and rotated array with duplicates
Find K-th smallest element in an array for multiple queries

Most visited in C
P(i) in C++ with Examples
Sprintf up to 1000 with help of Prgma in C/C++
Program to calculate Electricity Bill
Modulo Operator (%) in C/C++ with Examples
Role of SemiColon in various Programming Languages

Flexible Array Members in a structure in C

Flexible Array Member(FAM) is a feature introduced in the C99 standard of the C programming language.

- For the **structures** in C programming language from C99 standard onwards, we can declare an array **without a dimension** and whose size is flexible in nature.
- Such an array inside the structure should preferably be declared as the **last member** of structure and its size is variable(can be changed be at runtime).
- The structure must contain at least one more named member in addition to the flexible array member.

What must be the size of the structure below?

```
filter_none
edit
close

play_arrow

link
brightness_4
code

struct student
{
    int stud_id;
    int name_len;
    int struct_size;
    char stud_name[];
};

chevron_right
filter_none
```

The size of structure is = 4 + 4 + 4 + 0 = 12

In the above code snippet, the size i.e length of array "stud_name" isn't fixed and is an FAM.

The memory allocation using flexible array members(as per C99 standards) for the above example can be done as:

```
struct student *s = malloc( sizeof(*s) + sizeof(char [strlen(stud_name)]) );
```

Note: While using flexible array members in structures some convention regarding actual size of the member is defined.
In the above example the convention is that the member "stud_name" has character size.

For Example, Consider the following structure:

```
Input : id = 15, name = "Kartik"
Output : Student_id : 15
         Stud_Name : Kartik
         Name_Length: 6
         Allocated_Struct_size: 18
```

Memory allocation of above structure:

```
struct student *s =
    malloc( sizeof(*s) + sizeof(char [strlen("Kartik")]));
```

Its structure representation is equal to:

```
filter_none
edit
close

play_arrow

link
brightness_4
code

struct student
{
    int stud_id;
    int name_len;
    int struct_size;
```

```
char stud_name[6]; //character array of length 6
};
```

[chevron_right](#)

[filter_none](#)

Implementation

[filter_none](#)

[edit](#)

[close](#)

[play_arrow](#)

[link](#)

[brightness_4](#)

[code](#)

```
// C program for variable length members in
// structures in GCC
#include<string.h>
#include<stdio.h>
#include<stdlib.h>
```

```
// A structure of type student
struct student
```

{

```
    int stud_id;
    int name_len;
```

```
    // This is used to store size of flexible
    // character array stud_name[]
    int struct_size;
```

```
    // Flexible Array Member(FAM)
    // variable length array must be last
    // member of structure
    char stud_name[];
```

};

```
// Memory allocation and initialisation of structure
```

```
struct student *createStudent(struct student *s,
                               int id, char a[])
{
```

```
    // Allocating memory according to user provided
```

```
    // array of characters
```

```
    s =
        malloc( sizeof(*s) + sizeof(char) * strlen(a));
```

```
    s->stud_id = id;
```

```
    s->name_len = strlen(a);
```

```
    strcpy(s->stud_name, a);
```

```
    // Assigning size according to size of stud_name
    // which is a copy of user provided array a[].
```

```
    s->struct_size =
        (sizeof(*s) + sizeof(char) * strlen(s->stud_name));
```

```
    return s;
```

}

```
// Print student details
```

```
void printStudent(struct student *s)
```

{

```
    printf("Student_id : %d\n"
          "Stud_Name : %s\n"
          "Name_Length: %d\n"
          "Allocated_Struct_size: %d\n\n",
          s->stud_id, s->stud_name, s->name_len,
          s->struct_size);
```

```
    // Value of Allocated_Struct_size is in bytes here
```

}

```
// Driver Code
```

```
int main()
```

{

```
    struct student *s1 = createStudent(s1, 523, "Cherry");
    struct student *s2 = createStudent(s2, 535, "Sanjayulsha");
```

```
    printStudent(s1);
    printStudent(s2);
```

```
    // Size in struct student
    printf("Size of Struct student: %lu\n",
           sizeof(struct student));
```

```
    // Size in struct pointer
    printf("Size of Struct pointer: %lu",
           sizeof(s1));
```

```
    return 0;
```

}

[chevron_right](#)

[filter_none](#)

Output:

```
Student_id : 523
Stud_Name : SanjayKanna
Name_Length: 11
Allocated_Struct_size: 23

Student_id : 535
Stud_Name : Cherry
Name_Length: 6
Allocated_Struct_size: 18

Size of Struct student: 12
Size of Struct pointer: 8
```

Important Points:

1. Adjacent memory locations are used to store structure members in memory.
2. In previous standards of the C programming language, we were able to declare a zero size array member in place of a flexible array member. The GCC compiler with C89 standard considers it as zero size array.

This article is contributed by **Sanjay Kumar Ulsha** from **JNTUH College Of Engineering, Hyderabad**. If you like GeeksforGeeks and would like to contribute, you can also write an article using [contribute.geeksforgeeks.org](#) or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes [arrow_drop_up](#)

Add your personal notes here! (max 5000 chars)

[Save](#)

Recommended Posts:

- Are array members deeply copied?
- Different ways to Initialize all members of an array to the same value in C
- Static data members in C++
- Initialization of data members
- Difference between Structure and Array in C
- Can we access private data members of a class without using a member or a friend function?
- C | Structure & Union | Question 9
- C | Structure & Union | Question 7
- C | Structure & Union | Question 8
- C | Structure & Union | Question 10
- Anonymous Union and Structure in C
- C | Structure & Union | Question 1
- C | Structure & Union | Question 5
- C | Structure & Union | Question 2
- C | Structure & Union | Question 10

Improved By : [Akanksha_Rai](#)

Article Tags :

C
C-Struct-Union-Enum
C-Structure & Union
Practice Tags :

C

[thumb_up](#)
2

To-do Done

3.1

Based on 7 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) Wait System Call in C

Next

[last_page](#) C program that does not suspend when Ctrl+Z is pressed

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

[Load Comments](#)



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
Format specifiers in different Programming Languages
C program to find square root of a given number
Predefined Macros in C with Examples
C program to print odd line contents of a File followed by even line content

More related articles in C
Introduction to the C99 Programming Language : Part II
Features of Programming language
Problems in comparing Floating point numbers and how to compare them correctly?
<cfloat> float.h in C/C++ with Examples
Getting System and Process Information Using C Programming and Shell in Linux

Difference between Structure and Union in C

[structures in C](#)

A structure is a user-defined data type available in C that allows combining data items of different kinds. Structures are used to represent a record.

Defining a structure: To define a structure, you must use the **struct** statement. The struct statement defines a new data type, with more than or equal to one member. The format of the struct statement is as follows:

```
struct [structure name]
{
    member definition;
    member definition;
    ...
    member definition;
};
```

[union](#)

A union is a special data type available in C that allows storing different data types in the same memory location. You can define a union with many members, but only one member can contain a value at any given time. Unions provide an efficient way of using the same memory location for multiple purposes.

Defining a Union: To define a union, you must use the **union** statement in the same way as you did while defining a structure. The union statement defines a new data type with more than one member for your program. The format of the union statement is as follows:

```
union [union name]
{
    member definition;
    member definition;
    ...
    member definition;
};
```

[Similarities between Structure and Union](#)

- Both are user-defined data types used to store data of different types as a single unit.
- Their members can be objects of any type, including other structures and unions or arrays. A member can also consist of a bit field.
- Both structures and unions support only assignment = and sizeof operators. The two structures or unions in the assignment must have the same members and member types.
- A structure or a union can be passed by value to functions and returned by value by functions. The argument must have the same type as the function parameter. A structure or union is passed by value just like a scalar variable as a corresponding parameter.
- '.' operator is used for accessing members.

[Differences](#)

	STRUCTURE	UNION
Keyword	The keyword struct is used to define a structure.	The keyword union is used to define a union.
Size	When a variable is associated with a structure, the compiler allocates the memory for each member. The size of structure is greater than or equal to the sum of sizes of its members.	when a variable is associated with a union, the compiler allocates the memory by considering the size of the largest memory. So, size of union is equal to the size of largest member.
Memory	Each member within a structure is assigned unique storage area or location.	Memory allocated is shared by individual members of union.
Value Altering	Altering the value of a member will not affect other members of the structure.	Altering the value of any of the member will alter other member values.
Accessing members	Individual member can be accessed at a time.	Only one member can be accessed at a time.
Initialization of Members	Several members of a structure can initialize at once.	Only the first member of a union can be initialized.

filter_none
edit
close

play_arrow

link
brightness_4
code

```

// C program to illustrate differences
// between structure and Union
#include <stdio.h>
#include <string.h>

// declaring structure
struct struct_example
{
    int integer;
    float decimal;
    char name[20];
};

// declaraing union

union union_example
{
    int integer;
    float decimal;
    char name[20];
};

void main()
{
    // creating variable for structure
    // and initializing values difference
    // six
    struct struct_example s={18,38,"geeksforgeeks"};

    // creating variable for union
    // and initializing values
    union union_example u={18,38,"geeksforgeeks"};

    printf("structure data:\n integer: %d\n"
          "decimal: %.2f\n name: %s\n",
          s.integer, s.decimal, s.name);
    printf("\nunion data:\n integer: %d\n"
          "decimal: %.2f\n name: %s\n",
          u.integer, u.decimal, u.name);

    // difference two and three
    printf("\nsizeof structure : %d\n", sizeof(s));
    printf("sizeof union : %d\n", sizeof(u));

    // difference five
    printf("\n Accessing all members at a time:");
    s.integer = 183;
    s.decimal = 90;
    strcpy(s.name, "geeksforgeeks");

    printf("structure data:\n integer: %d\n "
          "decimal: %.2f\n name: %s\n",
          s.integer, s.decimal, s.name);

    u.integer = 183;
    u.decimal = 90;
    strcpy(u.name, "geeksforgeeks");

    printf("\nunion data:\n integer: %d\n "
          "decimal: %.2f\n name: %s\n",
          u.integer, u.decimal, u.name);

    printf("\n Accessing one member at time:");

    printf("\nstructure data:");
    s.integer = 240;
    printf("\ninteger: %d", s.integer);

    s.decimal = 120;
    printf("\ndecimal: %f", s.decimal);

    strcpy(s.name, "C programming");
    printf("\nname: %s\n", s.name);

    printf("\n union data:");
    u.integer = 240;
    printf("\ninteger: %d", u.integer);

    u.decimal = 120;
    printf("\ndecimal: %f", u.decimal);

    strcpy(u.name, "C programming");
    printf("\nname: %s\n", u.name);

    //difference four
    printf("\naltering a member value:\n");
    s.integer = 1218;
    printf("structure data:\n integer: %d\n "
          "decimal: %.2f\n name: %s\n",
          s.integer, s.decimal, s.name);
}

```

```

s.integer, s.decimal, s.name);

u.integer = 1218;
printf("union data:\n integer: %d\n"
      " decimal: %.2f\n name: %s\n",
      u.integer, u.decimal, u.name);
}
chevron_right
filter_none

```

Output:

```

structure data:
integer: 18
decimal: 38.00
name: geeksforgeeks

union data:
integer: 18
decimal: 0.00
name: ?

sizeof structure: 28
sizeof union: 20

Accessing all members at a time: structure data:
integer: 183
decimal: 99.00
name: geeksforgeeks

union data:
integer: 1801887207
decimal: 277322871721159510000000000.00
name: geeksforgeeks

Accessing one member at a time:
structure data:
integer: 240
decimal: 120.000000
name: C programming

union data:
integer: 240
decimal: 120.000000
name: C programming

Altering a member value:
structure data:
integer: 1218
decimal: 120.00
name: C programming
union data:
integer: 1218
decimal: 0.00
name: ?

```

In my opinion, structure is better because as memory is shared in union ambiguity is more.

Quiz on Structures and Union

This article is contributed by **Harish Kumar**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up

[Save](#)

Recommended Posts:

- [C | Structure & Union | Question 1](#)
- [C | Structure & Union | Question 10](#)
- [C | Structure & Union | Question 10](#)
- [C | Structure & Union | Question 8](#)
- [C | Structure & Union | Question 7](#)
- [C | Structure & Union | Question 4](#)
- [C | Structure & Union | Question 5](#)
- [C | Structure & Union | Question 2](#)
- [Anonymous Union and Structure in C](#)
- [C | Structure & Union | Question 9](#)
- [C | Structure & Union | Question 6](#)
- [Difference between Single Bus Structure and Double Bus Structure](#)
- [Difference between JOIN and UNION in SQL](#)
- [Difference between Structure and Array in C](#)
- [Difference between data type and data structure](#)

Improved By : [arvindchoudhary6572](#)

Article Tags :

C
Difference Between
Practice Tags :
C

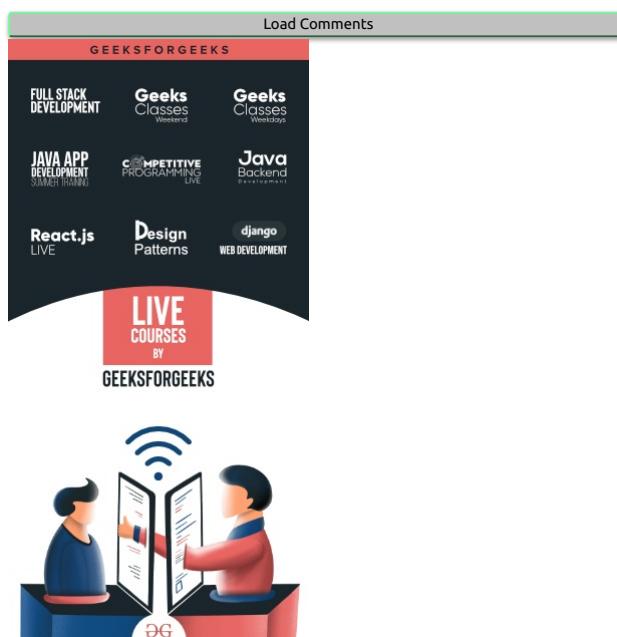
[thumb_up](#)
22

To-do Done
2.1

Based on 27 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Difference between C structures and C++ structures

In C++, struct and class are exactly the same things, except for that struct defaults to public visibility and class defaults to private visibility.

Some important differences between the C and C++ structures:

- Member functions inside structure:** Structures in C cannot have member functions inside structure but Structures in C++ can have member functions along with data members.
- Direct Initialization:** We cannot directly initialize structure data members in C but we can do it in C++.

C

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// C program to demonstrate that direct
// member initialization is not possible in C
#include <stdio.h>

struct Record {
    int x = 7;
};

// Driver Program
int main()
{
    struct Record s;
    printf("%d", s.x);
    return 0;
}

/* Output : Compiler Error
6:8: error: expected ',', ';', ';' or
'_attribute_' before '=' token
int x = 7;
^

In function 'main': */
chevron_right
```

filter_none

C++

```
filter_none
edit
close
play_arrow
```

```

link
brightness_4
code

// CPP program to initialize data member in C++
#include <iostream>
using namespace std;

struct Record {
    int x = 7;
};

// Driver Program
int main()
{
    Record s;
    cout << s.x << endl;
    return 0;
}
// Output
// 7
chevron_right
filter_none

```

Output:

7

3. **Using struct keyword:** In C, we need to use struct to declare a struct variable. In C++, struct is not necessary. For example, let there be a structure for Record. In C, we must use "struct Record" for Record variables. In C++, we need not use struct and using 'Record' only would work.

4. **Static Members:** C structures cannot have static members but is allowed in C++.

C

```

filter_none
edit
close

play_arrow
link
brightness_4
code

// C program with structure static member
struct Record {
    static int x;
};

// Driver program
int main()
{
    return 0;
}
/* 6:5: error: expected specifier-qualifier-list
   before 'static'
       static int x;
       ^*/
chevron_right
filter_none

```

C++

```

filter_none
edit
close

play_arrow
link
brightness_4
code

// C++ program with structure static member

struct Record {
    static int x;
};

// Driver program
int main()
{
    return 0;
}
chevron_right
filter_none

```

This will generate an error in C but no error in C++.

5. **Constructor creation in structure:** Structures in C cannot have constructor inside structure but Structures in C++ can have Constructor creation.

C

```

filter_none
edit

```

```

close
play_arrow
link
brightness_4
code

// C program to demonstrate that Constructor is not allowed
#include <stdio.h>

struct Student {
    int roll;
    Student(int x)
    {
        roll = x;
    }
};

// Driver Program
int main()
{
    struct Student s(2);
    printf("%d", s.x);
    return 0;
}

/* Output : Compiler Error
[Error] expected specifier-qualifier-list
before 'Student'
[Error] expected declaration specifiers or
'...' before numeric constant
[Error] 's' undeclared (first use
555555555in this function)
In function 'main': */
chevron_right
filter_none

```

C++

```

filter_none
edit
close
play_arrow
link
brightness_4
code

// CPP program to initialize data member in c++
#include <iostream>
using namespace std;

struct Student {
    int roll;
    Student(int x)
    {
        roll = x;
    }
};

// Driver Program
int main()
{
    struct Student s(2);
    cout << s.roll;
    return 0;
}

// Output
// 2
chevron_right
filter_none

```

Output:

2

6. **sizeof operator:** This operator will generate **0** for an empty structure in C whereas **1** for an empty structure in C++.

```

filter_none
edit
close
play_arrow
link
brightness_4
code

// C program to illustrate empty structure
#include <stdio.h>

// empty structure
struct Record {
};

// Driver program
int main()

```

```
{  
    struct Record s;  
    printf("%d\n", sizeof(s));  
    return 0;  
}
```

chevron_right

filter_none

Output in C:

0

Output in C++:

1

7. **Data Hiding:** C structures do not allow concept of Data hiding but is permitted in C++ as C++ is an object oriented language whereas C is not.

8. **Access Modifiers:** C structures do not have access modifiers as these modifiers are not supported by the language. C++ structures can have this concept as it is inbuilt in the language.

Related Article: Structure vs Class in C++

This article is contributed by **Shubham Chaudhary**. If you like GeeksforGeeks and would like to contribute, you can also write an article using [contribute.geeksforgeeks.org](#) or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes *arrow_drop_up*

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- Difference between Stack and Queue Data Structures
- Difference between Linear and Non-linear Data Structures
- Structures in C++
- Structures in C
- Policy based data structures in g++
- Slack Bytes in Structures : Explained with Example
- C program to store Student records as Structures and Sort them by Name
- C/C++ program to add N distances given in inch-feet system using Structures
- Difference between C and C#
- Difference between Tor and VPN
- Difference between USB 2.0 and USB 3.0
- Difference between CPU and GPU
- Difference between TCP and RTP
- Difference between PNG and GIF
- Difference between LAN and WAN

Improved By : [gyanendra371](#)

Article Tags :

C
C++
Difference Between
cpp-structure
Practice Tags :
C
CPP

thumb_up
32

To-do Done
2.2

Based on 31 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) Error Handling in C programs

Next

[last_page](#) Named Pipe or FIFO with example C program

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

[Load Comments](#)



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string ctype.h<ctype.h> library in C/C++ with Examples
Implicit Type Conversion in C with Examples
How to call function within function in C or C++
Format specifiers in different Programming Languages

Most visited in C++
Vector of Vectors in C++ STL with Examples
C++ STL with Examples
Which C++ libraries are useful for competitive programming?
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

Anonymous Union and Structure in C

In C11 standard of C, anonymous Unions and structures were added.

Anonymous unions/structures are also known as unnamed unions/structures as they don't have names. Since there is no name, direct objects(variables) of them are not created and we use them in nested structure or unions.

Definition is just like that of a normal union just without a name or tag. For example,

```
// Anonymous union example
union
{
    char alpha;
    int num;
};

// Anonymous structure example
struct
{
    char alpha;
    int num;
};
```

Since there is no variable and no name, we can directly access members. This accessibility works only inside the scope where the anonymous union is defined.

Following is a complete working example of anonymous union.

```
filter_none
edit
close

play_arrow
link
brightness_4
code

// C Program to demonstrate working of anonymous union
#include<stdio.h>
struct Scope
{
    // Anonymous union
    union
    {
        char alpha;
        int num;
    };
};

int main()
{
    struct Scope x;
    x.num = 65;

    // Note that members of union are accessed directly
    printf("x.alpha = %c, x.num = %d", x.alpha, x.num);

    return 0;
}
```

Output:

x.alpha = A, x.num = 65

filter_none

```

edit
close
play_arrow
link
brightness_4
code

// C Program to demonstrate working of anonymous struct
#include<stdio.h>
struct Scope
{
    // Anonymous structure
    struct
    {
        char alpha;
        int num;
    };
};

int main()
{
    struct Scope x;
    x.num = 65;
    x.alpha = 'B';

    // Note that members of structure are accessed directly
    printf("x.alpha = %c, x.num = %d", x.alpha, x.num);

    return 0;
}
chevron_right
filter_none

```

Output:

```
x.alpha = B, x.num = 65
```

What about C++?

Anonymous Unions and Structures are NOT part of C++ 11 standard, but most of the C++ compilers support them. Since this is a C only feature, the C++ implementations don't allow to anonymous struct/union to have private or protected members, static members, and functions.

This article is contributed by **Nikita Raj**. If you like GeeksforGeeks and would like to contribute, you can also write an article and mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up

Add your personal notes
here! (max 5000 chars)

[Save](#)

Recommended Posts:

- [C | Structure & Union | Question 1](#)
- [C | Structure & Union | Question 2](#)
- [C | Structure & Union | Question 10](#)
- [C | Structure & Union | Question 8](#)
- [C | Structure & Union | Question 7](#)
- [C | Structure & Union | Question 6](#)
- [C | Structure & Union | Question 5](#)
- [C | Structure & Union | Question 10](#)
- [C | Structure & Union | Question 9](#)
- [C | Structure & Union | Question 4](#)
- [Difference between Structure and Union in C](#)
- [Output of C programs | Set 44 \(Structure & Union\)](#)
- [Output of C++ programs | Set 41 \(Structure and Union\)](#)
- [Union in C](#)
- [How to avoid Structure Padding in C?](#)

Article Tags :

[C](#)
[C-Structure & Union](#)
Practice Tags :
[C](#)

[thumb_up](#)
8

To-do Done
2.4

Based on 5 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.
Post navigation

Previous [first_page](#) [nextafter\(\)](#) and [nexttoward\(\)](#) in C/C++

Next [last_page](#)

[Floating Point Operations & Associativity in C, C++ and Java](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

[Load Comments](#)

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Weekend Geeks Classes Weekdays

JAVA APP DEVELOPMENT COMPETITIVE PROGRAMMING LITE Java Backend Development

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
Format specifiers in different Programming Languages
C program to find square root of a given number
Predefined Macros in C with Examples
C program to print odd line contents of a File followed by even line content

More related articles in C
Introduction to the C99 Programming Language : Part II
Features of Programming language
Problems in comparing Floating point numbers and how to compare them correctly?
<cfloat> float.h in C/C++ with Examples
Getting System and Process Information Using C Programming and Shell in Linux

Compound Literals in C

Consider below program in C.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// Please make sure that you compile this program
// using a C compiler, not a C++ compiler (Save your
// file .cpp). If using online compiler, select "C"
#include <stdio.h>
int main()
{
    // Compound literal (an array is created without
    // any name and address of first element is assigned
    // to p. This is equivalent to:
    // int arr[] = {2, 4, 6};
    // int *p = arr;
    int *p = (int []){2, 4, 6};

    printf("%d %d %d", p[0], p[1], p[2]);

    return 0;
}
chevron_right
filter_none
```

Output:
2 4 6

The above example is an example of compound literals. Compound literals were introduced in [C99 standard of C](#). Compound literals feature allows us to create unnamed objects with given list of initialized values. In the above example, an array is created without any name. Address of first element of array is assigned to pointer p.

What is the use of it?

Compound literals are mainly used with structures and are particularly useful when passing structures variables to functions. We can pass a structure object without defining it. For example, consider the below code.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// Please make sure that you compile this program
// using a C compiler, not a C++ compiler (Save your
// file .cpp). If using online compiler, select "C"
#include <stdio.h>

// Structure to represent a 2D point
struct Point
{
    int x, y;
};
```

```

// Utility function to print point
void printPoint(struct Point p)
{
    printf("%d, %d", p.x, p.y);
}

int main()
{
    // Calling printPoint() without creating any temporary
    // Point variable in main()
    printPoint((struct Point){2, 3});

    /* Without compound literal, above statement would have
       been written as
       struct Point temp = {2, 3};
       printPoint(temp); */

    return 0;
}

```

[chevron_right](#)

[filter_none](#)

Output:

2, 3

This article is contributed by **Shivam Gupta**. If you like GeeksforGeeks and would like to contribute, you can also write an article and mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes [arrow_drop_up](#)

Add your personal notes here! (max 5000 chars)

[Save](#)

Recommended Posts:

- Octal literals in C
- Types of Literals in C/C++ with Examples
- Format specifiers in different Programming Languages
- C program to find square root of a given number
- C program to print odd line contents of a File followed by even line content
- Getting System and Process Information Using C Programming and Shell in Linux
- Program to calculate Electricity Bill
- Program to print half Diamond star pattern
- C/C++ program for calling main() in main()
- Print all possible combinations of the string by replacing '\$' with any other digit from the string
- How to use make utility to build C projects?
- How to call function within function in C or C++
- Role of SemiColon in various Programming Languages
- C program to display month by month calendar for a given year

Article Tags :

C
C-Data Types
Practice Tags :
C

[thumb_up](#)
4

To-do Done
2.7

Based on 4 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) Extra brackets with function names in C/C++

Next

[last_page](#) nextafter() and nexttoward() in C/C++

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

[Load Comments](#)



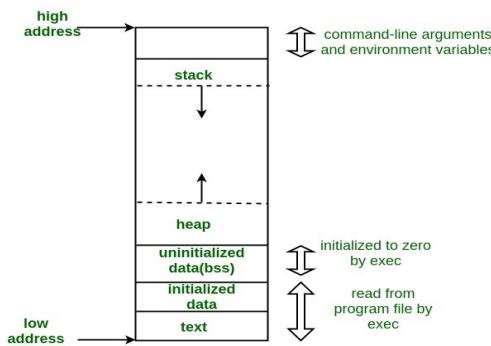
Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
Predefined Macros in C with Examples
Format specifiers in different Programming Languages
C program to print odd line contents of a file followed by even line content
Introduction to the C99 Programming Language : Part II

More related articles in C
C program to find square root of a given number
Problems of Comparing Floating Point numbers and how to compare them correctly?
Features of C Programming Language
Pointer Expressions in C with Examples
Introduction to the C99 Programming Language : Part III

Memory Layout of C Programs

A typical memory representation of C program consists of following sections.

1. Text segment
2. Initialized data segment
3. Uninitialized data segment
4. Stack
5. Heap



A typical memory layout of a running process

1. Text Segment:

A text segment, also known as a code segment or simply as text, is one of the sections of a program in an object file or in memory, which contains executable instructions.

As a memory region, a text segment may be placed below the heap or stack in order to prevent heaps and stack overflows from overwriting it.

Usually, the text segment is sharable so that only a single copy needs to be in memory for frequently executed programs, such as text editors, the C compiler, the shells, and so on. Also, the text segment is often read-only, to prevent a program from accidentally modifying its instructions.

2. Initialized Data Segment:

Initialized data segment, usually called simply the Data Segment. A data segment is a portion of virtual address space of a program, which contains the global variables and static variables that are initialized by the programmer.

Note that, data segment is not read-only, since the values of the variables can be altered at run time.

This segment can be further classified into initialized read-only area and initialized read-write area.

For instance the global string defined by `char s[] = "hello world"` in C and a C statement like `int debug=1` outside the main (i.e. global) would be stored in initialized read-write area. And a global C statement like `const char* string = "hello world"` makes the string literal "hello world" to be stored in initialized read-only area and the character pointer variable `string` in initialized read-write area.

Ex: `static int i = 10` will be stored in data segment and `global int i = 10` will also be stored in data segment

3. Uninitialized Data Segment:

Uninitialized data segment, often called the "bss" segment, named after an ancient assembler operator that stood for "block started by symbol." Data in this segment is initialized by the kernel to arithmetic 0 before the program starts executing

uninitialized data starts at the end of the data segment and contains all global variables and static variables that are initialized to zero or do not have explicit initialization in source code.

For instance a variable declared `static int i;` would be contained in the BSS segment.

For instance a global variable declared `int j;` would be contained in the BSS segment.

4. Stack:

The stack area traditionally adjoined the heap area and grew the opposite direction; when the stack pointer met the heap pointer, free memory was exhausted. (With modern large address spaces and virtual memory techniques they may be placed almost anywhere, but they still typically grow opposite directions.)

The stack area contains the program stack, a LIFO structure, typically located in the higher parts of memory. On the standard PC x86 computer architecture it grows toward address zero; on some other architectures it grows the opposite direction. A "stack pointer" register tracks the top of the stack; it is adjusted each time a value is "pushed" onto the stack. The set of values pushed for one function call is termed a "stack frame"; A stack frame consists at minimum of a return address.

Stack, where automatic variables are stored, along with information that is saved each time a function is called. Each time a function is called, the address of where to return to and certain information about the caller's environment, such as some of the machine registers, are saved on the stack. The newly called function then allocates room on the stack for its automatic and temporary variables. This is how recursive functions in C can work. Each time a recursive function calls itself, a new stack frame is used, so one set of variables doesn't interfere with the variables from another instance of the function.

5. Heap:

Heap is the segment where dynamic memory allocation usually takes place.

The heap area begins at the end of the BSS segment and grows to larger addresses from there. The Heap area is managed by malloc, realloc, and free, which may use the brk and sbrk system calls to adjust its size (note that the use of brk/sbrk and a single "heap area" is not required to fulfill the contract of malloc/realloc/free; they may also be implemented using mmap to reserve potentially non-contiguous regions of virtual memory into the process' virtual address space). The Heap area is shared by all shared libraries and dynamically loaded modules in a process.

Examples.

The size(1) command reports the sizes (in bytes) of the text, data, and bss segments. (for more details please refer man page of size(1))

1. Check the following simple C program

```
filter_none
edit
close

play_arrow

link
brightness_4
code

#include <stdio.h>
```

```
int main(void)
{
    return 0;
}

chevron_right
filter_none
```

```
[narendra@CentOS]$ gcc memory-layout.c -o memory-layout
[narendra@CentOS]$ size memory-layout
text      data      bss      dec      hex   filename
960       248        8     1216     4C0   memory-layout
```

2. Let us add one global variable in program, now check the size of bss (highlighted in red color).

```
filter_none
edit
close

play_arrow

link
brightness_4
code

#include <stdio.h>
```

```
int global; /* Uninitialized variable stored in bss*/

int main(void)
{
    return 0;
}

chevron_right
filter_none
```

```
[narendra@CentOS]$ gcc memory-layout.c -o memory-layout
[narendra@CentOS]$ size memory-layout
text      data      bss      dec      hex   filename
960       248       12     1220     4C4   memory-layout
```

3. Let us add one static variable which is also stored in bss.

```
filter_none
edit
close

play_arrow

link
brightness_4
code

#include <stdio.h>
```

```
int global; /* Uninitialized variable stored in bss*/

int main(void)
{
    static int i; /* Uninitialized static variable stored in bss */
    return 0;
}

chevron_right
filter_none
```

```
[narendra@CentOS]$ gcc memory-layout.c -o memory-layout
[narendra@CentOS]$ size memory-layout
text      data      bss      dec      hex   filename
960       248       16     1224     4C8   memory-layout
```

4. Let us initialize the static variable which will then be stored in Data Segment (DS)

```
filter_none
edit
close

play_arrow

link
brightness_4
code

#include <stdio.h>
```

```
int global; /* Uninitialized variable stored in bss*/

int main(void)
```

```
{  
    static int i = 100; /* Initialized static variable stored in DS*/  
    return 0;  
}
```

chevron_right

filter_none

```
[narendra@CentOS]$ gcc memory-layout.c -o memory-layout  
[narendra@CentOS]$ size memory-layout  
text      data      bss      dec      hex      filename  
960       252        12     1224      4c8      memory-layout
```

5. Let us initialize the global variable which will then be stored in Data Segment (DS)

filter_none
edit
close

play_arrow

link
brightness_4
code

```
#include <stdio.h>
```

```
int global = 10; /* initialized global variable stored in DS*/  
  
int main(void)  
{  
    static int i = 100; /* Initialized static variable stored in DS*/  
    return 0;  
}
```

chevron_right

filter_none

```
[narendra@CentOS]$ gcc memory-layout.c -o memory-layout  
[narendra@CentOS]$ size memory-layout  
text      data      bss      dec      hex      filename  
960       256        8      1224      4c8      memory-layout
```

This article is compiled by **Narendra Kangalkar**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Source:

http://en.wikipedia.org/wiki/Data_segment
http://en.wikipedia.org/wiki/Code_segment
<http://en.wikipedia.org/wiki/bss>
<http://www.amazon.com/Advanced-Programming-UNIX-Environment-2nd/dp/0201433079>

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes *arrow_drop_up*

Add your personal notes
here! (max 5000 chars)

Recommended Posts:

- [Output of C programs | Set 66 \(Accessing Memory Locations\)](#)
- [Common Memory/Pointer Related bug in C Programs](#)
- [C/C++ Tricky Programs](#)
- [Output of C programs | Set 63](#)
- [Error Handling in C programs](#)
- [Programs to print Interesting Patterns](#)
- [Output of C programs | Set 30 \(Switch Case\)](#)
- [How to Compile and Run C/C++/Java Programs in Linux](#)
- [8085 programs to find 2's compliment with carry | Set 2](#)
- [Facts and Question related to Style of writing programs in C/C++](#)
- [IPC through shared memory](#)
- [How to deallocate memory without using free\(\) in C?](#)
- [Memory leak in C++ and How to avoid it?](#)
- [What is Memory Leak? How can we avoid?](#)
- [MCQ on Memory allocation and compilation process](#)

Article Tags :

C
C-Dynamic Memory Allocation
system-programming

Practice Tags :

C

thumb_up
115

To-do Done
2.6

Based on 176 vote(s)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

first_page Hiding of all overloaded methods with same name in base class

Next

last_page Templates and Default Arguments

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.



Most popular in C
Program to calculate Electricity Bill
Program to print half Diamond Star pattern
C/C++ program for calling main() in main()
C/C++ #include directive with Examples
Output of C programs | Set 66 (Accessing Memory Locations)

More related articles in C
Difference between Type Casting and Type Conversion
C/C++ Headers with Examples
Print all possible combinations of the string by replacing 's' with any other digit from the string
getch() function in C with Examples
Jagged Array or Array of Arrays in C with Examples

How to deallocate memory without using free() in C?

Question: How to deallocate dynamically allocated memory without using "free()" function.

Solution: Standard library function `realloc()` can be used to deallocate previously allocated memory. Below is function declaration of "realloc()" from "stdlib.h"

```
filter_none
edit
close

play_arrow

link
brightness_4
code

void *realloc(void *ptr, size_t size);
chevron_right
filter_none
```

If "size" is zero, then call to `realloc` is equivalent to "`free(ptr)`". And if "ptr" is NULL and size is non-zero then call to `realloc` is equivalent to "`malloc(size)`".

Let us check with simple example.

```
filter_none
edit
close

play_arrow

link
brightness_4
code

/* code with memory leak */
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
    int *ptr = (int*)malloc(10);

    return 0;
}
chevron_right
filter_none
```

Check the leak summary with valgrind tool. It shows memory leak of 10 bytes, which is highlighted in red colour.

```
[narendra@ubuntu]$ valgrind --leak-check=full ./free
==1238== LEAK SUMMARY:
==1238==   definitely lost: 10 bytes in 1 blocks.
==1238==   possibly lost: 0 bytes in 0 blocks.
==1238==   still reachable: 0 bytes in 0 blocks.
==1238==   suppressed: 0 bytes in 0 blocks.
[narendra@ubuntu]$
```

Let us modify the above code.

```
filter_none
edit
close

play_arrow

link
brightness_4
code

#include <stdio.h>
```

```
#include <stdlib.h>

int main(void)
{
    int *ptr = (int*) malloc(10);

    /* we are calling realloc with size = 0 */
    realloc(ptr, 0);

    return 0;
}
```

chevron_right

filter_none

Check the valgrind's output. It shows no memory leaks are possible, highlighted in red color.

```
[narendra@ubuntu]$ valgrind --leak-check=full ./a.out
==1435== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 11 from 1)
==1435== malloc/free: in use at exit: 0 bytes in 0 blocks.
==1435== malloc/free: 1 allocs, 1 frees, 10 bytes allocated.
==1435== For counts of detected errors, rerun with: -v
==1435== All heap blocks were freed -- no leaks are possible.
[narendra@ubuntu]$
```

This article is compiled by "Narendra Kangarkar" and reviewed by GeeksforGeeks team. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes *arrow_drop_up*

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- [How does free\(\) know the size of memory to be deallocated?](#)
- [Dynamic Memory Allocation in C using malloc\(\), calloc\(\), free\(\) and realloc\(\)](#)
- [delete and free\(\) in C++](#)
- [IPC through shared memory](#)
- [Memory leak in C++ and How to avoid it?](#)
- [What is Memory Leak? How can we avoid?](#)
- [Memory Layout of C Programs](#)
- [MCQ on Memory allocation and compilation process](#)
- [C | Dynamic Memory Allocation | Question 5](#)
- [C | Dynamic Memory Allocation | Question 2](#)
- [C | Dynamic Memory Allocation | Question 1](#)
- [C | Dynamic Memory Allocation | Question 6](#)
- [C | Dynamic Memory Allocation | Question 7](#)
- [C | Dynamic Memory Allocation | Question 8](#)
- [C | Dynamic Memory Allocation | Question 8](#)

Article Tags :

C
C-Dynamic Memory Allocation
Practice Tags :
C

thumb_up
10

To-do Done
2.5

Based on 29 vote(s)

Basic **Easy** **Medium** **Hard** **Expert**

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) [Exception handling and object destruction | Set 1](#)

Next

[last_page](#) [Catch block and type conversion in C++](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments



Most popular in C
Program to calculate Electricity Bill
Program to print half Diamond Star pattern
C/C++ program for calling main() in main()
C/C++ #include directive with Examples
Output of C programs | Set 66 (Accessing Memory Locations)

More related articles in C
Difference between Type Casting and Type Conversion
C/C++ Headers with Examples
Print all possible combinations of the string by replacing 's' with any other digit from the string
getch() function in C with Examples
Jagged Array or Array of Arrays in C with Examples

Difference Between malloc() and calloc() with Examples

Pre-requisite: Dynamic Memory Allocation in C using malloc(), calloc(), free() and realloc()

The name **malloc** and **calloc**) are library functions that allocate memory dynamically. It means that memory is allocated during runtime(execution of the program) from the heap segment.

- **Initialization:** malloc() allocates memory block of given size (in bytes) and returns a pointer to the beginning of the block. malloc() doesn't initialize the allocated memory. If we try to access the content of memory block(before initializing) then we'll get segmentation fault error(or maybe garbage values).

filter_none

edit

close

play_arrow

link

brightness_4

code

```
void* malloc(size_t size);
chevron_right
```

filter_none

calloc() allocates the memory and also initializes the allocated memory block to zero. If we try to access the content of these blocks then we'll get 0.

filter_none

edit

close

play_arrow

link

brightness_4

code

```
void* calloc(size_t num, size_t size);
chevron_right
```

filter_none

- **Number of arguments:** Unlike malloc(), calloc() takes two arguments:

- 1) Number of blocks to be allocated.
- 2) Size of each block.

- **Return Value:** After successful allocation in malloc() and calloc(), a pointer to the block of memory is returned otherwise **NULL** value is returned which indicates the failure of allocation.

For instance, If we want to allocate memory for array of 5 integers, see the following program:-

filter_none

edit

close

play_arrow

link

brightness_4

code

```
// C program to demonstrate the use of calloc()
```

```
// and malloc()
```

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
int main()
```

```
{
```

```
    int* arr;
```

```
    // malloc() allocate the memory for 5 integers
```

```
    // containing garbage values
```

```
    arr = (int*)malloc(5 * sizeof(int)); // 5*4bytes = 20 bytes
```

```

// Deallocates memory previously allocated by malloc() function
free(arr);

// calloc() allocate the memory for 5 integers and
// set 0 to all of them
arr = (int*)calloc(5, sizeof(int));

// Deallocates memory previously allocated by calloc() function
free(arr);

return (0);
}
chevron_right
filter_none

```

We can achieve same functionality as calloc() by using malloc() followed by memset().

```

filter_none
edit
close
play_arrow
link
brightness_4
code

ptr = malloc(size);
memset(ptr, 0, size);
chevron_right
filter_none

```

Note: It would be better to use malloc over calloc, unless we want the zero-initialization because malloc is faster than calloc. So if we just want to copy some stuff or do something that doesn't require filling of the blocks with zeros, then malloc would be a better choice.

This article is contributed by [Shubham Bansal](#). If you like GeeksforGeeks and would like to contribute, you can also write an article using [contribute.geeksforgeeks.org](#) or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up

Add your personal notes
here! (max 5000 chars)

Save

Recommended Posts:

- [Dynamic Memory Allocation in C using malloc\(\), calloc\(\), free\(\) and realloc\(\)](#)
- [malloc\(\) vs new](#)
- [Function Interposition in C with an example of user defined malloc\(\)](#)
- [Difference between Iterators and Pointers in C/C++ with Examples](#)
- [Difference between Argument and Parameter in C/C++ with Examples](#)
- [Difference between C and C++](#)
- [Difference between ++p, *p++ and ++p](#)
- [Difference between C and C#](#)
- [Difference between C and Python](#)
- [Difference between while\(1\) and while\(0\) in C language](#)
- [Difference between scanf\(\) and gets\(\) in C](#)
- [Difference between Structure and Array in C](#)
- [Difference between float and double in C/C++](#)
- [Difference between Java and C language](#)
- [Difference between while and do-while loop in C, C++, Java](#)

Improved By : [chakradharkandula](#), [shubham_singh](#), [Navfal](#), [HarshitHiremath](#)

Article Tags :

C

C-Dynamic Memory Allocation

Practice Tags :

C

thumb_up

32

To-do Done
1.5

Based on 80 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) What is return type of getchar(), fgetc() and getc() ?

Next

[last_page](#) Storage for Strings in C

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments



Most popular in C
Program to calculate Electricity Bill
Program to print half Diamond star pattern
C/C++ program for calling main() in main()
C/C++ #include directive with Examples
Output of C programs | Set 66 (Accessing Memory Locations)

More related articles in C
getch() function in C with Examples
Difference between Type Casting and Type Conversion
Chain of Pointers in C with Examples
Print all possible combinations of the string by replacing '\$' with any other digit from the string
ctype.h(<cctype>) library in C/C++ with Examples

How does free() know the size of memory to be deallocated?

Consider the following prototype of `free()` function which is used to free memory allocated using `malloc()` or `calloc()` or `realloc()`.

```
filter_none
edit
close
play_arrow
link
brightness_4
code
void free(void *ptr);
chevron_right
filter_none
```

Note that the `free` function does not accept size as a parameter. How does `free()` function know how much memory to free given just a pointer?

Following is the most common way to store size of memory so that `free()` knows the size of memory to be deallocated.

When memory allocation is done, the actual heap space allocated is one word larger than the requested memory. The extra word is used to store the size of the allocation and is later used by `free()`

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

References:

<http://www.cs.cmu.edu/afs/cs/academic/class/15213-f10/www/lectures/17-allocation-basic.pptx>
<http://en.wikipedia.org/wiki/Malloc>

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes [arrow_drop_up](#)

Add your personal notes
here! (max 5000 chars)

[Save](#)

Recommended Posts:

- [How to deallocate memory without using free\(\) in C?](#)
- [Dynamic Memory Allocation in C using malloc\(\), calloc\(\), free\(\) and realloc\(\)](#)
- [delete and free\(\) in C++](#)
- [Get the stack size and set the stack size of thread attribute in C](#)
- [IPC through shared memory](#)
- [Memory leak in C++ and How to avoid it?](#)
- [Memory Layout of C Programs](#)
- [What is Memory Leak? How can we avoid?](#)
- [MCQ on Memory allocation and compilation process](#)
- [C | Dynamic Memory Allocation | Question 2](#)
- [C | Dynamic Memory Allocation | Question 3](#)
- [C | Dynamic Memory Allocation | Question 8](#)
- [C | Dynamic Memory Allocation | Question 5](#)
- [C | Dynamic Memory Allocation | Question 6](#)
- [C | Dynamic Memory Allocation | Question 7](#)

Article Tags :

C
C-Dynamic Memory Allocation
Practice Tags :

[thumb_up](#)

16

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

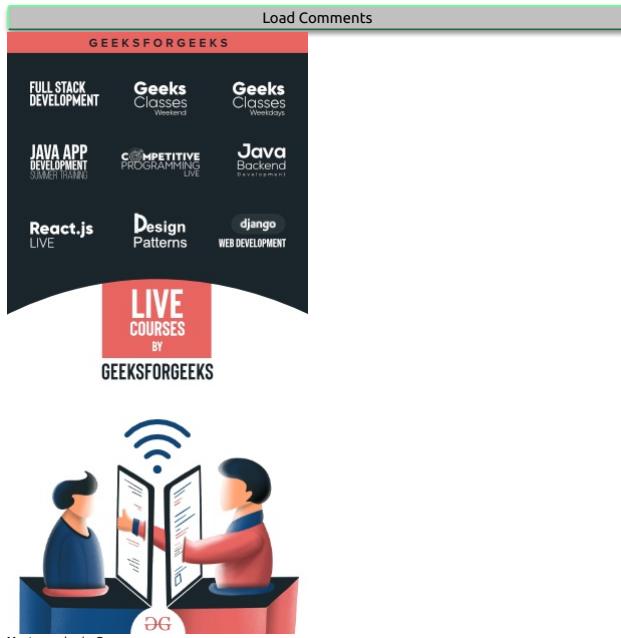
Previous

[first_page](#) [fseek\(\) vs rewind\(\)](#) in C

Next

[last_page](#) Multidimensional Pointer Arithmetic in C/C++

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C
[Program to calculate Electricity Bill](#)
[Program to print half Diamond star pattern](#)
[C program for calling main\(\) in main\(\)](#)
[C/C++ #include directive with Examples](#)
[Output of C programs | Set 66 \(Accessing Memory Locations\)](#)

More related articles in C
[getch\(\) function in C with Examples](#)
[Difference between Type Casting and Type Conversion](#)
[Chain of Pointers in C with Examples](#)
[Print all possible combinations of the string by replacing 's' with any other digit from the string](#)
[ctype.h<<ctype>> library in C/C++ with Examples](#)

Use of realloc()

Size of dynamically allocated memory can be changed by using realloc().

As per the C99 standard:

```
filter_none
edit
close

play_arrow

link
brightness_4
code

void *realloc(void *ptr, size_t size);
chevron_right
filter_none
```

realloc deallocates the old object pointed to by ptr and returns a pointer to a new object that has the size specified by size. The contents of the new object is identical to that of the old object prior to deallocation, up to the lesser of the new and old sizes. Any bytes in the new object beyond the size of the old object have indeterminate values.

The point to note is that **realloc()** should only be used for dynamically allocated memory. If the memory is not dynamically allocated, then behavior is undefined. For example, program 1 demonstrates incorrect use of realloc() and program 2 demonstrates correct use of realloc().

Program 1:

```
filter_none
edit
close

play_arrow

link
brightness_4
code

#include <stdio.h>
#include <stdlib.h>
int main()
{
    int arr[2], i;
    int *ptr = arr;
    int *ptr_new;

    arr[0] = 10;
    arr[1] = 20;

    // incorrect use of new_ptr: undefined behaviour
    ptr_new = (int *)realloc(ptr, sizeof(int)*3);
```

```

*(ptr_new + 2) = 30;

for(i = 0; i < 3; i++)
    printf("%d ", *(ptr_new + i));

getchar();
return 0;
}
chevron_right
filter_none

```

Output:
Undefined Behavior

Program 2:

```

filter_none
edit
close
play_arrow
link
brightness_4
code

#include <stdio.h>
#include <stdlib.h>
int main()
{
    int *ptr = (int *)malloc(sizeof(int)*2);
    int i;
    int *ptr_new;

    *ptr = 10;
    *(ptr + 1) = 20;

    ptr_new = (int *)realloc(ptr, sizeof(int)*3);
    *(ptr_new + 2) = 30;
    for(i = 0; i < 3; i++)
        printf("%d ", *(ptr_new + i));

    getchar();
    return 0;
}
chevron_right
filter_none

```

Output:
10 20 30

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up

[Save](#)

Recommended Posts:

- Dynamic Memory Allocation in C using malloc(), calloc(), free() and realloc()
- Format specifiers in different Programming Languages
- C program to find square root of a given number
- C program to print odd line contents of a File followed by even line content
- Getting System and Process Information Using C Programming and Shell in Linux
- Program to calculate Electricity Bill
- Program to print half Diamond star pattern
- C/C++ program for calling main() in main()
- Print all possible combinations of the string by replacing '\$' with any other digit from the string
- How to use make utility to build C projects?
- How to call function within function in C or C++
- Role of Semicolon in various Programming Languages
- C program to display month by month calendar for a given year
- Difference between Type Casting and Type Conversion

Article Tags :

C
C-Dynamic Memory Allocation

Practice Tags :

C

[thumb_up](#)

17

To-do Done
2.2

Based on 27 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) How to Count Variable Numbers of Arguments in C?

Next

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Weekend Geeks Classes Weekdays

JAVA APP DEVELOPMENT COMPETITIVE PROGRAMMING DAY Java Backend

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS



Most popular in C
Program to calculate Electricity Bill
Program to print half Diamond star pattern
C/C++ program for calling main() in main()
C/C++ #include directive with Examples
Output of C programs | Set 66 (Accessing Memory Locations)

More related articles in C
getchar() Function with Examples
Difference between Type Casting and Type Conversion
Chain of Pointers in C with Examples
Print all possible combinations of the string by replacing '\$' with any other digit from the string
ctype.h(<cctype>) library in C/C++ with Examples

What is Memory Leak? How can we avoid?

Memory leak occurs when programmers create a memory in heap and forget to delete it.

Memory leaks are particularly serious issues for programs like daemons and servers which by definition never terminate.

filter_none
edit
close

play_arrow

link
brightness_4
code

/* Function with memory leak */
#include <stdlib.h>

void f()
{
 int *ptr = (int *) malloc(sizeof(int));

 /* Do some work */

 return; /* Return without freeing ptr*/
}

To avoid memory leaks, memory allocated on heap should always be freed when no longer needed.

filter_none
edit
close

play_arrow

link
brightness_4
code

/* Function without memory leak */
#include <stdlib.h>;

void f()
{
 int *ptr = (int *) malloc(sizeof(int));

 /* Do some work */

 free(ptr);
 return;
}

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes [arrow_drop_up](#)

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- Memory leak in C++ and How to avoid it?
- What are Wild Pointers? How can we avoid?
- How to avoid Structure Padding in C?
- IPC through shared memory
- POSIX shared-memory API
- How to deallocate memory without using free() in C?
- Memory Layout of C Programs
- C | Dynamic Memory Allocation | Question 5
- C | Dynamic Memory Allocation | Question 3
- C | Dynamic Memory Allocation | Question 1
- C | Dynamic Memory Allocation | Question 8
- C | Dynamic Memory Allocation | Question 2
- How will you show memory representation of C variables?
- How does free() know the size of memory to be deallocated?
- MCQ on Memory allocation and compilation process

Article Tags :

Articles

C

C-Dynamic Memory Allocation

Practice Tags :

C

thumb_up

10

To-do Done

1.5

Based on 52 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) Pointer vs Array in C

Next

[last_page](#) Operands for sizeof operator

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Westernd Geeks Classes Weekdays

JAVA APP DEVELOPMENT COMPETITIVE PROGRAMMING LIVE Java Backend Development

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS



Most popular in Articles
new vs malloc() and free() vs delete in C++
How to change default icon of Android App
Difference Between sum of degrees of odd and even degree nodes in an Undirected Graph
When to use each sorting algorithms
Group all co-prime numbers from 1 to N

Most visited in C
P(n) in C++ with Examples
Speed up Code executions with help of Pragma in C/C++
Program to calculate Electricity Bill
Modulo Operator (%) In C/C++ with Examples
Role of SemiColon in various Programming Languages

fseek() vs rewind() in C

In C, fseek() should be preferred over rewind().

Note the following text C99 standard:

The `rewind` function sets the file position indicator for the stream pointed to by `stream` to the beginning of the file. It is equivalent to

filter_none
edit
close

play_arrow

link
brightness_4
code

```
(void)fseek(stream, 0L, SEEK_SET)
```

```
chevron_right
```

```
filter_none
```

except that the error indicator for the stream is also cleared.

This following code example sets the file position indicator of an input stream back to the beginning using rewind(). But there is no way to check whether the rewind() was successful.

```
filter_none
```

```
edit
```

```
close
```

```
play_arrow
```

```
link
```

```
brightness_4
```

```
code
```

```
int main()
```

```
{
```

```
FILE *fp = fopen("test.txt", "r");
```

```
if ( fp == NULL ) {
```

```
/* Handle open error */
```

```
}
```

```
/* Do some processing with file*/
```

```
rewind(fp); /* no way to check if rewind is successful */
```

```
/* Do some more precessing with file */
```

```
return 0;
```

```
}
```

```
chevron_right
```

```
filter_none
```

In the above code, fseek() can be used instead of rewind() to see if the operation succeeded. Following lines of code can be used in place of rewind(fp);

```
filter_none
```

```
edit
```

```
close
```

```
play_arrow
```

```
link
```

```
brightness_4
```

```
code
```

```
if ( fseek(fp, 0L, SEEK_SET) != 0 ) {
```

```
/* Handle repositioning error */
```

```
}
```

```
chevron_right
```

```
filter_none
```

Source:

<https://www.securecoding.cert.org/confluence/display/seccode/FI007-C.+Prefer+fseek%28%29+to+rewind%28%29>

This article is contributed by **Rahul Gupta**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes 

Add your personal notes here! (max 5000 chars)

Recommended Posts:

- [fseek\(\) in C/C++ with example](#)
- [Format specifiers in different Programming Languages](#)
- [C program to find square root of a given number](#)
- [C program to print odd line contents of a File followed by even line content](#)
- [Getting System and Process Information Using C Programming and Shell in Linux](#)
- [Program to calculate Electricity Bill](#)
- [Program to print half Diamond star pattern](#)
- [C/C++ program for calling main\(\) in main\(\)](#)
- [Interesting and Cool Tricks in Java](#)
- [Print all possible combinations of the string by replacing '\\$' with any other digit from the string](#)
- [C program to find and replace a word in a File by another given word](#)
- [How to use make utility to build C projects?](#)
- [How to call function within function in C or C++](#)
- [Role of SemiColon in various Programming Languages](#)

Article Tags :

C

GFacts

C-File Handling

secure-coding

Practice Tags :

C



10

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page Advanced C++ | Conversion Operators](#)

Next

[last_page How does free\(\) know the size of memory to be deallocated?](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
Copy粘贴操作库在C/C++与示例
C程序以月显示月历为给定年份
预定义宏在C与示例
隐式类型转换在C与示例

Most visited in GFACTS
有趣和酷炫技巧在Java
有趣的事实关于Perl
如何安装和配置MongoDB在Ubuntu?

EOF, getc() and feof() in C

In C/C++, `getc()` returns EOF when end of file is reached. `getc()` also returns EOF when it fails. So, only comparing the value returned by `getc()` with EOF is not sufficient to check for actual end of file. To solve this problem, C provides `feof()` which returns non-zero value only if end of file has reached, otherwise it returns 0.

For example, consider the following C program to print contents of file test.txt on screen. In the program, returned value of `getc()` is compared with EOF first, then there is another check using `feof()`. By putting this check, we make sure that the program prints "End of file reached" only if end of file is reached. And if `getc()` returns EOF due to any other reason, then the program prints "Something went wrong"

```
filter_none
edit
close

play_arrow
link
brightness_4
code

#include <stdio.h>

int main()
{
    FILE *fp = fopen("test.txt", "r");
    int ch = getc(fp);
    while (ch != EOF)
    {
        /* display contents of file on screen */
        putchar(ch);

        ch = getc(fp);
    }

    if (feof(fp))
        printf("\n End of file reached.");
    else
        printf("\n Something went wrong.");
    fclose(fp);

    getchar();
    return 0;
}

chevron_right
filter_none
```

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes [arrow_drop_up](#)

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- What is return type of getchar(), fgetc() and getc() ?
- Difference between getc(), getchar(), getch() and getche()
- Format specifiers in different Programming Languages
- C program to find square root of a given number
- C program to print odd line contents of a File followed by even line content
- Getting System and Process Information Using C Programming and Shell in Linux
- Program to calculate Electricity Bill
- Program to print half Diamond star pattern
- C/C++ program for calling main() in main()
- Print all possible combinations of the string by replacing '\$' with any other digit from the string
- C program to find and replace a word in a File by another given word
- How to use make utility to build C projects?
- How to call function within function in C or C++
- Role of SemiColon in various Programming Languages

Article Tags :

C
C-File Handling
Practice Tags :

thumb_up
8

To-do Done
2.3

Based on 24 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

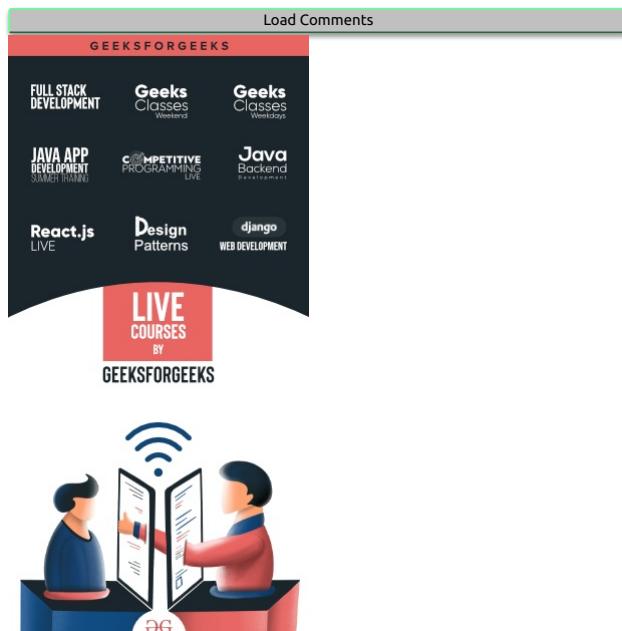
Previous

first_page Order of operands for logical operators

Next

last_page Result of comma operator as l-value in C and C++

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
ctype.h <ctype.h> library in C/C++ with Examples
C program to display month by month calendar for a given year
Predefined Macros in C with Examples
Implicit Type Conversion in C with Examples

More related articles in C
How to call function within function in C or C++
C program to print odd line contents of a File followed by even line content
Introduction to C Programming Language - Part II
How to create GUI in C programming using GTK Toolkit
C program to find square root of a given number

fopen() for an existing file in write mode

In C, `fopen()` is used to open a file in different modes. To open a file in write mode, "w" is specified. When mode "w" is specified, it creates an empty file for output operations.

What if the file already exists?

If a file with the same name already exists, its contents are discarded and the file is treated as a new empty file. For example, in the following program, if "test.txt" already exists, its contents are removed and "GeeksforGeeks" is written to it.

filter_none
edit
close
play_arrow
link
brightness_4
code

#include <stdio.h>
#include <stdlib.h>

int main()
{

```

FILE *fp = fopen("test.txt", "w");
if (fp == NULL)
{
    puts("Couldn't open file");
    exit(0);
}
else
{
    fputs("GeeksforGeeks", fp);
    puts("Done");
    fclose(fp);
}
return 0;
}

```

chevron_right

filter_none

The above behavior may lead to unexpected results. If programmer's intention was to create a new file and a file with same name already exists, the existing file's contents are overwritten.

The latest C standard C11 provides a new mode "x" which is exclusive create-and-open mode. Mode "x" can be used with any "w" specifier, like "wx", "wbx". **When x is used with w, fopen() returns NULL if file already exists or could not open.** Following is modified C11 program that doesn't overwrite an existing file.

```

filter_none
edit
close
play_arrow
link
brightness_4
code

#include <stdio.h>
#include <stdlib.h>

```

```

int main()
{
    FILE *fp = fopen("test.txt", "wx");
    if (fp == NULL)
    {
        puts("Couldn't open file or file already exists");
        exit(0);
    }
    else
    {
        fputs("GeeksforGeeks", fp);
        puts("Done");
        fclose(fp);
    }
    return 0;
}

```

chevron_right

filter_none

References:

Do not make assumptions about `fopen()` and file creation
[http://en.wikipedia.org/wiki/C11_\(C_standard_revision\)](http://en.wikipedia.org/wiki/C11_(C_standard_revision))
<http://www.cplusplus.com/reference/cstdio/freopen/>

This article is compiled by **Abhay Rathi**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes *arrow_drop_up*

Save

Recommended Posts:

- [How to write your own header file in C?](#)
- [Read/Write structure to a file in C](#)
- [lseek\(\) in C/C++ to read the alternate nth byte and write it in another file](#)
- [Write a C program that displays contents of a given file like 'more' utility in Linux](#)
- [C fopen\(\) function with Examples](#)
- [C program to copy contents of one file to another file](#)
- [When should we write our own assignment operator in C++?](#)
- [Write your own memcpy\(\) and memmove\(\)](#)
- [When should we write our own copy constructor?](#)
- [Write a C program that won't compile in C++](#)
- [How to write a running C code without main\(\)?](#)
- [Write a C macro PRINT\(x\) which prints x](#)
- [Write a program that produces different results in C and C++](#)
- [C program to write an image in PGM format](#)
- [Write one line functions for strcat\(\) and strcmp\(\)](#)

Article Tags :

C
C-File Handling
cpp-file-handling
Practice Tags :
C

thumb_up

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

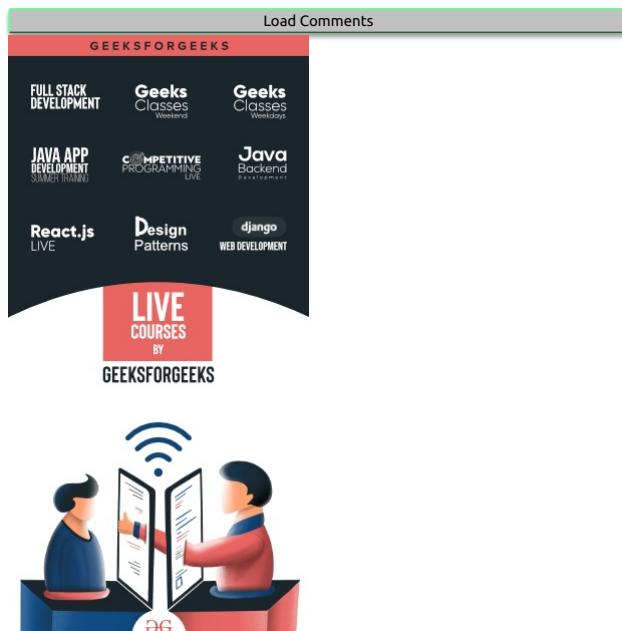
Previous

[first_page](#) C++ | Operator Overloading | Question 2

Next

[last_page](#) C++ | References | Question 4

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Read/Write structure to a file in C

Prerequisite: [Structure in C](#)

For writing in file, it is easy to write string or int to file using **fprintf** and **putc**, but you might have faced difficulty when writing contents of struct. **fwrite** and **fread** make task easier when you want to write and read blocks of data.

1. **fwrite** : Following is the declaration of fwrite function

```
size_t fwrite(const void *ptr, size_t size, size_t nmemb, FILE *stream)
ptr - This is pointer to array of elements to be written
size - This is the size in bytes of each element to be written
nmemb - This is the number of elements, each one with a size of size bytes
stream - This is the pointer to a FILE object that specifies an output stream
```

```
filter_none
edit
close
play_arrow
link
brightness_4
code
```

```
// C program for writing
// struct to file
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

```
// a struct to read and write
```

```
struct person
```

```
{
```

```
    int id;
```

```
    char fname[20];
```

```
    char lname[20];
```

```
};
```

```
int main ()
```

```
{
```

```
    FILE *outfile;
```

```
    // open file for writing
```

```
    outfile = fopen ("person.dat", "w");
```

```
    if (outfile == NULL)
```

```
{
```

```
        fprintf(stderr, "\nError opened file\n");
```

```

    exit (1);
}

struct person input1 = {1, "rohan", "sharma"};
struct person input2 = {2, "mahendra", "dhoni"};

// write struct to file
fwrite (&input1, sizeof(struct person), 1, outfile);
fwrite (&input2, sizeof(struct person), 1, outfile);

if(fwrite != 0)
    printf("contents to file written successfully !\n");
else
    printf("error writing file !\n");

// close file
fclose (outfile);

return 0;
}
chevron_right

```

filter_none

Output:

```

gcc demowrite.c
./a.out
contents to file written successfully!

```

2. **fread** : Following is the declaration of fread function

```

size_t fread(void *ptr, size_t size, size_t nmemb, FILE *stream)
ptr - This is the pointer to a block of memory with a minimum size of size*nmemb bytes.
size - This is the size in bytes of each element to be read.
nmemb - This is the number of elements, each one with a size of size bytes.
stream - This is the pointer to a FILE object that specifies an input stream.

```

filter_none

edit

close

play_arrow

link

brightness_4

code

// C program for reading

// struct from a file

#include <stdio.h>

#include <stdlib.h>

// struct person with 3 fields

struct person

{

int id;

char fname[20];

char lname[20];

};

// Driver program

int main ()

{

FILE *infile;

struct person input;

// Open person.dat for reading

infile = fopen ("person.dat", "r");

if (infile == NULL)

{

fprintf(stderr, "\nError opening file\n");

exit (1);

}

// read file contents till end of file

while(fread(&input, sizeof(struct person), 1, infile))

printf ("id = %d name = %s %s\n", input.id,

input.fname, input.lname);

// close file

fclose (infile);

return 0;

}

chevron_right

filter_none

Output:

```

gcc demoread.c
./a.out
id = 1  name = rohan sharma
id = 2  name = mahendra dhoni

```

My Personal Notes

Add your personal notes here! (max 5000 chars)

Recommended Posts:

- C program to copy contents of one file to another file
- C | Structure & Union | Question 10
- C | Structure & Union | Question 6
- C | Structure & Union | Question 5
- C | Structure & Union | Question 7
- C | Structure & Union | Question 8
- C | Structure & Union | Question 8
- C | Structure & Union | Question 9
- C | Structure & Union | Question 4
- C | Structure & Union | Question 10
- C | Structure & Union | Question 2
- C | Structure & Union | Question 1
- Anonymous Union and Structure in C
- Difference between Structure and Union in C
- Difference between Structure and Array in C
- How to avoid Structure Padding in C?

Improved By : [KathiNotapplicable](#)

Article Tags :

C
C-Structure & Union

Practice Tags :

C



8

To-do Done
2.5

Based on 2 vote(s)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) [Input-output system calls in C | Create, Open, Close, Read, Write](#)

Next

[last_page](#) [pipe\(\) System call](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Weekend Geeks Classes Weekdays

JAVA APP DEVELOPMENT COMPETITIVE PROGRAMMING DN Java Backend

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
ctype.h<ctype> library in C/C++ with Examples
C program to display month by month calendar for a given year
Predefined Macros in C with Examples
Implicit Type Conversion in C with Examples

More related articles in C
How to use function with function in C or C++
C program to read line contents of a file followed by even line content
Introduction to the C99 Programming Language : Part II
How to create GUI in C programming using GTK Toolkit
C program to find square root of a given number

fgets() and gets() in C language

For reading a string value with spaces, we can use either gets() or fgets() in C programming language. Here, we will see what is the difference between gets() and fgets().

fgets()

It reads a line from the specified stream and stores it into the string pointed to by str. It stops when either (n-1) characters are read, the newline character is read, or the end-of-file is reached, whichever comes first.

Syntax :

```
char *fgets(char *str, int n, FILE *stream)
str : Pointer to an array of chars where the string read is copied.
n : Maximum number of characters to be copied into str
(including the terminating null-character).
*stream : Pointer to a FILE object that identifies an input stream.
stdin can be used as argument to read from the standard input.

returns : the function returns str
```

- It follows some parameters such as Maximum length, buffer, input device reference.
- It is **safe** to use because it checks the array bound.
- It keeps on reading until a new line character is encountered or maximum limit of character array.

Example : Let's say the maximum number of characters are 15 and input length is greater than 15 but still fgets() will read only 15 characters and print it.

```
filter_none
edit
close

play_arrow
link
brightness_4
code

// C program to illustrate
// fgets()
#include <stdio.h>
#define MAX 15
int main()
{
    char buf[MAX];
    fgets(buf, MAX, stdin);
    printf("string is: %s\n", buf);

    return 0;
}
```

chevron_right
filter_none

Since fgets() reads input from user, we need to provide input during runtime.

```
Input:
Hello and welcome to GeeksforGeeks

Output:
Hello and welc
```

gets()

Reads characters from the standard input (stdin) and stores them as a C string into str until a newline character or the end-of-file is reached.

Syntax:

```
char * gets ( char * str );
str : Pointer to a block of memory (array of char)
where the string read is copied as a C string.
returns : the function returns str
```

- It is not safe to use because it does not check the array bound.
- It is used to read strings from user until a newline character is not encountered.

Example : Suppose we have a character array of 15 characters and input is greater than 15 characters, gets() will read all these characters and store them into variable. Since, gets() does not check the maximum limit of input characters, so at any time compiler may return buffer overflow error.

```
filter_none
edit
close

play_arrow
link
brightness_4
code

// C program to illustrate
// gets()
#include <stdio.h>
#define MAX 15
```

```
int main()
{
    char buf[MAX];

    printf("Enter a string: ");
    gets(buf);
    printf("string is: %s\n", buf);

    return 0;
}
```

chevron_right
filter_none

Since gets() reads input from user, we need to provide input during runtime.

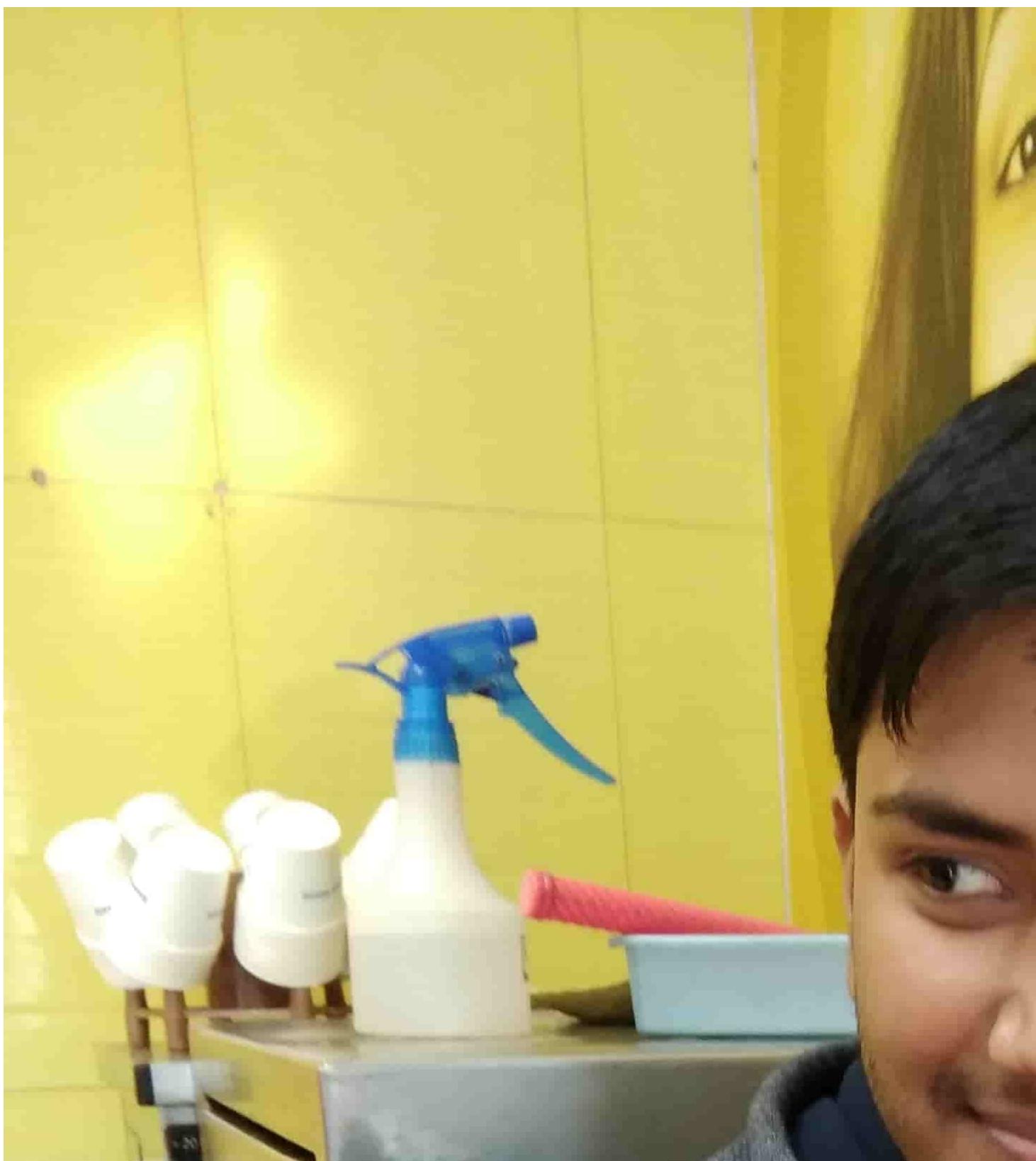
```
Input:
Hello and welcome to GeeksforGeeks

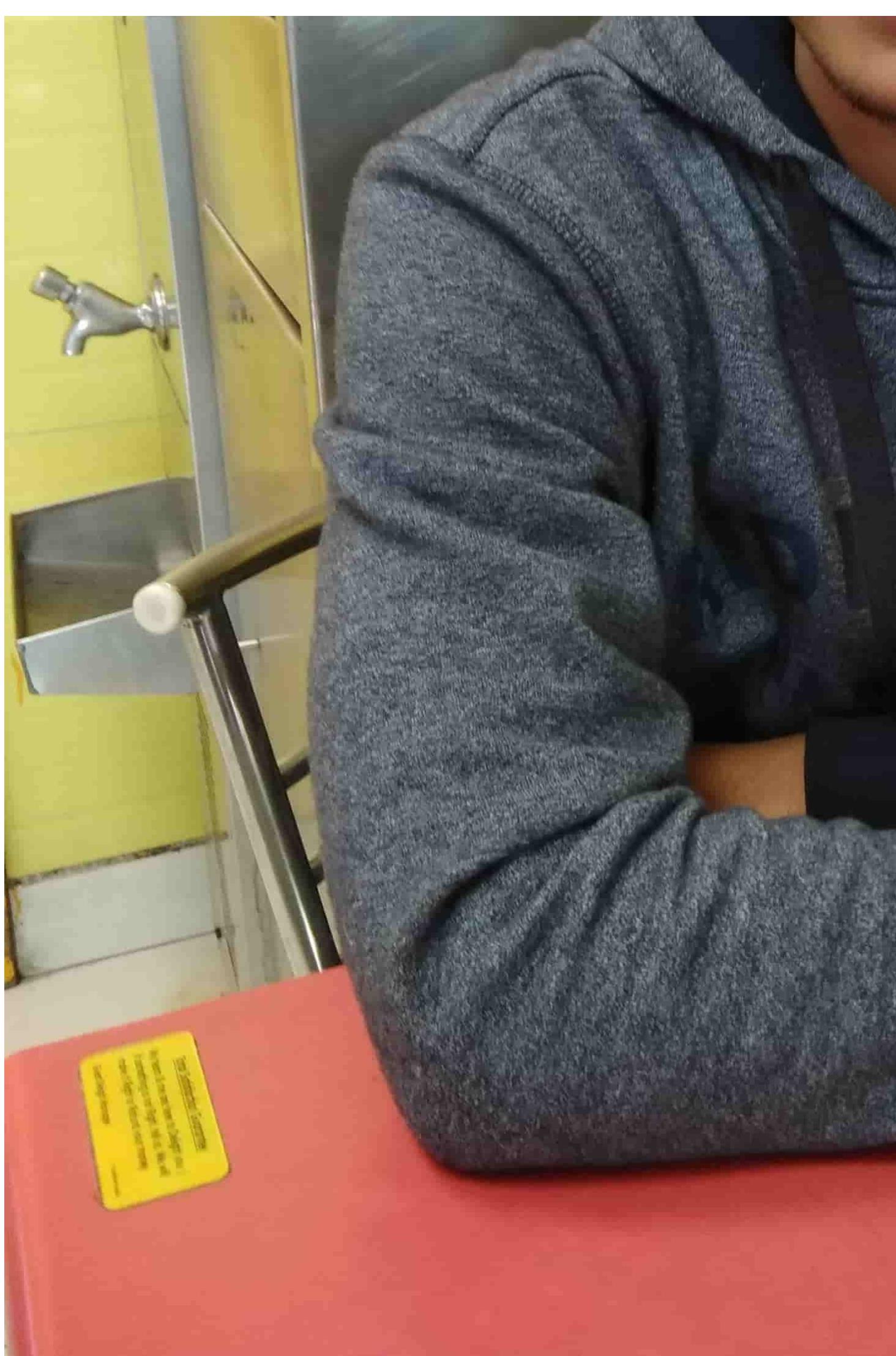
Output:
Hello and welcome to GeeksforGeeks
```

Save

Recommended Posts:

- Why to use fgets() over scanf() in C?
 - Problem with scanf() when there is fgets()/gets()/scanf() after it
 - Difference between while(1) and while(0) in C language
 - kbhit in C language
 - Stopwatch using C language
 - C Language Introduction
 - Signals in C language
 - chdir() in C language with Examples
 - Interesting facts about C Language
 - Difference between Java and C language
 - How to use POSIX semaphores in C language
 - Features of C Programming Language
 - Ivalue and rvalue in C language
 - isalnum() function in C Language
 - isupper() function in C Language
-





Kanchan_Ray

Check out this Author's contributed articles.

If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please Improve this article if you find anything incorrect by clicking on the "Improve Article" button below.

Article Tags :

C

Misc

C-File Handling

Practice Tags :

Misc

Misc

C



20

To-do Done
1.1

Based on 10 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) Count the number of objects using Static member function

Next

[last_page](#) rename function in C/C++

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

[Load Comments](#)

GEEKSFORGEEKS

FULL STACK DEVELOPMENT

Java App Development SUMMER TRAINING

React.js LIVE

Geeks Classes Weekdays

Competitive Programming LIVE

Design Patterns

Geeks Classes Weekdays

Java Backend Development

django WEB DEVELOPMENT



LIVE COURSES BY GEEKSFORGEEKS

Most popular in C

[Print all possible combinations of the string by replacing '\\$' with any other digit from the string](#)

[Program to display month by month calendar for a given year](#)

[Copy & Paste library in C/C++ with Examples](#)

[Implicit Type Conversion in C with Examples](#)

[How to call function within function in C or C++](#)

Most visited in Misc

[Egg dropping puzzle | Set 2](#)

[Introduction of JIRA](#)

[Types of Models in Object Oriented Modeling and Design](#)

[Components of Image Processing System](#)

[Features of Blockchain](#)

Basics of File Handling in C

So far the operations using C program are done on a prompt / terminal which is not stored anywhere. But in the software industry, most of the programs are written to store the information fetched from the program. One such way is to store the fetched information in a file. Different operations that can be performed on a file are:

1. Creation of a new file (**fopen with attributes as "a" or "a+" or "w" or "w++"**)
2. Opening an existing file (**fopen**)
3. Reading from file (**fscanf or fgets**)
4. Writing to a file (**fprintf or fputs**)
5. Moving to a specific location in a file (**fseek, rewind**)
6. Closing a file (**fclose**)

The text in the brackets denotes the functions used for performing those operations.

Functions in File Operations:

File operation	Declaration & Description
fopen() - To open a file	<p>Declaration: FILE *fopen (const char *filename, const char *mode)</p> <p>fopen() function is used to open a file to perform operations such as reading, writing etc. In a C program, we declare a file pointer and use fopen() as below. fopen() function creates a new file if the mentioned file name does not exist.</p> <pre>FILE *fp; fp=fopen ("filename", "mode"); Where, fp - file pointer to the data type "FILE". filename - the actual file name with full path of the file. mode - refers to the operation that will be performed on the file. Example: r, w, a, r+, w+ and a+. Please refer below the description for these mode of operations.</pre>
fclose() - To close a file	<p>Declaration: int fclose(FILE *fp);</p> <p>fclose() function closes the file that is being pointed by file pointer fp. In a C program, we close a file as below.</p> <pre>fclose (fp);</pre>
fgets() - To read a file	<p>Declaration: char *fgets(char *string, int n, FILE *fp)</p> <p>fgets function is used to read a file line by line. In a C program, we use fgets function as below.</p> <pre>fgets (buffer, size, fp); where, buffer - buffer to put the data in. size - size of the buffer fp - file pointer</pre>
fprintf() - To write into a file	<p>Declaration:</p> <pre>int fprintf(FILE *fp, const char *format, ...);</pre> <p>fprintf() function writes string into a file pointed by fp. In a C program, we write string into a file as below.</p> <pre>fprintf (fp, "some data"); or fprintf (fp, "text %d", variable_name);</pre>

Opening or creating file

For opening a file, fopen function is used with the required access modes. Some of the commonly used file access modes are mentioned below.

File opening modes in C:

- “r” – Searches file. If the file is opened successfully fopen() loads it into memory and sets up a pointer which points to the first character in it. If the file cannot be opened fopen() returns NULL.
- “w” – Searches file. If the file exists, its contents are overwritten. If the file doesn’t exist, a new file is created. Returns NULL, if unable to open file.
- “a” – Searches file. If the file is opened successfully fopen() loads it into memory and sets up a pointer that points to the last character in it. If the file doesn’t exist, a new file is created. Returns NULL, if unable to open file.
- “r+” – Searches file. If is opened successfully fopen() loads it into memory and sets up a pointer which points to the first character in it. Returns NULL, if unable to open the file.
- “w+” – Searches file. If the file exists, its contents are overwritten. If the file doesn’t exist a new file is created. Returns NULL, if unable to open file.
- “a+” – Searches file. If the file is opened successfully fopen() loads it into memory and sets up a pointer which points to the last character in it. If the file doesn’t exist, a new file is created. Returns NULL, if unable to open file.

As given above, if you want to perform operations on a binary file, then you have to append ‘b’ at the last. For example, instead of “w”, you have to use “wb”, instead of “a+” you have to use “a+b”. For performing the operations on the file, a special pointer called File pointer is used which is declared as

```
FILE *filePointer;
So, the file can be opened as
filePointer = fopen("fileName.txt", "w")
```

The second parameter can be changed to contain all the attributes listed in the above table.

▪ Reading from a file -

The file read operations can be performed using functions fscanf or fgets. Both the functions performed the same operations as that of scanf and gets but with an additional parameter, the file pointer. So, it depends on you if you want to read the file line by line or character by character.

And the code snippet for reading a file is as:

```
FILE * filePointer;
filePointer = fopen("fileName.txt", "r");
fscanf(filePointer, "%s %s %d", str1, str2, str3, &year);
```

▪ Writing a file :-

The file write operations can be performed by the functions printf and fputs with similarities to read operations. The snippet for writing to a file is as :

```
FILE *filePointer ;
filePointer = fopen("fileName.txt", "w");
fprintf(filePointer, "%s %s %d", "We", "are", 2012);
```

▪ Closing a file :-

After every successful file operations, you must always close a file. For closing a file, you have to use fclose function. The snippet for closing a file is given as :

```
FILE *filePointer ;
filePointer= fopen("fileName.txt", "w");
----- Some file Operations -----
fclose(filePointer)
```

Example 1: Program to Open a File, Write in it, And Close the File

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// C program to Open a File,
// Write in it, And Close the File

# include <stdio.h>
# include <string.h>

int main( )
{
    // Declare the file pointer
```

```

FILE *filePointer ;

// Get the data to be written in file
char dataToBeWritten[50]
    = "GeeksforGeeks-A Computer Science Portal for Geeks";

// Open the existing file GfgTest.c using fopen()
// in write mode using "w" attribute
filePointer = fopen("GfgTest.c", "w") ;

// Check if this filePointer is null
// which maybe if the file does not exist
if( filePointer == NULL )
{
    printf( "GfgTest.c file failed to open." ) ;
}
else
{

    printf("The file is now opened.\n" );

    // Write the dataToBeWritten into the file
    if( strlen( dataToBeWritten ) > 0 )
    {

        // writing in the file using fputs()
        fputs(dataToBeWritten, filePointer) ;
        fputs("\n", filePointer) ;
    }

    // Closing the file using fclose()
    fclose(filePointer) ;

    printf("Data successfully written in file GfgTest.c\n");
    printf("The file is now closed." );
}

return 0;
}

```

Example 2: Program to Open a File, Read from it, And Close the File

```

filter_none
edit
close
play_arrow
link
brightness_4
code

// C program to Open a File,
// Read from it, And Close the File

# include <stdio.h>
# include <string.h>

int main( )
{

    // Declare the file pointer
    FILE *filePointer ;

    // Declare the variable for the data to be read from file
    char dataToBeRead[50];

    // Open the existing file GfgTest.c using fopen()
    // in read mode using "r" attribute
    filePointer = fopen("GfgTest.c", "r" ) ;

    // Check if this filePointer is null
    // which maybe if the file does not exist
    if( filePointer == NULL )
    {
        printf( "GfgTest.c file failed to open." ) ;
    }
    else
    {

        printf("The file is now opened.\n" );

        // Read the dataToBeRead from the file
        // using fgets() method
        while( fgets( dataToBeRead, 50, filePointer ) != NULL )
        {

            // Print the dataToBeRead
            printf( "%s" , dataToBeRead ) ;
        }

        // Closing the file using fclose()
        fclose(filePointer) ;
    }
}

```

```
    printf("Data successfully read from file GfgTest.c\n");
    printf("The file is now closed.\n");
}
chevron_right
```

If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes [arrow_drop_up](#)

[Save](#)

Recommended Posts:

- [C | File Handling | Question 3](#)
- [C | File Handling | Question 4](#)
- [C | File Handling | Question 5](#)
- [C | File Handling | Question 2](#)
- [C | File Handling | Question 1](#)
- [Four File Handling Hacks which every C/C++ Programmer should know](#)
- [C program to copy contents of one file to another file](#)
- [C | Pointer Basics | Question 1](#)
- [C | Pointer Basics | Question 11](#)
- [C | Pointer Basics | Question 17](#)
- [C | Pointer Basics | Question 17](#)
- [C | Pointer Basics | Question 8](#)
- [C | Pointer Basics | Question 7](#)
- [C | Pointer Basics | Question 6](#)
- [C | Pointer Basics | Question 17](#)

Improved By : [RishabhPrabhu](#), [RajanKumar3](#), [istwasanders](#), [RuturajM](#), [shreyashagrawal](#)

Article Tags :

[C](#)

[C Programs](#)

Practice Tags :

[C](#)

[thumb_up](#)

24

To-do Done
2.7

Based on 19 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) [Substring in C++](#)

Next

[last_page](#) [Output of C programs | Set 63](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

[Load Comments](#)



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
ctype.h<ctype.h> library in C/C++ with Examples
C program to display month by month calendar for a given year
Predefined Macro in C with Examples
Implicit Type Conversion in C with Examples

Most visited in C Programs
C program to sort an array in ascending order
Check whether the given string is Palindrome using Stack
C/C++ program to print Hello World without using main() and semicolon
Value of Pi() up to 50 decimal places
Program to find absolute value of a given number

fsetpos() (Set File Position) in C

The **fsetpos()** function moves the file position indicator to the location specified by the object pointed to by position. When **fsetpos()** is executed ,the end-of-file indicator is reset.

Declaration

```
int fsetpos(FILE *stream, const fpos_t *position)
```

Parameters -

- **stream** - This is the pointer to a FILE object that identifies the stream.
- **position** - This is the pointer to a fpos_t object containing a position previously obtained with fgetpos.

Return - If it successful, it return **zero** otherwise returns nonzero value.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// c code to demonstrate fsetpos() function.
#include <stdio.h>
int main () {
    FILE *fp;
    fpos_t position;

    /*write your own file name.
     My file name is "myfile.txt"*/
    fp = fopen("myfile.txt","w+");
    fgetpos(fp, &position);
    fputs("HelloWorld!", fp);

    fsetpos(fp, &position);

    // previous function is override
    fputs("geeksforgeeks", fp);
    fclose(fp);

    return(0);
}
```

Output -

```
geeksforgeeks
```

This article is contributed by **Shivani Ghugtyal**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- C program to copy contents of one file to another file
- How to get the current position of cursor from output screen in C?
- What is data type of FILE in C ?
- Basics of File Handling in C
- How to write your own header file in C?
- Comment in header file name?
- C | File Handling | Question 1
- C | File Handling | Question 3
- C | File Handling | Question 4
- C | File Handling | Question 5
- C | File Handling | Question 2
- C program to delete a file
- <complex.h> header file in C with Examples
- time.h header file in C with Examples
- LEX program to add line numbers to a given file

Article Tags :

C
C-File Handling
Practice Tags :

C

2

To-do Done
2

Based on 1 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

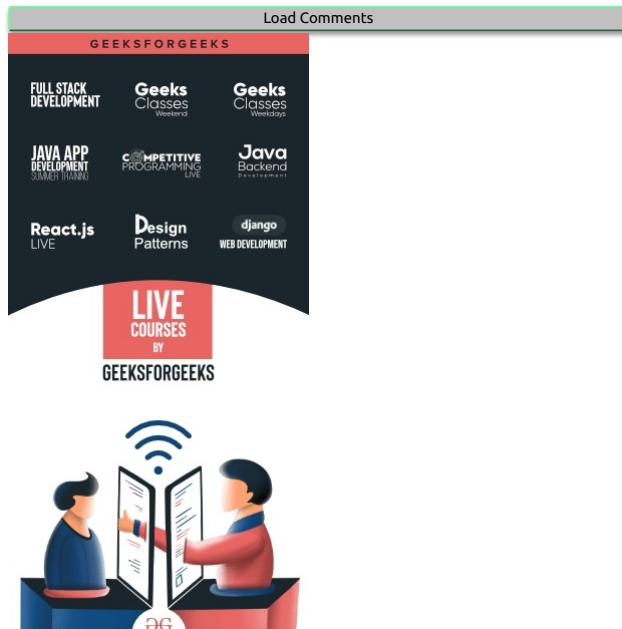
Previous

[first_page](#) How to print a number 100 times without using loop and recursion in C?

Next

[last_page](#) strpbrk() in C

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
Predefined Macros in C with Examples
Format specifier in different Programming Languages
How to call function within function in C or C++
C program to print odd line contents of a File followed by even line content

More related articles in C
C program to print root of a given number
Introduction to the C99 Programming Language : Part II
Problem in comparing Floating point numbers and how to compare them correctly?
Features of C Programming Language
Pointer Expressions in C with Examples

rename function in C/C++

rename() function is used to change the name of the file or directory i.e. from **old_name** to **new_name** without changing the content present in the file. This function takes name of the file as its argument. If **new_name** is the name of an existing file in the same folder then the function may either fail or override the existing file, depending on the specific system and library implementation.

Syntax:

```
int rename (const char *old_name, const char *new_name);
```

Parameters:
old_name : Name of an existing file to be renamed.
new_name : String containing new name of the file.

Return:

Return type of function is an integer. If the file is renamed successfully, zero is returned. On failure, a nonzero value is returned.

Assume that we have a text file having name **geeks.txt**, having some content. So, we are going to rename this file, using the below C program present in the same folder where this file is present.

Name	Date modified	Type	Size
geeks.txt	16-11-2017 18:07	Text Document	0 KB
Rename.c	16-11-2017 18:14	C File	1 KB

Before running program

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// C program to demonstrate use of rename()
#include<stdio.h>

int main()
{
    // Old file name
    char old_name[] = "geeks.txt";

    // Any string
    char new_name[] = "geeksforgeeks.txt";
    int value;

    // File name is changed here
    value = rename(old_name, new_name);

    // Print the result
    if(!value)
    {
        printf("%s", "File name changed successfully");
    }
    else
    {
        perror("Error");
    }
    return 0;
}
chevron_right
filter_none
```

Output:

```
If file name changed
File name changed successfully
OR
If file name not changed
Error: No such file or directory
```

Name	Date modified	Type	Size
geeksforgeeks.txt	16-11-2017 18:07	Text Document	0 KB
Rename.c	16-11-2017 18:14	C File	1 KB
Rename.exe	16-11-2017 18:23	Application	28 KB
Rename.o	16-11-2017 18:23	O File	1 KB

After running code

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes 

Recommended Posts:

- Function Overloading vs Function Overriding in C++
- How to call function within function in C or C++
- What happens when a virtual function is called inside a non-virtual function in C++
- Difference between Virtual function and Pure virtual function in C++
- div() function in C++
- fma() function in C++
- log() function in C++
- exp() function C++
- arc function in C
- wctob() function in C++
- isinf() function in C++
- wcscll() function in C++
- Modulus function in C++ STL
- transform_inclusive_scan() function in C++
- mbstowcs() function in C/C++

**AKASH GUPTA 6**

Check out this Author's contributed articles.

If you like GeeksforGeeks and would like to contribute, you can also write an article using [contribute.geeksforgeeks.org](#) or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please Improve this article if you find anything incorrect by clicking on the "Improve Article" button below.

Article Tags :

C
C++
C-File Handling
C-Library
cpp-file-handling
CPP-Library

Practice Tags :

C
CPP

3

To-do Done
1.5

Based on 2 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) [rint\(\)](#), [rintf\(\)](#), [rintl\(\)](#) in C++

Next

[last_page](#) [map vs unordered_map](#) in C++

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Online Geeks Classes Onsite

JAVA APP DEVELOPMENT COMPETITIVE PROGRAMMING LIVE Java Backend Development

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS

Most popular in C

Print all possible combinations of the string by replacing 's' with any other digit from the string

Predefined Macros in C with Examples

How to call function within function in C or C++

C program to print odd line contents of a file followed by even line content

Introduction to the C99 Programming Language : Part II

Most visited in C++

Vector of Vectors in C++ STL with Examples

C++ Tutorial

Which C++ libraries are useful for competitive programming?

Array of Vectors in C++ STL

Map of Vectors in C++ STL with Examples

tmpfile() function in C

In C Programming Language, the tmpfile() function is used to produce/create a temporary file.

- tmpfile() function is defined in the "stdio.h" header file.
- The created temporary file will automatically be deleted after the termination of program.
- It opens file in binary update mode i.e., wb+ mode.
- The syntax of tmpfile() function is:

```
FILE *tmpfile(void)
```

- The tmpfile() function always returns a pointer after the creation of file to the temporary file. If by chance temporary file can not be created, then the tmpfile() function returns NULL pointer.

```

close
play_arrow
link
brightness_4
code

// C program to demonstrate working of tmpfile()
#include <stdio.h>
int main()
{
    char str[] = "Hello GeeksforGeeks";
    int i = 0;

    FILE* tmp = tmpfile();
    if (tmp == NULL)
    {
        puts("Unable to create temp file");
        return 0;
    }

    puts("Temporary file is created\n");
    while (str[i] != '\0')
    {
        fputc(str[i], tmp);
        i++;
    }

    // rewind() function sets the file pointer
    // at the beginning of the stream.
    rewind(tmp);

    while (!feof(tmp))
        putchar(fgetc(tmp));
}

```

chevron_right

filter_none

Output:

```

Temporary file is created
Hello GeeksforGeeks

```

This article is contributed by **Bishal Kumar Dubey**. If you like GeeksforGeeks and would like to contribute, you can also write an article using [contribute.geeksforgeeks.org](#) or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes **arrow_drop_up**

Add your personal notes
here! (max 5000 chars)

Save

Recommended Posts:

- [How to call function within function in C or C++](#)
- [arc function in C](#)
- [Inline function in C](#)
- [putchar\(\) function in C](#)
- [atexit\(\) function in C/C++](#)
- [strcspn\(\) function in C/C++](#)
- [strrchr\(\) function in C/C++](#)
- [towupper\(\) function in C/C++](#)
- [iswpunct\(\) function in C/C++](#)
- [iswblank\(\) function in C/C++](#)
- [strtoumax\(\) function in C++](#)
- [wctype\(\) function in C/C++](#)
- [strtoimax\(\) function in C++](#)
- [iswctype\(\) function in C/C++](#)
- [c16rtomb\(\) function in C/C++](#)

Article Tags :

C

C-File Handling

Practice Tags :

C

thumb_up

2

To-do Done
1.5

Based on 2 vote(s)

Basic **Easy** **Medium** **Hard** **Expert**

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) [pthread_self\(\)](#) in C with Example

Next

[last_page](#) Interesting facts about data-types and modifiers in C/C++

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Wednesday

JAVA APP DEVELOPMENT SUMMER TRAINING Geeks Classes Wednesday

COMPETITIVE PROGRAMMING LIVE Java Backend Development

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS

Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
Predefined Macros in C with Examples
Format Specifiers in different Programming Languages
C program to print odd line contents of a file followed by even line content
C program to find square root of a given number

More related articles in C
Introduction to the C99 Programming Language : Part II
Problem in comparing Floating point numbers and how to compare them correctly?
Format of C Programs and Languages
Pointer Expressions in C with Examples
Introduction to the C99 Programming Language : Part III

fgetc() and fputc() in C

fgetc()

fgetc() is used to obtain input from a file single character at a time. This function returns the number of characters read by the function. It returns the character present at position indicated by file pointer. After reading the character, the file pointer is advanced to next character. If pointer is at end of file or if an error occurs EOF file is returned by this function.

Syntax:

```
int fgetc(FILE *pointer)
pointer: pointer to a FILE object that identifies
the stream on which the operation is to be performed.
```

```
filter_none
edit
close

play_arrow
link
brightness_4
code

// C program to illustrate fgetc() function
#include <stdio.h>
```

```
int main ()
{
    // open the file
    FILE *fp = fopen("test.txt","r");

    // Return if could not open file
    if (fp == NULL)
        return 0;

    do
    {
        // Taking input single character at a time
        char c = fgetc(fp);

        // Checking for end of file
        if (feof(fp))
            break;

        printf("%c", c);
    } while(1);

    fclose(fp);
    return(0);
}
```

Output:

```
The entire content of file is printed character by
character till end of file. It reads newline character
as well.
```

Using fputc()

fputc() is used to write a single character at a time to a given file. It writes the given character at the position denoted by the file pointer and then advances the file pointer.
This function returns the character that is written in case of successful write operation else in case of error EOF is returned.

Syntax:

```
int fputc(int char, FILE *pointer)
char: character to be written.
This is passed as its int promotion.
pointer: pointer to a FILE object that identifies the
stream where the character is to be written.
```

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// C program to illustrate fputc() function
#include<stdio.h>
int main()
{
    int i = 0;
    FILE *fp = fopen("output.txt","w");

    // Return if could not open file
    if (fp == NULL)
        return 0;

    char string[] = "good bye", received_string[20];

    for (i = 0; string[i]!='\0'; i++)

        // Input string into the file
        // single character at a time
        fputc(string[i], fp);

    fclose(fp);
    fp = fopen("output.txt","r");

    // Reading the string from file
    fgets(received_string,20,fp);

    printf("%s", received_string);

    fclose(fp);
    return 0;
}
```

[chevron_right](#)

filter_none

Output:

```
good bye
```

When fputc() is executed characters of string variable are written into the file one by one. When we read the line from the file we get the same string that we entered.

This article is contributed by **Hardik Gaur**. If you like GeeksforGeeks and would like to contribute, you can also write an article using [contribute.geeksforgeeks.org](#) or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes [arrow_drop_up](#)

Add your personal notes
here! (max 5000 chars)

[Save](#)

Recommended Posts:

- [What is return type of getchar\(\), fgetc\(\) and getc\(\) ?](#)
- [Format specifiers in different Programming Languages](#)
- [C program to find square root of a given number](#)
- [C program to print odd line contents of a File followed by even line content](#)
- [Getting System and Process Information Using C Programming and Shell in Linux](#)
- [Program to calculate Electricity Bill](#)
- [Program to print half Diamond star pattern](#)
- [C/C++ program for calling main\(\) in main\(\)](#)
- [Print all possible combinations of the string by replacing '\\$' with any other digit from the string](#)
- [How to use make utility to build C projects?](#)
- [How to call function within function in C or C++](#)
- [Role of SemiColon in various Programming Languages](#)
- [C program to display month by month calendar for a given year](#)
- [Difference between Type Casting and Type Conversion](#)

Article Tags :

C
cpp-file-handling
CPP-Library
Practice Tags :
C

[thumb_up](#)

6

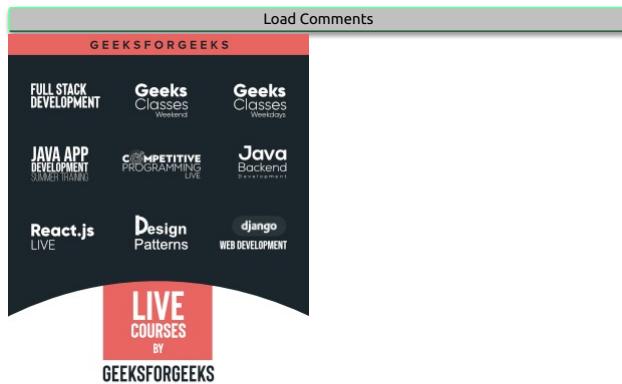
To-do Done

2

Based on 1 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C
Print all possible combinations of the string by replacing 'S' with any other digit from the string
Format specifiers in different Programming Languages
C program to find square root of a given number
Predefined Macros in C with Examples
C program to print odd line contents of a File followed by even line content

More related articles in C
Introduction to the C99 Programming Language : Part II
Features of C Programming Language
Problems in comparing floating point numbers and how to compare them correctly?
<cfloat> float.h in C/C++ with Examples
Getting System and Process Information Using C Programming and Shell in Linux

fseek() in C/C++ with example

fseek() is used to move file pointer associated with a given file to a specific position.

Syntax:

```
int fseek(FILE *pointer, long int offset, int position)
pointer: pointer to a FILE object that identifies the stream.
offset: number of bytes to offset from position
position: position from where offset is added.

returns:
zero if successful, or else it returns a non-zero value
```

position defines the point with respect to which the file pointer needs to be moved. It has three values:

SEEK_END : It denotes end of the file.

SEEK_SET : It denotes starting of the file.

SEEK_CUR : It denotes file pointer's current position.

```
filter_none
edit
close

play_arrow

link
brightness_4
code

// C Program to demonstrate the use of fseek()
#include <stdio.h>
```

```
int main()
{
    FILE *fp;
    fp = fopen("test.txt", "r");

    // Moving pointer to end
    fseek(fp, 0, SEEK_END);

    // Printing position of pointer
    printf("%ld", ftell(fp));

    return 0;
}
```

Output:

"Someone over there is calling you.
we are going for work.
take care of yourself."

When we implement fseek() we move the pointer by 0 distance with respect to end of file i.e pointer now points to end of the file. Therefore the output is 81.

Related article: fseek vs rewind in C

This article is contributed by **Hardik Gaur**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes [arrow_drop_up](#)

[Save](#)

Recommended Posts:

- [fseek\(\) vs rewind\(\) in C](#)
- [std::basic_istream::ignore in C++ with Examples](#)
- [std::basic_istream::getline in C++ with Examples](#)
- [Format Specifiers in different Programming Languages](#)
- [C program to find square root of a given number](#)
- [C program to print odd line contents of a File followed by even line content](#)
- [std::is_heap\(\) in C++ with Examples](#)
- [std::basic_istream::gcount\(\) in C++ with Examples](#)
- [new vs malloc\(\) and free\(\) vs delete in C++](#)
- [std::string::rfind in C++ with Examples](#)
- [Sum of factorials of Prime numbers in a Linked list](#)
- [Sum of all Palindrome Numbers present in a Linked list](#)
- [Generate an array of given size with equal count and sum of odd and even numbers](#)
- [Namespaces in C++ | Set 4 \(Overloading, and Exchange of Data in different Namespaces\)](#)

Article Tags :

[C](#)

[C++](#)

[C-File Handling](#)

[C-Library](#)

[CPP-Library](#)

Practice Tags :

[C](#)

[CPP](#)

[thumb_up](#)

2

To-do Done
2.2

Based on 5 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

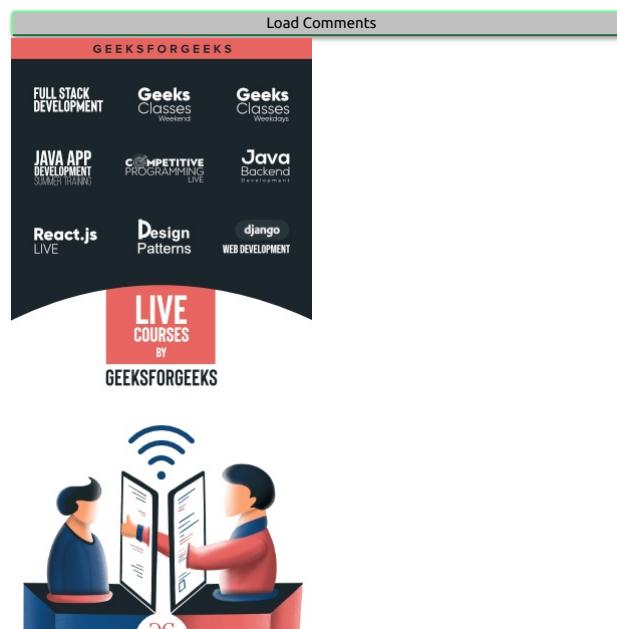
Previous

[first_page](#) C Program to find direction of growth of stack

Next

[last_page](#) dup() and dup2() Linux system call

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C
Combinations of the string by replacing 's' with any other digit from the string
Predefined Macros in C with Examples
C program to print odd line contents of a File followed by even line content
Introduction to the C99 Programming Language : Part II
C program to find square root of a given number

Most visited in C++
Vector of Vectors in C++ STL with Examples
C++ Tutorial
Which C++ libraries are useful for competitive programming?
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

ftell() in C with example

ftell() in C is used to find out the position of file pointer in the file with respect to starting of the file. Syntax of ftell() is:

```
long ftell(FILE *pointer)
```

Consider below C program. The file taken in the example contains the following data :

"Someone over there is calling you. We are going for work. Take care of yourself." (without the quotes)

When the fscanf statement is executed word "Someone" is stored in string and the pointer is moved beyond "Someone". Therefore ftell(fp) returns 7 as length of "someone" is 6.

```
filter_none
edit
close

play_arrow

link
brightness_4
code

// C program to demonstrate use of ftell()
#include<stdio.h>

int main()
{
    /* Opening file in read mode */
    FILE *fp = fopen("test.txt","r");

    /* Reading first string */
    char string[20];
    fscanf(fp,"%s",string);

    /* Printing position of file pointer */
    printf("%ld", ftell(fp));
    return 0;
}

chevron_right
filter_none
```

Output : Assuming test.txt contains "Someone over there".

7

This article is contributed by **Hardik Gaur**. If you like GeeksforGeeks and would like to contribute, you can also write an article using [contribute.geeksforgeeks.org](#) or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes [arrow_drop_up](#)

Add your personal notes
here! (max 5000 chars)

Save

Recommended Posts:

- Format specifiers in different Programming Languages
- C program to find square root of a given number
- C program to print odd line contents of a File followed by even line content
- Getting System and Process Information Using C Programming and Shell in Linux
- Program to calculate Electricity Bill
- Program to print half Diamond star pattern
- C/C++ program for calling main() in main()
- Print all possible combinations of the string by replacing '\$' with any other digit from the string
- How to use make utility to build C projects?
- How to call function within function in C or C++
- Role of SemiColon in various Programming Languages
- C program to display month by month calendar for a given year
- Difference between Type Casting and Type Conversion
- Pi(π) in C++ with Examples

Article Tags :

C
cpp-file-handling
CPP-Library

Practice Tags :

C

[thumb_up](#)

2

To-do Done
1.3

Based on 3 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) Output of C programs | Set 30 (Switch Case)

Next

[last_page](#) C Program to find direction of growth of stack

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments



Most popular in C
 Print all possible combinations of the string by replacing '\$' with any other digit from the string
 Predefined Macros in C with Examples
 C program to print odd line contents of a File followed by even line content
 Introduction to the C99 Programming Language : Part II
 C program to find square root of a given number
 More related articles in C
 Format specifiers in different Programming Languages
 Problem in Comparing Floating point numbers and how to compare them correctly?
 Features of C Programming Language
 Pointer Expressions in C with Examples
 Introduction to the C99 Programming Language : Part III

Iseek() in C/C++ to read the alternate nth byte and write it in another file

From a given file (e.g. input.txt) read the alternate nth byte and write it on another file with the help of "lseek".

Iseek (C System Call): lseek is a system call that is used to change the location of the read/write pointer of a file descriptor. The location can be set either in absolute or relative terms.

Function Definition

```
off_t lseek(int fd, off_t offset, int whence);
```

Field Description

int fd : The file descriptor of the pointer that is going to be moved
off_t offset : The offset of the pointer (measured in bytes).
int whence : The method in which the offset is to be interpreted (rela, absolute, etc.). Legal value r this variable are provided at the end.
return value : Returns the offset of the pointer (in bytes) from the beginning of the file. If the return value is -1, then there was an error moving the pointer.

For example, say our Input file is as follows:

```
start.txt (~/) - gedit
Open  Save
start.txt  x  end.txt  x
We encounter various problems like Maximum length palindrome in a string, number of palindromic substrings and many more interesting problems on palindromic substrings . Mostly of these palindromic substring problems have some DP O(n^2) solution (n is length of the given string) or then we have a complex algorithm like Manacher's algorithm which solves the Palindromic problems in linear time.In this article, we will study an interesting Data Structure, which will solve all the above similar problems in much more simpler way. This data structure is invented by Mikhail Rubinchik.
```

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// C program to read nth byte of a file and
// copy it to another file using lseek
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <fcntl.h>

void func(char arr[], int n)
```

```

{
    // Open the file for READ only.
    int f_write = open("start.txt", O_RDONLY);

    // Open the file for WRITE and READ only.
    int f_read = open("end.txt", O_WRONLY);

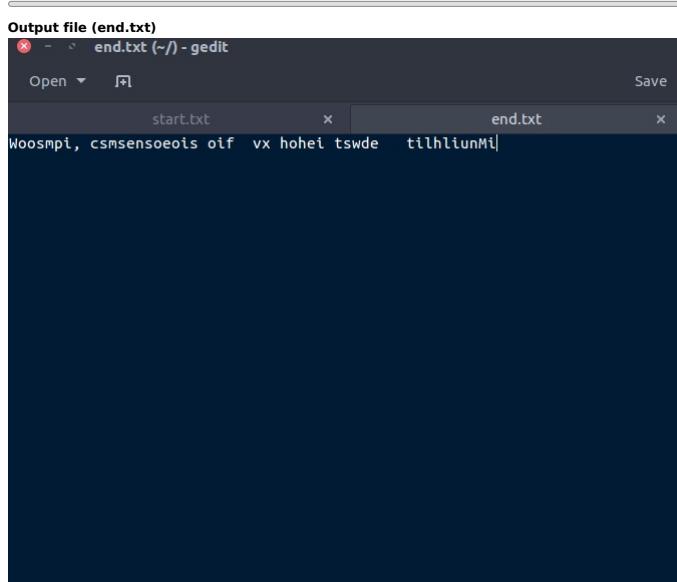
    int count = 0;
    while (read(f_write, arr, 1))
    {
        // to write the 1st byte of the input file in
        // the output file
        if (count < n)
        {
            // SEEK_CUR specifies that
            // the offset provided is relative to the
            // current file position
            lseek (f_write, n, SEEK_CUR);
            write (f_read, arr, 1);
            count = n;
        }

        // After the nth byte (now taking the alternate
        // nth byte)
        else
        {
            count = (2*n);
            lseek(f_write, count, SEEK_CUR);
            write(f_read, arr, 1);
        }
    }
    close(f_write);
    close(f_read);
}

// Driver code
int main()
{
    char arr[100];
    int n;
    n = 5;

    // Calling for the function
    func(arr, n);
    return 0;
}

```



This article is contributed by **Kishlay Verma**. If you like GeeksforGeeks and would like to contribute, you can also write an article using [contribute.geeksforgeeks.org](#) or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes [arrow_drop_up](#)

[Save](#)

Recommended Posts:

- [Read/Write structure to a file in C](#)
- [Read/Write Class Objects from/to File in C++](#)
- [Input-output system calls in C | Create, Open, Close, Read, Write](#)

- Read a record from a File in C++ using seekg() and tellg()
- Implement your own tail (Read last n lines of a huge file)
- How to write your own header file in C?
- fopen() for an existing file in write mode
- Write a C program that displays contents of a given file like 'more' utility in Linux
- C++ program to read file word by word
- C program to copy contents of one file to another file
- How to Read and Print an Integer value in C
- How to Read and Print an Integer value in C++
- Write a URL in a C++ program
- When should we write our own copy constructor?
- Write your own memcpy() and memmove()

Article Tags :

C
C++
cpp-file-handling
Practice Tags :
C
CPP

thumb_up
1

To-do Done
3.2

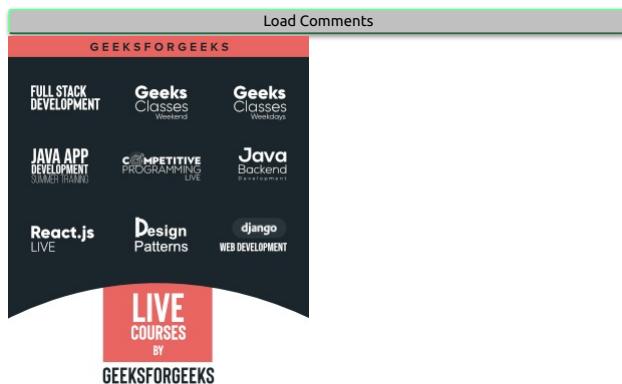
Based on 7 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation
Previous
[first_page](#) random header | Set 2 (Distributions)
Next
[last_page](#) Zombie Processes and their Prevention

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
Predefined Macros in C with Examples
Format Specifiers in different programming Languages
How to call function within function in C or C++
C program to print odd line contents of a File followed by even line content

Most visited in C++
Vector of Vectors in C++ STL with Examples
C++ Tutorial
Which C++ libraries are useful for competitive programming?
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

C program to delete a file

The `remove` function in C/C++ can be used to delete a file. The function returns 0 if files is deleted successfully, other returns a non-zero value.

```
filter_none
edit
close

play_arrow
link
brightness_4
code

#include<stdio.h>

int main()
{
    if (remove("abc.txt") == 0)
        printf("Deleted successfully");
    else
        printf("Unable to delete the file");

    return 0;
}
```

```
}
```

```
chevron_right
```

```
filter_none
```

Using remove() function in C, we can write a program which can destroy itself after it is compiled and executed.

Explanation: This can be done using the [remove function](#) in C. Note that, this is done in Linux environment. So, the remove function is fed the first parameter in command line argument i.e. **a.out** file (executable file) created after compiling . Hence the program will be destroyed.

```
filter_none  
edit  
close  
  
play_arrow  
  
link  
brightness_4  
code  
  
#include<stdio.h>  
#include<stdlib.h>
```

```
int main(int c, char *argv[])
{
    printf("By the time you will compile me I will be destroyed \n");

    // array of pointers to command line arguments
    remove(argv[0]);

    // Note: argv[0] will contain the executable file i.e. 'a.out'

    return 0;
}
```

```
// This code is contributed by MAZHAR IMAM KHAN.  
chevron_right
```

```
filter_none
```

Output:

```
By the time you will compile me I will be destroyed
```

After the output shown above, the **a.out** file will be removed.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up

```
Add your personal notes  
here! (max 5000 chars)
```

```
Save
```

Recommended Posts:

- [Program to delete a line given the line number from a file](#)
- [C program to copy contents of one file to another file](#)
- [C++ program to create a file](#)
- [C Program to find size of a File](#)
- [C Program to print contents of file](#)
- [LEX program to add line numbers to a given file](#)
- [C Program to count number of lines in a file](#)
- [C++ program to print unique words in a file](#)
- [C Program to merge contents of two files into a third file](#)
- [C++ program to append content of one text file to another](#)
- [C Program to count the Number of Characters in a File](#)
- [Write a C program that displays contents of a given file like 'more' utility in Linux](#)
- [C Program for Lower Case to Uppercase and vice-versa in a file](#)
- [Lex program to take input from file and remove multiple spaces, lines and tabs](#)
- [C program to print odd line contents of a File followed by even line content](#)

Article Tags :

C
cpp-file-handling
Practice Tags :
C

thumb_up
4

To-do Done
2.1

Based on 9 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) C Program to merge contents of two files into a third file

Next

[last_page](#) Multithreading in C

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Weekend Geeks Classes Weekdays

JAVA APP DEVELOPMENT SUMMER TRAINING COMPETITIVE PROGRAMMING LIVE Java Backend Development

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS

Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
Predefined Macros in C with Examples
How to call function within function in C or C++
C program to print odd line contents of a file followed by even line content
Introduction to the C99 Programming Language : Part II

More related articles in C
C program to find square root of a given number
Format specifiers in different Programming Languages
Problems with Floating point numbers and how to compare them correctly?
Features of C Programming Language
Pointer Expressions in C with Examples

C Program to merge contents of two files into a third file

Let the given two files be file1.txt and file2.txt. The following are steps to merge.

- 1) Open file1.txt and file2.txt in read mode.
- 2) Open file3.txt in write mode.
- 3) Run a loop to one by one copy characters of file1.txt to file3.txt.
- 4) Run a loop to one by one copy characters of file2.txt to file3.txt.
- 5) Close all files.

To successfully run the below program file1.txt and file2.txt must exists in same folder.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

#include <stdio.h>
#include <stdlib.h>

int main()
{
    // Open two files to be merged
    FILE *fp1 = fopen("file1.txt", "r");
    FILE *fp2 = fopen("file2.txt", "r");

    // Open file to store the result
    FILE *fp3 = fopen("file3.txt", "w");
    char c;

    if (fp1 == NULL || fp2 == NULL || fp3 == NULL)
    {
        puts("Could not open files");
        exit(0);
    }

    // Copy contents of first file to file3.txt
    while ((c = fgetc(fp1)) != EOF)
        fputc(c, fp3);

    // Copy contents of second file to file3.txt
    while ((c = fgetc(fp2)) != EOF)
        fputc(c, fp3);

    printf("Merged file1.txt and file2.txt into file3.txt");

    fclose(fp1);
    fclose(fp2);
    fclose(fp3);
    return 0;
}
```

Output:

Merged file1.txt and file2.txt into file3.txt

Related Articles :

- Java program to merge two files alternatively into third file
- Java program to merge two files into a third file

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes [arrow_drop_up](#)

Add your personal notes here! (max 5000 chars)

[Save](#)

Recommended Posts:

- C program to copy contents of one file to another file
- C Program to print contents of file
- Write a C program that displays contents of a given file like 'more' utility in Linux
- C program to print odd line contents of a File followed by even line content
- Program to copy the contents of one array into another in the reverse order
- C program to compare two files and report mismatches
- C Program to list all files and sub-directories in a directory
- Merge operations using STL in C++ | merge(), includes(), set_union(), set_intersection(), set_difference(), .. inplace_merge,
- C++ program to create a file
- C program to delete a file
- C Program to find size of a File
- LEX program to add line numbers to a given file
- C++ program to append content of one text file to another
- C Program to count number of lines in a file
- C Program to count the Number of Characters in a File

Article Tags :

C
cpp-file-handling

Practice Tags :

C

[thumb_up](#)
2

To-do Done
2

Based on 5 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

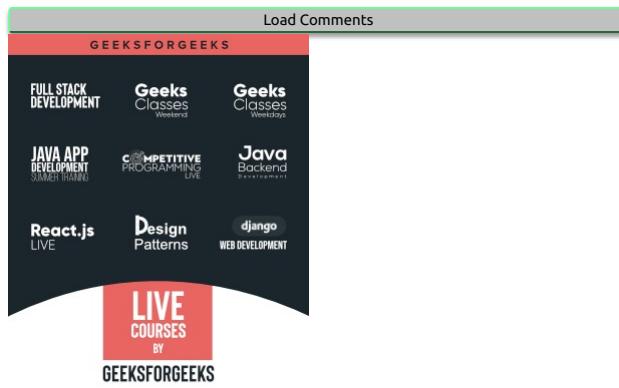
Previous

[first_page](#) How to print range of basic data types without any library function and constant in C?

Next

[last_page](#) C program to delete a file

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
[ctype.h\(<cctype>\)](#) library in C/C++ with Examples
Implicit Type Conversion in C with Examples
Format specifiers in different Programming Languages
C program to find square root of a given number

More related articles in C
Predefined Macros in C with Examples
How to cat function with examples in C or C++
C program to print line contents of a File followed by even line content
How to create GUI in C programming using GTK Toolkit
Introduction to the C99 Programming Language : Part II

C Program to print contents of file

[fopen\(\)](#) is used to open and [fclose\(\)](#) is used to close a file in C

[filter_none](#)
[edit](#)
[close](#)

```

play_arrow
link
brightness_4
code

#include <stdio.h>
#include <stdlib.h> // For exit()

int main()
{
    FILE *fptr;

    char filename[100], c;

    printf("Enter the filename to open \n");
    scanf("%s", filename);

    // Open file
    fptr = fopen(filename, "r");
    if (fptr == NULL)
    {
        printf("Cannot open file \n");
        exit(0);
    }

    // Read contents from file
    c = fgetc(fptr);
    while (c != EOF)
    {
        printf ("%c", c);
        c = fgetc(fptr);
    }

    fclose(fptr);
    return 0;
}
chevron_right
filter_none

```

Output:

```

Enter the filename to open
a.txt
/*Contents of a.txt*/

```

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

[My Personal Notes](#) 

Add your personal notes here! (max 5000 chars)

[Save](#)

Recommended Posts:

- C program to print odd line contents of a File followed by even line content
- C program to copy contents of one file to another file
- C Program to merge contents of two files into a third file
- Write a C program that displays contents of a given file like 'more' utility in Linux
- C++ program to print unique words in a file
- Program to copy the contents of one array into another in the reverse order
- Print "Hello World" in C/C++ without using any header file
- C program to delete a file
- C++ program to create a file
- C program to print a string without any quote (single or double) in the program
- C Program to find size of a File
- LEX program to add line numbers to a given file
- C Program to count the Number of Characters in a File
- C Program to count number of lines in a file
- C++ program to append content of one text file to another

Article Tags :

C
cpp-file-handling

Practice Tags :

C

 3

To-do Done
1.6

Based on 3 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) How to write a running C code without main()?

Next

[last_page](#) C program to copy contents of one file to another file

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

[Load Comments](#)



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
ctype.h <ctype.h> library in C/C++ with Examples
C program to display month by month calendar for a given year
Predefined Macro in C with Examples
Implicit Type Conversion in C with Examples

More related articles in C
How to call function within function in C or C++
C programs to print contents of file followed by even line content
Introduction to the C99 Programming Language - Part II
C program to find square root of a given number
Format specifiers in different Programming Languages

C Program to print numbers from 1 to N without using semicolon?

How to print numbers from 1 to N without using any semicolon in C.

```
filter_none
edit
close
play_arrow
link
brightness_4
code
#include<stdio.h>
#define N 100

// Add your code here to print numbers from 1
// to N without using any semicolon
chevron_right
```

What code to add in above snippet such that it doesn't contain semicolon and prints numbers from 1 to N?

We strongly recommend you to minimize your browser and try this yourself first

Method 1 (Recursive)

```
filter_none
edit
close
play_arrow
link
brightness_4
code
// A recursive C program to print all numbers from 1
// to N without semicolon
#include<stdio.h>
#define N 10
```

```
int main(int num)
{
    if (num <= N && printf("%d ", num) && main(num + 1))
    {
    }
}
```

chevron_right

filter_none

Output:

```
1 2 3 4 5 6 7 8 9 10
```

See [this](#) for complete run. Thanks to Utkarsh Trivedi for suggesting this solution.

Method 2 (Iterative)

```
filter_none
edit
close
play_arrow
link
brightness_4
```

code

```
// An iterative C program to print all numbers from 1  
// to N without semicolon  
#include<stdio.h>  
#define N 10
```

```
int main(int num, char *argv[]){  
    while (num <= N && printf("%d ", num) && num++)  
    {  
    }  
}  
chevron_right
```

filter_none

Output:

```
1 2 3 4 5 6 7 8 9 10
```

See [this](#) for complete run. Thanks to Rahul Huria for suggesting this solution.

How do these solutions work?

main() function can receive arguments. The first argument is argument count whose value is 1 if no argument is passed to it. The first argument is always program name.

filter_none
edit
close

play_arrow

link
brightness_4
code

```
#include<stdio.h>
```

```
int main(int num, char *argv[]){  
    printf("num = %d\n", num);  
    printf("argv[0] = %s ", argv[0]);  
}
```

chevron_right

filter_none

Output:

```
num = 1  
argv[0] = <file_name>
```

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- [How to print a semicolon\(;\) without using semicolon in C/C++?](#)
- [Write a C program to print "Geeks for Geeks" without using a semicolon](#)
- [Print Hello World without semicolon in C/C++](#)
- [Program to print a pattern of numbers](#)
- [Role of SemiColon in various Programming Languages](#)
- [C program to print a string without any quote \(single or double\) in the program](#)
- [How will you print numbers from 1 to 100 without using loop? | Set-2](#)
- [How will you print numbers from 1 to 100 without using loop?](#)
- [Print numbers in sequence using thread synchronization](#)
- [Program to print last 10 lines](#)
- [C Program to print environment variables](#)
- [Program to print Happy Birthday](#)
- [C Program to print contents of file](#)
- [C Program to print Floyd's triangle](#)
- [Program to print the diamond shape](#)

Improved By : [maveriek](#)

Article Tags :

C
c-puzzle
Practice Tags :

thumb_up
21

To-do Done

3

Based on 52 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) How to check whether a number is in the range[low, high] using one comparison ?

Next

[last_page](#) Print individual digits as words without using if or switch

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Weekend Geeks Classes Weekdays

JAVA APP DEVELOPMENT SUMMER TRAINING COMPETITIVE PROGRAMMING LIVE Java Backend Development

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
Predefined Macros in C with Examples
How to call function within function in C or C++
C program to print odd line contents of a file followed by even line content
Introduction to the C99 Programming Language : Part II

More related articles in C
C program to find square root of a given number
Format specifiers in different Programming Languages
Problem in comparing Floating point numbers and how to compare them correctly?
Features of C Programming Language
Pointer Expressions in C with Examples

To find sum of two numbers without using any operator

Write a program to find sum of positive integers without using any operator. Only use of printf() is allowed. No other library function can be used.

Solution

It's a trick question. We can use printf() to find sum of two numbers as printf() returns the number of characters printed. The **width field in printf()** can be used to find the sum of two numbers. We can use '*' which indicates the minimum width of output. For example, in the statement "printf("%*d", width, num)", the specified 'width' is substituted in place of *, and 'num' is printed within the minimum width specified. If number of digits in 'num' is smaller than the specified 'width', the output is padded with blank spaces. If number of digits are more, the output is printed as it is (not truncated). In the following program, add() returns sum of x and y. It prints 2 spaces within the width specified using x and y. So total characters printed is equal to sum of x and y. That is why add() returns x+y.

filter_none
edit
close

play_arrow

link
brightness_4
code


```
#include <stdio.h>

int add(int x, int y)
{
    return printf("%c%c", x, ' ', y, ' ');
}

// Driver code
int main()
{
    printf("Sum = %d", add(3, 4));
    return 0;
}
```

Output:

Sum = 7

The output is seven spaces followed by "Sum = 7". We can avoid the leading spaces by using carriage return. Thanks to **krazyCoder** and **Sandeep** for suggesting this. The following program prints output without any leading spaces.

filter_none
edit
close

play_arrow

link
brightness_4
code


```
#include <stdio.h>

int add(int x, int y)
{
    return printf("%c%c", x, '\r', y, '\r');
}

// Driver code
```

```
int main()
{
    printf("Sum = %d", add(3, 4));
    return 0;
}
```

[chevron_right](#)

[filter_none](#)

Output:

```
Sum = 7
```

Another Method :

C++

```
filter_none
edit
close

play_arrow
link
brightness_4
code

#include <iostream>
using namespace std;

int main()
{
    int a = 10, b = 5;
    if (b > 0) {
        while (b > 0) {
            a++;
            b--;
        }
    }
    if (b < 0) { // when 'b' is negative
        while (b < 0) {
            a--;
            b++;
        }
    }
    cout << "Sum = " << a;
    return 0;
}

// This code is contributed by SHUBHAMSINGH10
// This code is improved & fixed by Abhijeet Soni.
```

[chevron_right](#)

[filter_none](#)

C

```
filter_none
edit
close

play_arrow
link
brightness_4
code

#include <stdio.h>

int main()
{
    int a = 10, b = 5;
    if (b > 0) {
        while (b > 0) {
            a++;
            b--;
        }
    }
    if (b < 0) { // when 'b' is negative
        while (b < 0) {
            a--;
            b++;
        }
    }
    printf("Sum = %d", a);
    return 0;
}

// This code is contributed by Abhijeet Soni
```

[chevron_right](#)

[filter_none](#)

Java

```
filter_none
edit
close

play_arrow
```

```
link  
brightness_4  
code  
  
// Java code  
class GfG {  
  
    public static void main(String[] args)  
    {  
        int a = 10, b = 5;  
        if (b > 0) {  
            while (b > 0) {  
                a++;  
                b--;  
            }  
        }  
        if (b < 0) { // when 'b' is negative  
            while (b < 0) {  
                a--;  
                b++;  
            }  
        }  
        System.out.println("Sum is: " + a);  
    }  
}  
  
// This code is contributed by Abhijeet Soni
```

[chevron_right](#)
[filter_none](#)

Python 3

```
filter_none  
edit  
close  
  
play_arrow  
  
link  
brightness_4  
code  
  
# Python 3 Code
```

```
if __name__ == '__main__':  
  
    a = 10  
    b = 5  
  
    if b > 0:  
        while b > 0:  
            a = a + 1  
            b = b - 1  
    if b < 0:  
        while b < 0:  
            a = a - 1  
            b = b + 1  
  
    print("Sum is: ", a)
```

```
# This code is contributed by Akanksha Rai  
# This code is improved & fixed by Abhijeet Soni  
chevron\_right  
filter\_none
```

C#

```
filter_none  
edit  
close  
  
play_arrow  
  
link  
brightness_4  
code  
  
// C# code  
using System;
```

```
class GFG {  
    static public void Main()  
    {  
        int a = 10, b = 5;  
        if (b > 0) {  
            while (b > 0) {  
                a++;  
                b--;  
            }  
        }  
        if (b < 0) { // when 'b' is negative  
            while (b < 0) {  
                a--;  
                b++;  
            }  
        }  
    }  
}
```

```

        }
        Console.WriteLine("Sum is: " + a);
    }

// This code is contributed by Tushil
// This code is improved & fixed by Abhijeet Soni.
chevron_right
filter_none

```

PHP

```

filter_none
edit
close
play_arrow
link
brightness_4
code

<?php
// PHP Code
$a = 10;
$b = 5;

if ($b > 0) {
while($b > 0)
{
    $a++;
    $b--;
}
}

if ($b < 0) {
while($b < 0)
{
    $a--;
    $b++;
}
}

echo "Sum is: ", $a;

```

```

// This code is contributed by Dinesh
// This code is improved & fixed by Abhijeet Soni.
?>
chevron_right
filter_none

```

Output:

```
Sum = 15
```

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up

Add your personal notes here! (max 5000 chars)

Recommended Posts:

- [Program to find remainder without using modulo or % operator](#)
- [Find the remainder when N is divided by 4 using Bitwise AND operator](#)
- [Program to Find the Largest Number using Ternary Operator](#)
- [Given two numbers a and b find all x such that a % x = b](#)
- [Find two numbers whose sum and GCD are given](#)
- [Program to find LCM of two numbers](#)
- [Find XOR of numbers from the range \[L, R\]](#)
- [Program to find GCD or HCF of two numbers](#)
- [Find the XOR of first N Prime Numbers](#)
- [Find the numbers from 1 to N that contains exactly k non-zero digits](#)
- [Find the sum of the all amicable numbers up to N](#)
- [Find the sum of the first Nth Icosagonal Numbers](#)
- [Program to find LCM of 2 numbers without using GCD](#)
- [Find two prime numbers with given sum](#)
- [Find k numbers which are powers of 2 and have sum N | Set 1](#)

Improved By : [falgunatara](#), [prerna saini](#), [jit_t](#), [Akanksha_Rai](#), [SHUBHAMSINGH10](#), [more](#)

Article Tags :

C

Mathematical

C-Operators

Practice Tags :

Mathematical

C

28

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page Variable length arguments for Macros](#)

Next

[last_page Copy elision in C++](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

The screenshot shows the GeeksforGeeks homepage with a focus on their live courses. At the top, there's a navigation bar with links like 'Load Comments', 'GEEKSFORGEEKS', 'FULL STACK DEVELOPMENT', 'Geeks Classes', 'Java Development', 'JAVA APP DEVELOPMENT', 'COMPETITIVE PROGRAMMING', 'React.js LIVE', 'Design Patterns', and 'django WEB DEVELOPMENT'. Below this is a large red banner with the text 'LIVE COURSES BY GEEKSFORGEEKS'. Underneath the banner, there's an illustration of two people looking at a screen with a Wi-Fi signal. The text on the page includes:

- Most popular in C
- getchar function in C with Examples
- Print all possible combinations of the string by replacing 'S' with any other digit from the string `ctype.h <cctype>` library in C/C++ with Examples
- How to use make utility to build C projects?
- C programs to display month by month calendar for a given year
- Most visited in Mathematical
- Must do Math for Competitive Programming
- Count of subarrays having exactly K perfect square numbers
- Product of all Subarrays of an Array
- Perfect Sum Problem
- Logarithm tricks for Competitive Programming

How will you show memory representation of C variables?

Write a C program to show memory representation of C variables like int, float, pointer, etc.

Algorithm:

Get the address and size of the variable. Typecast the address to char pointer. Now loop for size of the variable and print the value at the typecasted pointer.

Program:

```
filter_none
edit
close
play_arrow
link
brightness_4
code
#include <stdio.h>
typedef unsigned char *byte_pointer;

/*show_bytes takes byte pointer as an argument
and prints memory contents from byte_pointer
to byte_pointer + len */
void show_bytes(byte_pointer start, int len)
{
    int i;
    for (i = 0; i < len; i++)
        printf(" %.2x", start[i]);
    printf("\n");
}

void show_int(int x)
{
    show_bytes((byte_pointer) &x, sizeof(int));
}

void show_float(float x)
{
    show_bytes((byte_pointer) &x, sizeof(float));
}

void show_pointer(void *x)
{
```

```
show_bytes((byte_pointer) &x, sizeof(void *));
```

```
}
```

```
/* Driver program to test above functions */
```

```
int main()
```

```
{
```

```
    int i = 1;
    float f = 1.0;
    int *p = &i;
    show_float(f);
    show_int(i);
    show_pointer(p);
    show_int(i);
    getchar();
    return 0;
}
```

```
chevron_right
```

```
filter_none
```

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes [arrow_drop_up](#)

Add your personal notes here! (max 5000 chars)

[Save](#)

Recommended Posts:

- Difference between Static variables and Register variables in C
- Variables and Keywords in C
- Static Variables in C
- Can Global Variables be dangerous ?
- Operations on struct variables in C
- Initialization of static variables in C
- Constants vs Variables in C language
- Implicit initialization of variables with 0 or 1 in C
- Initialization of global and static variables in C
- C Program to print environment variables
- Initialization of variables sized arrays in C
- What are the default values of static variables in C?
- How are variables scoped in C - Static or Dynamic?
- Linking Files having same variables with different data types in C
- Swap two variables in one line in C/C++, Python, PHP and Java

Article Tags :

C

C-puzzle

Practice Tags :

C

[thumb_up](#)

5

To-do Done
4

Based on 27 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) [What is the best way in C to convert a number to a string?](#)

Next

[last_page](#) [Understanding “extern” keyword in C](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

[Load Comments](#)



Most popular in C

Condition To Print “HelloWord”

What should be the “condition” so that the following code snippet prints both HelloWorld!

```
if "condition"
    printf ("Hello");
else
    printf("World");
```

Method 1:

```
filter_none
edit
close
play_arrow
link
brightness_4
code
#include<stdio.h>
int main()
{
    if(!printf("Hello"))
        printf("Hello");
    else
        printf("World");
    getchar();
}
chevron_right
filter_none
```

Explanation: Printf returns the number of character it has printed successfully. So, following solutions will also work

if (printf("Hello") < 0) or if (printf("Hello") < 1) etc

Method 2: Using fork()

```
filter_none
edit
close
play_arrow
link
brightness_4
code
#include<stdio.h>
#include<unistd.h>
int main()
{
    if(fork())
        printf("Hello");
    else
        printf("World");
    getchar();
}
chevron_right
filter_none
```

This method is contributed by Aravind Alapati.

Please comment if you find more solutions of this.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up

Save

Recommended Posts:

- Count elements in a vector that match a target value or condition
- How to print % using printf?
- How to print a variable name in C?
- Print 1 2 3 infinitely using threads in C
- How will you print numbers from 1 to 100 without using loop? | Set-2
- Print a long int in C using putchar() only
- Print "Even" or "Odd" without using conditional statement
- Print Hello World without semicolon in C/C++
- How will you print numbers from 1 to 100 without using loop?
- C++ program to print all Even and Odd numbers from 1 to N
- Print calendar for a given year in C++
- How to Read and Print an Integer value in C
- Program to print last 10 lines
- Print 1 to 100 in C++, without loop and recursion
- How to Read and Print an Integer value in C++

Article Tags :

C
C++
c-puzzle

Practice Tags :

C

CPP

thumb_up

7

To-do Done
2.3

Based on 36 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

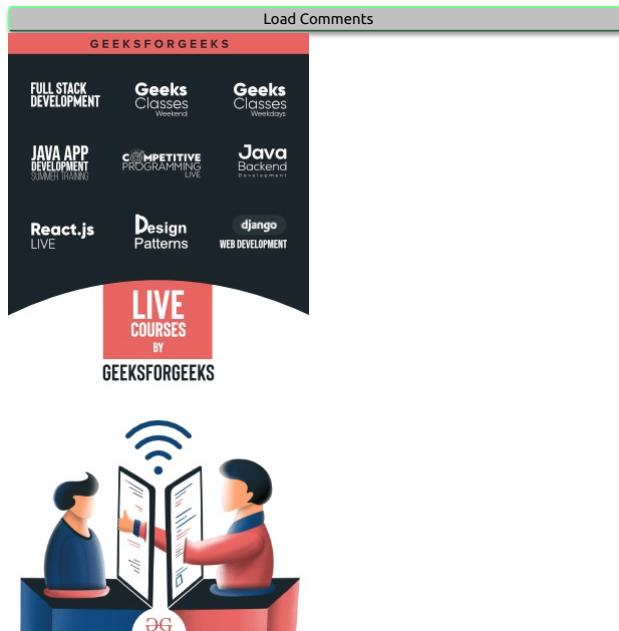
Previous

first_page Implement Your Own sizeof

Next

last_page Change/add only one character and print '*' exactly 20 times

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.



Change/add only one character and print '*' exactly 20 times

In the below code, change/add only one character and print '*' exactly 20 times.

```
int main()
{
    int i, n = 20;
    for (i = 0; i < n; i--)
        printf(" *");
    getchar();
    return 0;
}
```

Solutions:

1. Replace i by n in for loop's third expression

C++

```
filter_none
edit
close
play_arrow
link
brightness_4
code
#include <iostream>
using namespace std;
int main()
{
    int i, n = 20;
    for (i = 0; i < n; n--)
        cout << " *";
    getchar();
    return 0;
}
chevron_right
filter_none
```

C

```
filter_none
edit
close
play_arrow
link
brightness_4
code

#include <stdio.h>
int main()
{
    int i, n = 20;
    for (i = 0; i < n; n--)
        printf("*");
    getchar();
    return 0;
}
chevron_right
filter_none
```

Java

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// Java code
class GfG {
public static void main(String[] args)
{
    int i, n = 20;
    for (i = 0; i < n; n--)
        System.out.print("*");
}
}
chevron_right
filter_none
```

C#

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// C# code
using System;
class GfG
{
    public static void Main()
    {
        int i, n = 20;
        for (i = 0; i < n; n--)
            Console.Write("*");
    }
}

// This code is contributed by SoumikMondal
chevron_right
filter_none
```

2. Put '-' before i in for loop's second expression

```
filter_none
edit
close
play_arrow
link
brightness_4
code

#include <stdio.h>
int main()
{
    int i, n = 20;
    for (i = 0; -i < n; i--)
        printf("*");
    getchar();
    return 0;
}
chevron_right
filter_none
```

3. Replace < by + in for loop's second expression

```
filter_none
edit
close
play_arrow
link
brightness_4
code

#include <stdio.h>
int main()
{
    int i, n = 20;
    for (i = 0; i + n; i--)
        printf("**");
    getchar();
    return 0;
}
chevron_right
filter_none
```

Let's extend the problem little.

Change/add only one character and print '**' exactly 21 times.

Solution: Put negation operator before i in for loop's second expression.

Explanation: Negation operator converts the number into its one's complement.

No.	One's complement
0 (00000..00)	-1 (1111..11)
-1 (11..1111)	0 (00..0000)
-2 (11..1110)	1 (00..0001)
-3 (11..1101)	2 (00..0010)
.....
-20 (11..01100)	19 (00..10011)

```
filter_none
edit
close
```

```
play_arrow
link
brightness_4
code
```

```
#include <stdio.h>
int main()
{
    int i, n = 20;
    for (i = 0; ~i < n; i--)
        printf("**");
    getchar();
    return 0;
}
```

chevron_right

filter_none

Please comment if you find more solutions of above problems.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up

Save

Recommended Posts:

- Print a character n times without using loop, recursion or goto in C++
- How to print N times without using loops or recursion ?
- How to print a number 100 times without using loop and recursion in C?
- Print a number 100 times without using loop, recursion and macro expansion in C?
- Check input character is alphabet, digit or special character
- Character arithmetic in C and C++
- Character Classification in C++ : ctype
- Type difference of character literals in C and C++
- Data type of character constants in C and C++
- Convert character array to string in C++
- Frequency of each character in a String using unordered_map in C++
- Differentiate printable and control character in C ?
- How to convert a single character to string in C++?
- getline() function and character array
- Storage of integer and character values in C

Improved By : prerna saini, SoumikMondal, SHUBHAMSINGH10

Article Tags :

C
C++

c-puzzle
Practice Tags :

C
CPP

thumb_up
2

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

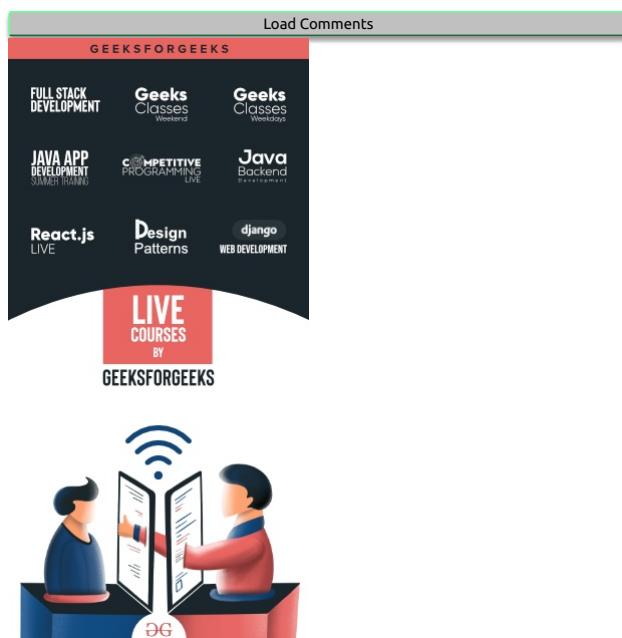
Previous

[first_page Condition To Print "HelloWord"](#)

Next

[last_page What is the best way in C to convert a number to a string?](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Program for Sum of the digits of a given number

Given a number, find sum of its digits.

Examples :

```
Input : n = 687
Output : 21

Input : n = 12
Output : 3
```

Recommended: Please solve it on "[PRACTICE](#)" first, before moving on to the solution.

General Algorithm for sum of digits in a given number:

1. Get the number
2. Declare a variable to store the sum and set it to 0
3. Repeat the next two steps till the number is not 0
4. Get the rightmost digit of the number with help of remainder '%' operator by dividing it with 10 and add it to sum.
5. Divide the number by 10 with help of '/' operator
6. Print or return the sum

Below are the solutions to get sum of the digits.

1. Iterative:

C++

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// C program to compute sum of digits in
// number.
# include<iostream>
using namespace std;
/* Function to get sum of digits */
class gfg
{
    public:
        int getSum(int n)
```

```
{  
int sum = 0;  
while (n != 0)  
{  
    sum = sum + n % 10;  
    n = n/10;  
}  
return sum;  
}  
};  
//driver code  
int main()  
{  
    gfg g;  
    int n = 687;  
    cout<< g.getSum(n);  
    return 0;  
}  
//This code is contributed by Soumik  
chevron_right  
filter_none
```

C

```
filter_none  
edit  
close  
play_arrow  
link  
brightness_4  
code  
// C program to compute sum of digits in  
// number.  
# include<stdio.h>  
  
/* Function to get sum of digits */  
int getSum(int n)  
{  
    int sum = 0;  
    while (n != 0)  
    {  
        sum = sum + n % 10;  
        n = n/10;  
    }  
    return sum;  
}  
  
int main()  
{  
    int n = 687;  
    printf(" %d ", getSum(n));  
    return 0;  
}  
chevron_right  
filter_none
```

Java

```
filter_none  
edit  
close  
play_arrow  
link  
brightness_4  
code  
// Java program to compute  
// sum of digits in number.  
import java.io.*;  
  
class GFG {  
  
    /* Function to get sum of digits */  
    static int getSum(int n)  
    {  
        int sum = 0;  
  
        while (n != 0)  
        {  
            sum = sum + n % 10;  
            n = n/10;  
        }  
  
        return sum;  
    }  
  
    // Driver program  
    public static void main(String[] args)  
    {
```

```
int n = 687;
System.out.println(getSum(n));
}

// This code is contributed by Gitanjali
chevron_right
filter_none
```

Python3

```
filter_none
edit
close
play_arrow
link
brightness_4
code

# Python 3 program to
# compute sum of digits in
# number.

# Function to get sum of digits
def getSum(n):

    sum = 0
    while (n != 0):

        sum = sum + int(n % 10)
        n = int(n/10)

    return sum

n = 687
print(getSum(n))
chevron_right
filter_none
```

C#

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// C# program to compute
// sum of digits in number.
using System;

class GFG
{
    /* Function to get sum of digits */
    static int getSum(int n)
    {
        int sum = 0;

        while (n != 0)
        {
            sum = sum + n % 10;
            n = n/10;
        }

        return sum;
    }

    // Driver program
    public static void Main()
    {
        int n = 687;
        Console.Write(getSum(n));
    }
}

// This code is contributed by Sam007
chevron_right
filter_none
```

PHP

```
filter_none
edit
close
play_arrow
link
brightness_4
```

```
code
<?php
// PHP Code to compute sum
// of digits in number.

// Function to get
// $sum of digits
function getsum($n)
{
    $sum = 0;
    while ($n != 0)
    {
        $sum = $sum + $n % 10;
        $n = $n/10;
    }
    return $sum;
}

// Driver Code
$n = 687;
$res = getsum($n);
echo("$res");

// This code is contributed by
// Smitha Dinesh Semwal.
?>
chevron_right
filter_none
```

Output :

21

How to compute in single line?

Below function has three lines instead of one line but it calculates sum in line. It can be made one line function if we pass pointer to sum.

C++

```
filter_none
edit
close
play_arrow
link
brightness_4
code

# include<iostream>
using namespace std;
/* Function to get sum of digits */
class gfg
{
public:
int getSum(int n)
{
    int sum;

    /* Single line that calculates sum */
    for (sum = 0; n > 0; sum += n % 10, n /= 10);

    return sum;
}
};

//driver code
int main()
{
gfg g;
int n = 687;
cout<< g.getSum(n);
return 0;
}
```

C

```
filter_none
edit
close
play_arrow
link
brightness_4
code

# include<stdio.h>
/* Function to get sum of digits */
int getSum(int n)
{
    int sum;

    /* Single line that calculates sum */
```

```
for (sum = 0; n > 0; sum += n % 10, n /= 10);

return sum;
}

int main()
{
    int n = 687;
    printf("%d ", getSum(n));
    return 0;
}
chevron_right
filter_none
```

Java

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// Java program to compute
// sum of digits in number.
import java.io.*;

class GFG {

    /* Function to get sum of digits */
    static int getSum(int n)
    {
        int sum;

        /* Single line that calculates sum */
        for (sum = 0; n > 0; sum += n % 10,
            n /= 10);

        return sum;
    }

    // Driver code
    public static void main(String[] args)
    {
        int n = 687;

        System.out.println(getSum(n));
    }
}

// This code is contributed by Gitanjali
chevron_right
filter_none
```

Python3

```
filter_none
edit
close
play_arrow
link
brightness_4
code

# Function to get sum of digits

def getSum(n):

    sum = 0

    # Single line that calculates sum
    while(n > 0):
        sum += int(n%10)
        n = int(n/10)

    return sum

# Driver code

n = 687
print(getSum(n))

# This code is contributed by
# Smitha Dinesh Semwal
chevron_right
filter_none
```

C#

```

filter_none
edit
close
play_arrow
link
brightness_4
code

// C# program to compute
// sum of digits in number.
using System;

class GFG
{
    static int getSum(int n)
    {
        int sum;

        /* Single line that calculates sum */
        for (sum = 0; n > 0; sum += n % 10,
            n /= 10);

        return sum;
    }

    // Driver code
    public static void Main()
    {
        int n = 687;
        Console.Write(getSum(n));
    }
}

// This code is contributed by Sam007
chevron_right
filter_none

```

PHP

```

filter_none
edit
close
play_arrow
link
brightness_4
code

<?php
// PHP Code for Sum the
// digits of a given number

// Function to get sum of digits
function getsum($n)
{

    // Single line that calculates $sum
    for ($sum = 0; $n > 0; $sum += $n % 10,
        $n /= 10);

    return $sum;
}

// Driver Code
$n = 687;
echo(getsum($n));

// This code is contributed by
// Smitha Dinesh Semwal.
?>
chevron_right
filter_none

```

Output :

21

2. Recursive

Thanks to ayesha for providing the below recursive solution.

C++

```

filter_none
edit
close
play_arrow
link
brightness_4
code

#include<iostream>
using namespace std;
class gfg
{
public: int sumDigits(int no)

```

```
{  
    return no == 0 ? 0 : no%10 + sumDigits(no/10) ;  
}  
};  
  
//driver code  
int main(void)  
{  
    GFG g;  
    cout<<g.sumDigits(687);  
    return 0;  
}  
chevron_right  
filter_none
```

C

```
filter_none  
edit  
close  
play_arrow  
link  
brightness_4  
code  
  
int sumDigits(int no)  
{  
    return no == 0 ? 0 : no%10 + sumDigits(no/10) ;  
}  
  
int main(void)  
{  
    printf("%d", sumDigits(687));  
    return 0;  
}  
chevron_right  
filter_none
```

Java

```
filter_none  
edit  
close  
play_arrow  
link  
brightness_4  
code  
  
// Java program to compute  
// sum of digits in number.  
import java.io.*;  
  
class GFG {  
  
    /* Function to get sum of digits */  
    static int sumDigits(int no)  
    {  
        return no == 0 ? 0 : no%10 +  
               sumDigits(no/10) ;  
    }  
  
    // Driver code  
    public static void main(String[] args)  
    {  
        System.out.println(sumDigits(687));  
    }  
}  
  
// This code is contributed by Gitanjali
```

```
chevron_right  
filter_none
```

Python3

```
filter_none  
edit  
close  
play_arrow  
link  
brightness_4  
code  
  
# Python program to compute  
# sum of digits in number.  
  
def sumDigits(no):  
    return 0 if no == 0 else int(no%10) + sumDigits(int(no/10))  
  
# Driver code  
print(sumDigits(687))
```

This code is contributed by

Smitha Dinesh Semwal

[chevron_right](#)

[filter_none](#)

C#

[filter_none](#)

[edit](#)

[close](#)

[play_arrow](#)

[link](#)

brightness_4

[code](#)

// C# program to compute

// sum of digits in number.

using System;

class GFG

{

/* Function to get sum of digits */

static int sumDigits(int no)

{

return no == 0 ? 0 : no % 10 +
sumDigits(no / 10);

}

// Driver code

public static void Main()

{

Console.WriteLine(sumDigits(687));

}

}

// This code is contributed by Sam007

[chevron_right](#)

[filter_none](#)

PHP

[filter_none](#)

[edit](#)

[close](#)

[play_arrow](#)

[link](#)

brightness_4

[code](#)

<?php

// PHP program to compute

// sum of digits in number.

function sumDigits(\$no)

{

return \$no == 0 ? 0 : \$no % 10 +
sumDigits(\$no / 10) ;

}

// Driver Code

echo sumDigits(687);

// This code is contributed by aj_36

?>

[chevron_right](#)

[filter_none](#)

Output :

21

Please write comments if you find the above codes/algorithms incorrect, or find better ways to solve the same problem.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes [arrow_drop_up](#)

Add your personal notes
here! (max 5000 chars)

[Save](#)

Recommended Posts:

- [C program to Count the digits of a number](#)
- [C Program to Print all digits of a given number](#)
- [Program to check if a given number is Lucky \(all digits are different\)](#)
- [Program to calculate product of digits of a number](#)
- [Write a program to reverse digits of a number](#)
- [Program to check if a number is divisible by sum of its digits](#)
- [Program to check if a number is divisible by any of its digits](#)
- [Count of integers in a range which have even number of odd digits and odd number of even digits](#)
- [Find smallest number with given number of digits and sum of digits under given constraints](#)

- Check whether product of digits at even places is divisible by sum of digits at odd place of a number
- Count of numbers between range having only non-zero digits whose sum of digits is N and number is divisible by M
- Number formed by deleting digits such that sum of the digits becomes even and the number odd
- Find the Largest number with given number of digits and sum of digits
- Find smallest number with given number of digits and sum of digits
- Minimum number of digits to be removed so that no two consecutive digits are same

Improved By : jit_t, SoumikMondal, RishabhPrabhu

Article Tags :

C
C++
Recursion
cpp-puzzle
number-digits
Practice Tags :
Recursion
C
CPP

10

To-do Done
1.5

Based on 55 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

first_page How will you print numbers from 1 to 100 without using loop?

Next

last_page Output of the program | Dereference, Reference, Dereference, Reference....

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Weekend Geeks Classes Weekdays

JAVA APP DEVELOPMENT SUMMER TRAINING COMPETITIVE PROGRAMMING LIVE Java Backend Development

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS

Most popular in C
Print all possible combinations of the string by replacing 's' with any other digit from the string
C Program to print every month by month calendar for a given year
ctype.h<<cctype>> library in C/C++ with Examples
Implicit Type Conversion in C with Examples
How to call function within function in C or C++

Most visited in C++
Vector of Vectors in C++ STL with Examples
C++ Tutorial
Which C++ libraries are useful for competitive programming?
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

What is the best way in C to convert a number to a string?

Solution: Use `sprintf()` function.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

#include<stdio.h>
int main()
{
    char result[50];
    float num = 23.34;
    sprintf(result, "%f", num);
    printf("\n The string for the num is %s", result);
    getchar();
}
chevron_right
filter_none
```

You can also write your own function using ASCII values of numbers.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes [arrow_drop_up](#)

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- Convert a floating point number to string in C
- `gcvf()` | Convert float value to string in C
- How to convert C style strings to std::string and vice versa?
- Converting string to number and vice-versa in C++
- C program to count number of vowels and consonants in a String
- Print substring of a given string without using any string function and loop in C
- Print all possible combinations of the string by replacing '\$' with any other digit from the string
- How to find length of a string without string.h and loop in C?
- Convert C/C++ code to assembly language
- Program to Convert Hexadecimal to Octal
- Convert C/C++ program to Preprocessor code
- C | String | Question 5
- C | String | Question 4
- C | String | Question 3
- C | String | Question 6

Article Tags :

C
C-Data Types

c-puzzle

Practice Tags :

C

3

To-do Done
2.1

Based on 18 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

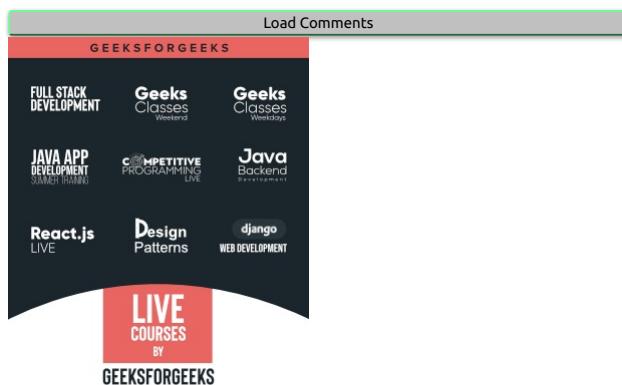
Previous

[first_page](#) Change/add only one character and print '*' exactly 20 times

Next

[last_page](#) How will you show memory representation of C variables?

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
Copy and Paste library in C/C++ with Examples
C program to display month by month calendar for a given year
Predefined Macros in C with Examples
Implicit Type Conversion in C with Examples

More related articles in C
Format specifiers in different Programming Languages
How to call function within function in C or C++
C program to print odd line contents of a file followed by even line content
Introduction to the C99 Programming Language : Part II
How to create GUI in C programming using GTK Toolkit

Program to compute Log n

Write a one line C function that calculates and returns $\log_2 n$. For example, if $n = 64$, then your function should return 6, and if $n = 129$, then your function should return 7.

Using Recursion

filter_none
edit
close
play_arrow
link
brightness_4
code

```
// C program to find log(n) using Recursion
#include <stdio.h>

unsigned int Log2n(unsigned int n)
{
    return (n > 1) ? 1 + Log2n(n / 2) : 0;
}

int main()
{
    unsigned int n = 32;
    printf("%u", Log2n(n));
    getchar();
    return 0;
}
```

chevron_right

filter_none

Java

filter_none
edit
close
play_arrow
link
brightness_4
code

```
// Java program to find log(n)
// using Recursion
class Gfg1
{

    static int Log2n(int n)
    {
        return (n > 1) ? 1 + Log2n(n / 2) : 0;
    }

    // Driver Code
    public static void main(String args[])
    {
        int n = 32;
        System.out.println(Log2n(n));
    }
}

// This code is contributed by Niraj_Pandey
```

chevron_right

filter_none

Python3

filter_none
edit
close
play_arrow
link
brightness_4
code

```
# Python 3 program to
# find log(n) using Recursion

def Log2n(n):

    return 1 + Log2n(n / 2) if (n > 1) else 0

# Driver code
n = 32
print(Log2n(n))

# This code is contributed by
# Smitha Dinesh Semwal
```

chevron_right

filter_none

C#

filter_none
edit
close
play_arrow

```
link  
brightness_4  
code  
  
// C# program to find log(n)  
// using Recursion  
using System;  
  
class GFG {  
  
    static int Log2n(int n)  
    {  
        return (n > 1) ? 1 +  
            Log2n(n / 2) : 0;  
    }  
  
    // Driver Code  
    public static void Main()  
    {  
        int n = 32;  
  
        Console.WriteLine(Log2n(n));  
    }  
}
```

// This code is contributed by
// nitin mittal.

[chevron_right](#)
[filter_none](#)

Output :

5

Time complexity: O(log n)

Auxiliary space: O(log n) if the stack size is considered during recursion otherwise O(1)

Using inbuilt log function

We can use the inbuilt function of standard library which is available in library.

C

```
filter_none  
edit  
close  
  
play_arrow  
  
link  
brightness_4  
code  
  
// C program to find log(n) using Inbuilt  
// function of <math.h> library  
#include <math.h>  
#include <stdio.h>  
int main()  
{  
    unsigned int n = 32;  
    printf("%d", (int)log2(n));  
    return 0;  
}
```

[chevron_right](#)
[filter_none](#)

Java

```
filter_none  
edit  
close  
  
play_arrow  
  
link  
brightness_4  
code  
  
// Java program to find log(n) using Inbuilt  
// function of java.util.Math library  
import java.util.*;  
  
class Gfg2  
{  
    public static void main(String args[])  
    {  
        int n = 32;  
        System.out.println((int)(Math.log(n) / Math.log(2)));  
    }  
}
```

// This code is contributed by Niraj_Pandey
[chevron_right](#)

[filter_none](#)

Output :

5

Time complexity: O(1)
Auxiliary space: O(1)

Let us try an extended version of the problem.

Write a one line function $\text{Log}(n, r)$ which returns $\lfloor \log_r n \rfloor$.

Using Recursion

C

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// C program to find log(n) on arbitrary base using Recursion
#include <stdio.h>

unsigned int Log(unsigned int n, unsigned int r)
{
    return (n > r - 1) ? 1 + Log(n / r, r) : 0;
}

int main()
{
    unsigned int n = 256;
    unsigned int r = 3;
    printf("%u", Log(n, r));
    return 0;
}
chevron_right
filter_none
```

Java

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// Java program to find log(n) on
// arbitrary base using Recursion
class Gfg3
{
    static int Log(int n, int r)
    {
        return (n > r - 1) ? 1 + Log(n / r, r) : 0;
    }

    // Driver Code
    public static void main(String args[])
    {
        int n = 256;
        int r = 3;
        System.out.println(Log(n, r));
    }
}

// This code is contributed by Niraj_Pandey
chevron_right
filter_none
```

Output :

5

Time complexity: O(log n)
Auxiliary space: O(log n) if the stack size is considered during recursion otherwise O(1)

Using inbuilt log function

We only need to use logarithm property to find the value of $\log(n)$ on arbitrary base r . i.e., $\log_r n = \frac{\log_k(n)}{\log_k(r)}$ where k can be any anything, which for standard log functions are either **e** or **10**

C

```
filter_none
edit
close
play_arrow
link
brightness_4
```

code

```
// C program to find log(n) on arbitrary base
// using log() function of maths library
#include <math.h>
#include <stdio.h>

unsigned int Logn(unsigned int n, unsigned int r)
{
    return log(n) / log(r);
}

int main()
{
    unsigned int n = 256;
    unsigned int r = 3;
    printf("%u", Logn(n, r));

    return 0;
}
```

chevron_right

filter_none

Java

filter_none

edit

close

play_arrow

link

brightness_4

code

```
// Java program to find log(n) on arbitrary base
// using log() function of java.util.Math library
import java.util.*;
```

class Gfg4 {

```
    public static void main(String args[])
    {
        int n = 256;
        int r = 3;
        System.out.println((int)(Math.log(n) / Math.log(r)));
    }
}
```

// This code is contributed by Niraj_Pandey

chevron_right

filter_none

Output :

5

Time complexity: O(1)

Auxiliary space: O(1)

This article is contributed by Shubham Bansal. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer Placement 100 for details

My Personal Notes arrow_drop_up

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- [C Program to Compute Quotient and Remainder](#)
- [C program to print a string without any quote \(single or double\) in the program](#)
- [Hello World Program : First program while learning Programming](#)
- [C program to detect tokens in a C program](#)
- [Output of C Program | Set 29](#)
- [Write a URL in a C++ program](#)
- [Program to compare m^n and n^m](#)
- [How does a C program executes?](#)
- [How to compile 32-bit program on 64-bit gcc in C and C++](#)
- [Output of C++ Program | Set 14](#)
- [Output of C++ Program | Set 10](#)
- [Program for n-th even number](#)
- [Output of C++ Program | Set 16](#)
- [Output of C++ Program | Set 18](#)
- [Program to validate an IP address](#)

Improved By : [Niraj_Pandey](#), nitin mittal

Article Tags :

C

C++

Practice Tags :

C

CPP

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

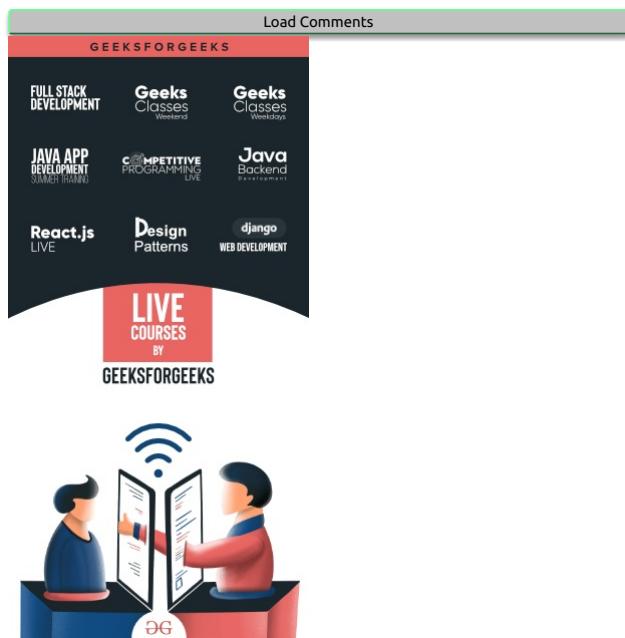
Previous

[first_page](#) C++ default constructor | Built-in types

Next

[last_page](#) Advanced C++ | Conversion Operators

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C

- Print all possible combinations of the string by replacing '\$' with any other digit from the string
- C program to display month by month calendar for a given year
- ctype.h(<cctype>) library in C/C++ with Examples
- Implicit Type Conversion in C with Examples
- How to call function within function in C or C++

Most visited in C++

- Vector of Vectors in C++ STL with Examples
- C++ Tutorial
- Which C++ libraries are useful for competitive programming?
- Array of Vectors in C++ STL
- Map of Vectors in C++ STL with Examples



- 5th Floor, A-118,
- Sector-136, Noida, Uttar Pradesh - 201305
- feedback@geeksforgeeks.org

▪ COMPANY

- About Us
- Careers
- Privacy Policy
- Contact Us

▪ LEARN

- Algorithms
- Data Structures
- Languages
- CS Subjects
- Video Tutorials

▪ PRACTICE

- Courses
- Company-wise
- Topic-wise
- How to begin?

▪ CONTRIBUTE

- Write an Article
- Write Interview Experience
- Internships
- Videos

Print “Even” or “Odd” without using conditional statement

Write a C/C++ program that accepts a number from the user and prints “Even” if the entered number is even and prints “Odd” if the number is odd. You are not allowed to use any comparison (==, <, >, ...etc) or conditional (if, else, switch, ternary operator,..etc) statement.

Method 1

Below is a tricky code can be used to print “Even” or “Odd” accordingly.

C++

```
filter_none
edit
close

play_arrow
link
brightness_4
code

#include<iostream>
#include<conio.h>

using namespace std;

int main()
{
    char arr[2][5] = {"Even", "Odd"};
    int no;
    cout << "Enter a number: ";
    cin >> no;
    cout << arr[no%2];
    getch();
    return 0;
}
chevron_right
filter_none
```

Python3

```
filter_none
edit
close

play_arrow
link
brightness_4
code

arr = ["Even", "Odd"]
print ("Enter the number")
no = input()
print (arr[int(no) % 2])
chevron_right
filter_none
```

PHP

```
filter_none
edit
close

play_arrow
link
brightness_4
code

<?php
$arr = ["Even", "Odd"];
$input = 5;
echo ($arr[$input % 2]);

// This code is contributed
// by Aman ojha
?>
chevron_right
filter_none
```

Method 2

Below is another tricky code can be used to print “Even” or “Odd” accordingly. Thanks to [student](#) for suggesting this method.

```
filter_none
edit
close

play_arrow
link
brightness_4
code

#include<stdio.h>
int main()
{
    int no;
    printf("Enter a no: ");
    scanf("%d", &no);
```

```
(no & 1 && printf("odd"))|| printf("even");
return 0;
}
```

chevron_right

filter_none

Please write comments if you find the above code incorrect, or find better ways to solve the same problem

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes *arrow_drop_up*

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- Implementing ternary operator without any conditional statement
- Conditional or Ternary Operator (?) in C/C++
- Set a variable without using Arithmetic, Relational or Conditional Operator
- Conditionally assign a value without using conditional and arithmetic operators
- Switch Statement in C/C++
- C/C++ if statement with Examples
- goto statement in C/C++
- Continue Statement in C/C++
- C/C++ if else statement with Examples
- Break Statement in C/C++
- return statement in C/C++ with Examples
- Nested switch statement in C++
- Interesting facts about switch statement in C
- return statement vs exit() in main()
- Programming puzzle (Assign value without any control statement)

Improved By : Prateek Bajaj, Aman ojha

Article Tags :

C

C++

Practice Tags :

C

CPP

thumb_up

7

To-do Done
2.3

Based on 21 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) Does C++ compiler create default constructor when we write our own?

Next

[last_page](#) Can we call an undeclared function in C++?

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Weekend Geeks Classes Weekdays

JAVA APP DEVELOPMENT COMPETITIVE PROGRAMMING LIVE Java Backend Evolution

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS



Most popular in C

Print all possible combinations of the string by replacing '\$' with any other digit from the string

C program to display month by month calendar for a given year

cyclicALLYCyclic Library in C/C++ with Examples

Implicit Type Conversion with Examples

How to call function within function in C or C++

Most visited in C++

Vectors of Vectors in C++ STL with Examples

C++ Tutorial

Which C++ libraries are useful for competitive programming?

Array of Vectors in C++ STL

Map of Vectors in C++ STL with Examples

How will you print numbers from 1 to 100 without using loop?

If we take a look at this problem carefully, we can see that the idea of "loop" is to track some counter value e.g. "i=0" till "i <= 100". So if we aren't allowed to use loop, how else can be track something in C language!

Well, one possibility is the use of 'recursion' provided we use the terminating condition carefully. Here is a solution that prints numbers using recursion.

C++

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// C++ program to How will you print
// numbers from 1 to 100 without using loop?
#include <iostream>
using namespace std;

class gfg
{

    // Prints numbers from 1 to n
public:
void printNos(unsigned int n)
{
    if(n > 0)
    {
        printNos(n - 1);
        cout << n << " ";
    }
    return;
}
};

// Driver code
int main()
{
    gfg g;
    g.printNos(100);
    return 0;
}

// This code is contributed by SoM15242
chevron_right
filter_none
```

C

```
filter_none
edit
close
play_arrow
link
brightness_4
code

#include <stdio.h>

// Prints numbers from 1 to n
void printNos(unsigned int n)
{
    if(n > 0)
    {
        printNos(n - 1);
        printf("%d ", n);
    }
    return;
}

// Driver code
int main()
{
    printNos(100);
    getchar();
    return 0;
}
```

Java

```
filter_none
edit
close
play_arrow
```

```
link  
brightness_4  
code  
  
import java.io.*;  
import java.util.*;  
import java.text.*;  
import java.math.*;  
import java.util.regex.*;  
  
class GFG  
{  
    // Prints numbers from 1 to n  
    static void printNos(int n)  
    {  
        if(n > 0)  
        {  
            printNos(n - 1);  
            System.out.print(n + " ");  
        }  
        return;  
    }  
  
    // Driver Code  
    public static void main(String[] args)  
    {  
        printNos(100);  
    }  
}  
  
// This code is contributed by Manish_100
```

[chevron_right](#)

[filter_none](#)

Python3

```
filter_none  
edit  
close  
play_arrow  
link  
brightness_4  
code  
  
# Python3 program to Print  
# numbers from 1 to n  
  
def printNos(n):  
    if n > 0:  
        printNos(n - 1)  
        print(n, end = ' ')  
  
# Driver code  
printNos(100)
```

[# This code is contributed by Smitha Dinesh Semwal](#)

[chevron_right](#)

[filter_none](#)

C#

```
filter_none  
edit  
close  
play_arrow  
link  
brightness_4  
code  
  
// C# code for print numbers from  
// 1 to 100 without using loop  
using System;  
  
class GFG  
{  
  
    // Prints numbers from 1 to n  
    static void printNos(int n)  
    {  
        if(n > 0)  
        {  
            printNos(n - 1);  
            Console.Write(n + " ");  
        }  
        return;  
    }  
  
    // Driver Code  
    public static void Main()  
    {  
        printNos(100);  
    }  
}
```

```
}
```

```
// This code is contributed by Ajit
chevron_right
```

```
filter_none
```

PHP

```
filter_none
edit
close

play_arrow

link
brightness_4
code

<?php
// PHP program print numbers
// from 1 to 100 without
// using loop

// Prints numbers from 1 to n
function printNos($n)
{
    if($n > 0)
    {
        printNos($n - 1);
        echo $n, " ";
    }
    return;
}

// Driver code
printNos(100);
```

```
// This code is contributed by vt_m
?>
chevron_right
```

```
filter_none
```

Output :

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
16 17 18 19 20 21 22 23 24 25 26 27
28 29 30 31 32 33 34 35 36 37 38 39
40 41 42 43 44 45 46 47 48 49 50 51
52 53 54 55 56 57 58 59 60 61 62 63
64 65 66 67 68 69 70 71 72 73 74 75
76 77 78 79 80 81 82 83 84 85 86 87
88 89 90 91 92 93 94 95 96 97 98 99
100
```

Time Complexity : O(n)

Now try writing a program that does the same but without any “if” construct.

Hint — use some operator which can be used instead of “if”.

Please note that recursion technique is good but every call to the function creates one “stack-frame” in program stack. So if there’s constraint to the limited memory and we need to print large set of numbers, “recursion” might not be a good idea. So what could be the other alternative?

Another alternative is “goto” statement. Though use of “goto” is not suggestible as a general programming practice as “goto” statement changes the normal program execution sequence yet in some cases, use of “goto” is the best working solution.

So please give a try printing numbers from 1 to 100 with “goto” statement. You can use [GfG IDE!](#)

[Print 1 to 100 in C++, without loop and recursion](#)

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up

Save

Recommended Posts:

- [How will you print numbers from 1 to 100 without using loop? | Set-2](#)
- [Print 1 to 100 in C++, without loop and recursion](#)
- [Print a pattern without using any loop](#)
- [Print pattern using only one loop | Set 1 \(Using setw\)](#)
- [How to print a number 100 times without using loop and recursion in C?](#)
- [Print a character n times without using loop, recursion or goto in C++](#)
- [Print a number 100 times without using loop, recursion and macro expansion in C?](#)
- [Write a C program to print “GfG” repeatedly without using loop, recursion and any control structure?](#)
- [Print all distinct integers that can be formed by K numbers from a given array of N numbers](#)
- [Print substring of a given string without using any string function and loop in C](#)
- [C++ program to print all Even and Odd numbers from 1 to N](#)
- [Print prime numbers in a given range using C++ STL](#)
- [Program to print first N Stepping numbers](#)
- [Print N-bit binary numbers having more 1's than 0's in all prefixes](#)
- [Program to print a pattern of numbers](#)

Improved By : [Manish_100](#), [vt_m](#), [jit_t](#), [SoumikMondal](#)

Article Tags :

C
C++
Recursion
cpp-puzzle
Practice Tags :
Recursion
C
CPP

thumb_up
4

To-do Done
1.3

Based on 46 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

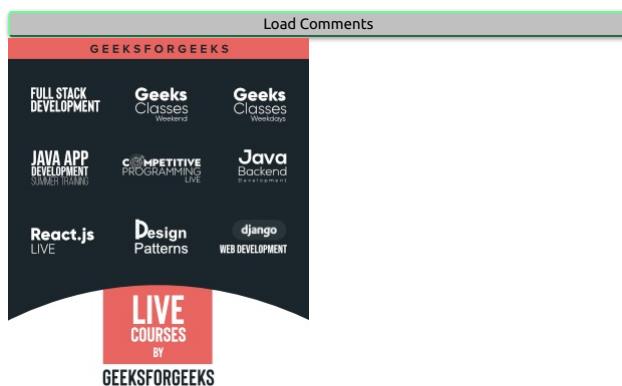
Previous

first_page

Next

last_page Program for Sum of the digits of a given number

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.



Most popular in C
Print all possible combinations of the string by replacing 'S' with any other digit from the string
ctype.h & cctype.h library in C with Examples
C program to display month by month calendar for a given year
Predefined Macros in C with Examples
Implicit Type Conversion in C with Examples

Most visited in C++
Vector of Vectors in C++ STL with Examples
C++ Tutorial
Which C++ libraries are useful for competitive programming?
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

Program for Sum of the digits of a given number

Given a number, find sum of its digits.

Examples :

```
Input : n = 687
Output : 21

Input : n = 12
Output : 3
```

Recommended: Please solve it on "*PRACTICE* " first, before moving on to the solution.

General Algorithm for sum of digits in a given number:

1. Get the number
2. Declare a variable to store the sum and set it to 0
3. Repeat the next two steps till the number is not 0
4. Get the rightmost digit of the number with help of remainder '%' operator by dividing it with 10 and add it to sum.
5. Divide the number by 10 with help of '/' operator
6. Print or return the sum

Below are the solutions to get sum of the digits.

1. Iterative:

C++

```
filter_none
edit
close
play_arrow
link
brightness_4
```

code

```
// C program to compute sum of digits in
// number.

# include<iostream>
using namespace std;
/* Function to get sum of digits */
class gfg
{
    public:
        int getSum(int n)
    {
        int sum = 0;
        while (n != 0)
        {
            sum = sum + n % 10;
            n = n/10;
        }
        return sum;
    }
};

//driver code
int main()
{
    gfg g;
    int n = 687;
    cout<< g.getSum(n);
    return 0;
}

//This code is contributed by Soumik
chevron_right
```

filter_none

C

filter_none
edit
close
play_arrow
link
brightness_4
code

```
// C program to compute sum of digits in
// number.

# include<stdio.h>

/* Function to get sum of digits */
int getSum(int n)
{
    int sum = 0;
    while (n != 0)
    {
        sum = sum + n % 10;
        n = n/10;
    }
    return sum;
}

int main()
{
    int n = 687;
    printf("%d ", getSum(n));
    return 0;
}
```

chevron_right

filter_none

Java

filter_none
edit
close
play_arrow
link
brightness_4
code

```
// Java program to compute
// sum of digits in number.

import java.io.*;

class GFG {

    /* Function to get sum of digits */
    static int getSum(int n)
    {
        int sum = 0;

        while (n != 0)
        {
```

```

        sum = sum + n % 10;
        n = n/10;
    }

    return sum;
}

// Driver program
public static void main(String[] args)
{
    int n = 687;

    System.out.println(getSum(n));
}
}

// This code is contributed by Gitanjali
chevron_right
filter_none

```

Python3

```

filter_none
edit
close
play_arrow
link
brightness_4
code

# Python 3 program to
# compute sum of digits in
# number.

# Function to get sum of digits
def getSum(n):

    sum = 0
    while (n != 0):

        sum = sum + int(n % 10)
        n = int(n/10)

    return sum

n = 687
print(getSum(n))
chevron_right
filter_none

```

C#

```

filter_none
edit
close
play_arrow
link
brightness_4
code

// C# program to compute
// sum of digits in number.
using System;

class GFG
{
    /* Function to get sum of digits */
    static int getSum(int n)
    {
        int sum = 0;

        while (n != 0)
        {
            sum = sum + n % 10;
            n = n/10;
        }
    }

    return sum;
}

// Driver program
public static void Main()
{
    int n = 687;
    Console.WriteLine(getSum(n));
}
}

// This code is contributed by Sam007
chevron_right

```

[filter_none](#)

PHP

```
filter_none
edit
close

play_arrow
link
brightness_4
code

<?php
// PHP Code to compute sum
// of digits in number.

// Function to get
// $sum of digits
function getsum($n)
{
    $sum = 0;
    while ($n != 0)
    {
        $sum = $sum + $n % 10;
        $n = $n/10;
    }
    return $sum;
}

// Driver Code
$n = 687;
$res = getsum($n);
echo("$res");

// This code is contributed by
// Smitha Dinesh Semwal.
?>
```

[chevron_right](#)

[filter_none](#)

[Output :](#)

(21)

How to compute in single line?

Below function has three lines instead of one line but it calculates sum in line. It can be made one line function if we pass pointer to sum.

C++

```
filter_none
edit
close

play_arrow
link
brightness_4
code

# include<iostream>
using namespace std;
/* Function to get sum of digits */
class gfg
{
public:
int getSum(int n)
{
    int sum;

    /* Single line that calculates sum */
    for (sum = 0; n > 0; sum += n % 10, n /= 10);

    return sum;
}
};

//driver code
int main()
{
gfg g;
int n = 687;
cout<< g.getSum(n);
return 0;
}
```

[chevron_right](#)

[filter_none](#)

C

```
filter_none
edit
close

play_arrow
```

link
brightness_4
code

```
# include<stdio.h>
/* Function to get sum of digits */
int getSum(int n)
{
    int sum;

    /* Single line that calculates sum */
    for (sum = 0; n > 0; sum += n % 10, n /= 10);

    return sum;
}

int main()
{
    int n = 687;
    printf("%d ", getSum(n));
    return 0;
}
```

chevron_right
filter_none

Java

filter_none
edit
close
play_arrow
link
brightness_4
code

```
// Java program to compute
// sum of digits in number.
import java.io.*;

class GFG {

    /* Function to get sum of digits */
    static int getSum(int n)
    {
        int sum;

        /* Single line that calculates sum */
        for (sum = 0; n > 0; sum += n % 10,
                         n /= 10);

        return sum;
    }

    // Driver code
    public static void main(String[] args)
    {
        int n = 687;

        System.out.println(getSum(n));
    }
}

// This code is contributed by Gitanjali
```

chevron_right
filter_none

Python3

filter_none
edit
close
play_arrow
link
brightness_4
code

```
# Function to get sum of digits

def getSum(n):

    sum = 0

    # Single line that calculates sum
    while(n > 0):
        sum += int(n%10)
        n = int(n/10)

    return sum

# Driver code
```

```
n = 687
print(getSum(n))
```

This code is contributed by

Smitha Dinesh Semwal

 chevron_right

 filter_none

C#

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// C# program to compute
// sum of digits in number.
using System;

class GFG
{
    static int getSum(int n)
    {
        int sum;

        /* Single line that calculates sum */
        for (sum = 0; n > 0; sum += n % 10,
            n /= 10);

        return sum;
    }
}
```

// Driver code

public static void Main()

```
{
    int n = 687;
    Console.Write(getSum(n));
}
```

 chevron_right

 filter_none

PHP

```
filter_none
edit
close
play_arrow
link
brightness_4
code

<?php
// PHP Code for Sum the
// digits of a given number

// Function to get sum of digits
function getsum($n)
{

    // Single line that calculates $sum
    for ($sum = 0; $n > 0; $sum += $n % 10,
        $n /= 10);

    return $sum;
}
```

// Driver Code

\$n = 687;

echo(getsum(\$n));

 chevron_right

 filter_none

Output :

21

2. Recursive

Thanks to ayesha for providing the below recursive solution.

C++

```
filter_none
edit
```

close
play_arrow
link
brightness_4
code

```
#include<iostream>
using namespace std;
class gfg
{
public: int sumDigits(int no)
{
    return no == 0 ? 0 : no%10 + sumDigits(no/10) ;
}
};

//driver code
int main(void)
{
    gfg g;
    cout<.sumDigits(687);
    return 0;
}
```

chevron_right
filter_none

C

filter_none
edit
close
play_arrow
link
brightness_4
code

```
int sumDigits(int no)
{
    return no == 0 ? 0 : no%10 + sumDigits(no/10) ;
}

int main(void)
{
    printf("%d", sumDigits(687));
    return 0;
}
```

chevron_right
filter_none

Java

filter_none
edit
close
play_arrow
link
brightness_4
code

```
// Java program to compute
// sum of digits in number.
import java.io.*;

class GFG {

    /* Function to get sum of digits */
    static int sumDigits(int no)
    {
        return no == 0 ? 0 : no%10 +
            sumDigits(no/10) ;
    }

    // Driver code
    public static void main(String[] args)
    {
        System.out.println(sumDigits(687));
    }
}

// This code is contributed by Gitanjali
```

chevron_right
filter_none

Python3

filter_none
edit
close
play_arrow
link

```
brightness_4
code

# Python program to compute
# sum of digits in number.

def sumDigits(no):
    return 0 if no == 0 else int(no%10) + sumDigits(int(no/10))

# Driver code
print(sumDigits(687))

# This code is contributed by
# Smitha Dinesh Semwal
chevron_right
filter_none
```

C#

```
filter_none
edit
close

play_arrow

link
brightness_4
code

// C# program to compute
// sum of digits in number.

using System;

class GFG
{
    /* Function to get sum of digits */
    static int sumDigits(int no)
    {
        return no == 0 ? 0 : no % 10 +
            sumDigits(no / 10);
    }

    // Driver code
    public static void Main()
    {
        Console.WriteLine(sumDigits(687));
    }
}
```

```
// This code is contributed by Sam007
chevron_right
```

```
filter_none
```

PHP

```
filter_none
edit
close

play_arrow

link
brightness_4
code

<?php
// PHP program to compute
// sum of digits in number.
function sumDigits($no)
{
    return $no == 0 ? 0 : $no % 10 +
        sumDigits($no / 10) ;
}

// Driver Code
echo sumDigits(687);

// This code is contributed by aj_36
?>
chevron_right
```

```
filter_none
```

Output :

(21)

Please write comments if you find the above codes/algorithms incorrect, or find better ways to solve the same problem.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up

Add your personal notes
here! (max 5000 chars)

Recommended Posts:

- C program to Count the digits of a number
- C Program to Print all digits of a given number
- Program to check if a given number is Lucky (all digits are different)
- Program to calculate product of digits of a number
- Write a program to reverse digits of a number
- Program to check if a number is divisible by sum of its digits
- Program to check if a number is divisible by any of its digits
- Count of integers in a range which have even number of odd digits and odd number of even digits
- Find smallest number with given number of digits and sum of digits under given constraints
- Check whether product of digits at even places is divisible by sum of digits at odd place of a number
- Count of numbers between range having only non-zero digits whose sum of digits is N and number is divisible by M
- Number formed by deleting digits such that sum of the digits becomes even and the number odd
- Find the Largest number with given number of digits and sum of digits
- Find smallest number with given number of digits and sum of digits
- Minimum number of digits to be removed so that no two consecutive digits are same

Improved By : jit_t, SoumikMondal, RishabhPrabhu

Article Tags :

C
C++
Recursion
cpp-puzzle
number-digits
Practice Tags :
Recursion
C
CPP

10

To-do Done
1.5

Based on 55 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

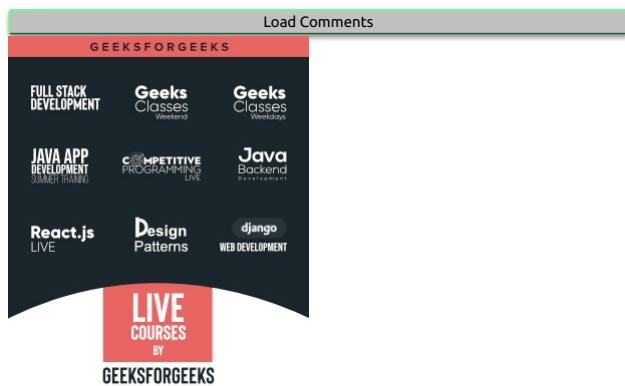
Previous

[first_page](#) How will you print numbers from 1 to 100 without using loop?

Next

[last_page](#) Output of the program | Dereference, Reference, Dereference, Reference....

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C
Print all possible combinations of the string by replacing 'S' with any other digit from the string
C Program to display month wise calendar for a given year
Copy & Paste C++ library in C/C++ with Examples
Implicit Type Conversion in C with Examples
How to call function within function in C or C++

Most visited in C++
Vector of Vectors in C++ STL with Examples
C++ Tutorial
Which C++ libraries are useful for competitive programming?
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

Write a C program to print “Geeks for Geeks” without using a semicolon

First of all we have to understand how printf() function works.

Prototype of printf() function is:

```
int printf( const char *format , ... )
```

Parameter

- **format:** This is a string that contains a text to be written to stdout.
- **Additional arguments:** ... (Three dots are called ellipses) which indicates the variable number of arguments depending upon the format string.

printf() returns the total number of characters written to stdout. Therefore it can be used as a condition check in an if condition, while condition, switch case and Macros.

Let's see each of these conditions one by one.

1. Using if condition:

```
filter_none
edit
close

play_arrow
link
brightness_4
code

#include<stdio.h>
int main()
{
    if (printf("Geeks for Geeks") )
    {
    }
}
chevron_right
filter_none
```

2. Using while condition:

```
filter_none
edit
close

play_arrow
link
brightness_4
code

#include<stdio.h>
int main(){
    while (!printf( "Geeks for Geeks" ))
    {
    }
}
chevron_right
filter_none
```

3. Using switch case:

```
filter_none
edit
close

play_arrow
link
brightness_4
code

#include<stdio.h>
int main(){
    switch (printf("Geeks for Geeks"))
    {
    }
}
chevron_right
filter_none
```

4. Using Macros:

```
filter_none
edit
close

play_arrow
link
brightness_4
code

#include<stdio.h>
#define PRINT printf("Geeks for Geeks")
int main()
{
    if (PRINT)
    {
    }
}
chevron_right
filter_none
```

Output: Geeks for Geeks

One trivial extension of the above problem: Write a C program to print ";" without using a semicolon

```
filter_none
edit
close

play_arrow
link
brightness_4
code

#include<stdio.h>
int main()
{
    // ASCII value of ; is 59
    if (printf("%c", 59))
    {
    }
}
```

chevron_right

filter_none

Output: :

This blog is contributed by [Shubham Bansal](#). If you like GeeksforGeeks and would like to contribute, you can also write an article using [contribute.geeksforgeeks.org](#) or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes *arrow_drop_up*

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- C Program to print numbers from 1 to N without using semicolon?
- How to print a semicolon(;) without using semicolon in C/C++?
- Print Hello World without semicolon in C/C++
- Write a C program to print "GFG" repeatedly without using loop, recursion and any control structure?
- Write a C macro PRINT(x) which prints x
- Role of SemiColon in various Programming Languages
- Write a C program that won't compile in C++
- C program to write an image in PGM format
- Write a program that produces different results in C and C++
- Write a C program that does not terminate when Ctrl+C is pressed
- Write a C program that displays contents of a given file like 'more' utility in Linux
- C program to print a string without any quote (single or double) in the program
- Program to print last 10 lines
- Program to print Happy Birthday
- C Program to print environment variables

Article Tags :

C

c-puzzle

Practice Tags :

C

thumb_up

14

To-do Done
1.5

Based on 46 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) Output of the program | Dereference, Reference, Dereference, Reference....

Next

[last_page](#) Write a one line C function to round floating point numbers

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

[GEEKSFORGEEKS](#)

FULL STACK DEVELOPMENT
Geeks Classes Wednesday
Geeks Classes Wednesday

JAVA APP DEVELOPMENT SUMMER TRAINING
COMPETITIVE PROGRAMMING LIVE
Java Backend Development

React.js LIVE
Design Patterns
django WEB DEVELOPMENT

LIVE COURSES BY **GEEKSFORGEEKS**



Most popular in C

[Program to calculate Electricity Bill](#)

[Program to print half Diamond star pattern](#)

[C/C++ program for calling main\(\) in main\(\)](#)

[exit\(0\) vs exit\(1\) in C/C++ with Examples](#)

[C/C++ #include directive with Examples](#)

More related articles in C

[Output of C programs | Set 66 \(Accessing Memory Locations\)](#)

[Difference between Type Casting and Type Conversion](#)

[getch\(\) function in C with Examples](#)

[Character arrays in C with Examples](#)

[Jagged Array or Array of Arrays in C with Examples](#)

Write a one line C function to round floating point numbers

Algorithm: roundNo(num)

1. If num is positive then add 0.5.
2. Else subtract 0.5.
3. Type cast the result to int and return.

Example:

num = 1.67, (int) num + 0.5 = (int)2.17 = 2
num = -1.67, (int) num - 0.5 = -(int)2.17 = -2

Implementation:

```
filter_none
edit
close
play_arrow
link
brightness_4
code

/* Program for rounding floating point numbers */
#include<stdio.h>

int roundNo(float num)
{
    return num < 0 ? num - 0.5 : num + 0.5;
}

int main()
{
    printf("%d", roundNo(-1.777));
    getchar();
    return 0;
}
chevron_right
filter_none
```

Output: -2

Time complexity: O(1)

Space complexity: O(1)

Now try rounding for a given precision. i.e., if given precision is 2 then function should return 1.63 for 1.63322 and -1.63 for 1.6332.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes 

Add your personal notes here! (max 5000 chars)

 Save

Recommended Posts:

- Precision of floating point numbers in C++ (floor(), ceil(), trunc(), round() and setprecision())
- C Program to Multiply two Floating Point Numbers
- Problem in comparing Floating point numbers and how to compare them correctly?
- Convert a floating point number to string in C
- How to count set bits in a floating point number in C?
- Floating Point Operations & Associativity in C, C++ and Java
- Rounding Floating Point Number To two Decimal Places in C and C++
- Write one line functions for strcat() and strcmp()
- LEX program to add line numbers to a given file
- C program to print odd line contents of a File followed by even line content
- getopt() function in C to parse command line arguments
- sizeof() for Floating Constant in C
- How to write your own header file in C?
- Write your own memcpy() and memmove()
- When should we write our own copy constructor?

Article Tags :

C
C-Data Types
c-puzzle

Practice Tags :

C

 1

To-do Done
1.5

Based on 19 vote(s)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) Write a C program to print "Geeks for Geeks" without using a semicolon

Next

[last_page](#) Implement Your Own sizeof

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

 Load Comments



Most popular in C
Program to calculate Electricity Bill
C/C++ program for calling main() in main()
Program to print half Diamond star pattern
exit(0) vs exit(1) in C/C++ with Examples
C/C++ #include directive with Examples

More related articles in C
Output of C programs | Set 66 (Accessing Memory Locations)
getchar() function with Examples
Difference between Type Casting and Type Conversion
Chain of Pointers in C with Examples
ctype.h <ctype> library in C/C++ with Examples

Implement Your Own sizeof

Here is an implementation.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

#include<stdio.h>
#define my_sizeof(type) (char *)(&type+1)-(char *)(&type)
int main()
{
    double x;
    printf("%ld", my_sizeof(x));
    getchar();
    return 0;
}
```

Type is like a local variable to the macro. &type gives the address of the variable (double x) declared in the program and incrementing it with 1 gives the address where the next variable of type x can be stored (here `addr_of(x)` + 8, for the size of a double is 8B).

The difference gives the result that how many variables of type of x can be stored in that amount of memory which will obviously be 1 for the type x (for incrementing it with 1 and taking the difference is what we've done). Typecasting it into char* and taking the difference will tell us how many variables of type char can be stored in the given memory space (the difference). Since each char requires the only 1B of memory, therefore (amount of memory)/1 will give the number of bytes between two successive memory locations of the type of the variable passed on to the macro and hence the amount of memory that the variable of type x requires.

But you won't be able to pass any literal to this macro and know their size.

You can also implement using the function instead of a macro, but function implementation cannot be done in C as C doesn't support function overloading and `sizeof()` is supposed to receive parameters of all data types.

Note that the above implementation assumes that the size of character is one byte.

Time Complexity: O(1)

Space Complexity: O(1)

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes [arrow_drop_up](#)

Add your personal notes here! (max 5000 chars)

[Save](#)

Recommended Posts:

- Is sizeof for a struct equal to the sum of sizeof of each member?
- sizeof operator in C
- Why does sizeof(x++) not increment x in C?
- sizeof() for Floating Constant in C
- Anything written in sizeof() is never executed in C
- Do not use sizeof for array parameters
- Operands for sizeof operator
- How to find size of array in C/C++ without using sizeof?
- Difference between strlen() and sizeof() for string in C
- Implement your own itoa()
- How to implement Min Heap using STL?
- How to implement our own Vector Class in C++?
- Using class to implement Vector Quantities in C++

- How to implement user defined Shared Pointers in C++
- Implement your own tail (Read last n lines of a huge file)

Improved By : romeosaurabh

Article Tags :

C
C++
c-puzzle
Practice Tags :
C
CPP

thumb_up
8

To-do Done
3.1

Based on 43 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

first_page Write a one line C function to round floating point numbers

Next

last_page Condition To Print "HelloWord"

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

The image shows the top navigation bar of the GeeksforGeeks website. It features a red header with the text 'Load Comments' and 'GEEKSFORGEEKS'. Below the header, there are several course categories: 'FULL STACK DEVELOPMENT', 'Geeks Classes Weekend', 'Geeks Classes Weekdays', 'JAVA APP DEVELOPMENT SUMMER TRAINING', 'COMPETITIVE PROGRAMMING LIVE', 'Java Backend Development', 'React.js LIVE', 'Design Patterns', and 'django WEB DEVELOPMENT'. A large red banner in the center says 'LIVE COURSES BY GEEKSFORGEEKS'. Below the banner, there are two cartoon figures looking at a screen displaying a Wi-Fi signal icon. At the bottom left, there's a list of most popular and visited C++ topics.

Most popular in C
Program to calculate Electricity Bill
Program to print half Diamond star pattern
C/C++ program for calling main() in main()
exit(0) vs exit(1) in C/C++ with Examples
C/C++ #include directive with Examples

Most visited in C++
Vector of Vectors in C++ STL with Examples
C++ Tutorials
Which C++ libraries are useful for competitive programming?
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

How to count set bits in a floating point number in C?

Given a [floating point](#) number, write a function to count set bits in its binary representation.

For example, floating point representation of 0.15625 has 6 set bits (See [this](#)). A typical C compiler uses single precision floating point format.

We can use the idea discussed [here](#). The idea is to take address of the given floating point number in a pointer variable, typecast the pointer to char * type and process individual bytes one by one. We can easily count set bits in a char using the techniques discussed [here](#).

Following is C implementation of the above idea.

```
filter_none
edit
close
play_arrow
link
brightness_4
code
#include <stdio.h>

// A utility function to count set bits in a char.
// Refer http://goo.gl/eHF6Y8 for details of this function.
unsigned int countSetBitsChar(char n)
{
    unsigned int count = 0;
    while (n)
    {
        n &= (n-1);
        count++;
    }
}
```

```

}

return count;
}

// Returns set bits in binary representation of x
unsigned int countSetBitsFloat(float x)
{
    // Count number of chars (or bytes) in binary representation of float
    unsigned int n = sizeof(float)/sizeof(char);

    // typcast address of x to a char pointer
    char *ptr = (char *)&x;

    int count = 0;    // To store the result
    for (int i = 0; i < n; i++)
    {
        count += countSetBitsChar(*ptr);
        ptr++;
    }
    return count;
}

```

// Driver program to test above function

```

int main()
{
    float x = 0.15625;
    printf ("Binary representation of %f has %u set bits ", x,
           countSetBitsFloat(x));
    return 0;
}

```

chevron_right

filter_none

Output:

Binary representation of 0.156250 has 6 set bits

This article is contributed by **Vineet Gupta**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes *arrow_drop_up*

Add your personal notes
here! (max 5000 chars)

Save

Recommended Posts:

- Convert a floating point number to string in C
- Rounding Floating Point Number To two Decimal Places in C and C++
- Can we use % operator on floating point numbers?
- Floating Point Operations & Associativity in C, C++ and Java
- C Program to Multiply two Floating Point Numbers
- Problem in comparing Floating point numbers and how to compare them correctly?
- Write a one line C function to round floating point numbers
- C++ Floating Point Manipulation (fmod(), remainder(), remquo() ... in cmath)
- Precision of floating point numbers in C++ (floor(), ceil(), trunc(), round() and setprecision())
- Find a number containing N - 1 set bits at even positions from the right
- Maximize the number by rearranging bits
- Program to find the Nth natural number with exactly two bits set
- sizeof() for Floating Constant in C
- Rotation of a point about another point in C++
- Count the number of 1's and 0's in a binary array using STL in C++ ?

Article Tags :

C

C++

cpp-puzzle

Practice Tags :

C

CPP

thumb_up

1

To-do Done
3.9

Based on 11 vote(s)

Basic **Easy** **Medium** **Hard** **Expert**

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) C++ | Templates | Question 10

Next

[last_page](#) C | Variable Declaration and Scope | Question 1

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments



Most popular in C
[Program to calculate Electricity Bill](#)
[Program to print half Diamond star pattern](#)
[C/C++ program for calling main\(\) in main\(\)](#)
[exit\(0\) vs exit\(1\) in C/C++ with Examples](#)
[C/C++ #include directive with Examples](#)

Most visited in C++
[Vector of Vectors in C++ STL with Examples](#)
[C++ STL](#)
[Which C++ libraries are useful for competitive programming?](#)
[Array of Vectors in C++ STL](#)
[Map of Vectors in C++ STL with Examples](#)

How to change the output of printf() in main() ?

Consider the following program. Change the program so that the output of printf is always 10. It is not allowed to change main(). Only fun() can be changed.

```
filter_none
edit
close
play_arrow
link
brightness_4
code
void fun()
{
    // Add something here so that the printf in main prints 10
}

int main()
{
    int i = 10;
    fun();
    i = 20;
    printf("%d", i);
    return 0;
}
```

We can use **Macro Arguments** to change the output of printf.

```
filter_none
edit
close
play_arrow
link
brightness_4
code
#include <stdio.h>

void fun()
{
    #define printf(x, y) printf(x, 10);
}

int main()
{
    int i = 10;
    fun();
    i = 20;
    printf("%d", i);
    return 0;
}
```

Output:

My Personal Notes [arrow_drop_up](#)

Add your personal notes
here! (max 5000 chars)

Save

Recommended Posts:

- Nested printf (printf inside printf) in C
- Is it fine to write "void main()" or "main()" in C/C++?
- Difference between "int main()" and "int main(void)" in C/C++?
- C/C++ program for calling main() in main()
- What is use of %n in printf()?
- How to print % using printf()?
- Use of & in scanf() but not in printf()
- Cin-Cout vs Scanf-Printf
- Execution of printf with ++ operators
- Passing NULL to printf in C
- puts() vs printf() for printing a string
- Return values of printf() and scanf() in C/C++
- What is the difference between printf, sprintf and fprintf?
- Can main() be overloaded in C++?
- What does main() return in C and C++?

Article Tags :

C
C++
cpp-input-output
cpp-puzzle

Practice Tags :

C
CPP

thumb_up

11

To-do Done
2.8

Based on 32 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) C++ | Inheritance | Question 5

Next

[last_page](#) Implement your own itoa()

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Weekend Geeks Classes Weekdays

JAVA APP DEVELOPMENT COMPETITIVE PROGRAMMING LIVE Java Backend Development

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS

Most popular in C
Program to calculate Electricity Bill
Program to print half Diamond star pattern
C/C++ program for calling main() in main()
exit(0) vs exit(1) in C/C++ with Examples
C/C++ #include directive with Examples

Most visited in C++
Vector of Vectors in C++ STL with Examples
C++ Tutorial
Why STL libraries are useful for competitive programming
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

How to find length of a string without string.h and loop in C?

Find the length of a string without using any loops and string.h in C. Your program is supposed to behave in following way:

```
Enter a string: GeeksforGeeks (Say user enters GeeksforGeeks)
Entered string is: GeeksforGeeks
Length is: 13
```

You may assume that the length of entered string is always less than 100.

The following is solution.

```
filter_none
edit
close
play_arrow
link
brightness_4
code
#include <stdio.h>

int main()
{
    char str[100];
    printf("Enter a string: \n");
    gets(str);
    printf("Entered string is:%s\n", str);
    printf( "\rLength is: %d",strlen(str));

    return 0;
}
```

chevron_right

filter_none

Output:

```
Enter a string: GeeksforGeeks
Entered string is: GeeksforGeeks
Length is: 13
```

The idea is to use return values of printf() and gets().

gets() returns the entered string.

printf() returns the number of characters successfully written on output.

In the above program, gets() returns the entered string. We print the length using the first printf. The second printf() calls gets() and prints the entered string using returned value of gets(), it also prints 20 extra characters for printing "Entered string is: " and "\n". That is why we subtract 20 from the returned value of second printf and get the length.

This article is contributed by **Umesh Sharma**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Another way of finding the length of a string without using string.h or loops is Recursion.

The following program does the work of finding a length of a string using recursion.

```
filter_none
edit
close
play_arrow
link
brightness_4
code
#include <stdio.h>

void LengthofString(int n,char *string)
{
    if(string[n] == '\0')
    {
        printf("%i",n);
        return;
    }

    LengthofString(n+1,string);
    //printf("%c",string[n]);
}

int main()
{
    char string[100];
    printf("Give a string : \n");
    scanf("%s",string);
    printf("Entered string is:%s\n", string);
    LengthofString(0,string);

    return 0;
}
```

The Function LengthofString calls itself until the character of string is'nt a null character it calls itself, when it calls itself it increases the value of the variable 'n' which stores number of times the function has been called and when it encounters the null character the function prints the value of 'n' and returns back in the same direction in which it was executed.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- [C program to find the length of a string](#)
- [Print substring of a given string without using any string function and loop in C](#)
- [The length of a string using pointers](#)
- [Counts of distinct consecutive sub-string of length two using C++ STL](#)
- [C/C++ For loop with Examples](#)

- C/C++ while loop with Examples
- C/C++ do while loop with Examples
- Difference between while and do-while loop in C, C++, Java
- How will you print numbers from 1 to 100 without using loop? | Set-2
- Print 1 to 100 in C++, without loop and recursion
- A nested loop puzzle
- How will you print numbers from 1 to 100 without using loop?
- How to concatenate two integer arrays without using loop in C ?
- What happens if loop till Maximum of Signed and Unsigned in C/C++?
- How to print a number 100 times without using loop and recursion in C?

Improved By : Zeeking99

Article Tags :

C
C-String
Practice Tags :
C

thumb_up
1

To-do Done
2.4

Based on 10 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) How to print "GeeksforGeeks" with empty main() in C, C++ and Java?

Next

[last_page](#) C | String | Question 14

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Most popular in C

Print all possible combinations of the string by replacing '\$' with any other digit from the string

Predefined Macros in C with Examples

Format specifiers in different Programming Languages

How to call function within function in C or C++

C program to print odd line contents of a file followed by even line content

Implement your own itoa()

[itoa](#) function converts integer into null-terminated string. It can convert negative numbers too. The standard definition of itoa function is give below:-

```
filter_none
edit
close
play_arrow
link
brightness_4
code
char* itoa(int num, char* buffer, int base)
chevron_right
filter_none
```

The third parameter base specify the conversion base. For example:- if base is 2, then it will convert the integer into its binary compatible string or if base is 16, then it will create hexadecimal converted string form of integer number.

If base is 10 and value is negative, the resulting string is preceded with a minus sign (-). With any other base, value is always considered unsigned.

Reference: <http://www.cplusplus.com/reference/cstdlib/itoa/?kw=itoa>

Examples:

```
itoa(1567, str, 10) should return string "1567"
itoa(-1567, str, 10) should return string "-1567"
itoa(1567, str, 2) should return string "11000011111"
itoa(1567, str, 16) should return string "61f"
```

Individual digits of the given number must be processed and their corresponding characters must be put in the given string. Using repeated division by the given base, we get individual digits from least significant to most significant digit. But in the output, these digits are needed in reverse order. Therefore, we reverse the string obtained after repeated division and return it.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

/* A C++ program to implement itoa() */
#include <iostream>
using namespace std;

/* A utility function to reverse a string */
void reverse(char str[], int length)
{
    int start = 0;
    int end = length -1;
    while (start < end)
    {
        swap(*(str+start), *(str+end));
        start++;
        end--;
    }
}

// Implementation of itoa()
char* itoa(int num, char* str, int base)
{
    int i = 0;
    bool isNegative = false;

    /* Handle 0 explicitly, otherwise empty string is printed for 0 */
    if (num == 0)
    {
        str[i++] = '0';
        str[i] = '\0';
        return str;
    }

    // In standard itoa(), negative numbers are handled only with
    // base 10. Otherwise numbers are considered unsigned.
    if (num < 0 && base == 10)
    {
        isNegative = true;
        num = -num;
    }

    // Process individual digits
    while (num != 0)
    {
        int rem = num % base;
        str[i++] = (rem > 9)? (rem-10) + 'a' : rem + '0';
        num = num/base;
    }

    // If number is negative, append '-'
    if (isNegative)
        str[i++] = '-';

    str[i] = '\0'; // Append string terminator

    // Reverse the string
    reverse(str, i);

    return str;
}

// Driver program to test implementation of itoa()
int main()
{
    char str[100];
    cout << "Base:10 " << itoa(1567, str, 10) << endl;
    cout << "Base:10 " << itoa(-1567, str, 10) << endl;
    cout << "Base:2 " << itoa(1567, str, 2) << endl;
    cout << "Base:8 " << itoa(1567, str, 8) << endl;
    cout << "Base:16 " << itoa(1567, str, 16) << endl;
    return 0;
}
```

chevron_right

filter_none

Output:

```
Base:10 1567
Base:10 -1567
Base:2 11000011111
Base:8 3037
```

This article is contributed by **Neha Mahajan**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes

[Save](#)

Recommended Posts:

- [Implement Your Own sizeof](#)
- [How to implement Min Heap using STL?](#)
- [How to implement our own Vector Class in C++?](#)
- [Using class to implement Vector Quantities in C++](#)
- [How to implement user defined Shared Pointers in C++](#)
- [Implement your own tail \(Read last n lines of a huge file\)](#)
- [Rust vs C++: Will Rust Replace C++ in Future ?](#)
- [Priority queue of pairs in C++ with ordering by first and second element](#)
- [Implementation of lower_bound\(\) and upper_bound\(\) in Vector of Pairs in C++](#)
- [Generating RGBA portable graphic images through C++](#)
- [std::basic_istream::ignore in C++ with Examples](#)
- [std::basic_istream::getline in C++ with Examples](#)
- [Format specifiers in different Programming Languages](#)
- [C program to find square root of a given number](#)

Article Tags :

[C](#)
[C++](#)
[CPP-Library](#)

Practice Tags :

[C](#)
[CPP](#)

 [thumb_up](#)

1

To-do Done
3.9

Based on 17 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) [How to change the output of printf\(\) in main\(\) ?](#)

Next

[last_page](#) [C++ | Inheritance | Question 6](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

[Load Comments](#)

GEEKSFORGEEKS

FULL STACK DEVELOPMENT **Geeks Classes** Wednesday **Geeks Classes** Wednesday

JAVA APP DEVELOPMENT COMPETITIVE PROGRAMMING LIVE Java Backend Development

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS



Most popular in C

[Print all possible combinations of the string by replacing '\\$' with any other digit from the string](#)

[How to implement Min Heap using STL in C or C++](#)

[Format specifiers in different Programming Languages](#)

[Predefined Macros in C with Examples](#)

[C program to print odd line contents of a File followed by even line content](#)

Most visited in C++

[Vector of Vectors in C++ STL with Examples](#)

[C++ Tutorial](#)

[Which C++ libraries are useful for competitive programming?](#)

[Array of Vectors in C++ STL with Examples](#)

[Map of Vectors in C++ STL with Examples](#)

Write a C program that does not terminate when Ctrl+C is pressed

Write a C program that doesn't terminate when Ctrl+C is pressed. It prints a message "Cannot be terminated using Ctrl+c" and continues execution.

We can use [signal handling](#) in C for this. When [Ctrl+C](#) is pressed, SIGINT signal is generated, we can catch this signal and run our defined signal handler. C standard defines following 6 signals in [signal.h](#) header file.

SIGABRT - abnormal termination.
SIGFPE - floating point exception.
SIGILL - invalid instruction.
SIGINT - interactive attention request sent to the program.
SIGSEGV - invalid memory access.
SIGTERM - termination request sent to the program.

Additional signals are specified Unix and Unix-like operating systems (such as Linux) defines more than 15 additional signals. See http://en.wikipedia.org/wiki/Unix_signal#POSIX_signals
The standard C library function `signal()` can be used to set up a handler for any of the above signals.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

/* A C program that does not terminate when Ctrl+C is pressed */
#include <stdio.h>
#include <signal.h>

/* Signal Handler for SIGINT */
void sigintHandler(int sig_num)
{
    /* Reset handler to catch SIGINT next time.
       Refer http://en.cppreference.com/w/c/program/signal */
    signal(SIGINT, sigintHandler);
    printf("\n Cannot be terminated using Ctrl+C \n");
    fflush(stdout);
}

int main ()
{
    /* Set the SIGINT (Ctrl-C) signal handler to sigintHandler
       Refer http://en.cppreference.com/w/c/program/signal */
    signal(SIGINT, sigintHandler);

    /* Infinite loop */
    while(1)
    {
    }
    return 0;
}
chevron_right
```

filter_none

Output: When Ctrl+C was pressed two times

```
Cannot be terminated using Ctrl+C
Cannot be terminated using Ctrl+C
```

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes [arrow_drop_up](#)

Add your personal notes
here! (max 5000 chars)

[Save](#)

Recommended Posts:

- [C program that does not suspend when Ctrl+Z is pressed](#)
- [Write a URL in a C++ program](#)
- [Write a C program that won't compile in C++](#)
- [C program to write an image in PGM format](#)
- [Write a program that produces different results in C and C++](#)
- [Write a C program that displays contents of a given file like 'more' utility in Linux](#)
- [Write a C program to print "GFG" repeatedly without using loop, recursion and any control structure?](#)
- [Write a C program to print "Geeks for Geeks" without using a semicolon](#)
- [How to write your own header file in C?](#)
- [When should we write our own assignment operator in C++?](#)
- [Write your own memcpy\(\) and memmove\(\)](#)
- [When should we write our own copy constructor?](#)
- [Read/Write structure to a file in C](#)
- [How to write a running C code without main\(\)?](#)
- [Write a C macro PRINT\(x\) which prints x](#)

Improved By : [nidhi_biet](#)

Article Tags :

C
C++
cpp-puzzle
system-programming
Practice Tags :
C
CPP

[thumb_up](#)
Be the First to upvote.

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page C | Structure & Union | Question 6](#)

Next

[last_page C | Structure & Union | Question 7](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS



FULL STACK
DEVELOPMENT



Geeks
Classes
Weekend



Geeks
Classes
Weekdays



JAVA APP
DEVELOPMENT



COMPETITIVE
PROGRAMMING



Java
Backend
Development



React.js
LIVE



Design
Patterns



django
WEB DEVELOPMENT



LIVE
COURSES
BY
GEEKSFORGEEKS



Most popular in C
[Print all possible combinations of the string by replacing '\\$' with any other digit from the string](#)
[Predefined Macros in C with Examples](#)
[Format specifiers in different Programming Languages](#)
[C program to print odd line contents of a file followed by even line content](#)
[Introduction to the C99 Programming Language : Part II](#)

Most visited in C++
[Vector vs Vectors in C++ STL with Examples](#)
[C++ Tutorials](#)
[Which C++ libraries are useful for competitive programming?](#)
[Array of Vectors in C++ STL](#)
[Map of Vectors in C++ STL with Examples](#)

How to measure time taken by a function in C?

To calculate time taken by a process, we can use `clock()` function which is available `time.h`. We can call the clock function at the beginning and end of the code for which we measure time, subtract the values, and then divide by `CLOCKS_PER_SEC` (the number of clock ticks per second) to get processor time, like following.

```
#include <time.h>

clock_t start, end;
double cpu_time_used;

start = clock();
... /* Do the work. */
end = clock();
cpu_time_used = ((double) (end - start)) / CLOCKS_PER_SEC;
```

Following is a sample C program where we measure time taken by `fun()`. The function `fun()` waits for enter key press to terminate.

```
filter_none
edit
close

play_arrow

link
brightness_4
code

/* Program to demonstrate time taken by function fun() */
#include <stdio.h>
#include <time.h>

// A function that terminates when enter key is pressed
void fun()
{
    printf("fun() starts \n");
    printf("Press enter to stop fun \n");
    while(1)
    {
        if (getchar())
            break;
    }
    printf("fun() ends \n");
}

// The main program calls fun() and measures time taken by fun()
int main()
{
    // Calculate the time taken by fun()
    clock_t t;
    t = clock();
    fun();
    t = clock() - t;
```

```
double time_taken = ((double)t)/CLOCKS_PER_SEC; // in seconds  
printf("fun() took %f seconds to execute \n", time_taken);  
return 0;  
}
```

chevron_right

filter_none

Output: The following output is obtained after waiting for around 4 seconds and then hitting enter key.

```
fun() starts  
Press enter to stop fun  
  
fun() ends  
fun() took 4.017000 seconds to execute
```

How to find time taken by a command/program on Linux Shell?

References:

http://www.gnu.org/software/libc/manual/html_node/CPU-Time.html
<http://www.cplusplus.com/reference/ctime/clock/?kw=clock>

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes *arrow_drop_up*

Add your personal notes
here! (max 5000 chars)

Save

Recommended Posts:

- Measure execution time of a function in C++
- Measure execution time with high precision in C/C++
- time() function in C
- time.h localtime() function in C with Examples
- Time delay in C
- Print system time in C++ (3 different ways)
- time.h header file in C with Examples
- RTTI (Run-time type Information) in C++
- C++ Program to print current Day, Date and Time
- C program to print digital clock with current time
- Amadeus Labs R & D | On Campus (freshers) | Full time+Internship
- Cvent Interview Experience (On campus for Internship and Full Time)
- Program to convert time from 12 hour to 24 hour format
- Function Overloading vs Function Overriding in C++
- How to call function within function in C or C++

Article Tags :

C

C++

cpp-puzzle

Practice Tags :

C

CPP

thumb_up

7

To-do Done

2

Based on 14 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

first_page C | Operators | Question 8

Next

last_page C | String | Question 1

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
Predefined Macros in C with Examples
How to call function within function in C or C++
C program to print odd line contents of a File followed by even line content
Introduction to the C99 Programming Language : Part II

Most visited in C++
Vector of Vectors in C++ STL with Examples
C++ STL
Which C++ libraries are useful for competitive programming?
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

Print a long int in C using putchar() only

Write a C function `print(n)` that takes a long int number `n` as argument, and prints it on console. The only allowed library function is `putchar()`, no other function like `itoa()` or `printf()` is allowed. Use of loops is also not allowed.

We strongly recommend to minimize the browser and try this yourself first.

This is a simple trick question. Since `putchar()` prints a character, we need to call `putchar()` for all digits. Recursion can always be used to replace iteration, so using recursion we can print all digits one by one. Now the question is `putchar()` prints a character, how to print digits using `putchar()`. We need to convert every digit to its corresponding ASCII value, this can be done by using ASCII value of '0'. Following is complete C program.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

/* C program to print a long int number
   using putchar() only*/
#include <stdio.h>

void print(long n)
{
    // If number is smaller than 0, put a - sign
    // and change number to positive
    if (n < 0) {
        putchar('-');
        n = -n;
    }

    // Remove the last digit and recur
    if (n/10)
        print(n/10);

    // Print the last digit
    putchar(n%10 + '0');
}

// Driver program to test above function
int main()
{
    long int n = 12045;
    print(n);
    return 0;
}
```

chevron_right

filter_none

Output:

12045

One important thing to note is the sequence of `putchar()` and recursive call `print(n/10)`. Since the digits should be printed left to right, the recursive call must appear before `putchar()` (The rightmost digit should be printed at the end, all other digits must be printed before it).

This article is contributed by **Abhay Rathi**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- putchar() function in C
- Is there any need of "long" data type in C and C++?
- Working of Keyword long in C programming
- How to write long strings in Multi-lines C/C++?
- Write your own strlen() for a long string padded with '\0's
- How to print a variable name in C?
- How to print % using printf()?
- Print 1 2 3 infinitely using threads in C
- How to Read and Print an Integer value in C
- Print Hello World without semicolon in C/C++
- Print calendar for a given year in C++
- Print 1 to 100 in C++, without loop and recursion
- Print "Even" or "Odd" without using conditional statement
- How will you print numbers from 1 to 100 without using loop? | Set-2
- Program to print last 10 lines

Improved By : Prashant_Singh

Article Tags :

C
C-Library
C-String
Practice Tags :
C

Be the First to upvote.

To-do Done
2

Based on 15 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) Bitwise Operators in C/C++

Next

[last_page](#) Integer Promotions in C

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes (Weekend) Geeks Classes (Weekdays)

JAVA APP DEVELOPMENT SUMMER TRAINING COMPETITIVE PROGRAMMING LIVE Java Backend Development

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS

Most popular in C

[Print all possible combinations of the string by replacing '\\$' with any other digit from the string](#)
[Print Macro Examples](#)
[Format specifier in different Programming Languages](#)
[How to call function within function in C or C++](#)
[C program to print odd line contents of a File followed by even line content](#)

More related articles in C

[Introduction to the C99 Programming Language : Part II](#)
[C program to find square root of a given number](#)
[Problems in comparing Floating point numbers and how to compare them correctly?](#)
[Features of C Programming Language](#)
[Pointer Expressions in C with Examples](#)

Convert a floating point number to string in C

Write a C function ftoa() that converts a given floating-point number or a double to a string. Use of standard library functions for direct conversion is not allowed. The following is prototype of ftoa(). The article provides insight of conversion of C double to string.

```
ftoa(n, res, afterpoint)
n      --> Input Number
res[]    --> Array where output string to be stored
afterpoint --> Number of digits to be considered after the point.
```

Example:

- ftoa(1.555, str, 2) should store "1.55" in res.

■ `ftoa(1.555, str, 0)` should store "1" in `res`.

We strongly recommend to minimize the browser and try this yourself first.

A simple way is to use `sprintf()`, but use of standard library functions for direct conversion is not allowed.

Approach: The idea is to separate integral and fractional parts and convert them to strings separately. Following are the detailed steps.

1. Extract integer part from floating-point or double number.
2. First, convert integer part to the string.
3. Extract fraction part by exacted integer part from `n`.
4. If `d` is non-zero, then do the following.
 - a. Convert fraction part to an integer by multiplying it with `pow(10, d)`
 - b. Convert the integer value to string and append to the result.

Following is C implementation of the above approach.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// C program for implementation of ftoa()
#include <math.h>
#include <stdio.h>

// Reverses a string 'str' of length 'len'
void reverse(char* str, int len)
{
    int i = 0, j = len - 1, temp;
    while (i < j) {
        temp = str[i];
        str[i] = str[j];
        str[j] = temp;
        i++;
        j--;
    }
}

// Converts a given integer x to string str[].
// d is the number of digits required in the output.
// If d is more than the number of digits in x,
// then 0s are added at the beginning.
int intToStr(int x, char str[], int d)
{
    int i = 0;
    while (x) {
        str[i++] = (x % 10) + '0';
        x = x / 10;
    }

    // If number of digits required is more, then
    // add 0s at the beginning
    while (i < d)
        str[i++] = '0';

    reverse(str, i);
    str[i] = '\0';
    return i;
}

// Converts a floating-point/double number to a string.
void ftoa(float n, char* res, int afterpoint)
{
    // Extract integer part
    int ipart = (int)n;

    // Extract floating part
    float fpart = n - (float)ipart;

    // convert integer part to string
    int i = intToStr(ipart, res, 0);

    // check for display option after point
    if (afterpoint != 0) {
        res[i] = '.'; // add dot

        // Get the value of fraction part upto given no.
        // of points after dot. The third parameter
        // is needed to handle cases like 233.007
        fpart = fpart * pow(10, afterpoint);

        intToStr((int)fpart, res + i + 1, afterpoint);
    }
}

// Driver program to test above function
int main()
{
```

```
char res[20];
float n = 233.007;
ftoa(n, res, 4);
printf("%s\n", res);
return 0;
}
```

chevron_right

filter_none

Output:

{233.0070}

Note: The program performs similar operation if instead of float, a double type is taken.

This article is contributed by Jayasantosh Samal. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes 

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- [How to count set bits in a floating point number in C?](#)
- [Rounding Floating Point Number To two Decimal Places in C and C++](#)
- [Can we use % operator on floating point numbers?](#)
- [C Program to Multiply two Floating Point Numbers](#)
- [Floating Point Operations & Associativity in C, C++ and Java](#)
- [C++ Floating Point Manipulation \(fmod\(\), remainder\(\), remquo\(\) ... in cmath\)](#)
- [Write a one line C function to round floating point numbers](#)
- [Problem in comparing Floating point numbers and how to compare them correctly?](#)
- [Precision of floating point numbers in C++ \(floor\(\), ceil\(\), trunc\(\), round\(\) and setprecision\(\)\)](#)
- [What is the best way in C to convert a number to a string?](#)
- [Convert string to char array in C++](#)
- [Convert character array to string in C++](#)
- [C++ Program to Convert String to Integer](#)
- [How to convert a single character to string in C++?](#)
- [gcvt\(\) | Convert float value to string in C](#)

Improved By : shubham_singh

Article Tags :

C
C++
cpp-data-types
cpp-string

Practice Tags :
C
CPP

 4

To-do Done
3.4

Based on 13 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) C | File Handling | Question 5

Next

[last_page](#) How to dynamically allocate a 2D array in C?

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

[Load Comments](#)



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
Format specifiers in different Programming Languages
Predefined Macros in C with Examples
C program to print odd line contents of a File followed by even line content
C program to find square root of a given number

Most visited in C++
Vector of Vectors in C++ STL with Examples
C++ STL
Which C++ libraries are useful for competitive programming?
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

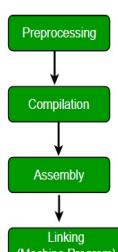
How to write a running C code without main()?

Write a C language code that prints **GeeksforGeeks** without any main function.

Logically it's seems impossible to write a C program without using a main() function. Since every program must have a main() function because:-

- It's an entry point of every C/C++ program.
- All Predefined and User-defined Functions are called directly or indirectly through the main.

Therefore we will use preprocessor(a program which processes the source code before compilation) directive #define with arguments to give an impression that the program runs without main. But in reality it runs with a hidden main function. Let's see how the preprocessor doing their job:-



Hence it can be solved in following ways:-

1. Using a macro that defines main

```

filter_none
edit
close

play_arrow

link
brightness_4
code

#include<stdio.h>
#define fun main
int fun(void)
{
    printf("Geeksforgeeks");
    return 0;
}
chevron_right
filter_none
  
```

2. Using Token-Pasting Operator

The above solution has word 'main' in it. If we are not allowed to even write main, we can use token-pasting operator (see [this](#) for details)

```

filter_none
edit
close

play_arrow

link
brightness_4
code

#include<stdio.h>
#define fun m##a##i##n
  
```

```
int fun()
{
    printf("Geeksforgeeks");
    return 0;
}
chevron_right
filter_none
Output: Geeksforgeeks
```

3. Using Argumented Macro

```
filter_none
edit
close

play_arrow
link
brightness_4
code

#include<stdio.h>
#define begin(m,a,i,n) m##a##i##n
#define start begin(m,a,i,n)

void start() {
    printf("Geeksforgeeks");
}
```

```
chevron_right
filter_none
Output: Geeksforgeeks
```

4. Modify the entry point during compilation

```
filter_none
edit
close

play_arrow
link
brightness_4
code

#include<stdio.h>
#include<stdlib.h>

// entry point function
int nomain();

void _start(){

    // calling entry point
    nomain();
    exit(0);
}

int nomain()
{
    puts("Geeksforgeeks");
    return 0;
}
```

```
chevron_right
filter_none
Output:
Geeksforgeeks
```

Compilation using command :

gcc filename.c -nostartfiles
(nostartfiles option tells the compiler to avoid standard linking)

Explanation:

Under normal compilation the body of _start() will contain a function call to main() [this _start() will be appended to every code during normal compilation], so if that main() definition is not present it will result in error like "In function `_start':
.text+0x20: undefined reference to `main'.

In the above code what we have done is that we have defined our own _start() and defined our own entry point i.e nomain()

- This method is contributed by Aravind Alapati

Refer Executing main() in C - behind the scene for another solution.

References:

Macros and Preprocessors in C

This article is contributed by **Abhay Rathi** and improved by **Shubham Bansal**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes

Add your personal notes here! (max 5000 chars)

Recommended Posts:

- Is it fine to write "void main()" or "main()" in C/C++?
- Difference between "int main()" and "int main(void)" in C/C++?
- C/C++ program for calling main() in main()
- What does main() return in C and C++?

- Can main() be overloaded in C++?
- Functions that are executed before and after main() in C
- Executing main() in C/C++ - behind the scene
- How to change the output of printf() in main() ?
- return statement vs exit() in main()
- How to print "GeeksforGeeks" with empty main() in C, C++ and Java?
- How to write your own header file in C?
- When should we write our own assignment operator in C++?
- Write your own memcpy() and memmove()
- Write a C program that won't compile in C++
- When should we write our own copy constructor?

Improved By : ManasChhabra2

Article Tags :

C

c-puzzle

Practice Tags :

C



8

To-do Done
1.5

Based on 17 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

first_page Can virtual functions be inlined?

Next

last_page C Program to print contents of file

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

The image shows the top navigation bar of the GeeksforGeeks website. It includes links for 'FULL STACK DEVELOPMENT', 'JAVA APP DEVELOPMENT SUMMER TRAINING', 'React.js LIVE', 'Geeks Classes Weekend', 'COMPETITIVE PROGRAMMING LIVE', 'Design Patterns', 'Java Backend Development', and 'django WEB DEVELOPMENT'. Below the navigation is a red banner for 'LIVE COURSES BY GEEKSFORGEEKS' featuring two stylized figures interacting with tablets.

Most popular in C

- Print all possible combinations of the string by replacing '\$' with any other digit from the string
- Format specifiers in different Programming Languages
- C program to find square root of a given number
- Predefined Macros in C with Examples
- How to call function within function in C or C++

More related articles in C

- C program to print odd line contents of a File followed by even line content
- Introduction to the C99 Programming Language : Part II
- Features of C Programming Language
- Problem in comparing Floating point numbers and how to compare them correctly?
- <cfloat> float.h in C/C++ with Examples

GeeksforGeeks
A computer science portal for geeks

- 5th Floor, A-118,
- Sector-136, Noida, Uttar Pradesh - 201305
- feedback@geeksforgeeks.org
- **COMPANY**
 - About Us
 - Careers
 - Privacy Policy
 - Contact Us
- **LEARN**
 - Algorithms

- Data Structures
- Languages
- CS Subjects
- Video Tutorials
- **PRACTICE**
- Courses
- Company-wise
- Topic-wise
- How to begin?
- **CONTRIBUTE**
- Write an Article
- Write Interview Experience
- Internships
- Videos
-

@geeksforgeeks,

Write your own memcpy() and memmove()

The `memcpy` function is used to copy a block of data from a source address to a destination address. Below is its prototype.

```
void * memcpy(void * destination, const void * source, size_t num);
```

The idea is to simply typecast given addresses to `char *`(char takes 1 byte). Then one by one copy data from source to destination. Below is implementation of this idea.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// A C implementation of memcpy()
#include<stdio.h>
#include<string.h>

void myMemcpy(void *dest, void *src, size_t n)
{
    // Typecast src and dest addresses to (char *)
    char *csrc = (char *)src;
    char *cdest = (char *)dest;

    // Copy contents of src[] to dest[]
    for (int i=0; i<n; i++)
        cdest[i] = csrc[i];
}

// Driver program
int main()
{
    char csrc[] = "GeeksforGeeks";
    char cdest[100];
    myMemcpy(cdest, csrc, strlen(csrc)+1);
    printf("Copied string is %s", cdest);

    int isrc[] = {10, 20, 30, 40, 50};
    int n = sizeof(isrc)/sizeof(isrc[0]);
    int idest[n], i;
    myMemcpy(idest, isrc, sizeof(isrc));
    printf("\nCopied array is ");
    for (i=0; i<n; i++)
        printf("%d ", idest[i]);
    return 0;
}
chevron_right
```

filter_none

Output:

```
Copied string is GeeksforGeeks
Copied array is 10 20 30 40 50
```

What is memmove()?

`memmove()` is similar to `memcpy()` as it also copies data from a source to destination. `memcpy()` leads to problems when source and destination addresses overlap as `memcpy()` simply copies data one by one from one location to another. For example consider below program.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// Sample program to show that memcpy() can loose data.
#include <stdio.h>
#include <string.h>
int main()
{
    char csrc[100] = "Geeksfor";
    memcpy(csrc+5, csrc, strlen(csrc)+1);
    printf("%s", csrc);
    return 0;
}
```

chevron_right

filter_none

Output:

GeeksGeeksfor

Since the input addresses are overlapping, the above program overwrites the original string and causes data loss.

filter_none

edit

close

play_arrow

link

brightness_4

code

```
// Sample program to show that memmove() is better than memcpy()
// when addresses overlap.
#include <stdio.h>
#include <string.h>
int main()
{
    char csrc[100] = "Geeksfor";
    memmove(csrc+5, csrc, strlen(csrc)+1);
    printf("%s", csrc);
    return 0;
}
```

chevron_right

filter_none

Output:

GeeksGeeksfor

How to implement memmove()?

The trick here is to use a temp array instead of directly copying from src to dest. The use of temp array is important to handle cases when source and destination addresses are overlapping.

filter_none

edit

close

play_arrow

link

brightness_4

code

```
//C++ program to demonstrate implementation of memmove()
#include<stdio.h>
#include<string.h>

// A function to copy block of 'n' bytes from source
// address 'src' to destination address 'dest'.
void myMemMove(void *dest, void *src, size_t n)
{
    // Typecast src and dest addresses to (char *)
    char *csrc = (char *)src;
    char *cdest = (char *)dest;

    // Create a temporary array to hold data of src
    char *temp = new char[n];

    // Copy data from csrc[] to temp[]
    for (int i=0; i<n; i++)
        temp[i] = csrc[i];

    // Copy data from temp[] to cdest[]
    for (int i=0; i<n; i++)
        cdest[i] = temp[i];

    delete [] temp;
}

// Driver program
int main()
{
    char csrc[100] = "Geeksfor";
    myMemMove(csrc+5, csrc, strlen(csrc)+1);
    printf("%s", csrc);
    return 0;
}
```

chevron_right

filter_none

Output:

GeeksGeeksfor

Optimizations:

The algorithm is inefficient (and honestly double the time if you use a temporary array). Double copies should be avoided unless it is really impossible.

In this case though it is easily possible to avoid double copies by picking a direction of copy. In fact this is what the memmove() library function does.

By comparing the src and the dst addresses you should be able to find if they overlap.

- If they do not overlap, you can copy in any direction
- If they do overlap, find which end of dest overlaps with the source and choose the direction of copy accordingly.
- If the beginning of dest overlaps, copy from end to beginning
- If the end of dest overlaps, copy from beginning to end

- Another optimization would be to copy by word size. Just be careful to handle the boundary conditions.
- A further optimization would be to use vector instructions for the copy since they're contiguous.

This article is contributed by Saurabh Jain. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes 

Add your personal notes here! (max 5000 chars)

Recommended Posts:

- [memmove\(\) in C/C++](#)
- [memcpy\(\) in C/C++](#)
- [Write a URL in a C++ program](#)
- [How to write your own header file in C?](#)
- [Write a C program that won't compile in C++](#)
- [When should we write our own assignment operator in C++?](#)
- [When should we write our own copy constructor?](#)
- [C program to write an image in PGM format](#)
- [Write a program that produces different results in C and C++](#)
- [Write a C macro PRINT\(x\) which prints x](#)
- [How to write a running C code without main\(\)](#)
- [Read/Write structure to a file in C](#)
- [Does C++ compiler create default constructor when we write our own?](#)
- [Write one line functions for strcat\(\) and strcmp\(\)](#)
- [Read/Write Class Objects from/to File in C++](#)

Improved By : [SwatiGangwar](#)

Article Tags :

C
C++
CPP-Library
cpp-pointer

Practice Tags :
C
CPP

 4

To-do Done
3.4

Based on 13 vote(s)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) Difference between getc(), getchar(), getch() and getche()

Next

[last_page](#) What happens when a virtual function is called inside a non-virtual function in C++

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Weekend Geeks Classes Weekdays

JAVA APP DEVELOPMENT SUMMER TRAINING COMPETITIVE PROGRAMMING LIVE Java Backend Development

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
How to call function within function in C or C++
Format specifiers in different Programming Languages
Predefined Macros in C with Examples
C program to print odd line contents of a File followed by even line content

Most visited in C++
Vector of Vectors in C++ STL with Examples
C++ STL
Which C++ libraries are useful for competitive programming?
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

C program to print characters without using format specifiers

As we know that there are various **format specifiers** in C like %d, %f, %c etc, to help us print characters or other data types. We normally use these specifiers along with the **printf()** function to print any variables. But there is also a way to print characters specifically without the use of %c format specifier. This can be obtained by using the below-shown method to get the character value of any ASCII codes of any particular character.

Example:

```
filter_none
edit
close

play_arrow
link
brightness_4
code

// Prints characters without format specifiers
#include <stdio.h>
```

int main()

{

```
    printf("\x47 \n");
    printf("\x45 \n");
    printf("\x45 \n");
    printf("\x4b \n");
    printf("\x53");
}
```

chevron_right

filter_none

Output:

```
G
E
E
K
S
```

This article is contributed by **Chinmoy Lenka**. If you like GeeksforGeeks and would like to contribute, you can also write an article using [contribute.geeksforgeeks.org](#) or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up

Add your personal notes
here! (max 5000 chars)

Save

Recommended Posts:

- Format specifiers in C
- Format specifiers in different Programming Languages
- C program to write an image in PGM format
- C program to invert (making negative) an image content in PGM format
- Print * in place of characters for reading passwords in C
- C program to print a string without any quote (single or double) in the program
- C Program to count the Number of Characters in a File
- Program to print last 10 lines
- Program to print the diamond shape
- Program to print Happy Birthday
- C Program to print Floyd's triangle
- C Program to print contents of file
- C Program to print numbers from 1 to N without using semicolon?
- Program to print a pattern of numbers
- C Program to print environment variables

Article Tags :

C
C-puzzle

Practice Tags :

C

thumb_up

Be the First to upvote.

To-do Done
0

No votes yet.

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

first_page CHAR_BIT in C

Next

last_page Benefits of C language over other programming languages

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
Predefined Macros in C with Examples
Format specifiers in different Programming Languages
How to call function within function in C or C++
C program to print odd line contents of a file followed by even line content
More related articles in C
Introduction to the C99 Programming Language : Part II
C programs to find sum and product of given number
Program in comparing Floating point numbers and how to compare them correctly?
Features of C Programming Language
Pointer Expressions in C with Examples

C program to print a string without any quote (single or double) in the program

Print a string without using quotes anywhere in the program using C or C++.

Note : should not read input from the console.

Recommended: Please try your approach on {IDE} first, before moving on to the solution.

The idea is to use macro processor in C (Refer point 6 of this article). A token passed to macro can be converted to a string literal by using # before it.

```
filter_none
edit
close

play_arrow

link
brightness_4
code

// C program to print a string without
// quote in the program
#include <stdio.h>
#define get(x) #x
int main()
{
    printf(get(vignesh));
    return 0;
}
chevron_right
```

filter_none

Output:

```
vignesh
```

This article is contributed by **Vignesh**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up

Save

Recommended Posts:

- C/C++ program to find the size of int, float, double and char
- Program to print last 10 lines
- Program to print the diamond shape
- Program to print Happy Birthday
- C Program to print contents of file
- C Program to print numbers from 1 to N without using semicolon?
- Program to print a pattern of numbers
- C Program to print environment variables
- C Program to print Floyd's triangle
- C program to print characters without using format specifiers
- Program to Print Mirrored Hollow Parallelogram
- C program to print digital clock with current time
- Program to print half Diamond star pattern
- Program to print hollow pyramid and diamond pattern
- Program to print hollow rectangle or square star patterns

Article Tags :

C
C-Macro & Preprocessor
c-puzzle
Practice Tags :
C

thumb_up
2

To-do Done
1.8

Based on 6 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

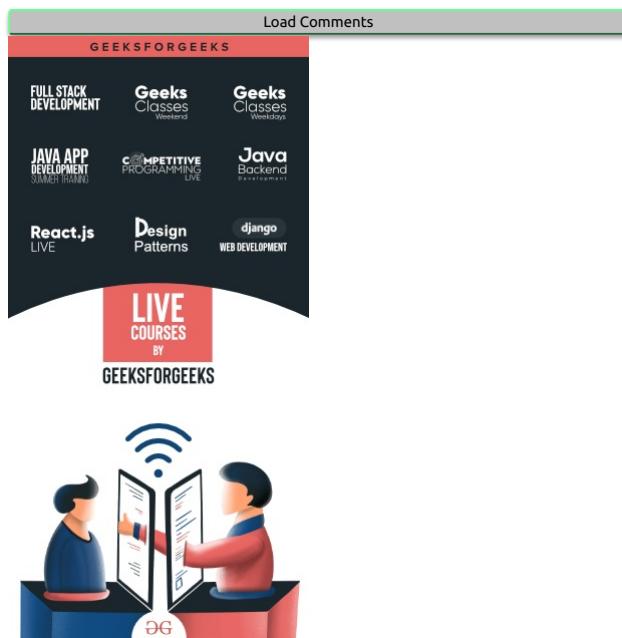
Previous

[first_page](#) Anything written in sizeof() is never executed in C

Next

[last_page](#) ispunct() function in C

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
Predefined Macros in C with Examples
Format specifiers in different Programming Languages
How to call function within function in C or C++
C program to print odd line contents of a file followed by even line content

More related articles in C
C program to calculate square root of a given number
Introduction to the C99 Programming Language : Part II
Problem in comparing Floating point numbers and how to compare them correctly
Feature of C Programming Language
Pointer Expressions in C with Examples

Execute both if and else statements in C/C++ simultaneously

Write a C/C++ program that execute both if-else block statements simultaneously.

```
Syntax of if-else statement in C/C++ language is:  
if (Boolean expression)  
{  
    // Statement will execute only  
    // if Boolean expression is true  
}  
else  
{  
    // Statement will execute only if  
    // the Boolean expression is false  
}
```

Hence we can conclude that only one of the block of if-else statement will execute according to the condition of Boolean expression. But we can able to make our program so that both the statements inside if and else will execute simultaneously.

Recommended: Please try your approach on [{IDE}](#) first, before moving on to the solution.

The trick is to use goto statement which provides an unconditional jump from the 'goto' to a labelled statement in the same function.

Below is C/C++ program to execute both statements simultaneously:-

```
filter_none  
edit  
close  
play_arrow  
link  
brightness_4  
code  
  
#include <stdio.h>  
int main()  
{  
    if (1) //Replace 1 with 0 and see the magic  
    {
```

```

label_1: printf("Hello ");

// Jump to the else statement after
// executing the above statement
goto label_2;
}

else
{
    // Jump to 'if block statement' if
    // the Boolean condition becomes false
    goto label_1;

    label_2: printf("Geeks");
}

return 0;
}

```

chevron_right

filter_none

Output:
Hello Geeks

Therefore both the statements of if and else block executed simultaneously. Another interesting fact can be seen that Output will always remain same and will not depend upon the whether the Boolean condition is true or false.

NOTE – Use of goto statement is highly discouraged in any programming language because it makes difficult to trace the control flow of a program, making the program hard to understand and hard to modify. As a programmer we should avoid the use of goto statement in C/C++.

This blog is contributed by [Shubham Bansal](#). If you like GeeksforGeeks and would like to contribute, you can also write an article using [contribute.geeksforgeeks.org](#) or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes *arrow_drop_up*

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- [fork\(\) to execute processes from bottom to up using wait\(\)](#)
- [How to execute zombie and orphan process in a single program?](#)
- [Difference between continue and break statements in C++](#)
- [C++17 new feature : If Else and Switch Statements with initializers](#)
- [std::basic_istream::ignore in C++ with Examples](#)
- [std::basic_istream::getline in C++ with Examples](#)
- [Format specifiers in different Programming Languages](#)
- [C program to find square root of a given number](#)
- [C program to print odd line contents of a File followed by even line content](#)
- [std::is_heap\(\) in C++ with Examples](#)
- [std::basic_istream::gcount\(\) in C++ with Examples](#)
- [new vs malloc\(\) and free\(\) vs delete in C++](#)
- [std::string::rfind in C++ with Examples](#)
- [Sum of factorials of Prime numbers in a Linked list](#)

Article Tags :

C
C++
c-puzzle
CPP-Basics
cpp-puzzle

Practice Tags :
C
CPP

thumb_up
34

To-do Done
1.8

Based on **70** vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) [Output of C Program | Set 29](#)

Next

[last_page](#) [Programs to print Interesting Patterns](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

[Load Comments](#)



Most popular in C
Program to calculate Electricity Bill
Program to print half Diamond Star pattern
C/C++ program for calling main() in main()
exit(0) vs exit(1) in C/C++ with Examples
C/C++ #include directive with Examples

Most visited in C++
Vector of Vectors in C++ STL with Examples
C++ STL
Which C++ libraries are useful for competitive programming?
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

Print “Hello World” in C/C++ without using any header file

Write a C/C++ program that prints **Hello World** without including any header file.

*Conceptually it's seems impractical to write a C/C++ program that print **Hello World** without using a header file of "stdio.h". Since the declaration of printf() function contains in the "stdio.h" header file.*

But we can easily achieve this by taking the advantage of C pre-processor directives. The fact is at the time of compiling a program, the first phase of C preprocessing expands all header files into a single file and after that compiler itself compiles the expanded file. Therefore we just need to extract the declaration of printf() function from header file and use it in our main program like that:-

- **C language:** Just declare the printf() function taken from "stdio.h" header file.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

//Declare the printf() function
int printf(const char *format, ...);

int main()
{
    printf( "Hello World" );
    return 0;
}
```

Output: Hello World

- **C++ language:** We can't directly put the declaration of printf() function as in previous case due to the problem of **Name mangling** in C++. See [this](#) to know more about Name mangling. Therefore we just need to declare the printf() inside extern keyword like that:-

```
filter_none
edit
close
play_arrow
link
brightness_4
code

//Declare the printf() function inside
//extern "C" for C++ compiler
extern "C"
{
    int printf(const char *format, ...);
}

int main()
{
    printf( "Hello World" );
    return 0;
}
```

Output: Hello World

See [this](#) to know more about all phases of compilation of C program.

This blog is contributed by [Shubham Bansal](#). If you like GeeksforGeeks and would like to contribute, you can also write an article using [contribute.geeksforgeeks.org](#) or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes

Add your personal notes here! (max 5000 chars)

Recommended Posts:

- [clocale header file in C++](#)
- [Comment in header file name?](#)
- [How to write your own header file in C?](#)
- [Difference between Header file and Library](#)
- [time.h header file in C with Examples](#)
- [<complex.h> header file in C with Examples](#)
- [C Program to print contents of file](#)
- [C++ program to print unique words in a file](#)
- [C program to print odd line contents of a File followed by even line content](#)
- [random header | Set 2 \(Distributions\)](#)
- [random header in C++ | Set 1\(Generators\)](#)
- [random header in C++ | Set 3 \(Distributions\)](#)
- [C program to copy contents of one file to another file](#)
- [accumulate\(\) and partial_sum\(\) in C++ STL : numeric header](#)
- [What's difference between header files "stdio.h" and "stdlib.h" ?](#)

Article Tags :

C

C++

c-puzzle

Practice Tags :

C

CPP



7

To-do Done
1.9

Based on 14 vote(s)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) [Zombie Processes and their Prevention](#)

Next

[last_page](#) [strftime\(\) function in C/C++](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

GEEKSFORGEEKS

FULL STACK DEVELOPMENT **Geeks Classes** Frontend **Geeks Classes** Backend

JAVA APP DEVELOPMENT COMPETITIVE PROGRAMMING LIVE Java Backend Backend

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
Predefined Macros in C with Examples
Format specifiers in different Programming Languages
C program to print odd line contents of a File followed by even line content
Introduction to the C99 Programming Language : Part II

Most visited in C++
Vector of Vectors in C++ STL with Examples
C++ Tutorial
Which C++ libraries are useful for competitive programming?
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

Quine – A self-reproducing program

A quine is a program which prints a copy of its own as the only output. A quine takes no input. Quines are named after the American mathematician and logician Willard Van Orman Quine (1908-2000). The interesting thing is you are not allowed to use open and then print file of the program.

To the best of our knowledge, below is the shortest quine in C.

[filter_none](#)
[edit](#)

close
play_arrow
link
brightness_4
code

```
main() { char *s="main() { char *s=%c%c; printf(s,34,s,34); }"; printf(s,34,s,34); }
```

chevron_right

filter_none

This program uses the printf function without including its corresponding header (#include), which can result in undefined behavior. Also, the return type declaration for main has been left off to reduce the length of the program. Two 34s are used to print double quotes around the string s.

Following is a shorter version of the above program suggested by Narendra.

filter_none
edit
close
play_arrow
link
brightness_4
code

```
main(a){printf(a=="main(a){printf(a=%c%c,34,a,34);}"",34,a,34);}
```

chevron_right

filter_none

If you find a shorter C quine or you want to share quine in other programming languages, then please do write in the comment section.

Quine in Python

Source:
http://en.wikipedia.org/wiki/Quine_%28computing%29

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer Placement 100 for details

My Personal Notes arrow_drop_up

Add your personal notes
here! (max 5000 chars)

Save

Recommended Posts:

- [C program to print a string without any quote \(single or double\) in the program](#)
- [Hello World Program : First program while learning Programming](#)
- [C program to detect tokens in a C program](#)
- [Output of C Program | Set 29](#)
- [Program to compare m^n and n^m](#)
- [Write a URL in a C++ program](#)
- [How does a C program executes?](#)
- [Output of C++ Program | Set 14](#)
- [Output of C++ Program | Set 16](#)
- [Output of C++ Program | Set 18](#)
- [Program to compute Log n](#)
- [Output of C++ Program | Set 10](#)
- [How to compile 32-bit program on 64-bit gcc in C and C++](#)
- [Program for n-th even number](#)
- [Splint -- A C program verifier](#)

Article Tags :

C
C++
Practice Tags :
C
CPP

thumb_up
2

To-do Done
4.2

Based on 14 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page When is copy constructor called?](#)

Next

[last_page What are the operators that can be and cannot be overloaded in C++?](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments



LIVE COURSES
BY GEEKSFORGEEKS



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
Format specifiers in different Programming Languages
C program to find square root of a given number
Predefined Macros in C with Examples
C program to print odd line contents of a File followed by even line content

Most visited in C++
Vector of Vectors in C++ STL with Examples
C++ STL with Examples
Which C++ libraries are useful for competitive programming?
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

Complicated declarations in C

Most of the times declarations are simple to read, but it is hard to read some declarations which involve pointer to functions. For example, consider the following declaration from "signal.h".

```
filter_none
edit
close
play_arrow
link
brightness_4
code

void (*bsd_signal(int, void (*)(int)))(int);
chevron_right
filter_none
```

Let us see the steps to read complicated declarations.

- 1) Convert C declaration to postfix format and read from right to left.
- 2) To convert expression to postfix, start from innermost parenthesis. If innermost parenthesis is not present then start from declarations name and go right first. When first ending parenthesis encounters then go left. Once whole parenthesis is parsed then come out from parenthesis.
- 3) Continue until complete declaration has been parsed.

Let us start with simple example. Below examples are from "K & R" book.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

1) int (*fp) ();
chevron_right
filter_none
```

Let us convert above expression to postfix format. For the above example, there is no innermost parenthesis, that's why, we will print declaration name i.e. "fp". Next step is, go to right side of expression, but there is nothing on right side of "fp" to parse, that's why go to left side. On left side we found "*", now print "*" and come out of parenthesis. We will get postfix expression as below.

```
fp * () int
```

Now read postfix expression from left to right. e.g. fp is pointer to function returning int

Let us see some more examples.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

2) int (*daytab)[13]
chevron_right
filter_none
```

Postfix : daytab * [13] int
Meaning : daytab is pointer to array of 13 integers.

```
filter_none
edit
close
```

```

play_arrow
link
brightness_4
code
3) void (*f[10]) (int, int)
chevron_right
filter_none
Postfix : f[10] * (int, int) void
Meaning : f is an array of 10 pointer to function(which takes 2 arguments of type int) returning void
filter_none
edit
close
play_arrow
link
brightness_4
code
4) char (*(*x())[{}]) ()
chevron_right
filter_none
Postfix : x () * [] * () char
Meaning : x is a function returning pointer to array of pointers to function returning char
filter_none
edit
close
play_arrow
link
brightness_4
code
5) char (*(*x[3])())[5]
chevron_right
filter_none
Postfix : x[3] * () * [5] char
Meaning : x is an array of 3 pointers to function returning pointer to array of 5 char's
filter_none
edit
close
play_arrow
link
brightness_4
code
6) int *(*arr[5])()
chevron_right
filter_none
Postfix : arr[5] * () * () * int
Meaning : arr is an array of 5 pointers to functions returning pointer to function returning pointer to integer
filter_none
edit
close
play_arrow
link
brightness_4
code
7) void (*bsd_signal(int sig, void (*func)(int)))(int);
chevron_right
filter_none
Postfix : bsd_signal(int sig, void(*func)(int)) * (int) void
Meaning : bsd_signal is a function that takes integer & a pointer to a function(that takes integer as argument and returns void) and returns pointer to a function(that takes integer as argument and returns void)

```

This article is compiled by "Narendra Kangalkar" and reviewed by GeeksforGeeks team. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up

Recommended Posts:

- Format specifiers in different Programming Languages
- C program to find square root of a given number
- C program to print odd line contents of a File followed by even line content
- Getting System and Process Information Using C Programming and Shell in Linux
- Program to calculate Electricity Bill
- Program to print half Diamond star pattern
- C/C++ program for calling main() in main()
- Print all possible combinations of the string by replacing '\$' with any other digit from the string
- How to use make utility to build C projects?
- How to call function within function in C or C++
- Role of SemiColon in various Programming Languages
- C program to display month by month calendar for a given year
- Difference between Type Casting and Type Conversion

Improved By : Karthiik, aakash nandi

Article Tags :

C
C-Variable Declaration and Scope
pointer
Practice Tags :

C

61

To-do Done

3.7

Based on 214 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

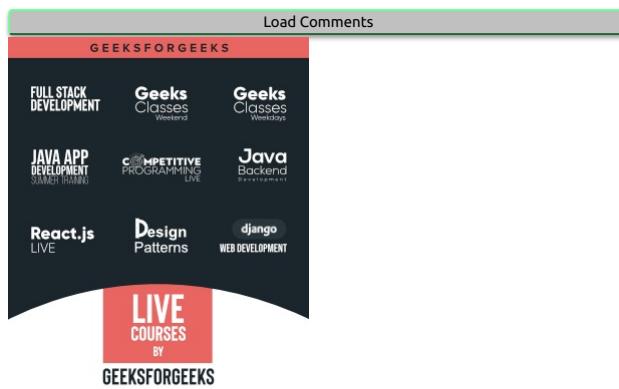
Previous

[first_page](#) Scope rules in C

Next

[last_page](#) Initialization of variables sized arrays in C

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C
Program to calculate Electricity Bill
Program to print half Diamond star pattern
C/C++ program for calling main() in main()
C/C++ #include directive with Examples
Output of C programs | Set 66 (Accessing Memory Locations)

More related articles in C
Chain of Pointers in C with Examples
Difference between Type Casting and Type Conversion
C program to display month by month calendar for a given year
Permute all possible combinations of the string by replacing 's' with any other digit from the string
getch() function in C with Examples

Use of bool in C

Prerequisite: [Bool Data Type in C++](#)

The C99 standard for C language supports bool variables. Unlike C++, where no header file is needed to use bool, a header file "stdbool.h" must be included to use bool in C. If we save the below program as .c, it will not compile, but if we save it as .cpp, it will work fine.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

int main()
{
    bool arr[2] = {true, false};
    return 0;
}
chevron_right
filter_none
```

If we include the header file "stdbool.h" in the above program, it will work fine as a C program.

```
filter_none
edit
close
play_arrow
link
brightness_4
```

```
code
#include <stdbool.h>
int main()
{
    bool arr[2] = {true, false};
    return 0;
}
```

chevron_right

filter_none

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes *arrow_drop_up*

Add your personal notes
here! (max 5000 chars)

Recommended Posts:

- Format specifiers in different Programming Languages
- C program to find square root of a given number
- C program to print odd line contents of a File followed by even line content
- Getting System and Process Information Using C Programming and Shell in Linux
- Program to calculate Electricity Bill
- Program to print half Diamond star pattern
- C/C++ program for calling main() in main()
- Print all possible combinations of the string by replacing '\$' with any other digit from the string
- How to use make utility to build C projects?
- How to call function within function in C or C++
- Role of SemiColon in various Programming Languages
- C program to display month by month calendar for a given year
- Difference between Type Casting and Type Conversion
- Pi(π) in C++ with Examples

Article Tags :

C

C-Data Types

Practice Tags :

C

thumb_up

28

To-do Done
1.3

Based on 105 vote(s)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

first_page Static data members in C++

Next

last_page Private Destructor

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Weekend Geeks Classes Weekdays

JAVA APP DEVELOPMENT SUMMER TRAINING COMPETITIVE PROGRAMMING LIVE Java Backend Development

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS



Most popular in C
Program to calculate Electricity Bill
Program to print half Diamond star pattern
C/C++ program for calling main() in main()
C/C++ #include directive with Examples
Output of C programs | Set 66 (Accessing Memory Locations)

More related articles in C

Chain of Pointers in C with Examples
Difference between Type Casting and Type Conversion
C program to display month by month calendar for a given year
Print all possible combinations of the string by replacing '\$' with any other digit from the string
getch() function in C with Examples

Sequence Points in C | Set 1

In this post, we will try to cover many ambiguous questions like following.

Guess the output of following programs.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// PROGRAM 1
#include <stdio.h>
int f1() { printf ("Geeks"); return 1;}
int f2() { printf ("forGeeks"); return 1;}
int main()
{
    int p = f1() + f2();
    return 0;
}

// PROGRAM 2
#include <stdio.h>
int x = 20;
int f1() { x = x+10; return x;}
int f2() { x = x-5; return x;}
int main()
{
    int p = f1() + f2();
    printf ("p = %d", p);
    return 0;
}

// PROGRAM 3
#include <stdio.h>
int main()
{
    int i = 8;
    int p = i++*i++;
    printf ("%d\n", p);
}
chevron_right
```

The output of all of the above programs is **undefined** or **unspecified**. The output may be different with different compilers and different machines. It is like asking the value of undefined automatic variable.

The reason for undefined behavior in PROGRAM 1 is, the operator '+' doesn't have standard defined order of evaluation for its operands. Either f1() or f2() may be executed first. So output may be either "GeeksforGeeks" or "forGeeksGeeks".

Similar to operator '+', most of the other similar operators like '-', '/', '*', Bitwise AND &, Bitwise OR |, .. etc don't have a standard defined order for evaluation for its operands.

Evaluation of an expression may also produce side effects. For example, in the above program 2, the final values of p is ambiguous. Depending on the order of expression evaluation, if f1() executes first, the value of p will be 55, otherwise 40.

The output of program 3 is also undefined. It may be 64, 72, or may be something else. The subexpression i++ causes a side effect, it modifies i's value, which leads to undefined behavior since i is also referenced elsewhere in the same expression.

Unlike above cases, *at certain specified points in the execution sequence called sequence points, all side effects of previous evaluations are guaranteed to be complete*. A sequence point defines any point in a computer program's execution at which it is guaranteed that all side effects of previous evaluations will have been performed, and no side effects from subsequent evaluations have yet been performed. Following are the sequence points listed in the C standard:

— **The end of the first operand of the following operators:**

- a) logical AND &&
- b) logical OR ||
- c) conditional ?
- d) comma ,

For example, the output of following programs is guaranteed to be "GeeksforGeeks" on all compilers/machines.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// Following 3 lines are common in all of the below programs
#include <stdio.h>
int f1() { printf ("Geeks"); return 1;}
int f2() { printf ("forGeeks"); return 1;}

// PROGRAM 4
int main()
{
    // Since && defines a sequence point after first operand, it is
    // guaranteed that f1() is completed first.
    int p = f1() && f2();
    return 0;
}

// PROGRAM 5
int main()
{
    // Since comma operator defines a sequence point after first operand, it is
```

```

// guaranteed that f1() is completed first.
int p = (f1(), f2());
return 0;
}

// PROGRAM 6
int main()
{
    // Since ? operator defines a sequence point after first operand, it is
    // guaranteed that f1() is completed first.
    int p = f1()? f2(): 3;
    return 0;
}
chevron_right
filter_none

```

— The end of a full expression. This category includes following expression statements

- a) Any full statement ended with semicolon like "a = b;"
- b) return statements
- c) The controlling expressions of if, switch, while, or do-while statements.
- d) All three expressions in a for statement.

The above list of sequence points is partial. We will be covering all remaining sequence points in the next post on Sequence Point.

References:

http://en.wikipedia.org/wiki/Sequence_point
<http://c-faq.com/expr/seqpoints.html>
[http://msdn.microsoft.com/en-us/library/d45c7a5d\(v=vs.110\).aspx](http://msdn.microsoft.com/en-us/library/d45c7a5d(v=vs.110).aspx)
<http://www.open-std.org/jtc1/sc22/wg14/www/docs/n925.htm>

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes 

Add your personal notes
here! (max 5000 chars)



Recommended Posts:

- Print numbers in sequence using thread synchronization
- Format specifiers in different Programming Languages
- C program to find square root of a given number
- C program to print odd line contents of a File followed by even line content
- Getting System and Process Information Using C Programming and Shell in Linux
- Program to calculate Electricity Bill
- Program to print half Diamond star pattern
- C/C++ program for calling main() in main()
- Print all possible combinations of the string by replacing '\$' with any other digit from the string
- How to use make utility to build C projects?
- How to call function within function in C or C++
- Role of SemiColon in various Programming Languages
- C program to display month by month calendar for a given year
- Difference between Type Casting and Type Conversion

Article Tags :

C
C-Operators
Practice Tags :
C

 16

To-do Done
3.4

Based on 56 vote(s)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) Importance of function prototype in C

Next

[last_page](#) Variable length arguments for Macros

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.





Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string ctype.h<ctype.h> library in C/C++ with Examples

Implicit Type Conversion in C with Examples
Format specifiers in different Programming Languages

C program to find square root of a given number

More related articles in C

Predefined Macros in C with Examples

How to call a function with return in C or C++

C program to print end line contents of a file followed by even line content

Introduction to the C99 Programming Language : Part II

Features of C Programming Language

Optimization Techniques | Set 2 (swapping)

How to swap two variables?

The question may look silly, neither geeky. See the following piece of code to swap two integers (**XOR swapping**),

```
void myswap(int *x, int *y)
{
    if (*x != *y)
    {
        *x ^= *y ^= *x ^= *y;
    }
}
```

At first glance, we may think nothing wrong with the code. However, when prompted for reason behind opting for XOR swap logic, the person was clue less. Perhaps any **commutative** operation can fulfill the need with some corner cases.

Avoid using fancy code snippets in production software. They create runtime surprises. We can observe the following notes on above code

1. The code behavior is undefined. The statement `*x^=*y^=*x^=*y;` modifying a variable more than once in without any **sequence point**.
2. It creates **pipeline stalls** when executed on a processor with pipeline architecture.
3. The compiler can't take advantage in optimizing the swapping operation. Some processors will provide single instruction to swap two variables. When we opted for standard library functions, there are more chances that the library would have been optimized. Even the compiler can recognize such standard function and generates optimum code.
4. Code readability is poor. It is very much important to write maintainable code.

Thanks to **Venki** for writing the above article. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes [arrow_drop_up](#)

Add your personal notes here! (max 5000 chars)

[Save](#)

Recommended Posts:

- Parameter Passing Techniques in C/C++
- Swapping of subranges from different containers in C++
- Number of ways to change the XOR of two numbers by swapping the bits
- XOR of elements in a given range with updates using Fenwick Tree
- Query to count odd and even parity elements in subarray after XOR with K
- Rust vs C++: Will Rust Replace C++ in Future ?
- Count of even and odd set bit Array elements after XOR with K for Q queries
- Priority queue of pairs in C++ with ordering by first and second element
- Implementation of `lower_bound()` and `upper_bound()` in Vector of Pairs in C++
- Generating RGBA portable graphic images through C++
- `std::basic_istream::ignore` in C++ with Examples
- `std::basic_istream::getline` in C++ with Examples
- Format specifiers in different Programming Languages
- `std::is_heap()` in C++ with Examples

Article Tags :

C++

Bitwise-XOR

Practice Tags :

CPP

[thumb_up](#)

Be the First to upvote.

To-do Done
3.6

Based on 6 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Weekend Geeks Classes Weekdays

JAVA APP DEVELOPMENT SUMMER TRAINING COMPETITIVE PROGRAMMING LIVE Java Backend Development

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS

Most popular in C++
Sum of all Palindrome Numbers present in a Linked list
C/C++ program for calling main() in main()
Sum of factorials of Prime numbers in a Linked list
std::string::find last of in C++ with Examples
Namespaces in C++ | Set 4 (Overloading, and Exchange of Data in different Namespaces)

More related articles in C++
new vs malloc() and free() vs delete in C++
How to initialize Array of objects with parameterized constructors in C++
std::equal in C++ with Examples
std::is_heap() in C++ with Examples
How to use make utility to build C projects?

ASCII NUL, ASCII 0 ('0') and Numeric literal 0

The ASCII NUL and zero are represented as 0x00 and 0x30 respectively. An ASCII NUL character serves as sentinel characters of strings in C/C++. When the programmer uses '0' in his code, it will be represented as 0x30 in hex form. What will be filled in the binary representation of 'integer' in the following program?

filter_none
edit
close
play_arrow
link
brightness_4
code

```
char charNUL = '\0';  
  
unsigned int integer = 0;  
  
char charBinary = '0';  
chevron_right
```

The binary form of `charNUL` will have all its bits set to logic 0. The binary form of `integer` will have all its bits set to logic 0, which means each byte will be filled with NUL character (\ 0). The binary form of `charBinary` will be set to binary equivalent of hex 0x30.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up

Save

Recommended Posts:

- [accumulate\(\) and partial_sum\(\) in C++ STL : numeric header](#)
- [numeric header in C++ STL | Set 2 \(adjacent_difference\(\), inner_product\(\) and iota\(\)\)](#)
- [Raw string literal in C++](#)
- [Split numeric, alphabetic and special symbols from a String](#)
- [Integer literal in C/C++ \(Prefixes and Suffixes\)](#)
- [<numeric> library in C++ STL](#)
- [Rust vs C++: Will Rust Replace C++ in Future ?](#)
- [std::basic_istream::ignore in C++ with Examples](#)
- [std::basic_istream::getline in C++ with Examples](#)
- [Format specifiers in different Programming Languages](#)
- [std::is_heap\(\) in C++ with Examples](#)
- [std::basic_istream::gcount\(\) in C++ with Examples](#)
- [new vs malloc\(\) and free\(\) vs delete in C++](#)
- [std::string::rfind in C++ with Examples](#)

Article Tags :

C++

Practice Tags :

CPP



Be the First to upvote.

To-do Done
2

Based on 6 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) Declare a C/C++ function returning pointer to array of integer pointers

Next

[last_page](#) Nested Classes in C++

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Weekend Geeks Classes Weekdays

JAVA APP DEVELOPMENT COMPETITIVE PROGRAMMING Java Backend

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS

Most popular in C++
 Sum of all Palindrome Numbers present in a Linked list
 Program to print half Diamond star pattern
 C/C++ program for calling main() in main()
 C/C++ #include directive with Examples
 Sum of factorials of Prime numbers in a Linked list

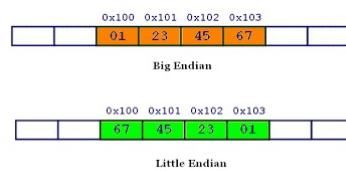
More related articles in C++
 std::string::find last of in C++ with Examples
 Namespaces in C++ | Set 1 (Introduction and Exchange of Data in different Namespaces)
 new operator (malloc) vs delete in C++
 Generate an array of given size with equal count and sum of odd and even numbers
 How to initialize Array of objects with parameterized constructors in C++

Little and Big Endian Mystery

What are these?

Little and big endian are two ways of storing multibyte data-types (int, float, etc). In little endian machines, last byte of binary representation of the multibyte data-type is stored first. On the other hand, in big endian machines, first byte of binary representation of the multibyte data-type is stored first.

Suppose integer is stored as 4 bytes (For those who are using DOS based compilers such as C++ 3.0 , integer is 2 bytes) then a variable x with value 0x01234567 will be stored as following.



Memory representation of integer 0x01234567 inside Big and little endian machines

How to see memory representation of multibyte data types on your machine?

Here is a sample C code that shows the byte representation of int, float and pointer.

```
filter_none
edit
close

play_arrow
link
brightness_4
code

#include <stdio.h>

/* function to show bytes in memory, from location start to start+n*/
void show_mem_rep(char *start, int n)
```

```

{
    int i;
    for (i = 0; i < n; i++)
        printf(" %2x", start[i]);
    printf("\n");
}

/*Main function to call above function for 0x01234567*/
int main()
{
    int i = 0x01234567;
    show_mem_rep((char *)&i, sizeof(i));
    getchar();
    return 0;
}

```

chevron_right

filter_none

When above program is run on little endian machine, gives "67 45 23 01" as output , while if it is run on big endian machine, gives "01 23 45 67" as output.

Is there a quick way to determine endianness of your machine?

There are n no. of ways for determining endianness of your machine. Here is one quick way of doing the same.

C++

```

filter_none
edit
close

play_arrow
link
brightness_4
code

#include <bits/stdc++.h>
using namespace std;
int main()
{
    unsigned int i = 1;
    char *c = (char*)&i;
    if (*c)
        cout<<"Little endian";
    else
        cout<<"Big endian";
    return 0;
}

// This code is contributed by rathbhupendra

```

chevron_right

filter_none

C

```

filter_none
edit
close

play_arrow
link
brightness_4
code

#include <stdio.h>
int main()
{
    unsigned int i = 1;
    char *c = (char*)&i;
    if (*c)
        printf("Little endian");
    else
        printf("Big endian");
    getchar();
    return 0;
}

```

chevron_right

filter_none

Output:

Little endian

In the above program, a character pointer c is pointing to an integer i. Since size of character is 1 byte when the character pointer is de-referenced it will contain only first byte of integer. If machine is little endian then *c will be 1 (because last byte is stored first) and if machine is big endian then *c will be 0.

Does endianness matter for programmers?

Most of the times compiler takes care of endianness, however, endianness becomes an issue in following cases.

It matters in network programming: Suppose you write integers to file on a little endian machine and you transfer this file to a big endian machine. Unless there is little endian to big endian transformation, big endian machine will read the file in reverse order. You can find such a practical example here.

Standard byte order for networks is big endian, also known as network byte order. Before transferring data on network, data is first converted to network byte order (big endian).

Sometimes it matters when you are using type casting, below program is an example.

filter_none

edit

close

play_arrow

```
link  
brightness_4  
code  
  
#include <stdio.h>  
int main()  
{  
    unsigned char arr[2] = {0x01, 0x00};  
    unsigned short int x = *(unsigned short int *) arr;  
    printf("%d", x);  
    getchar();  
    return 0;  
}  
chevron_right
```

In the above program, a char array is typecasted to an unsigned short integer type. When I run above program on little endian machine, I get 1 as output, while if I run it on a big endian machine I get 256. To make programs endianness independent, above programming style should be avoided.

What are bi-endians?

Bi-endian processors can run in both modes little and big endian.

What are the examples of little, big endian and bi-endian machines ?

Intel based processors are little endians. ARM processors were little endians. Current generation ARM processors are bi-endian.

Motorola 68K processors are big endians. PowerPC (by Motorola) and SPARK (by Sun) processors were big endian. Current version of these processors are bi-endians.

Does endianness affects file formats?

File formats which have 1 byte as a basic unit are independent of endianness e.g., ASCII files . Other file formats use some fixed endianness forrmatt e.g, JPEG files are stored in big endian format.

Which one is better — little endian or big endian?

The term little and big endian came from Gulliver's Travels by Jonathan Swift. Two groups could not agree by which end an egg should be opened -a- the little or the big. Just like the egg issue, there is no technological reason to choose one byte ordering convention over the other, hence the arguments degenerate into bickering about sociopolitical issues. As long as one of the conventions is selected and adhered to consistently, the choice is arbitrary.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- Find distance between two nodes in the given Binary tree for Q queries
- XOR of elements in a given range with updates using Fenwick Tree
- Query to count odd and even parity elements in subarray after XOR with K
- Maximum weighted edge in path between two nodes in an N-ary tree using binary lifting
- Split a binary string into K subsets minimizing sum of products of occurrences of 0 and 1
- Binary string with given frequencies of sums of consecutive pairs of characters
- Count of even and odd set bit Array elements after XOR with K for Q queries
- Count of the non-prime divisors of a given number
- Program to convert Hexa-Decimal Number to its equivalent BCD
- Priority queue of pairs in C++ with ordering by first and second element
- Program to convert a BCD to Hexa-Decimal Number
- Convert the given BCD to its equivalent Binary form
- Count of integers up to N which represent a Binary number
- How to build a simple music player app using Android Studio

Improved By : [skbarnwal](#), [louisportay](#), [rathbhupendra](#)

Article Tags :

Articles
Bit Magic
Big Endian
Endianness
Little Endian

Practice Tags :

Bit Magic

thumb_up
33

To-do Done
2.4

Based on 61 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) Check for Integer Overflow

Next

[last_page](#) Understanding “extern” keyword in C

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments



Most popular in Articles
[new vs malloc\(\) and free\(\) vs delete in C++](#)
[How to build a simple music player app using Android Studio](#)
[When to use each sorting algorithms](#)
[How to change the default icon of Android App](#)
[Difference Between sum of degrees of odd and even degree nodes in an Undirected Graph](#)
[Most visited in Bit Magic](#)
[Count of subarrays of an Array having all unique digits](#)
[Find XOR of all elements in an Array](#)
[Check if N number is Odd or Even using Bitwise Operators](#)
[Number of subsets with same AND, OR and XOR values in an Array](#)
[Range Queries for finding the Sum of all even parity numbers](#)

Comparator function of qsort() in C

Standard C library provides `qsort()` that can be used for sorting an array. As the name suggests, the function uses QuickSort algorithm to sort the given array. Following is prototype of `qsort()`

```
filter_none
edit
close
play_arrow
link
brightness_4
code

void qsort (void* base, size_t num, size_t size,
           int (*comparator)(const void*,const void*));
chevron_right
filter_none
```

The key point about `qsort()` is comparator function `comparator`. The comparator function takes two arguments and contains logic to decide their relative order in sorted output. The idea is to provide flexibility so that `qsort()` can be used for any type (including user defined types) and can be used to obtain any desired order (increasing, decreasing or any other).

The comparator function takes two pointers as arguments (both type-casted to `const void*`) and defines the order of the elements by returning (in a stable and transitive manner

```
int comparator(const void* p1, const void* p2);
Return value meaning
<0 The element pointed by p1 goes before the element pointed by p2
0 The element pointed by p1 is equivalent to the element pointed by p2
>0 The element pointed by p1 goes after the element pointed by p2
```

Source: <http://www.cplusplus.com/reference/cstdlib/qsort/>

For example, let there be an array of students where following is type of student.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

struct Student
{
    int age, marks;
    char name[20];
};

chevron_right
filter_none
```

Lets say we need to sort the students based on marks in ascending order. The comparator function will look like:

```
filter_none
edit
close
play_arrow
link
brightness_4
code

int comparator(const void *p, const void *q)
{
    int l = ((struct Student *)p)->marks;
    int r = ((struct Student *)q)->marks;
    return (l - r);
```

```
}
```

```
chevron_right
```

```
filter_none
```

See following posts for more sample uses of `qsort()`.

Given a sequence of words, print all anagrams together

Box Stacking Problem

Closest Pair of Points

Following is an interesting problem that can be easily solved with the help of `qsort()` and comparator function.

Given an array of integers, sort it in such a way that the odd numbers appear first and the even numbers appear later. The odd numbers should be sorted in descending order and the even numbers should be sorted in ascending order.

The simple approach is to first modify the input array such that the even and odd numbers are segregated followed by applying some sorting algorithm on both parts(odd and even) separately.

However, there exists an interesting approach with a little modification in comparator function of Quick Sort. The idea is to write a comparator function that takes two addresses p and q as arguments. Let l and r be the number pointed by p and q. The function uses following logic:

- 1) If both (l and r) are odd, put the greater of two first.
- 2) If both (l and r) are even, put the smaller of two first.
- 3) If one of them is even and other is odd, put the odd number first.

Following is C implementation of the above approach.

```
filter_none
```

```
edit
```

```
close
```

```
play_arrow
```

```
link
```

```
brightness_4
```

```
code
```

```
#include <stdio.h>
#include <stdlib.h>
```

```
// This function is used in qsort to decide the relative order
// of elements at addresses p and q.
```

```
int comparator(const void *p, const void *q)
```

```
{
```

```
    // Get the values at given addresses
```

```
    int l = *(const int *)p;
```

```
    int r = *(const int *)q;
```

```
    // both odd, put the greater of two first.
```

```
    if ((l&1) && (r&1))
```

```
        return (r-l);
```

```
    // both even, put the smaller of two first
```

```
    if ( !(l&1) && !(r&1) )
```

```
        return (l-r);
```

```
    // l is even, put r first
```

```
    if ( !(l&1) )
```

```
        return 1;
```

```
    // l is odd, put l first
```

```
    return -1;
```

```
}
```

```
// A utility function to print an array
```

```
void printArr(int arr[], int n)
```

```
{
```

```
    int i;
```

```
    for (i = 0; i < n; ++i)
```

```
        printf("%d ", arr[i]);
```

```
}
```

```
// Driver program to test above function
```

```
int main()
```

```
{
```

```
    int arr[] = {1, 6, 5, 2, 3, 9, 4, 7, 8};
```

```
    int size = sizeof(arr) / sizeof(arr[0]);
```

```
    qsort((void*)arr, size, sizeof(arr[0]), comparator);
```

```
    printf("Output array is\n");
```

```
    printArr(arr, size);
```

```
    return 0;
```

```
}
```

```
chevron_right
```

```
filter_none
```

Output:

```
Output array is
9 7 5 3 1 2 4 6 8
```

Exercise:

Given an array of integers, sort it in alternate fashion. Alternate fashion means that the elements at even indices are sorted separately and elements at odd indices are sorted separately.

This article is compiled by [Aashish Barnwal](#). Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes 

Add your personal notes
here! (max 5000 chars)

Save

Recommended Posts:

- C qsort() vs C++ sort()
- Count number of pairs with the given Comparator
- Sort an array of pairs using Java Pair and Comparator
- Sort an Array of dates in ascending order using Custom Comparator
- Function Overloading vs Function Overriding in C++
- How to call function within function in C or C++
- What happens when a virtual function is called inside a non-virtual function in C++
- Difference between Virtual function and Pure virtual function in C++
- div() function in C++
- log() function in C++
- exp() function C++
- fma() function in C++
- arc function in C
- mbsrtowcs() function in C/C++
- isunordered() function in C++

Article Tags :

C
C++
Sorting
C-Library
CPP-Library
Quick Sort
Sorting Quiz
Practice Tags :
Sorting
C
CPP

9

To-do Done
3.1

Based on 34 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

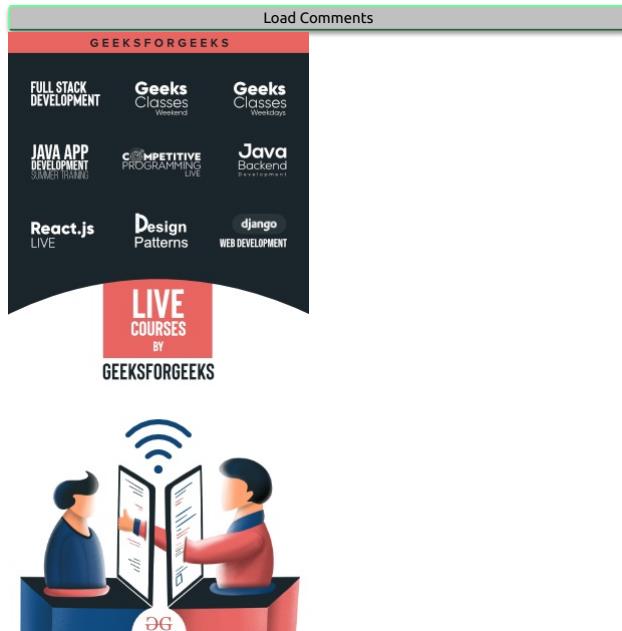
Previous

[first_page](#) C++ | Misc C++ | Question 1

Next

[last_page](#) C++ | Class and Object | Question 3

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
C program to display month by month calendar for a given year
ctype.h <ctype> library in C/C++ with Examples
Implicit Type Conversion in C with Examples
How to call function within function in C or C++

Most visited in C++
Vector of Vectors in C++ STL with Examples
C++ Tutorial
Which C++ libraries are useful for competitive programming?
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

Program to validate an IP address

Write a program to Validate an IPv4 Address.

According to Wikipedia, [IPv4 addresses](#) are canonically represented in dot-decimal notation, which consists of four decimal numbers, each ranging from 0 to 255, separated by dots, e.g., 172.16.254.1

Recommended: Please solve it on “PRACTICE” first, before moving on to the solution.

Following are steps to check whether a given string is valid IPv4 address or not:

step 1) Parse string with “.” as delimiter using “`strtok()`” function.

```
filter_none
edit
close
play_arrow
link
brightness_4
code
e.g.ptr = strtok(str, DELIM);
chevron_right
filter_none
```

step 2)

-a) If ptr contains any character which is not digit then return 0
-b) Convert “ptr” to decimal number say ‘NUM’
-c) If NUM is not in range of 0-255 return 0
-d) If NUM is in range of 0-255 and ptr is non-NULL increment “dot_counter” by 1
-e) if ptr is NULL goto step 3 else goto step 1

step 3) if dot_counter != 3 return 0 else return 1.

```
filter_none
edit
close
play_arrow
link
brightness_4
code
// Program to check if a given string is valid IPv4 address or not
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define DELIM "."

/* function to check whether the string passed is valid or not */
bool valid_part(char* s)
{
    int n = strlen(s);
    // if length of passed string is more than 3 then it is not valid
    if (n > 3)
        return false;
    // check if the string only contains digits
    // if not then return false
    for (int i = 0; i < n; i++)
        if ((s[i] >= '0' && s[i] <= '9') == false)
            return false;
    string str(s);
    // if the string is "00" or "001" or "05" etc then it is not valid
    if (str.find('0') == 0 && n > 1)
        return false;
    stringstream geek(str);
    int x;
    geek >> x;
    // the string is valid if the number generated is between 0 to 255
    return (x >= 0 && x <= 255);
}

/* return 1 if IP string is valid, else return 0 */
int is_valid_ip(char* ip_str)
{
    // if empty string then return false
    if (ip == NULL)
        return 0;
    int i, num, dots = 0;
    int len = strlen(ip);
    int count = 0;
    // the number dots in the original string should be 3
    // for it to be valid
    for (int i = 0; i < len; i++)
        if (ip[i] == '.')
            count++;
    if (count != 3)
        return false;
    // See following link for strtok()
    // http:// pubbs.opengroup.org/onlinepubs/009695399/functions/strtok_r.html
    char *ptr = strtok(ip_str, DELIM);
    if (ptr == NULL)
        return 0;
    while (ptr) {
        /* after parsing string, it must be valid */
        if (valid_part(ptr)) {
            /* parse remaining string */
            ptr = strtok(NULL, ".");
            if (ptr != NULL)
                ++dots;
        }
        else
            return 0;
    }
}
```

```

}

/* valid IP string must contain 3 dots */
// this is for the cases such as 1...1 where originally the
// no. of dots is three but after iteration of the string we find it is not valid
if (dots != 3)
    return 0;
return 1;
}

// Driver program to test above functions
int main()
{
    char ip1[] = "128.0.0.1";
    char ip2[] = "125.16.100.1";
    char ip3[] = "125.512.100.1";
    char ip4[] = "125.512.100.abc";
    is_valid_ip(ip1) ? printf("Valid\n") : printf("Not valid\n");
    is_valid_ip(ip2) ? printf("Valid\n") : printf("Not valid\n");
    is_valid_ip(ip3) ? printf("Valid\n") : printf("Not valid\n");
    is_valid_ip(ip4) ? printf("Valid\n") : printf("Not valid\n");
    return 0;
}

```

chevron_right

filter_none

Output:

```

Valid
Valid
Not valid
Not valid

```

This article is compiled by **Narendra Kangalkar**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes *arrow_drop_up*

Add your personal notes
here! (max 5000 chars)

Save

Recommended Posts:

- [C Program to display hostname and IP address](#)
- [C Program to find IP Address, Subnet Mask & Default Gateway](#)
- [How to validate CVV number using Regular Expression](#)
- [How to validate HTML tag using Regular Expression](#)
- [How to validate a domain name using Regular Expression](#)
- [How to validate PAN Card number using Regular Expression](#)
- [How to validate identifier using Regular Expression in Java](#)
- [How to validate IFSC Code using Regular Expression](#)
- [How to validate a Password using Regular Expressions in Java](#)
- [How to validate MasterCard number using Regular Expression](#)
- [How to validate SSN \(Social Security Number\) using Regular Expression](#)
- [How to validate Hexadecimal Color Code using Regular Expression](#)
- [How to validate image file extension using Regular Expression](#)
- [How to validate time in 12-hour format using Regular Expression](#)
- [How to validate time in 24-hour format using Regular Expression](#)

Improved By : [meeseeks_nits](#)

Article Tags :

C
C++
Strings
Microsoft
Qualcomm
Zoho
Practice Tags :
Zoho
Microsoft
Qualcomm
Strings
C
CPP

thumb_up
4

To-do Done
2.8

Based on 36 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) Facebook Interview | Set 1

Next

[last_page](#) Write your own strcmp that ignores cases

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
`ctype.h <ctype.h>` library in C/C++ with Examples
Predefined Macros in C with Examples
Format specifiers in different Programming Languages
How to call function within function in C or C++
Most visited in C++
Vector of Vectors in C++ STL with Examples
C++ STL
Which C++ libraries are useful for competitive programming?
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

Multithreading in C

What is a Thread?

A thread is a single sequence stream within in a process. Because threads have some of the properties of processes, they are sometimes called *lightweight processes*.

What are the differences between process and thread?

Threads are not independent of one other like processes as a result threads shares with other threads their code section, data section and OS resources like open files and signals. But, like process, a thread has its own program counter (PC), a register set, and a stack space.

Why Multithreading?

Threads are popular way to improve application through parallelism. For example, in a browser, multiple tabs can be different threads. MS word uses multiple threads, one thread to format the text, other thread to process inputs, etc.

Threads operate faster than processes due to following reasons:

- 1) Thread creation is much faster.
- 2) Context switching between threads is much faster.
- 3) Threads can be terminated easily
- 4) Communication between threads is faster.

See <http://www.personal.kent.edu/~rmuhamma/OpSystems/Myos/threads.htm> for more details.

Can we write multithreading programs in C?

Unlike Java, multithreading is not supported by the language standard. **POSIX Threads (or Pthreads)** is a POSIX standard for threads. Implementation of pthread is available with gcc compiler.

A simple C program to demonstrate use of pthread basic functions

Please note that the below program may compile only with C compilers with pthread library.

```
filter_none
edit
close

play_arrow
link
brightness_4
code

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h> //Header file for sleep(). man 3 sleep for details.
#include <pthread.h>

// A normal C function that is executed as a thread
// when its name is specified in pthread_create()
void *myThreadFun(void *vargp)
{
    sleep(1);
    printf("Printing GeeksQuiz from Thread \n");
    return NULL;
}

int main()
{
    pthread_t thread_id;
    printf("Before Thread\n");
    pthread_create(&thread_id, NULL, myThreadFun, NULL);
    pthread_join(thread_id, NULL);
    printf("After Thread\n");
    exit(0);
}
chevron_right
filter_none
```

In main() we declare a variable called `thread_id`, which is of type `pthread_t`, which is an integer used to identify the thread in the system. After declaring `thread_id`, we call `pthread_create()` function to create a thread.

`pthread_create()` takes 4 arguments.
The first argument is a pointer to `thread_id` which is set by this function.
The second argument specifies attributes. If the value is `NULL`, then default attributes shall be used.
The third argument is name of function to be executed for the thread to be created.
The fourth argument is used to pass arguments to the function, `myThreadFun`.
The `pthread_join()` function for threads is the equivalent of `wait()` for processes. A call to `pthread_join` blocks the calling thread until the thread with identifier equal to the first argument terminates.

How to compile above program?

To compile a multithreaded program using `gcc`, we need to link it with the `pthreads` library. Following is the command used to compile the program.

```
gfg@ubuntu:~/src$ gcc multithread.c -lpthread
gfg@ubuntu:~/src$ ./a.out
Before Thread
Printing GeeksQuiz from Thread
After Thread
gfg@ubuntu:~/src$
```

A C program to show multiple threads with global and static variables

As mentioned above, all threads share data segment. Global and static variables are stored in data segment. Therefore, they are shared by all threads. The following example program demonstrates the same.

```
filter_none
edit
close

play_arrow
link
brightness_4
code

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <pthread.h>

// Let us create a global variable to change it in threads
int g = 0;

// The function to be executed by all threads
void *myThreadFun(void *vargp)
{
    // Store the value argument passed to this thread
    int *myid = (int *)vargp;

    // Let us create a static variable to observe its changes
    static int s = 0;

    // Change static and global variables
    ++s; ++g;

    // Print the argument, static and global variables
    printf("Thread ID: %d, Static: %d, Global: %d\n", *myid, ++s, ++g);
}

int main()
{
    int i;
    pthread_t tid;

    // Let us create three threads
    for (i = 0; i < 3; i++)
        pthread_create(&tid, NULL, myThreadFun, (void *)&tid);

    pthread_exit(NULL);
    return 0;
}
```

chevron_right

filter_none

```
gfg@ubuntu:~/src$ gcc multithread.c -lpthread
gfg@ubuntu:~/src$ ./a.out
Thread ID: 3, Static: 2, Global: 2
Thread ID: 3, Static: 4, Global: 4
Thread ID: 3, Static: 6, Global: 6
gfg@ubuntu:~/src$
```

Please note that above is simple example to show how threads work. Accessing a global variable in a thread is generally a bad idea. What if thread 2 has priority over thread 1 and thread 1 needs to change the variable. In practice, if it is required to access global variable by multiple threads, then they should be accessed using a mutex.

References:

<http://www.csc.villanova.edu/~mdamian/threads/posixthreads.html>

Computer Systems : A Programmer

This article is contributed by **Rahul Jain**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes 

Recommended Posts:

- Multithreading in C++
- Format specifiers in different Programming Languages
- C program to find square root of a given number
- C program to print odd line contents of a File followed by even line content
- std::string::rfind in C++ with Examples
- Difference between Process and Kernel Thread

- Getting System and Process Information Using C Programming and Shell in Linux
- Program to calculate Electricity Bill
- Program to print half Diamond star pattern
- C/C++ program for calling main() in main()
- Print all possible combinations of the string by replacing '\$' with any other digit from the string
- C++ File Writer-Reader application using Windows Threads
- How to use make utility to build C projects?
- How to call function within function in C or C++

Improved By : [CristianBurungiu](#), [PratikRoy](#), [DipakAgrawal](#), [abhikbose23](#)

Article Tags :

C
C-Library
cpp-multithreading
system-programming

Practice Tags :

C

26

To-do Done
2.7

Based on 34 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

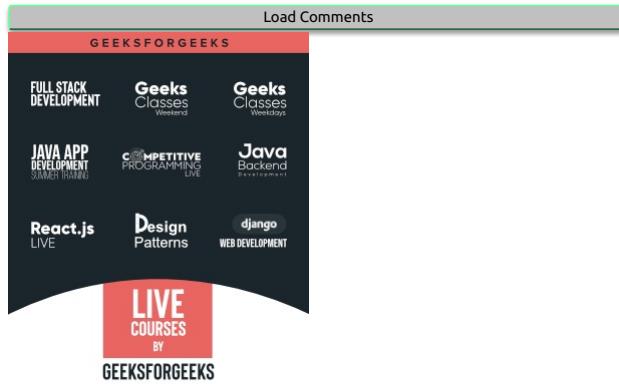
Previous

[first_page](#) C program to delete a file

Next

[last_page](#) Is it possible to call constructor and destructor explicitly?

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C
[Print all possible combinations of the string by replacing '\\$' with any other digit from the string](#)
[ctype.h & cctype.h library in C/C++ with Examples](#)
[How to call function within function in C or C++](#)
[Format specifiers in different Programming Languages](#)
[Predefined Macros in C with Examples](#)

More related articles in C
[C program to print odd line contents of a File followed by even line content](#)
[C program to find square root of a given number](#)
[Introduction to the C99 Programming Language : Part II](#)
[Problem in comparing Floating point numbers and how to compare them correctly?](#)
[Features of C Programming Language](#)

Assertions in C/C++

Assertions are statements used to test assumptions made by programmer. For example, we may use assertion to check if pointer returned by malloc() is NULL or not.

Following is syntax for assertion.

```
void assert( int expression );
```

If expression evaluates to 0 (false), then the expression, sourcecode filename, and line number are sent to the standard error, and then abort() function is called.

For example, consider the following program.

```
filter_none
edit
close

play_arrow

link
brightness_4
code

#include <stdio.h>
#include <assert.h>

int main()
{
```

```

int x = 7;

/* Some big code in between and let's say x
   is accidentally changed to 9 */
x = 9;

// Programmer assumes x to be 7 in rest of the code
assert(x==7);

/* Rest of the code */

return 0;
}

chevron_right
filter_none
Output

Assertion failed: x==7, file test.cpp, line 13
This application has requested the Runtime to terminate it in an unusual
way. Please contact the application's support team for more information.

```

Assertion Vs Normal Error Handling

Assertions are mainly used to check logically impossible situations. For example, they can be used to check the state a code expects before it starts running or state after it finishes running. Unlike normal error handling, assertions are generally disabled at run-time. Therefore, it is not a good idea to write statements in assert() that can cause side effects. For example writing something like assert(x = 5) is not a good idea as x is changed and this change won't happen when assertions are disabled. See [this](#) for more details.

Ignoring Assertions

In C/C++, we can completely remove assertions at compile time using the preprocessor NODEBUG.

```

filter_none
edit
close

play_arrow
link
brightness_4
code

// The below program runs fine because NDEBUG is defined
#define NDEBUG
#include <assert.h>

int main()
{
    int x = 7;
    assert (x==5);
    return 0;
}

```

The above program compiles and runs fine.

In Java, assertions are not enabled by default and we must pass an option to run-time engine to enable them.

Reference:

http://en.wikipedia.org/wiki/Assertion_%28software_development%29

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up

Add your personal notes
here! (max 5000 chars)

Save

Recommended Posts:

- Rust vs C++: Will Rust Replace C++ in Future ?
- Priority queue of pairs in C++ with ordering by first and second element
- Implementation of lower_bound() and upper_bound() in Vector of Pairs in C++
- Generating RGBA portable graphic images through C++
- std::basic_istream::ignore in C++ with Examples
- std::basic_istream::getline in C++ with Examples
- Format specifiers in different Programming Languages
- C program to find square root of a given number
- C program to print odd line contents of a File followed by even line content
- std::is_heap() in C++ with Examples
- std::basic_istream::gcount() in C++ with Examples
- new vs malloc() and free() vs delete in C++
- std::string::rfind in C++ with Examples
- Sum of factorials of Prime numbers in a Linked list

Improved By : [Jaideep Kaiwart](#)

Article Tags :

C

C++

Practice Tags :

C

CPP

thumb_up
10

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) Pure virtual destructor in C++

Next

[last_page](#) void pointer in C / C++

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

The screenshot shows the GeeksforGeeks homepage with a banner for 'LIVE COURSES BY GEEKSFORGEEKS'. Below the banner, there's a diagram of two people interacting with a large screen, representing live streaming or video courses. The page also features various course categories like Full Stack Development, Java App Development, React.js, Design Patterns, and Django.

Most popular in C
 Print all possible combinations of the string by replacing '\$' with any other digit from the string
 Format specifiers in different Programming Languages
 Predefined Macros in C with Examples
 C program to print odd line contents of a File followed by even line content
 C program to find square root of a given number

Most visited in C++
 Vector of Vectors in C++ STL with Examples
 C++ Tutorial
 Which C++ libraries are useful for competitive programming?
 Array of Vectors in C++ STL
 Map of Vectors in C++ STL with Examples

fork() in C

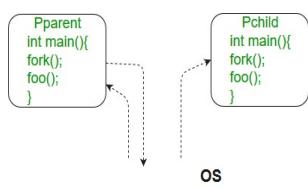
Fork system call is used for creating a new process, which is called **child process**, which runs concurrently with the process that makes the fork() call (parent process). After a new child process is created, both processes will execute the next instruction following the fork() system call. A child process uses the same pc(program counter), same CPU registers, same open files which use in the parent process.

It takes no parameters and returns an integer value. Below are different values returned by fork().

Negative Value: creation of a child process was unsuccessful.

Zero: Returned to the newly created child process.

Positive value: Returned to parent or caller. The value contains process ID of newly created child process.



Please note that the above programs don't compile in Windows environment.

- Predict the Output of the following program:

```
filter_none
edit
close
play_arrow
link
brightness_4
code

#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main()
{
    // make two process which run same
    // program after this instruction
    fork();

    printf("Hello world!\n");
    return 0;
}
```

chevron_right

filter_none

Output:

```
Hello world!
Hello world!
```

2. Calculate number of times hello is printed:

```
filter_none
edit
close
play_arrow
link
brightness_4
code

#include <stdio.h>
#include <sys/types.h>
int main()
{
    fork();
    fork();
    fork();
    printf("hello\n");
    return 0;
}
```

chevron_right

filter_none

Output:

```
hello
hello
hello
hello
hello
hello
hello
hello
```

The number of times 'hello' is printed is equal to number of process created. Total Number of Processes = 2^n , where n is number of fork system calls. So here n = 3, $2^3 = 8$

Let us put some label names for the three lines:

```
fork (); // Line 1
fork (); // Line 2
fork (); // Line 3

    L1      // There will be 1 child process
    / \    // created by line 1.
L2    L2 // There will be 2 child processes
/ \ / \
L3 L3 L3 L3 // There will be 4 child processes
              // created by line 3
```

So there are total eight processes (new child processes and one original process).

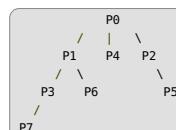
If we want to represent the relationship between the processes as a tree hierarchy it would be the following:

The main process: P0

Processes created by the 1st fork: P1

Processes created by the 2nd fork: P2, P3

Processes created by the 3rd fork: P4, P5, P6, P7



3. Predict the Output of the following program:

```
filter_none
edit
close
play_arrow
link
brightness_4
code

#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
void forkexample()
{
    // child process because return value zero
    if (fork() == 0)
        printf("Hello from Child!\n");

    // parent process because return value non-zero.
    else
        printf("Hello from Parent!\n");
}
int main()
{
    forkexample();
    return 0;
}
```

chevron_right

filter_none

Output:

```

1.
Hello from Child!
Hello from Parent!
(or)
2.
Hello from Parent!
Hello from Child!

```

In the above code, a child process is created. fork() returns 0 in the child process and positive integer in the parent process.

Here, two outputs are possible because the parent process and child process are running concurrently. So we don't know whether the OS will first give control to the parent process or the child process.

Important: Parent process and child process are running the same program, but it does not mean they are identical. OS allocate different data and states for these two processes, and the control flow of these processes can be different. See next example:

4. Predict the Output of the following program:

```

filter_none
edit
close

play_arrow
link
brightness_4
code

#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>

void forkexample()
{
    int x = 1;

    if (fork() == 0)
        printf("Child has x = %d\n", ++x);
    else
        printf("Parent has x = %d\n", --x);
}

int main()
{
    forkexample();
    return 0;
}
chevron_right
filter_none

```

Output:

```

Parent has x = 0
Child has x = 2
(or)
Child has x = 2
Parent has x = 0

```

Here, global variable change in one process does not affect two other processes because data/state of two processes are different. And also parent and child run simultaneously so two outputs are possible.

fork() vs exec()

The fork system call creates a new process. The new process created by fork() is a copy of the current process except for the returned value. The exec() system call replaces the current process with a new program.

Exercise:

1. A process executes the following code:

```

filter_none
edit
close

play_arrow
link
brightness_4
code

for (i = 0; i < n; i++)
    fork();
chevron_right
filter_none

```

The total number of child processes created is: (GATE-CS-2008)

- (A) n
- (B) $2^n - 1$
- (C) 2^n
- (D) $2^{(n+1)} - 1$

See [this](#) for solution.

2. Consider the following code fragment:

```

filter_none
edit
close

play_arrow
link
brightness_4
code

if (fork() == 0) {
    a = a + 5;
    printf("%d, %d\n", a, &a);
}
else {
    a = a -5;
    printf("%d, %d\n", a, &a);
}

```

chevron_right

filter_none

Let u, v be the values printed by the parent process, and x, y be the values printed by the child process. Which one of the following is TRUE? (GATE-CS-2005)

- (A) $u = x + 10$ and $v = y$
- (B) $u = x + 10$ and $v \neq y$
- (C) $u + 10 = x$ and $v = y$
- (D) $u + 10 = x$ and $v \neq y$

See [this](#) for solution.

3. Predict output of below program.

filter_none

edit

close

play_arrow

link

brightness_4

code

```
#include <stdio.h>
#include <unistd.h>
int main()
{
    fork();
    fork() && fork() || fork();
    fork();

    printf("forked\n");
    return 0;
}
```

chevron_right

filter_none

See [this](#) for solution

Related Articles :

[C program to demonstrate fork\(\) and pipe\(\)](#)

[Zombie and Orphan Processes in C](#)

[fork\(\) and memory shared b/w processes created using it](#).

References:

<http://www.csl.mtu.edu/cs4411.ck/www/NOTES/process/fork/create.html>

This article is contributed by **Team GeeksforGeeks** and **Kadam Patel**. If you like GeeksforGeeks and would like to contribute, you can also write an article using [contribute.geeksforgeeks.org](#) or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes *arrow_drop_up*

Recommended Posts:

- sorting in fork()
- searching in fork()
- Fork() Bomb
- fork() and Binary Tree
- Difference between fork() and exec()
- C vs BASH Fork bomb
- Fork() - Practice questions
- Creating multiple process using fork()
- Factorial calculation using fork() in C for Linux
- C program to demonstrate fork() and pipe()
- Creating child process using fork() in Python
- fork() and memory shared b/w processes created using it
- Calculation in parent and child process using fork()
- fork() to execute processes from bottom to up using wait()
- Create n-child process from same parent process using fork() in C

Improved By : [FunniestClown](#), [AndreiSas](#), [reddydheerajreddy](#), [4slan](#), [jimbotherisenclo](#)

Article Tags :

[C](#)
[C-Library](#)
[system-programming](#)

Practice Tags :

[C](#)

thumb_up

30

To-do Done
2.8

Based on 27 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

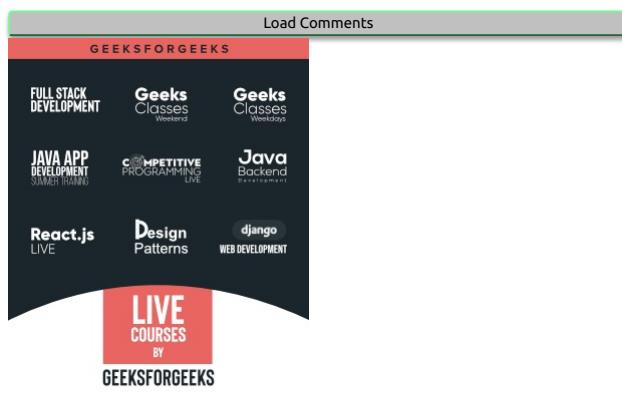
Post navigation

Previous

[first_page](#) Write your own strlen() for a long string padded with '\0's

Next

[last_page](#) Functions in C/C++



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
ctype.h<ctype.h> library in C++ with Examples
How to use printf function in C
Format specifiers in different Programming Languages
Predefined Macros in C with Examples

More related articles in C
C program to print odd line contents of a File followed by even line content
C program to find square root of a given number
Introduction to the C99 Programming Language : Part II
Problem in comparing Floating point numbers and how to compare them correctly?
Features of C Programming Language

Interesting Facts in C Programming

Below are some interesting facts about C programming:

1) The case labels of a switch statement can occur inside if-else statements.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

#include <stdio.h>

int main()
{
    int a = 2, b = 2;
    switch(a)
    {
        case 1:
            ;
            if (b==5)
            {
                case 2:
                    printf("GeeksforGeeks");
                }
            else case 3:
            {
                ;
            }
    }
}
```

chevron_right

filter_none

Output :

GeeksforGeeks

2) arr[index] is same as index[arr]

The reason for this to work is, array elements are accessed using pointer arithmetic.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// C program to demonstrate that arr[0] and
// 0[arr]
#include<stdio.h>
```

```
int main()
{
    int arr[10];
    arr[0] = 1;
    printf("%d", 0[arr] );
    return 0;
}
```

chevron_right

filter_none

Output :

1

3) We can use '<; :>' in place of '[,]' and '<%, %>' in place of '{,}'

filter_none

edit

close

play_arrow

link

brightness_4

code

```
#include<stdio.h>
```

```
int main()
```

```
<%
    int arr <:10:>;
    arr<:0:> = 1;
    printf("%d", arr<:0:>);

    return 0;
%>
```

chevron_right

filter_none

Output :

1

4) Using #include in strange places.

Let "a.txt" contains ("GeeksforGeeks");

filter_none

edit

close

play_arrow

link

brightness_4

code

```
#include<stdio.h>
```

```
int main()
```

```
{
```

```
    printf
```

```
    #include "a.txt"
```

```
;
```

```
}
```

chevron_right

filter_none

Output :

GeeksforGeeks

5) We can ignore input in scanf() by using '*' after '%' in format specifiers

filter_none

edit

close

play_arrow

link

brightness_4

code

```
#include<stdio.h>
```

```
int main()
```

```
{
```

```
    int a;
```

```
// Let we input 10 20, we get output as 20
```

```
// (First input is ignored)
```

```
// If we remove * from below line, we get 10.
```

```
scanf("%*d%d", &a);
```

```
printf( "%d ", a);
```

```
return 0;
}
```

chevron_right

filter_none

This article is contributed by **Harsh Agarwal**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes

Add your personal notes here! (max 5000 chars)

[Save](#)

Recommended Posts:

- Py-Facts - 10 interesting facts about Python
- Interesting Facts about C++
- Interesting Facts about C#
- Interesting Facts about Ubuntu
- Interesting Facts about Linux
- Interesting Facts About Java
- C++ bitset interesting facts
- Interesting facts about C Language
- Some Interesting facts about default arguments in C++
- Interesting Facts about Macros and Preprocessors in C
- Interesting facts about Fibonacci numbers
- Interesting facts about switch statement in C
- Interesting facts about strings in Python | Set 1
- Interesting facts about null in Java
- Interesting facts about strings in Python | Set 2 (Slicing)

Article Tags :

C
C-Input and Output Quiz

c-input-output
C-Loops & Control Statements

interesting-facts

Practice Tags :

C



4

To-do Done
2.2

Based on 15 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) Dangling, Void , Null and Wild Pointers

Next

[last_page](#) auto_ptr, unique_ptr, shared_ptr and weak_ptr

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Weekend Geeks Classes Weekend

JAVA APP DEVELOPMENT SUMMER TRAINING COMPETITIVE PROGRAMMING DIVE Java Backend Development

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS

Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
Predefined Macros in C with Examples
Format specifiers in different Programming Languages
C program to print odd line contents of a File followed by even line content
C program to find square root of a given number

More related articles in C
Introduction to the C99 Programming Language : Part II
Problem in comparing Floating point numbers and how to compare them correctly
Features of C Programming Language
Pointer Expressions in C with Examples
Introduction to the C99 Programming Language : Part III

Precision of floating point numbers in C++ (`floor()`, `ceil()`, `trunc()`, `round()` and `setprecision()`)

Decimal equivalent of 1/3 is 0.333333333333.... An infinite length number would require infinite memory to store, and we typically have 4 or 8 bytes. Therefore, Floating point numbers store only a certain number of significant digits, and the rest are lost. The **precision** of a floating point number defines how many significant digits it can represent without information loss. When outputting floating point numbers, cout has a default precision of 6 and it truncates anything after that.

Given below are few libraries and methods which are used to provide precision to floating point numbers in C++:

floor():

Floor rounds off the given value to the closest integer which is less than the given value.

filter_none
edit
close

play_arrow

link
brightness_4
code

```
// C++ program to demonstrate working of floor()
// in C/C++
#include<bits/stdc++.h>
using namespace std;

int main()
{
    double x = 1.411, y = 1.500, z = 1.711;
    cout << floor(x) << endl;
    cout << floor(y) << endl;
    cout << floor(z) << endl;

    double a = -1.411, b = -1.500, c = -1.611;
    cout << floor(a) << endl;
    cout << floor(b) << endl;
    cout << floor(c) << endl;
    return 0;
}
```

chevron_right

filter_none

Output:

```
1
1
1
-2
-2
-2
```

ceil():

Ceil rounds off the given value to the closest integer which is more than the given value.

filter_none

edit

close

play_arrow

link
brightness_4

code

chevron_right

filter_none

Output:

```
2
2
2
-1
-1
-1
```

trunc():

Trunc rounds removes digits after decimal point.

filter_none

edit

close

play_arrow

link
brightness_4

code

```
{  
    double x = 1.411, y = 1.500, z = 1.611;  
    cout << trunc(x) << endl;  
    cout << trunc(y) << endl;  
    cout << trunc(z) << endl;  
  
    double a = -1.411, b = -1.500, c = -1.611;  
    cout << trunc(a) << endl;  
    cout << trunc(b) << endl;  
    cout << trunc(c) << endl;  
    return 0;  
}  
chevron_right
```

[filter_none](#)

Output:

```
1  
1  
1  
-1  
-1  
-1
```

round():

Rounds given number to the closest integer.

[filter_none](#)

[edit](#)

[close](#)

[play_arrow](#)

[link](#)

[brightness_4](#)

[code](#)

```
// C++ program to demonstrate working of round()  
// in C/C++  
#include<bits/stdc++.h>  
using namespace std;
```

```
int main()  
{  
    double x = 1.411, y = 1.500, z = 1.611;  
  
    cout << round(x) << endl;  
    cout << round(y) << endl;  
    cout << round(z) << endl;  
  
    double a = -1.411, b = -1.500, c = -1.611;  
    cout << round(a) << endl;  
    cout << round(b) << endl;  
    cout << round(c) << endl;  
    return 0;  
}
```

chevron_right

[filter_none](#)

Output:

```
1  
2  
2  
-1  
-2  
-2
```

setprecision():

Setprecision when used along with 'fixed' provides precision to floating point numbers correct to decimal numbers mentioned in the brackets of the setprecision.

[filter_none](#)

[edit](#)

[close](#)

[play_arrow](#)

[link](#)

[brightness_4](#)

[code](#)

```
// C++ program to demonstrate working of setprecision()  
// in C/C++  
#include<bits/stdc++.h>  
using namespace std;
```

```
int main()  
{  
    double pi = 3.14159, npi = -3.14159;  
    cout << fixed << setprecision(0) << pi << " <<npi<<endl;  
    cout << fixed << setprecision(1) << pi << " <<npi<<endl;  
    cout << fixed << setprecision(2) << pi << " <<npi<<endl;  
    cout << fixed << setprecision(3) << pi << " <<npi<<endl;  
    cout << fixed << setprecision(4) << pi << " <<npi<<endl;  
    cout << fixed << setprecision(5) << pi << " <<npi<<endl;  
    cout << fixed << setprecision(6) << pi << " <<npi<<endl;  
}
```

chevron_right

[filter_none](#)

Output:

```
3 -3  
3.1 -3.1  
3.14 -3.14  
3.142 -3.142  
3.1416 -3.1416  
3.14159 -3.14159  
3.141590 -3.141590
```

Note: When the value mentioned in the setprecision() exceeds the number of floating point digits in the original number then 0 is appended to floating point digit to match the precision mentioned by the user.

There exists other methods too to provide precision to floating point numbers. The above mentioned are few of the most commonly used methods to provide precision to floating point numbers during competitive coding.

This article is contributed by **Aditya Gupta**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes 

Add your personal notes here! (max 5000 chars)

Recommended Posts:

- [Write a one line C function to round floating point numbers](#)
- [Can we use % operator on floating point numbers?](#)
- [C Program to Multiply two Floating Point Numbers](#)
- [Problem in comparing Floating point numbers and how to compare them correctly?](#)
- [Ceil and Floor functions in C++](#)
- [Finding Floor and Ceil of a Sorted Array using C++ STL](#)
- [Floating Point Operations & Associativity in C, C++ and Java](#)
- [Convert a floating point number to string in C](#)
- [How to count set bits in a floating point number in C?](#)
- [Rounding Floating Point Number To two Decimal Places in C and C++](#)
- [C++ Floating Point Manipulation \(fmod\(\), remainder\(\), remquo\(\) ... in cmath\)](#)
- [trunc\(\), truncf\(\), truncl\(\) in C language](#)
- [Setting decimal precision in C](#)
- [Measure execution time with high precision in C/C++](#)
- [sizeof\(\) for Floating Constant in C](#)

Article Tags :

C
C++
Technical Scripter
CPP-Library
Data Type
Operators
Practice Tags :
Operators
Data Type
C
CPP

 30

To-do Done
2.2

Based on 17 vote(s)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) Downloading gists from Github made simple

Next

[last_page](#) Namedtuple in Python

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
How to call function within function in C or C++
Format specifiers in different Programming Languages
Predefined Macros in C with Examples
C program to print odd line contents of a File followed by even line content

Most visited in C++
Vector of Vectors in C++ STL with Examples
C++ STL
Which C++ libraries are useful for competitive programming?
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

Concept of setjump and longjmp in C

"Setjump" and "Longjump" are defined in `setjmp.h`, a header file in C standard library.

- **`setjmp(jmp_buf buf)`** : uses buf to remember current position and returns 0.
- **`longjmp(jmp_buf buf, i)`** : Go back to place buf is pointing to and return i .

```
filter_none
edit
close

play_arrow

link
brightness_4
code

// A simple C program to demonstrate working of setjmp() and longjmp()
#include<stdio.h>
#include<setjmp.h>
jmp_buf buf;
void func()
{
    printf("Welcome to GeeksforGeeks\n");

    // Jump to the point setup by setjmp
    longjmp(buf, 1);

    printf("Geek2\n");
}

int main()
{
    // Setup jump position using buf and return 0
    if (setjmp(buf))
        printf("Geek3\n");
    else
    {
        printf("Geek4\n");
        func();
    }
    return 0;
}
chevron_right
filter_none
Output :

Geek4
Welcome to GeeksforGeeks
Geek3
```

The main feature of these functions is to provide a way that deviates from standard call and return sequence. This is mainly used to implement exception handling in C. `setjmp` can be used like `try` (in languages like C++ and Java). The call to `longjmp` can be used like `throw` (Note that `longjmp()` transfers control to the point set by `setjmp()`).

This article is contributed by **Aditya Chatterjee**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes

Save

Recommended Posts:

- Rust vs C++: Will Rust Replace C++ in Future ?
- std::basic_istream::ignore in C++ with Examples
- std::basic_istream::getline in C++ with Examples
- Format specifiers in different Programming Languages
- C program to find square root of a given number
- C program to print odd line contents of a File followed by even line content
- std::is_heap() in C++ with Examples
- std::basic_istream::gcount() in C++ with Examples
- new vs malloc() and free() vs delete in C++
- std::string::rfind in C++ with Examples
- Sum of factorials of Prime numbers in a Linked list
- Sum of all Palindrome Numbers present in a Linked list
- Generate an array of given size with equal count and sum of odd and even numbers
- Namespaces in C++ | Set 4 (Overloading, and Exchange of Data in different Namespaces)

Article Tags :

C
C++
C-Library
CPP-Library

Practice Tags :

C
CPP

thumb_up
5

To-do Done
2.3

Based on 3 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

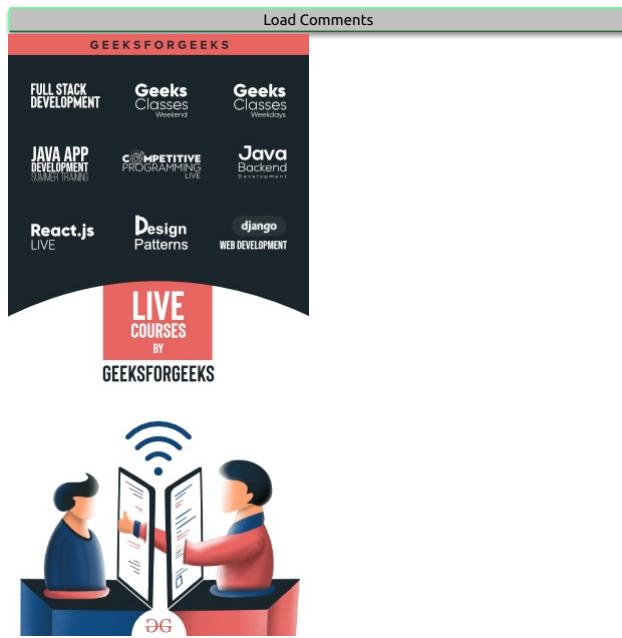
Previous

first_page A C Puzzle

Next

last_page memcpy() in C/C++

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.



nextafter() and nexttoward() in C/C++

How would you solve below problems in C/C++?

- What is the smallest representable positive floating point number in C/C++?
- What is the largest representable negative floating point number in C/C++?
- Given a positive floating point number x, find the largest representable floating point value smaller than x?

nextafter(x, y) and nexttoward(x,y)

In C and C++, both `nextafter(x, y)` and `nexttoward(x,y)` are similar functions defined in `math.h` or `cmath` header files. They both return next representable value after x in the direction of y. `nexttoward()` has more precise second parameter y.

The following program demonstrates the concept :

[filter none](#)
[edit](#)
[close](#)

```

play_arrow
link
brightness_4
code

// C program to demonstrate use of nextafter() and nexttoward()

#include <stdio.h>
#include <math.h>
int main ()
{
    // using nextafter
    printf ("Smallest positive floating point number : %e\n",
           nextafter(0.0, 1.0));
    printf ("Largest negative floating point number :%e\n",
           nextafter(0.0, -1.0));
    printf ("Largest positive floating point number smaller than 0.5 : %e\n",
           nextafter(0.5, 0.0));

    // using nexttoward
    printf ("Smallest positive floating point number : %e\n",
           nexttoward(0.0, 1.0));
    printf ("Largest negative floating point number : %e\n",
           nexttoward(0.0, -1.0));
    printf ("Largest positive floating point number smaller than 0.5 : %e\n",
           nexttoward(0.5, 0.0));
    return (0);
}

```

chevron_right

filter_none

Output:

```

nextafter first value greater than zero: 4.940656e-324
nextafter first value less than zero: -4.940656e-324
nexttoward first value greater than zero: 4.940656e-324
nexttoward first valnextafter first value greater than zero: 4.940656e-324
nexttoward first value less than zero: -4.940656e-324
nexttoward first value greater than zero: 4.940656e-324
nexttoward first value less than zero: -4.940656e-324 ue less than zero: -4.940656e-324

```

This article is contributed by **Aditya Chatterjee**. If you like GeeksforGeeks and would like to contribute, you can also write an article and mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes *arrow_drop_up*

Add your personal notes
here! (max 5000 chars)

Recommended Posts:

- [std::basic_istream::ignore in C++ with Examples](#)
- [std::basic_istream::getline in C++ with Examples](#)
- [Format specifiers in different Programming Languages](#)
- [C program to find square root of a given number](#)
- [C program to print odd line contents of a File followed by even line content](#)
- [std::is_heap\(\) in C++ with Examples](#)
- [std::basic_istream::gcount\(\) in C++ with Examples](#)
- [new vs malloc\(\) and free\(\) vs delete in C++](#)
- [std::string::rfind in C++ with Examples](#)
- [Sum of factorials of Prime numbers in a Linked list](#)
- [Sum of all Palindrome Numbers present in a Linked list](#)
- [Generate an array of given size with equal count and sum of odd and even numbers](#)
- [Namespaces in C++ | Set 4 \(Overloading, and Exchange of Data in different Namespaces\)](#)
- [Getting System and Process Information Using C Programming and Shell in Linux](#)

Article Tags :

C
C++
CPP-Library
Practice Tags :
C
CPP

thumb_up
Be the First to upvote.

To-do Done
1

Based on 1 vote(s)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

first_page Compound Literals in C

Next

last_page Anonymous Union and Structure in C

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
ctype.h<ctype> library in C/C++ with Examples
Predefined Macros in C with Examples
Implicit Type Conversion in C with Examples
How to call function within function in C or C++
Most visited in C++
Vector of Vectors in C++ STL with Examples
C++ STL with Examples
Which C++ libraries are useful for competitive programming?
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

pthread_cancel() in C with example

prerequisite: Multithreading, pthread_self() in C with Example

pthread_cancel() = This function cancel a particular thread using thread id. This function send a cancellation request to the thread.

Syntax : - int pthread_cancel(pthread_t thread);

First Program : - Cancel self thread

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// C program to demonstrates cancellation of self thread
// using thread id
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>

void* calls(void* ptr)
{
    printf("GeeksForGeeks");

    // To exit the current thread
    // pthread_self() return the particular thread id
    pthread_cancel(pthread_self());

    return NULL;
}

int main()
{
    // NULL when no attribute
    pthread_t thread;

    // calls is a function name
    pthread_create(&thread, NULL, calls, NULL);

    // Waiting for when thread is completed
    pthread_join(thread, NULL);

    return 0;
}
chevron_right
filter_none
```

Output:

```
GeeksForGeeks
```

If you use Linux then compile this program **gcc program_name.c -lpthread**

Second Program : - Cancel other thread

```

filter_none
edit
close
play_arrow
link
brightness_4
code

// C program to demonstrates cancellation of another thread
// using thread id
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <pthread.h>

// To Count
int counter = 0;

// for temporary thread which will be
// store thread id of second thread
pthread_t tmp_thread;

// thread_one call func
void* func(void* p)
{
    while (1) {

        printf("thread number one\n");
        sleep(1); // sleep 1 second
        counter++;

        // for exiting if counter == 5
        if (counter == 2) {

            // for cancel thread_two
            pthread_cancel(tmp_thread);

            // for exit from thread_one
            pthread_exit(NULL);
        }
    }
}

// thread_two call func2
void* func2(void* p)
{

    // store thread_two id to tmp_thread
    tmp_thread = pthread_self();

    while (1) {
        printf("thread Number two");
        sleep(1); // sleep 1 second
    }
}

// Driver code
int main()
{

    // declare two thread
    pthread_t thread_one, thread_two;

    // create thread_one
    pthread_create(&thread_one, NULL, func, NULL);

    // create thread_two
    pthread_create(&thread_two, NULL, func2, NULL);

    // waiting for when thread_one is completed
    pthread_join(thread_one, NULL);

    // waiting for when thread_two is completed
    pthread_join(thread_two, NULL);

}
chevron_right

```

Output:

```

thread number one
thread number two
thread number one
thread number two

```

If you use Linux then compile this program **gcc program_name.c -lpthread**

This article is contributed by **Devanshu Agarwal**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes *arrow_drop_up*

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- Format specifiers in different Programming Languages
- C program to find square root of a given number
- C program to print odd line contents of a File followed by even line content
 - std::string::find in C++ with Examples
- Getting System and Process Information Using C Programming and Shell in Linux
- Program to calculate Electricity Bill
- Program to print half Diamond star pattern
- C/C++ program for calling main() in main()
- Print all possible combinations of the string by replacing '\$' with any other digit from the string
- How to use make utility to build C projects?
- How to call function within function in C or C++
- Role of Semicolon in various Programming Languages
- C program to display month by month calendar for a given year
- Difference between Type Casting and Type Conversion
- Pi(π) in C++ with Examples

Article Tags :

C

C-Library

Practice Tags :

C



Be the First to upvote.

To-do Done

4.6

Based on 3 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

first_page pthread_equal() in C with example

Next

last_page Encode an ASCII string into Base-64 Format

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments

FULL STACK DEVELOPMENT

Geeks Classes Weekend

Geeks Classes Weekdays

JAVA APP DEVELOPMENT SUMMER TRAINING

COMPETITIVE PROGRAMMING LIVE

Java Backend Development

React.js LIVE

Design Patterns

django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS



Most popular in C

Print all possible combinations of the string by replacing '\$' with any other digit from the string
ctype.h <cmath> library in C/C++ with Examples
Predefining Macros in C with Examples
Implicit Type Conversion in C with Examples
Format specifiers in different Programming Languages

More related articles in C

How to call function within function in C or C++
C program to print odd line contents of a File followed by even line content
Introduction to the C99 Programming Language : Part II
C program to find square root of a given number
Problem in comparing Floating point numbers and how to compare them correctly?

pthread_equal() in C with example

Prerequisite : Multithreading, pthread_self() in C with Example

pthread_equal() = This compares two thread which is equal or not. This function compares two thread identifiers. It return '0' and non zero value. If it is equal then return non zero value else return 0.

Syntax:- `int pthread_equal (pthread_t t1, pthread_t t2);`

First Method :- Compare with self thread

```

filter_none
edit
close
play_arrow
link
brightness_4
code

// C program to demonstrate working of pthread_equal()
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <pthread.h>

pthread_t tmp_thread;

void* func_one(void* ptr)
{
    // in this field we can compare two thread
    // pthread_self gives a current thread id
    if (pthread_equal(tmp_thread, pthread_self())) {
        printf("equal\n");
    } else {
        printf("not equal\n");
    }
}

// driver code
int main()
{
    // thread one
    pthread_t thread_one;

    // assign the id of thread one in temporary
    // thread which is global declared    r
    tmp_thread = thread_one;

    // create a thread
    pthread_create(&thread_one, NULL, func_one, NULL);

    // wait for thread
    pthread_join(thread_one, NULL);
}
chevron_right

```

Output:

`equal`

Second Method :- Compare with other thread

```

filter_none
edit
close
play_arrow
link
brightness_4
code

// C program to demonstrate working of pthread_equal()
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <pthread.h>

// global declared pthread_t variable
pthread_t tmp_thread;

void* func_one(void* ptr)
{
    tmp_thread = pthread_self(); // assign the id of thread one in
    // temporary thread which is global declared
}

void* func_two(void* ptr)
{
    pthread_t thread_two = pthread_self();

    // compare two thread
    if (pthread_equal(tmp_thread, thread_two)) {
        printf("equal\n");
    } else {
        printf("not equal\n");
    }
}

int main()
{
    pthread_t thread_one, thread_two;

```

```

// creating thread one
pthread_create(&thread_one, NULL, func_one, NULL);

// creating thread two
pthread_create(&thread_two, NULL, func_two, NULL);

// wait for thread one
pthread_join(thread_one, NULL);

// wait for thread two
pthread_join(thread_two, NULL);
}

```

chevron_right

filter_none

Output:

not equal

This article is contributed by **Devanshu Agarwal**. If you like GeeksforGeeks and would like to contribute, you can also write an article using [contribute.geeksforgeeks.org](#) or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- Format specifiers in different Programming Languages
- C program to find square root of a given number
- C program to print odd line contents of a File followed by even line content
- std::string::find in C++ with Examples
- Getting System and Process Information Using C Programming and Shell in Linux
- Program to calculate Electricity Bill
- Program to print half Diamond star pattern
- C/C++ program for calling main() in main()
- Print all possible combinations of the string by replacing '\$' with any other digit from the string
- How to use make utility to build C projects?
- How to call function within function in C or C++
- Role of SemiColon in various Programming Languages
- C program to display month by month calendar for a given year
- Difference between Type Casting and Type Conversion
- Pi(π) in C++ with Examples

Improved By : [nidhi_biet](#)

Article Tags :

C

C-Library

Practice Tags :

C

thumb_up

Be the First to upvote.

To-do Done
4.5

Based on 4 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) [_Noreturn function specifier in C](#)

Next

[last_page](#) [pthread_cancel\(\) in C with example](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
ctype.h <ctype.h> library in C/C++ with Examples
Predefined Macros in C with Examples
Implicit Type Conversion in C with Examples
Format specifiers in different Programming Languages

More related articles in C
How to call function within function in C or C++
C program to print odd line contents in file followed by even line content
C program to find square root of a given number
Introduction to the C99 Programming Language : Part II
Problem in comparing Floating point numbers and how to compare them correctly?

pthread_self() in C with Example

Prerequisite : [Multithreading in C](#)

Syntax :- `pthread_t pthread_self(void);`

The `pthread_self()` function returns the ID of the thread in which it is invoked.

```
filter_none
edit
close

play_arrow

link
brightness_4
code

// C program to demonstrate working of pthread_self()
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
void* calls(void* ptr)
{
    // using pthread_self() get current thread id
    printf("In function \nthread id = %d\n", pthread_self());
    pthread_exit(NULL);
    return NULL;
}

int main()
{
    pthread_t thread; // declare thread
    pthread_create(&thread, NULL, calls, NULL);
    printf("In main \nthread id = %d\n", thread);
    pthread_join(thread, NULL);
    return 0;
}
chevron_right
filter_none
```

Output:

```
In function
thread id = 1
In main
thread id = 1
```

This article is contributed by **Devanshu Agarwal**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes [arrow_drop_up](#)

Add your personal notes here! (max 5000 chars)

[Save](#)

Recommended Posts:

- Format specifiers in different Programming Languages
- C program to find square root of a given number
- C program to print odd line contents of a File followed by even line content
- Getting System and Process Information Using C Programming and Shell in Linux
- Program to calculate Electricity Bill
- Program to print half Diamond star pattern
- C/C++ program for calling main() in main()
- Print all possible combinations of the string by replacing '\$' with any other digit from the string
- How to use make utility to build C projects?
- How to call function within function in C or C++
- Role of Semicolon in various Programming Languages
- C program to display month by month calendar for a given year
- Difference between Type Casting and Type Conversion
- Pi(π) in C++ with Examples
- Output of C programs | Set 66 (Accessing Memory Locations)

Improved By : shubham_singh

Article Tags :

C
Practice Tags :
C

thumb_up
Be the First to upvote.

To-do Done
4.6

Based on 3 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

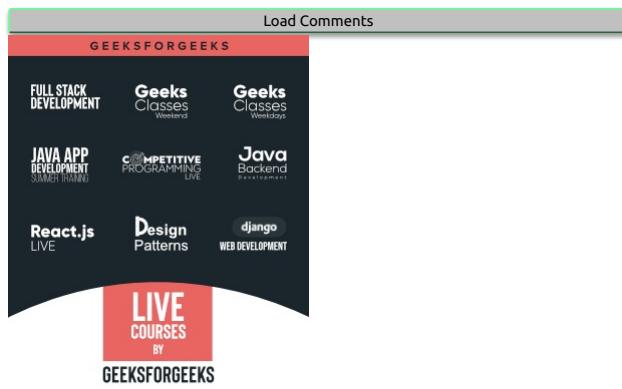
Previous

first_page Why variable name does not start with numbers in C ?

Next

last_page tmpfile() function in C

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
ctype.h Header Library in C/C++ with Examples
Predefining Macros in C with Examples
Implicit Type Conversion in C with Examples
Format specifiers in different Programming Languages

More related articles in C
How to call function within function in C or C++
C program to print odd line contents of a File followed by even line content
Introduction to the C99 Programming Language : Part II
C program to find square root of a given number
Problem in comparing Floating point numbers and how to compare them correctly?

Local Labels in C

Everybody who has programmed in C programming language must be aware about "goto" and "labels" used in C to jump within a C function. GCC provides an extension to C called "local labels".

Conventional Labels vs Local Labels

Conventional labels in C have function scope. Whereas local label can be scoped to an inner nested block. Therefore, conventional labels cannot be declared more than once in a function and that is where local labels are used.

A label can appear more than once in function when the label is inside a macro and macro is expanded multiple times in a function. For example if a macro funcMacro() has definition which involves jump instructions (goto statement) within the block and funcMacro is used by a function foo() several times.

```
filter_none
edit
close

play_arrow
link
brightness_4
code

#define funcMacro(params ...)
```

```

do {
    if (cond == true)
        \
        \
    <some code >
        \
        \
x:
    <some code>
        \
} while(0);                                \

```

```

Void foo()
{
    <some code>
    funcMacro(params ...);
    <some code >
    funcMacro(params...);
}

```

chevron_right

filter_none

In such a function foo() , the function macro will be expanded two times.

This will result in having more than one definition of label 'x' in a function, which leads to confusion to the compiler and results in compilation error. In such cases local labels are useful.

The above problem can be avoided by using local labels. A local label are declared as below:

label label;

Local label declarations must come at the beginning of the block, before any ordinary declarations or statements.

Below is C example where a macro IS_STR_EMPTY() is expanded multiple times. Since local labels have block scope and every expansion of macro causes a new do while block, the program compiles and runs fine.

```

filter_none
edit
close
play_arrow
link
brightness_4
code

#include <stdio.h>
#include <string.h>

//Function macro using local labels
#define IS_STR_EMPTY(str) \
do { \
    __label__ empty, not_empty, exit; \
    if (strlen(str)) \
        goto not_empty; \
    else \
        goto empty; \
    not_empty: \
        printf("string = %s\n", str); \
        goto exit; \
    empty: \
        printf("string is empty\n"); \
    exit: ; \
} while(0); \

```

```

int main()
{
    char string[20] = {'\0'};

    //Pass empty string to Macro function
    IS_STR_EMPTY(string);

```

```

    //Pass non-empty string to Macro function
    strcpy(string, "geeksForgeeks");
    IS_STR_EMPTY(string);

```

return 0;

chevron_right

filter_none

Output:

```

string is empty
string = geeksForgeeks

```

This article is contributed by **Kashish Bhatia**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes **arrow_drop_up**

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- Can we access global variable if there is a local variable with same name?
- Data type of case labels of switch statement in C++?
- Local Classes in C++

- Format specifiers in different Programming Languages
- C program to find square root of a given number
- C program to print odd line contents of a File followed by even line content
- Getting System and Process Information Using C Programming and Shell in Linux
- Program to calculate Electricity Bill
- Program to print half Diamond star pattern
- C/C++ program for calling main() in main()
- Print all possible combinations of the string by replacing '\$' with any other digit from the string
- How to use make utility to build C projects?
- How to call function within function in C or C++
- Role of SemiColon in various Programming Languages

Article Tags :

C

Practice Tags :

C



Be the First to upvote.

To-do Done
0

No votes yet.

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page Bit Fields in C](#)

Next

[last_page What are the differences between bitwise and logical AND operators in C/C++?](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

[Load Comments](#)

FULL STACK DEVELOPMENT
Geeks Classes Weekend
Geeks Classes Weekdays

JAVA APP DEVELOPMENT SUMMER TRAINING
COMPETITIVE PROGRAMMING LIVE
Java Backend Development

React.js LIVE
Design Patterns
django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS



Most popular in C
[Print all possible combinations of the string by replacing '\\$' with any other digit from the string](#)
[Predefined Macros in C with Examples](#)
[Format specifiers in different Programming Languages](#)
[How to call function within function in C or C++](#)
[C program to print odd line contents of a file followed by even line content](#)

More related articles in C

[Introduction to the C99 Programming Language : Part II](#)
[C program to find square root of a given number](#)
[Program in comparing Floating point numbers and how to compare them correctly?](#)
[Features of C Programming Language](#)
[Pointer Expressions in C with Examples](#)

Ivalue and rvalue in C language

I-value: "I-value" refers to memory location which identifies an object. I-value may appear as either left hand or right hand side of an assignment operator(=). I-value often represents as identifier.

Expressions referring to modifiable locations are called "**modifiable I-values**". A modifiable I-value cannot have an array type, an incomplete type, or a type with the **const** attribute. For structures and unions to be modifiable **Ivalues**, they must not have any members with the **const** attribute. The name of the identifier denotes a storage location, while the value of the variable is the value stored at that location.

An identifier is a modifiable **Ivalue** if it refers to a memory location and if its type is arithmetic, structure, union, or pointer. For example, if ptr is a pointer to a storage region, then ***ptr** is a modifiable I-value that designates the storage region to which **ptr** points.

In C, the concept was renamed as "**locator value**", and referred to expressions that locate (designate) objects. The I-value is one of the following:

1. The name of the variable of any type i.e, an identifier of integral, floating, pointer, structure, or union type.
2. A subscript ([]) expression that does not evaluate to an array.
3. A unary-indirection (*) expression that does not refer to an array.
4. An I-value expression in parentheses.
5. A **const** object (a nonmodifiable I-value).
6. The result of indirection through a pointer, provided that it isn't a function pointer.
7. The result of member access through pointer(> or .)

filter_none

edit

close

play_arrow

link

brightness_4

```

code
// declare a an object of type 'int'
int a;

// a is an expression referring to an
// 'int' object as l-value
a = 1;

int b = a; // Ok, as l-value can appear on right

// Switch the operand around '=' operator
9 = a;

// Compilation error:
// as assignment is trying to change the
// value of assignment operator
chevron_right
filter_none

```

R-value: r-value" refers to data value that is stored at some address in memory. A r-value is an expression that can't have a value assigned to it which means r-value can appear on right but not on left hand side of an assignment operator(=).

```

filter_none
edit
close
play_arrow
link
brightness_4
code
// declare a, b an object of type 'int'
int a = 1, b;

a + 1 = b; // Error, left expression is
            // is not variable(a + 1)

// declare pointer variable 'p', and 'q'
int *p, *q; // *p, *q are lvalue

*p = 1; // valid l-value assignment

// below is invalid - "p + 2" is not an l-value
// p + 2 = 18;

q = p + 5; // valid - "p + 5" is an r-value

// Below is valid - dereferencing pointer
// expression gives an l-value
*(p + 2) = 18;

p = &b;

int arr[20]; // arr[12] is an lvalue; equivalent
              // to *(arr+12)
              // Note: arr itself is also an lvalue

struct S { int m; };

struct S obj; // obj and obj.m are lvalues

// ptr-> is an lvalue; equivalent to (*ptr).m
// Note: ptr and *ptr are also lvalues
struct S* ptr = &obj;
chevron_right
filter_none

```

Note: The unary & (address-of) operator requires an lvalue as its operand. That is, &n is a valid expression only if n is an lvalue. Thus, an expression such as &12 is an error. Again, 12 does not refer to an object, so it's not addressable. For instance,

```

filter_none
edit
close
play_arrow
link
brightness_4
code
// declare a as int variable and
// 'p' as pointer variable
int a, *p;

p = &a; // ok, assignment of address
        // at l-value

&a = p; // error: &a is an r-value

int x, y;

(x < y ? y : x) = 0; // It's valid because the ternary
                      // expression preserves the "lvalue-ness"
                      // of both its possible return values
chevron_right
filter_none

```

Remembering the mnemonic, that **lvalues** can appear on the left of an assignment operator while **rvalues** can appear on the right

Reference:

<https://msdn.microsoft.com/en-us/library/bkbs2cds.aspx>

This article is contributed by Shubham Bansal. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes

Recommended Posts:

- [C Programming Language Standard](#)
- [A C Programming Language Puzzle](#)
- [C Language Introduction](#)
- [Arrays in C Language | Set 2 \(Properties\)](#)
- [Convert C/C++ code to assembly language](#)
- [Signals in C language](#)
- [Constants vs Variables in C language](#)
- [Difference between %d and %i format specifier in C language](#)
- [Difference between while\(1\) and while\(0\) in C language](#)
- [kbhit in C language](#)
- [How to use POSIX semaphores in C language](#)
- [Benefits of C language over other programming languages](#)
- [fgets\(\) and gets\(\) in C language](#)
- [isxdigit\(\) function in C Language](#)
- [isupper\(\) function in C Language](#)

Article Tags :

[C](#)
[Practice Tags :](#)
[C](#)

 11

To-do Done
2.1

Based on 11 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

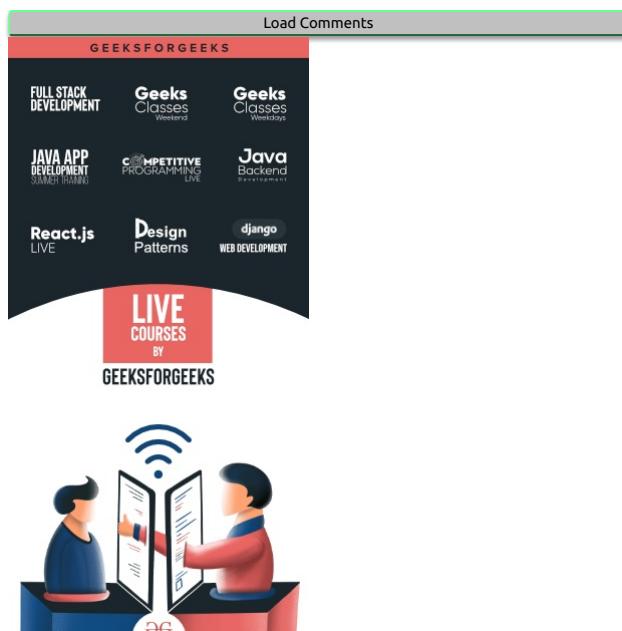
[first_page](#) MCQ on Memory allocation and compilation process

Next

[last_page](#) Difference between while(1) and while(0) in C language

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

[Load Comments](#)



The footer navigation bar includes links for Full Stack Development, Geeks Classes (Weekdays), Java App Development (Summer Training), Competitive Programming Live, Java Backend Development, React.js LIVE, Design Patterns, and django WEB DEVELOPMENT. It also features a 'LIVE COURSES BY GEEKSFORGEEKS' section with two stylized figures interacting with a large screen displaying code, and a 'GEEKSFORGEEKS' logo.

Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
Format specifiers in different Programming Languages
C program to find square root of a given number
Predefined Macros in C with Examples
How to call function within function in C or C++

More related articles in C
C program to print odd line contents of a File followed by even line content
Introduction to the C99 Programming Language : Part II
Frequently Asked Questions in C Language
Problem in comparing Floating point numbers and how to compare them correctly?
<cmath> float.h in C/C++ with Examples

Get the stack size and set the stack size of thread attribute in C

Prerequisite : [Multithreading](#)

Syntax :

`filter_none`

```

edit
close
play_arrow
link
brightness_4
code

// to get size of stack
int pthread_attr_getstacksize(const pthread_attr_t* restrict attr,
                             size_t* restrict stacksize);

// to set size of stack
int pthread_attr_setstacksize(pthread_attr_t* attr, size_t stacksize);
chevron_right
filter_none

```

pthread_attr_getstacksize() :

It is use for get threads stack size. The stacksize attribute gives the minimum stack size allocated to threads stack. When successfully run then it gives 0 otherwise gives any value.

First argument – It takes pthread attribute.

Second argument – It takes a variable and give the size of the thread attribute.

pthread_attr_setstacksize() :

It is use for set new threads stack size. The stacksize attribute gives the minimum stack size allocated to threads stack. When successfully run then it gives 0 otherwise if error gives any value.

First argument – It takes pthread attribute.

Second argument – It takes the size of new stack (In bytes)

```

filter_none
edit
close
play_arrow
link
brightness_4
code

```

```

#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>

int main()
{

```

```

    // for takes the size of threads stack
    size_t stksize;

```

```

    // attribute declaration
    pthread_attr_t atr;

```

```

    // it gets the threads stack size and give
    // value in stksize variable
    pthread_attr_getstacksize(&atr, &stksize);

```

```

    // print the current threads stack size
    printf("Current stack size - > %d\n", stksize);

```

```

    // then set the new threads stack size
    pthread_attr_setstacksize(&atr, 320000034);
    pthread_attr_getstacksize(&atr, &stksize);

```

```

    // print the new stack size
    printf("New stack size-> %d\n", stksize);
    return 0;
}
chevron_right
filter_none

```

Output :

```

Current stack size - > 4196464
New stack size-> 320000034

```

For compile use gcc program_name.c -lpthread

This article is contributed by **Devanshu Agarwal**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up

Add your personal notes
here! (max 5000 chars)

Save

Recommended Posts:

- [Stack vs Heap Memory Allocation](#)
- [Inter-process Communication using a shared stack](#)
- [C Program to find direction of growth of stack](#)
- [Why is the size of an empty class not zero in C++?](#)
- [How to find size of array in C/C++ without using sizeof ?](#)

- C Program to find size of a File
- How to print size of array parameter in C++?
- How does free() know the size of memory to be deallocated?
- C/C++ program to find the size of int, float, double and char
- Extended Integral Types (Choosing the correct integer size in C/C++)
- Heap overflow and Stack overflow
- size of char datatype and char array in C
- Thread functions in C/C++
- Print numbers in sequence using thread synchronization

Article Tags :

C
C-Library
system-programming
Practice Tags :

C

thumb_up
Be the First to upvote.

To-do Done
4.6

Based on 3 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

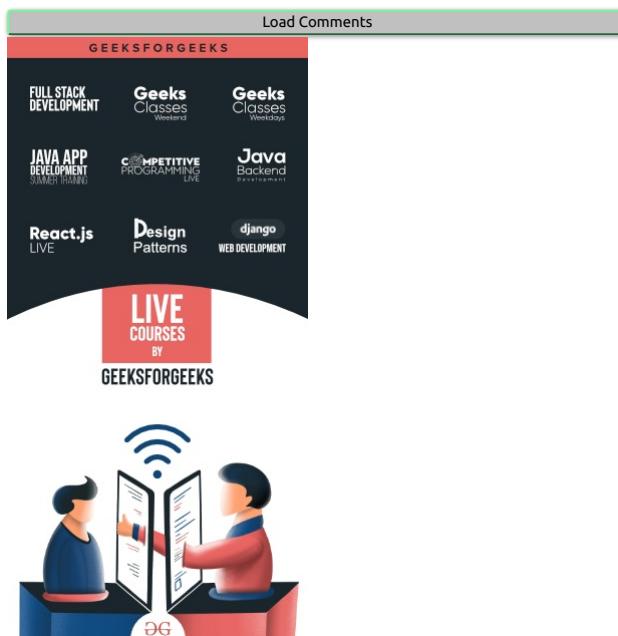
Previous

first_page Segmentation Fault (SIGSEGV) vs Bus Error (SIGBUS)

Next

last_page MCQ on Memory allocation and compilation process

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
Predefined Macros in C with Examples
Program to print odd line contents of a file followed by even line content
Introduction to the C99 Programming Language : Part II

More related articles in C
C program to find square root of a given number
Problem in comparing Floating point numbers and how to compare them correctly?
Features of C Programming Language
Pointer Expressions in C with Examples
Introduction to the C99 Programming Language : Part III

Difference between fork() and exec()

Every application(program) comes into execution through means of process, **process** is a running instance of a program. Processes are created through different system calls, most popular are **fork()** and **exec()**

fork()

```
pid_t pid = fork();
```

fork() creates a new process by duplicating the calling process, The new process, referred to as child, is an exact duplicate of the calling process, referred to as parent, except for the following :

1. The child has its own unique process ID, and this PID does not match the ID of any existing process group.
2. The child's parent process ID is the same as the parent's process ID.
3. The child does not inherit its parent's memory locks and semaphore adjustments.
4. The child does not inherit outstanding asynchronous I/O operations from its parent nor does it inherit any asynchronous I/O contexts from its parent.

Return value of fork()

On success, the PID of the child process is returned in the parent, and 0 is returned in the child. On failure, -1 is returned in the parent, no child process is created, and errno is set appropriately.

[Detailed article on fork system call](#)

exec()

The exec() family of functions **replaces** the current process image with a new process image. It loads the program into the current process space and runs it from the entry point.

The exec() family consists of following functions, I have implemented **execv()** in following C program, you can try rest as an exercise

```
int execl(const char *path, const char *arg, ...);
int execvp(const char *file, const char *arg, ...);
int execve(const char *path, const char *arg, ...,
           char * const envp[]);
```

```
int execv(const char *path, char *const argv[]);
int execvp(const char *file, char *const argv[]);
int execve(const char *file, char *const argv[],
          char *const envp[]);
```

fork vs exec

- fork starts a new process which is a copy of the one that calls it, while exec replaces the current process image with another (different) one.
- Both parent and child processes are executed simultaneously in case of fork() while Control never returns to the original program unless there is an exec() error.

```
filter_none
edit
close

play_arrow
link
brightness_4
code

// C program to illustrate use of fork() &
// exec() system call for process creation
```

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <stdlib.h>
#include <errno.h>
#include <sys/wait.h>

int main(){
    pid_t pid;
    int ret = 1;
    int status;
    pid = fork();

    if (pid == -1){

        // pid == -1 means error occurred
        printf("can't fork, error occurred\n");
        exit(EXIT_FAILURE);
    }
    else if (pid == 0){

        // pid == 0 means child process created
        // getpid() returns process id of calling process
        // Here It will return process id of child process
        printf("child process, pid = %u\n",getpid());
        // Here It will return Parent of child Process means Parent process it self
        printf("parent of child process, pid = %u\n",getppid());

        // the argv list first argument should point to
        // filename associated with file being executed
        // the array pointer must be terminated by NULL
        // pointer
        char * argv_list[] = {"ls","-lart","/home",NULL};

        // the execv() only return if error occurred.
        // The return value is -1
        execv("ls",argv_list);
        exit(0);
    }
    else{
        // a positive number is returned for the pid of
        // parent process
        // getppid() returns process id of parent of
        // calling process
        // Here It will return parent of parent process's ID
        printf("Parent Of parent process, pid = %u\n",getppid());
        printf("parent process, pid = %u\n",getpid());

        // the parent process calls waitpid() on the child
        // waitpid() system call suspends execution of
        // calling process until a child specified by pid
        // argument has changed state
        // see wait() man page for all the flags or options
        // used here
        if (waitpid(pid, &status, 0) > 0){

            if (WIFEXITED(status) && !WEXITSTATUS(status))
                printf("program execution successful\n");

            else if (WIFEXITED(status) && WEXITSTATUS(status)) {
                if (WEXITSTATUS(status) == 127){

                    // execv failed
                    printf("execv failed\n");
                }
                else
                    printf("program terminated normally,"
                           " but returned a non-zero status\n");
            }
            else
                printf("program didn't terminate normally\n");
        }
        else {
    }
```

```
// waitpid() failed
printf("waitpid() failed\n");
}
exit(0);
}
return 0;
}

chevron_right
filter_none
```

Output:

```
parent process, pid = 11523
child process, pid = 14188
Program execution successful
```

References :

Linux man pages

This article is contributed by **Mandeep Singh**. If you like GeeksforGeeks and would like to contribute, you can also write an article using [contribute.geeksforgeeks.org](#) or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up

Add your personal notes
here! (max 5000 chars)

Save

Recommended Posts:

- exec family of functions in C
- fork() in C
- searching in fork()
- Fork() Bomb
- sorting in fork()
- C vs BASH Fork bomb
- Fork() - Practice questions
- fork() and Binary Tree
- Factorial calculation using fork() in C for Linux
- C program to demonstrate fork() and pipe()
- Creating multiple process using fork()
- fork() to execute processes from bottom to up using wait()
- fork() and memory shared b/w processes created using it
- Creating child process using fork() in Python
- Calculation in parent and child process using fork()

Improved By : [nidhi_biet](#), [poojan4004](#)

Article Tags :

C
Difference Between
system-programming
Practice Tags :

C

thumb_up
4

To-do Done
2

Based on 5 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) Why quicksort is better than mergesort ?

Next

[last_page](#) Print Hello World without semicolon in C/C++

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
ctype.h<ctype.h> library in C/C++ with Examples
How to call function within function in C or C++
Format specifiers in different Programming Languages
Predefined Macro in C with Examples

Most visited in Difference Between
Difference between PostgreSQL and MongoDB
Difference between DELETE and TRUNCATE
Difference Between SMO and SEO
Difference between Prim's and Kruskal's algorithm for MST
Monolithic vs Microservices architecture

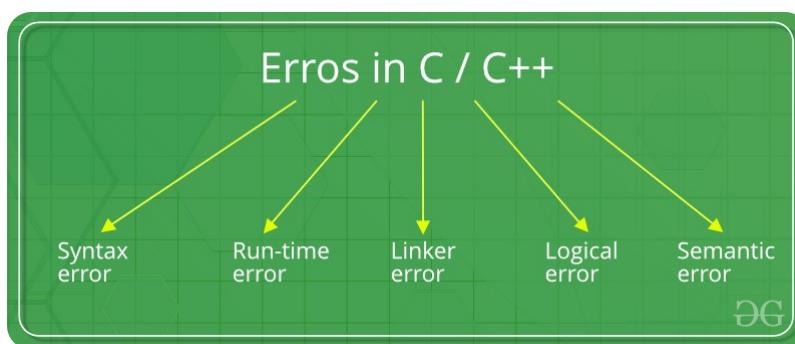
Errors in C/C++

Error is an illegal operation performed by the user which results in abnormal working of the program.

Programming errors often remain undetected until the program is compiled or executed. Some of the errors inhibit the program from getting compiled or executed. Thus errors should be removed before compiling and executing.

The most common errors can be broadly classified as follows.

Type of errors



- Syntax errors:** Errors that occur when you **violate the rules** of writing C/C++ syntax are known as syntax errors. This compiler error indicates something that must be fixed before the code can be compiled. All these errors are detected by compiler and thus are known as compile-time errors.

Most frequent syntax errors are:

- Missing Parenthesis { })
- Printing the value of variable without declaring it
- Missing semicolon like this:

filter_none

edit

close

play_arrow

link

brightness_4

code

// C program to illustrate

// syntax error

#include<stdio.h>

void main()

{

 int x = 10;

 int y = 15;

 printf("%d", (x, y)) // semicolon missed

}

chevron_right

filter_none

Error:

error: expected ';' before '}' token

- Syntax of a basic construct is written wrong. For example : while loop

filter_none

edit

close

play_arrow

```

link
brightness_4
code

// C program to illustrate
// syntax error
#include<stdio.h>
int main(void)
{
    // while() cannot contain ".." as an argument.
    while(..)
    {
        printf("hello");
    }
    return 0;
}
chevron_right
filter_none

```

Error:

```

error: expected expression before `..` token
while(..)

```

In the given example, the syntax of while loop is incorrect. This causes a syntax error.

2. **Run-time Errors :** Errors which occur during program execution(run-time) after successful compilation are called run-time errors. One of the most common run-time error is division by zero also known as Division error. These types of error are hard to find as the compiler doesn't point to the line at which the error occurs.

For more understanding run the example given below.

```

filter_none
edit
close
play_arrow

```

```

link
brightness_4
code

// C program to illustrate
// run-time error
#include<stdio.h>
void main()
{
    int n = 9, div = 0;

    // wrong logic
    // number is divided by 0,
    // so this program abnormally terminates
    div = n/0;

    printf("result = %d", div);
}
chevron_right

```

```

filter_none

```

Error:

```

warning: division by zero [-Wdiv-by-zero]
      div = n/0;

```

In the given example, there is Division by zero error. This is an example of run-time error i.e errors occurring while running the program.

3. **Linker Errors:** These error occurs when after compilation we link the different object files with main's object using **Ctrl+F9** key(RUN). These are errors generated when the executable of the program cannot be generated. This may be due to wrong function prototyping, incorrect header files. One of the most common linker error is writing **Main()** instead of **main()**.

```

filter_none
edit
close
play_arrow

```

```

link
brightness_4
code

// C program to illustrate
// linker error
#include<stdio.h>
```

```

void Main() // Here Main() should be main()
{
    int a = 10;
    printf("%d", a);
}
chevron_right

```

```

filter_none

```

Error:

```
(.text+0x20): undefined reference to `main'
```

4. **Logical Errors :** On compilation and execution of a program, desired output is not obtained when certain input values are given. These types of errors which provide incorrect output but appears to be error free are called logical errors. These are one of the most common errors done by beginners of programming.

These errors solely depend on the logical thinking of the programmer and are easy to detect if we follow the line of execution and determine why the program takes that path of execution.

```

filter_none
edit
close
play_arrow

```

```

link
brightness_4
code

```

```
// C program to illustrate
// logical error
int main()
{
    int i = 0;

    // logical error : a semicolon after loop
    for(i = 0; i < 3; i++);
    {
        printf("loop ");
        continue;
    }
    getchar();
    return 0;
}
```

chevron_right

filter_none

No output

5. **Semantic errors :** This error occurs when the statements written in the program are not meaningful to the compiler.

filter_none

edit

close

play_arrow

link

brightness_4

code

```
// C program to illustrate
```

```
// semantic error
```

```
void main()
```

{

```
    int a, b, c;
```

```
    a + b = c; //semantic error
```

}

chevron_right

filter_none

Error

```
error: lvalue required as left operand of assignment
a + b = c; //semantic error
```

This article is contributed by **Krishna Bhatia**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer Placement 100 for details

My Personal Notes *arrow_drop_up*

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- [std::basic_istream::ignore in C++ with Examples](#)
- [std::basic_istream::getline in C++ with Examples](#)
- [Format specifiers in different Programming Languages](#)
- [C program to find square root of a given number](#)
- [C program to print odd line contents of a File followed by even line content](#)
- [std::is_heap\(\) in C++ with Examples](#)
- [std::basic_istream::gcount\(\) in C++ with Examples](#)
- [new vs malloc\(\) and free\(\) vs delete in C++](#)
- [std::string::rfind in C++ with Examples](#)
- [Sum of factorials of Prime numbers in a Linked list](#)
- [Sum of all Palindrome Numbers present in a Linked list](#)
- [Generate an array of given size with equal count and sum of odd and even numbers](#)
- [Namespaces in C++ | Set 4 \(Overloading, and Exchange of Data in different Namespaces\)](#)
- [Getting System and Process Information Using C Programming and Shell in Linux](#)

Article Tags :

C
C++
CBSE - Class 11
school-programming

Practice Tags :

CPP
CPP

thumb_up
9

To-do Done

1.5

Based on 4 vote(s)

Basic *Easy* *Medium* *Hard* *Expert*

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

first_page std::string::find_first_not_of in C++

Next

last_page C vs BASH Fork bomb

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Weekend Geeks Classes Weekdays

Java APP DEVELOPMENT SUMMER TRAINING COMPETITIVE PROGRAMMING LIVE Java Backend Part-time

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
Predefined Macros in C with Examples
How to use printf with functions in C or C++
C program to print odd line contents of a file followed by even line content
Introduction to the C99 Programming Language : Part II

Most visited in C++
Vector of Vectors in C++ STL with Examples
C++ Tutorial
Which C++ libraries are useful for competitive programming?
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

Why is C considered faster than other languages ?

You might have come across these statements, C is more optimised or performance of C is better than higher languages, so I'll be discussing the reasons for this hypothesis.

First let's list out functionalities which are provided by languages like Java and not C :

1. Array index bound checking
2. Uninitialized variable values checking
3. Check for memory leaks
4. Check for null pointer dereference
5. Automatic garbage collection
6. Run-time type checking
7. Exception handling

and there are more such features which are not present in C.

Extra features comes at cost and the cost includes decreased **speed** and increased **size**.

Let's take an example for dynamic allocation in C and Java

Java

```
MyClass obj = new MyClass();
```

Did you consider size of **obj**, the answer is **No**. The reason being it is automatically handled by language itself in background and you don't have to write specific code for it.

But in case of **C**

```
struct MyStruct *obj = malloc(sizeof(struct MyStruct));
```

As you can see in above code the tasks of assigning reference to pointer, allocation of size is done explicitly by programmer and at last free up allocated memory.

The array bound check is supported by Thumb Execution Environment (ThumbEE), its other features includes automatic null pointer checks on every load and store instruction, an special instruction that call a handler.

Another reason is closeness of C to the Assembly language, in most of the cases its instructions directly map to assembly language, C is only one or level two levels of abstraction away from assembly language while Java is at minimum 3 levels abstraction away from assembler.

References :

- 1) [why-is-c-so-fast-and-why-arent-other-languages-as-fast-or-faster](#)
- 2) [ARM_architecture#Thumb_Execution_Environment_.28ThumbEE.29](#)
- 3) [Linus Torvalds view](#)

This article is contributed by **Mandeep Singh**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes 

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- [getchar_unlocked\(\) - faster input in C/C++ for Competitive Programming](#)
- [Benefits of C language over other programming languages](#)
- [Why are elementwise additions much faster in separate loops than in a combined loop?](#)
- [Role of SemiColon in various Programming Languages](#)
- [Format specifiers in different Programming Languages](#)
- [Format specifiers in different Programming Languages](#)

- C program to find square root of a given number
- C program to print odd line contents of a File followed by even line content
- Getting System and Process Information Using C Programming and Shell in Linux
- Program to calculate Electricity Bill
- Program to print half Diamond star pattern
- C/C++ program for calling main() in main()
- Print all possible combinations of the string by replacing '\$' with any other digit from the string
- How to use make utility to build C projects?

Article Tags :

C
Practice Tags :

C

thumb_up
1

To-do Done
2.7

Based on 9 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

first_page Non-blocking I/O with pipes in C

Next

last_page Accessing array out of bounds in C/C++

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT
Geeks Classes Webinars
Geeks Classes Workshops

JAVA APP DEVELOPMENT SUMMER TRAINING
COMPETITIVE PROGRAMMING LIVE
Java Backend Development

React.js LIVE
Design Patterns
django WEB DEVELOPMENT



LIVE COURSES
BY
GEEKSFORGEEKS



Most popular in C

Print all possible combinations of the string by replacing '\$' with any other digit from the string

Predefined Macros in C with Examples

Format specifiers in different Programming Languages

C program to print odd line contents of a File followed by even line content

C program to find square root of a given number

More related articles in C

Introduction to the C99 Programming Language : Part II

Problem in comparing Floating point numbers and how to compare them correctly?

Features of C Programming Language

Pointer Expressions in C with Examples

Introduction to the C99 Programming Language : Part III

Incompatibilities between C and C++ codes

The C/C++ incompatibilities that cause most real problems are not subtle. Most are easily caught by compilers.

This section gives examples of C code that is not C++ :

1) In C, functions can be defined using a syntax that optionally specifies argument types after the list of arguments:

```
filter_none
edit
close

play_arrow

link
brightness_4
code

// C code that uses argument types after
// the list of arguments.
#include<stdio.h>
void fun(a, b)int a;int b;      // Invalid in C++
{
    printf("%d %d", a, b);
}

// Driver code
int main()
{
    fun(8, 9);
    return 0;
}
```

chevron_right

filter_none

Output:
8 9

Error in C++ :- a and b was not declared in this scope

2) In C and in pre-standard versions of C++, the type specifier defaults to int.

filter_none
edit
close
play_arrow
link
brightness_4
code

```
// C code to show implicit int declaration.
#include<stdio.h>
int main()
{
    // implicit int in C, not allowed in C++
    const a = 7;

    printf("%d", a);
    return 0;
}
```

chevron_right

filter_none

Output:
7

Error in C++ :- a does not name a type

3) In C, a global data object may be declared several times without using the **extern** specifier. As long as at most one such declaration provides an initializer, the object is considered defined only once.

filter_none
edit
close
play_arrow
link
brightness_4
code

```
// C code to demonstrate multiple global
// declarations of same variable.
#include<stdio.h>

// Declares single integer a, not allowed in C++
int a;  int a;
int main()
{
    return 0;
}
```

chevron_right

filter_none

Error in C++ :- Redefinition of int a

4) In C, a void* may be used as the right-hand operand of an assignment to or initialization of a variable of any pointer type.

filter_none
edit
close
play_arrow
link
brightness_4
code

```
// C code to demonstrate implicit conversion
// of void* to int*
#include<stdio.h>
#include<malloc.h>
void f(int n)
{
    // Implicit conversion of void* to int*
    // Not allowed in C++.
    int* p = malloc(n* sizeof(int));
}

// Driver code
int main()
{
    f(7);
    return 0;
}
```

Error in C++ :- Invalid conversion of void* to int*

5) In C, an array can be initialized by an initializer that has more elements than the array requires.

filter_none
edit
close

```
play_arrow
link
brightness_4
code

// C code to demonstrate that extra character
// check is not done in C.
#include<stdio.h>
int main()
{
    // Error in C++
    char array[5] = "Geeks";

    printf("%s", array);
    return 0;
}
```

chevron_right

filter_none

Output:

Geeks

Error in C++ :- Initializer-string for array of chars is too long

6) In C, a function declared without specifying any argument types can take any number of arguments of any type at all. Click [here](#) to know more about this.

filter_none

edit

close

play_arrow

link

brightness_4

code

```
// In C, empty brackets mean any number
// of arguments can be passed
#include<stdio.h>
```

```
void fun() { }
```

```
int main(void)
{
    fun(10, "GfG", "GQ");
    return 0;
}
```

chevron_right

filter_none

Error in C++ :- Too many arguments to function 'void fun()'

Related Articles:

- [Is it fine to write "void main\(\)" or "main\(\)" in C/C++?](#)
- [Difference between "int main\(\)" and "int main\(void\)" in C/C++?](#)
- [Output of C++ programs | Set 24 \(C++ vs C\)](#)

This article is contributed by **Sakshi Tiwari**. If you like GeeksforGeeks (We know you do!) and would like to contribute, you can also write an article using [contribute.geeksforgeeks.org](#) or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- [Rust vs C++: Will Rust Replace C++ in Future ?](#)
- [Priority queue of pairs in C++ with ordering by first and second element](#)
- [Implementation of lower_bound\(\) and upper_bound\(\) in Vector of Pairs in C++](#)
- [Generating RGBA portable graphic images through C++](#)
- [std::basic_istream::ignore in C++ with Examples](#)
- [std::basic_istream::getline in C++ with Examples](#)
- [Format specifiers in different Programming Languages](#)
- [C program to find square root of a given number](#)
- [C program to print odd line contents of a File followed by even line content](#)
- [std::is_heap\(\) in C++ with Examples](#)
- [std::basic_istream::gcount\(\) in C++ with Examples](#)
- [new vs malloc\(\) and free\(\) vs delete in C++](#)
- [std::string::find in C++ with Examples](#)
- [Sum of factorials of Prime numbers in a Linked list](#)

Article Tags :

C

C++

Practice Tags :

C

CPP

thumb_up

Be the First to upvote.

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

[Previous](#)

[first_page](#) `std::string::compare()` in C++

[Next](#)

[last_page](#) [Input-output system calls in C](#) | [Create, Open, Close, Read, Write](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT

Geeks Classes Weekend

Geeks Classes Weekdays

JAVA APP DEVELOPMENT

COMPETITIVE PROGRAMMING DAY

Java Backend EXPERTISE

React.js LIVE

Design Patterns

django WEB DEVELOPMENT



LIVE COURSES BY GEEKSFORGEEKS



Most popular in C
[Print all possible combinations of the string by replacing '\\$' with any other digit from the string](#)
[Predefined Macros in C with Examples](#)
[Format specifiers in different Programming Languages](#)
[C program to print odd line contents of a file followed by even line content](#)
[C program to find square root of a given number](#)

Most viewed in C++
[Vector of Vectors in C++ STL with Examples](#)
[C++ Tutorial](#)
[Which C++ libraries are useful for competitive programming?](#)
[Array of Vectors in C++ STL](#)
[Map of Vectors in C++ STL with Examples](#)

Convert C/C++ code to assembly language

We use g++ compiler to turn provided C code into assembly language. To see the assembly code generated by the C compiler, we can use the “-S” option on the command line:

Syntax:

```
$ gcc -S filename.c
```

This will cause gcc to run the compiler, generating an assembly file. Suppose we write a C code and store it in a file name “geeks.c” .

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// C code stored in geeks.c file
#include <stdio.h>

// global string
char s[] = "GeeksforGeeks";

// Driver Code
int main()
{
    // Declaring variables
    int a = 2000, b =17;

    // Printing statement
    printf("%s %d \n", s, a+b);
}
```

Running the command:

```
$ gcc -S geeks.c
```

This will cause gcc to run the compiler, generating an assembly file **geeks.s**, and go no further. (Normally it would then invoke the assembler to generate an object- code file.)

The assembly-code file contains various declarations including the set of lines:

```
filter_none
edit
close
play_arrow
link
brightness_4
```

```

code
.section __TEXT, __text, regular, pure_instructions
.macosx_version_min 10, 12
.globl _main
.align 4, 0x90
_main:                                ## @main
    .cfi_startproc
## BB#0:
    pushq %rbp
.Ltmp0:
    .cfi_offset %rbp, 16
.Ltmp1:
    .cfi_offset %rbp, -16
    movq %rsp, %rbp
.Ltmp2:
    .cfi_offset %rbp, -16
    subq $16, %rsp
    leaq L_.str(%rip), %rdi
    leaq _S(%rip), %rsi
    movl $2000, -4(%rbp)      ## imm = 0x7D0
    movl $17, -8(%rbp)
    movl -4(%rbp), %eax
    addl -8(%rbp), %eax
    movl %eax, %edx
    movb $0, %al
    callq _printf
    xorl %edx, %edx
    movl %eax, -12(%rbp)     ## 4-byte Spill
    movl %edx, %eax
    addq $16, %rsp
    popq %rbp
    retq
    .cfi_endproc

.section __DATA, __data
.globl _S                         ## @S
.S:
    .asciz "GeeksforGeeks"

.section __TEXT, __cstring, cstring_literals
L_.str:                           ## @.str
    .asciz "%s %d \n"

```

.subsections_via_symbols
chevron_right

filter none

Each indented line in the above code corresponds to a single machine instruction. For example, the **pushq** instruction indicates that the contents of register **%rbp** should be pushed onto the program stack. All information about local variable names or data types has been stripped away. We still see a reference to the global variable **S[] = "GeeksforGeeks"**, since the compiler has not yet determined where in memory this variable will be stored.

This article is contributed by **Sahil Rajput**. If you like GeeksforGeeks and would like to contribute, you can also write an article using [contribute.geeksforgeeks.org](#) or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes *arrow_drop_up*

Recommended Posts:

- Assembly language program to find the range of bytes
- Assembly language program to find largest number in an array
- 8085 code to convert binary number to ASCII code
- 8085 program to convert ASCII code into HEX code
- Convert C/C++ program to Preprocessor code
- 8086 program to convert binary to Grey code
- 8085 program to convert 8 bit BCD number into ASCII Code
- 8086 program to convert 8 bit BCD number into ASCII Code
- 8085 program to convert a hexadecimal number into ASCII code
- Assembly program to transfer the status of switches
- Signals in C language
- kbhit in C language
- C Language Introduction
- Difference between while(1) and while(0) in C language
- Stopwatch using C language

Article Tags :

C
C++
system-programming
Practice Tags :
C
CPP

thumb_up
Be the first to upvote.

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

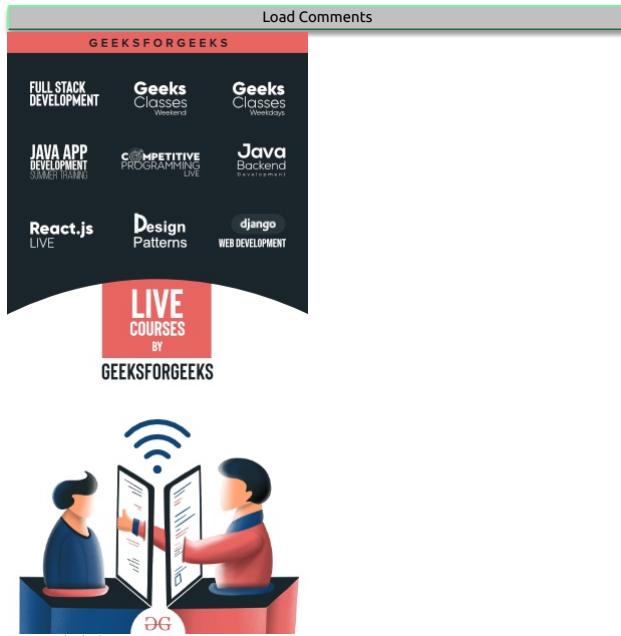
Previous

[first_page](#) Constructor Overloading in C++

Next

[last_page](#) std::gcd | C++ inbuilt function for finding GCD

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



The screenshot shows the GeeksforGeeks homepage with a focus on live courses. It features a banner for 'LIVE COURSES BY GEEKSFORGEEKS' with two stylized figures interacting with a screen displaying course content. Below this, there are sections for Full Stack Development, Java App Development, React.js, Design Patterns, and Django. A red box highlights the 'LIVE COURSES' section.

Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
Formulas and Examples in different Programming Languages
Predefined Macros in C with Examples
C program to find square root of a given number
C program print odd line contents of a File followed by even line content

Most visited in C++
Vector of Vectors in C++ STL with Examples
C++ Tutorial
Which C++ libraries are useful for competitive programming?
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

Error Handling in C programs

Although C does not provide direct support to error handling (or exception handling), there are ways through which error handling can be done in C. A programmer has to prevent errors at the first place and test return values from the functions.

A lot of C function calls return a -1 or NULL in case of an error, so quick test on these return values are easily done with for instance an 'if statement'. For example, In [Socket Programming](#), the returned value of the functions like socket(), listen() etc. are checked to see if there is an error or not.

Example: Error handling in Socket Programming

```
if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0)
{
    perror("socket failed");
    exit(EXIT_FAILURE);
}
```

Different methods of Error handling in C

1. **Global Variable errno:** When a function is called in C, a variable named as errno is automatically assigned a code (value) which can be used to identify the type of error that has been encountered. Its a global variable indicating the error occurred during any function call and defined in the header file errno.h.

Different codes (values) for errno mean different types of errors. Below is a list of few different errno values and its corresponding meaning:

errno value	Error
1	/* Operation not permitted */
2	/* No such file or directory */
3	/* No such process */
4	/* Interrupted system call */
5	/* I/O error */
6	/* No such device or address */
7	/* Argument list too long */
8	/* Exec format error */
9	/* Bad file number */
10	/* No child processes */
11	/* Try again */
12	/* Out of memory */
13	/* Permission denied */

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// C implementation to see how errno value is
// set in the case of any error in C
#include <stdio.h>
#include <errno.h>

int main()
{
```

```

// If a file is opened which does not exist,
// then it will be an error and corresponding
// errno value will be set
FILE * fp;

// opening a file which does
// not exist.
fp = fopen("GeeksForGeeks.txt", "r");

printf(" Value of errno: %d\n ", errno);

return 0;
}

```

chevron_right

filter_none

Output:

Value of errno: 2

Note: Here the errno is set to 2 which means - No such file or directory. On online IDE it may give errno 13, which says permission denied.

2. perror() and strerror():

The errno value got above indicate the types of error encountered. If it is required to show the error description, then there are two functions that can be used to display a text message that is associated with errno. The functions are:

- **perror:** It displays the string you pass to it, followed by a colon, a space, and then the textual representation of the current errno value.

Syntax:

```
void perror (const char *str)
str: is a string containing a custom message
to be printed before the error message itself.
```

- **strerror():** returns a pointer to the textual representation of the current errno value.

Syntax:

```
char *strerror (int errnum)
errnum: is the error number (errno).
```

filter_none

edit

close

play_arrow

link

brightness_4

code

```

// C implementation to see how perror() and strerror()
// functions are used to print the error messages.
#include <stdio.h>
#include <errno.h>
#include <string.h>
```

int main ()

{

FILE *fp;

```

// If a file is opened which does not exist,
// then it will be an error and corresponding
// errno value will be set
fp = fopen(" GeeksForGeeks.txt ", "r");
```

```

// opening a file which does
// not exist.
printf("Value of errno: %d\n ", errno);
printf("The error message is : %s\n",
strerror(errno));
perror("Message from perror");

```

return 0;

}

chevron_right

filter_none

Output:

On Personal desktop:

Value of errno: 2
The error message is : No such file or directory
Message from perror: No such file or directory

On online IDE:

Value of errno: 13
The error message is : Permission denied

Note: The function perror() displays a string passed to it, followed by a colon and the textual message of the current errno value.

3. Exit Status:

The C standard specifies two constants: EXIT_SUCCESS and EXIT_FAILURE, that may be passed to exit() to indicate successful or unsuccessful termination, respectively. These are macros defined in stdlib.h.

filter_none

edit

close

play_arrow

link

brightness_4

code

```

// C implementation which shows the
// use of EXIT_SUCCESS and EXIT_FAILURE.
#include <stdio.h>
#include <errno.h>
#include <string.h>
#include <stdlib.h>
```

```

int main ()
{
    FILE * fp;
    fp = fopen ("filedoesnotexist.txt", "rb");

    if (fp == NULL)
    {
        printf("Value of errno: %d\n", errno);
        printf("Error opening the file: %s\n",
               strerror(errno));
        perror("Error printed by perror");

        exit(EXIT_FAILURE);
        printf("I will not be printed\n");
    }

    else
    {
        fclose (fp);
        exit(EXIT_SUCCESS);
        printf("I will not be printed\n");
    }
    return 0;
}

```

chevron_right

filter_none

Output:

```

Value of errno: 2
Error opening the file: No such file or directory
Error printed by perror: No such file or directory

```

4. **Divide by Zero Errors:** A common pitfall made by C programmers is not checking if a divisor is zero before a division command. Division by zero leads to undefined behavior, there is no C language construct that can do anything about it. Your best bet is to not divide by zero in the first place, by checking the denominator.

filter_none

edit

close

play_arrow

link

brightness_4

code

// C program to check and rectify

// divide by zero condition

#include<stdio.h>

#include <stdlib.h>

void function(int);

int main()

{

int x = 0;

function(x);

return 0;

}

void function(int x)

{

float fx;

if (x==0)

{

printf("Division by Zero is not allowed");

fprintf(stderr, "Division by zero! Exiting...\n");

exit(EXIT_FAILURE);

}

else

{

fx = 10 / x;

printf("f(x) is: %.5f", fx);

}

}

chevron_right

filter_none

Output:

```

Division by Zero is not allowed

```

This article is contributed by **MAZHAR IMAM KHAN**. If you like GeeksforGeeks and would like to contribute, you can also write an article using [contribute.geeksforgeeks.org](#) or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes *arrow_drop_up*

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- Memory Layout of C Programs
- Exception handling and object destruction | Set 1
- Exception Handling in C++
- Four File Handling Hacks which every C/C++ Programmer should know
- Socket Programming in C/C++: Handling multiple clients on server without multi threading
- Programs to print Interesting Patterns
- C | File Handling | Question 1
- C | File Handling | Question 2
- C | File Handling | Question 3
- C | File Handling | Question 4
- C | File Handling | Question 5
- C++ | Exception Handling | Question 1
- C++ | Exception Handling | Question 2
- C++ | Exception Handling | Question 3
- C++ | Exception Handling | Question 4

Article Tags :

C
Practice Tags :
C

thumb_up
2

To-do Done
3.7

Based on 12 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

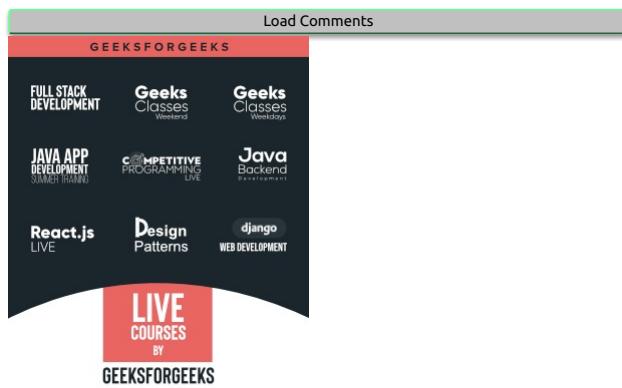
Previous

first_page How to add "graphics.h" C/C++ library to gcc compiler in Linux

Next

last_page Difference between C structures and C++ structures

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.



Most popular in C
Print all possible combinations of the string by replacing 's' with any other digit from the string
Format specifiers in different Programming Languages
C program to find square root of a given number
Predefined Macros in C with Examples
C program to print odd line contents of a file followed by even line content

More related articles in C
Introduction to the C99 Programming Language : Part II
Features of C Programming Language
Problem in comparing Floating point numbers and how to compare them correctly?
<cmath>> float.h in C/C++ with Examples
Getting System and Process Information Using C Programming and Shell in Linux

Executing main() in C/C++ – behind the scene

How to write a C program to print "Hello world" without main() function?

At first, it seems impractical to execute a program without a **main()** function because the **main()** function is the entry point of any program.

Let us first understand what happens under the hood while executing a C program in Linux system, how main() is called and how to execute a program without main().

Following setup is considered for the demonstration.

- Ubuntu 16.4 LTS operating system
- GCC 5.4.0 compiler
- objdump utility

From C/C++ programming perspective, the program entry point is main() function. From the perspective of program execution, however, it is not. Prior to the point when the execution flow reaches to the main(), calls to few other functions are made, which setup arguments, prepare environment variables for program execution etc.

The executable file created after compiling a C source code is a **Executable and Linkable Format (ELF) file**.

Every ELF file have a ELF header where there is a **e_entry** field which contains the program memory address from which the execution of executable will start. This memory address point to the **_start()** function.

After loading the program, loader looks for the **e_entry** field from the ELF file header. Executable and Linkable Format (ELF) is a common standard file format used in UNIX system for executable files, object code, shared libraries, and core dumps.

Let's see this using an example. I'm creating a **example.c** file to demonstrate this.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

int main()
{
    return(0);
}

chevron_right
filter_none
```

Now compiling this using following commands

```
gcc -o example example.c
```

Now an **example** executable is created, let us examine this using objdump utility

```
objdump -f example
```

This outputs following critical information of executable on my machine. Have a look at start address below, this is the address pointing to `_start()` function.

```
example:   file format elf64-x86-64
architecture: i386:x86-64, flags 0x000000112:
EXEC_P, HAS_SYMS, D_PAGED
start address 0x00000000004003e0
```

We can cross check this address by deassembling the executable, the output is long so I'm just pasting the output which shows where this address **0x00000000004003e0** is pointing

```
objdump --disassemble example
```

Output :

```
00000000004003e0 <_start>:
4003e0: 31 ed          xor    %ebp,%ebp
4003e2: 49 89 d1        mov    %rdx,%r9
4003e5: 5e              pop    %rsi
4003e6: 48 89 e2        mov    %rsp,%rdx
4003e9: 48 83 e4 f0      and   $0xfffffffffffffff0,%rsp
4003ed: 50              push   %rax
4003ee: 54              push   %rsp
4003ef: 49 c7 c0 60 05 40 00  mov   $0x400560,%r8
4003f6: 48 c7 c1 f0 04 40 00  mov   $0x4004f0,%rcx
4003fd: 48 c7 c7 d6 04 40 00  mov   $0x4004d6,%rdi
400404: e8 b7 ff ff ff    callq  4003c0
400409: f4              hlt
40040a: 66 0f 1f 44 00 00  nopw   0x0(%rax,%rax,1)
```

As we can clearly see this is pointing to the `_start()` function.

The role of `_start()` function

The `_start()` function prepare the input arguments for another function `_libc_start_main()` which will be called next. This is prototype of `_libc_start_main()` function. Here we can see the arguments which were prepared by `_start()` function.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

int __libc_start_main(int (*main) (int, char * *, char * *), /* address of main function*/
int argc, /* number of command line args*/
char ** ubp_av, /* command line arg array*/
void (*init) (void), /* address of init function*/
void (*fini) (void), /* address of fini function*/
void (*rtld_fini) (void), /* address of dynamic linker fini function */
void (* stack_end) /* end of the stack address*/
);
chevron_right
filter_none
```

The role of `_libc_start_main()` function

The role of `_libs_start_main()` function is following -

- Preparing environment variables for program execution
- Calls `_init()` function which performs initialization before the `main()` function starts.
- Register `_fini()` and `_rtld_fini()` functions to perform cleanup after program terminates

After all the prerequisite actions has been completed, `_libc_start_main()` calls the `main()` function.

Writing program without `main()`

Now we know how the call to the `main()` is made. To make it clear, `main()` is nothing but a agreed term for startup code. We can have any name for startup code it doesn't necessarily have to be "main". As `_start()` function by default calls `main()`, we have to change it if we want to execute our custom startup code. We can override the `_start()` function to make it call our custom startup code not `main()`. Let's have an example, save it as **nomain.c** -

```
filter_none
edit
close
play_arrow
link
brightness_4
code

#include<stdio.h>
```

```
#include<stdlib.h>
void _start()
{
    int x = my_fun(); //calling custom main function
    exit(x);
}

int my_fun() // our custom main function
{
    printf("Hello world!\n");
    return 0;
}
```

chevron_right

filter_none

Now we have to force compiler to not use it's own implementation of `_start()`. In GCC we can do this using `-nostartfiles`

```
gcc -nostartfiles -o nomain nomain.c
```

Execute the executable nomain

```
./nomain
```

Output:

```
Hello world!
```

References

- <http://dbp-consulting.com/tutorials/debugging/linuxProgramStartup.html>
- Advanced C/C++ Compiling by Milan Stevanovic

This article is contributed by **Atul Kumar**. If you like GeeksforGeeks and would like to contribute, you can also write an article using [contribute.geeksforgeeks.org](#) or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes *arrow_drop_up*

Add your personal notes
here! (max 5000 chars)

Save

Recommended Posts:

- Is it fine to write "void main()" or "main()" in C/C++?
- C/C++ program for calling main() in main()
- Difference between "int main()" and "int main(void)" in C/C++?
- What does main() return in C and C++?
- Can main() be overloaded in C++?
- Functions that are executed before and after main() in C
- return statement vs exit() in main()
- How to change the output of printf() in main() ?
- How to write a running C code without main()?
- How to print "GeeksforGeeks" with empty main() in C, C++ and Java?
- Print "Hello World" with empty or blank main in C++
- std::basic_istream::ignore in C++ with Examples
- std::basic_istream::getline in C++ with Examples
- Format specifiers in different Programming Languages

Article Tags :

C
C++
system-programming
Practice Tags :
C
CPP

thumb_up
21

To-do Done
3.5

Based on 18 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

first_page Double Pointer (Pointer to Pointer) in C

Next

last_page How does Duff's Device work?

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
Predefined Macros in C with Examples
C program to print odd line contents of a File followed by even line content
Introduction to the C99 Programming Language : Part II
C program to find square root of a given number

Most visited in C++
Vector of Vectors in C++ STL with Examples
C++ STL
Which C++ libraries are useful for competitive programming?
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

Hygienic Macros : An Introduction

We are all familiar with the working of [macros](#) in languages like C. There are certain situations in which macro expansions can lead to undesirable results because of accidental capture of identifiers.

For example:

```
filter_none
edit
close

play_arrow
link
brightness_4
code

// C program to illustrate a situation known as
// accidental capture of identifiers - an
// undesirable result caused by unhygienic macros
#define INCI(i) do { int x = 0; ++i; } while(0)
int main(void)
{
    int x = 4, y = 8;

    // macro called first time
    INCI(x);

    // macro called second time
    INCI(y);

    printf("x = %d, b = %d\n", x, y);
    return 0;
}
```

The code is actually equivalent to:

```
filter_none
edit
close

play_arrow
link
brightness_4
code

// C program to illustrate unhygienic macros
// with Macro definition substituted in source code.
int main(void)
{
    int x = 4, y = 8;

    //macro called first time
    do { int x = 0; ++x; } while(0);

    //macro called second time
    do { int x = 0; ++y; } while(0);

    printf("x = %d, b = %d\n", x, y);
    return 0;
}
```

Output:

x = 4, y = 9

The variable a declared in the scope of the main function is overshadowed by the variable a in the macro definition so a = 4 never gets updated (known as accidental capture).

Hygienic macros

Hygienic macros are macros whose expansion is guaranteed not to cause the accidental capture of identifiers. A hygienic macro doesn't use variable names that can risk interfering with the code under expansion.

The situation in the above code can be avoided simply by changing the name of the variable in the macro definition, which will produce a different output.

```
filter_none
edit
close

play_arrow
link
brightness_4
code

// C program to illustrate
// Hygienic macros using
// identifier names such that
// they do not cause
// the accidental capture of identifiers
#define INCI(i) do { int m = 0; ++i; } while(0)
int main(void)
{
    int x = 4, y = 8;

    // macro called first time
    INCI(x);

    // macro called second time
    INCI(y);

    printf("x = %d, y = %d\n", x, y);
    return 0;
}
```

chevron_right

filter_none

Output:

x = 5, y = 9

This article is contributed by [Palash Nigam](#). If you like GeeksforGeeks and would like to contribute, you can also write an article using [contribute.geeksforgeeks.org](#) or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- [X-Macros in C](#)
- [Macros vs Functions](#)
- [Multiline macros in C](#)
- [Branch prediction macros in GCC](#)
- [Predefined Macros in C with Examples](#)
- [Variable length arguments for Macros](#)
- [Data Type Ranges and their macros in C++](#)
- [Interesting Facts about Macros and Preprocessors in C](#)
- [p5.js | Introduction](#)
- [Introduction to SQLite](#)
- [AA Trees | Set 1 \(Introduction\)](#)
- [Introduction of a Router](#)
- [Introduction to WebRTC](#)
- [Introduction To APIs](#)
- [Introduction of Hyperledger](#)

Article Tags :

C
Technical Scripter
Practice Tags :
C

thumb_up
1

To-do Done
1

Based on 2 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) Determine the type of an image in Python using imgHDR

Next

[last_page](#) What happens when we turn on computer?



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
[ctype.h<ctype.h> library in C/C++ with Examples](#)
C program to display month by month calendar for a given year
[Predefined Macros in C with Examples](#)
[Implicit Type Conversion in C with Examples](#)

More related articles in C
[Format specifiers in different Programming Languages](#)
[How to call function within function in C or C++](#)
[C program to print odd line contents of a file followed by even line content](#)
[C program to find square root of a given number](#)
[Introduction to the C99 Programming Language : Part II](#)

Command line arguments in C/C++

The most important function of C/C++ is main() function. It is mostly defined with a return type of int and without parameters :

```
int main() { /* ... */ }
```

We can also give command-line arguments in C and C++. Command-line arguments are given after the name of the program in command-line shell of Operating Systems.

To pass command line arguments, we typically define main() with two arguments : first argument is the number of command line arguments and second is list of command-line arguments.

```
int main(int argc, char *argv[]) { /* ... */ }
```

or

```
int main(int argc, char **argv) { /* ... */ }
```

- **argc (ARGument Count)** is int and stores number of command-line arguments passed by the user including the name of the program. So if we pass a value to a program, value of argc would be 2 (one for argument and one for program name)
- The value of argc should be non negative.
- **argv(ARGUMENT Vector)** is array of character pointers listing all the arguments.
- If argc is greater than zero, the array elements from argv[0] to argv[argc-1] will contain pointers to strings.
- Argv[0] is the name of the program , After that till argv[argc-1] every element is command -line arguments.

For better understanding run this code on your linux machine.

```
filter_none
edit
close

play_arrow

link
brightness_4
code

// Name of program mainreturn.cpp
#include <iostream>
using namespace std;

int main(int argc, char** argv)
{
    cout << "You have entered " << argc
        << " arguments:" << "\n";

    for (int i = 0; i < argc; ++i)
        cout << argv[i] << "\n";
}
```

Input:

```
$ g++ mainreturn.cpp -o main
$ ./main geeks for geeks
```

Output:

```
You have entered 4 arguments:
./main
geeks
for
geeks
```

Note : Other platform-dependent formats are also allowed by the C and C++ standards; for example, Unix (though not POSIX.1) and Microsoft Visual C++ have a third argument giving the program's environment, otherwise accessible through getenv in stdlib.h: Refer [C program to print environment variables](#) for details.

Properties of Command Line Arguments:

1. They are passed to main() function.
2. They are parameters/arguments supplied to the program when it is invoked.
3. They are used to control program from outside instead of hard coding those values inside the code.
4. argv[argc] is a NULL pointer.
5. argv[0] holds the name of the program.
6. argv[1] points to the first command line argument and argv[n] points last argument.

Note : You pass all the command line arguments separated by a space, but if argument itself has a space then you can pass such arguments by putting them inside double quotes "" or single quotes ' '.

filter_none
edit
close

play_arrow
link
brightness_4
code

```
// C program to illustrate
// command line arguments
#include<stdio.h>

int main(int argc,char* argv[])
{
    int counter;
    printf("Program Name Is: %s",argv[0]);
    if(argc==1)
        printf("\nNo Extra Command Line Argument Passed Other Than Program Name");
    if(argc>=2)
    {
        printf("\nNumber Of Arguments Passed: %d",argc);
        printf("\n----Following Are The Command Line Arguments Passed----");
        for(counter=0;counter<argc;counter++)
            printf("\nargv[%d]: %s",counter,argv[counter]);
    }
    return 0;
}
```

chevron_right

filter_none

Output in different scenarios:

1. **Without argument:** When the above code is compiled and executed without passing any argument, it produces following output.

```
$ ./a.out
Program Name Is: ./a.out
No Extra Command Line Argument Passed Other Than Program Name
```

2. **Three arguments :** When the above code is compiled and executed with a three arguments, it produces the following output.

```
$ ./a.out First Second Third
Program Name Is: ./a.out
Number Of Arguments Passed: 4
----Following Are The Command Line Arguments Passed----
argv[0]: ./a.out
argv[1]: First
argv[2]: Second
argv[3]: Third
```

3. **Single Argument :** When the above code is compiled and executed with a single argument separated by space but inside double quotes, it produces the following output.

```
$ ./a.out "First Second Third"
Program Name Is: ./a.out
Number Of Arguments Passed: 2
----Following Are The Command Line Arguments Passed----
argv[0]: ./a.out
argv[1]: First Second Third
```

4. **Single argument in quotes separated by space :** When the above code is compiled and executed with a single argument separated by space but inside single quotes, it produces the following output.

```
$ ./a.out 'First Second Third'
Program Name Is: ./a.out
Number Of Arguments Passed: 2
----Following Are The Command Line Arguments Passed----
argv[0]: ./a.out
argv[1]: First Second Third
```

References:

<http://www.cprogramming.com/tutorial/lesson14.html>
<http://c0x.coding-guidelines.com/5.1.2.2.1.html>

This article is contributed by **Kartik Ahuja** and **Avadhut Patade**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- [Command line arguments example in C](#)
- [getopt\(\) function in C to parse command line arguments](#)
- [Equation of straight line passing through a given point which bisects it into two equal line segments](#)
- [C program to print odd line contents of a File followed by even line content](#)
- [Program to delete a line given the line number from a file](#)
- [Default Arguments in C++](#)
- [Templates and Default Arguments](#)
- [How to Count Variable Numbers of Arguments in C?](#)

- Some Interesting facts about default arguments in C++
- Variable length arguments for Macros
- When do we pass arguments by reference or pointer?
- Default arguments and virtual function
- C++ | Function Overloading and Default Arguments | Question 5
- C++ | Function Overloading and Default Arguments | Question 4
- C++ | Function Overloading and Default Arguments | Question 3

Article Tags :

C
C++
TCS
Practice Tags :
TCS
C
CPP

32

To-do Done
2.4

Based on 26 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

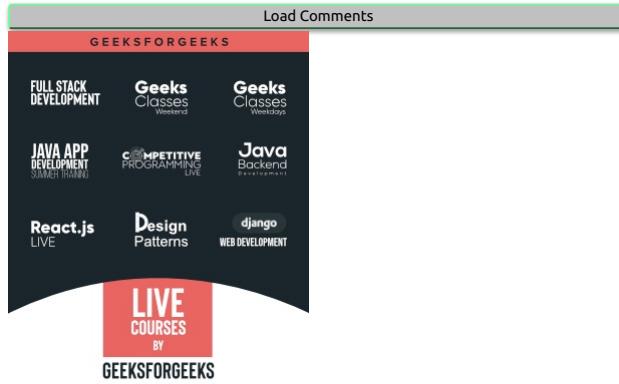
Previous

[first_page](#) [Builtin functions of GCC compiler](#)

Next

[last_page](#) [Using a variable as format specifier in C](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
Format specifiers in different Programming Languages
Predefined Macros in C with Examples
How to call function within function in C or C++
C program to print odd line contents of a file followed by even line content

Most visited in C++
Vector of Vectors in C++ STL with Examples
C++ Tutorials
Which C++ libraries are useful for competitive programming?
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

Inbuilt library functions for user Input | scanf, fscanf, sscanf, scanf_s, fscanf_s, sscanf_s

1. **scanf()** : The C library function int scanf (const char *format, ...) reads formatted input from stdin.

```
Syntax:  
int scanf(const char *format, ...)  
  
Return type: Integer  
  
Parameters:  
format: string that contains the type specifier(s)  
"..." (ellipsis): indicates that the function accepts  
a variable number of arguments
```

Each argument must be a memory address where the converted result is written to. On success, the function returns the number of variables filled. In case of an input failure, before any data could be successfully read, EOF is returned.

Type specifiers that can be used in scanf:

```
%c - Character  
%d - Signed integer  
%f - Floating point  
%s - String  
  
filter_none  
edit  
close  
  
play_arrow  
  
link  
brightness_4  
code
```

```
//C program t illustrate scanf statement
#include <stdio.h>
#include <stdlib.h>

int main()
{
    char a[10];
    printf("Please enter your name : \n");

    //scanf statement
    scanf("%s", a);

    printf("You entered: \n%s", a);

    return 0;
}
```

chevron_right

filter_none

Input:

Geek

Output:

```
Please enter your name :
You entered:
Geek
```

2. **sscanf()**: sscanf() is used to read formatted input from the string.

Syntax:
`int sscanf (const char * s, const char * format, ...);`

Return type: Integer

Parameters:

`s: string` used to retrieve data
`format: string` that contains the type specifier(s)
... : arguments contains pointers to allocate storage **with** appropriate type.

There should be at least **as many of** these arguments **as** the number of values stored **by** the format specifiers.

On success, the function returns the number of variables filled. In the case of an input failure, before any data could be successfully read, EOF is returned.

filter_none

edit

close

play_arrow

link

brightness_4

code

```
// C program to illustrate sscanf statement
#include <stdio.h>
```

```
int main ()
{
    // declaring array s
    char s [] = "3 red balls 2 blue balls";
    char str [10],str2 [10];
    int i;

    // %*s is used to skip a word
    sscanf (s,"%d %*s %*s %*s %*s", &i, str, str2);

    printf ("%d %s %s \n", i, str, str2);

    return 0;
}
```

chevron_right

filter_none

Output:

3 blue balls

3. **fscanf()**: fscanf() reads formatted data from file and stores it into variables.

Syntax:
`int fscanf(FILE *stream, const char *format, ...)`

Parameters:

`Stream: pointer to the File object` that identifies the stream.
`format : is a string` that contains the type specifier(s)

On success, the function returns the number of variables filled. In the case of an input failure, before any data could be successfully read, EOF is returned.

filter_none

edit

close

play_arrow

link

brightness_4

code

```
// C program to illustrate fscanf statement
// This program will run on system having the file file.txt
#include <stdio.h>
#include <stdlib.h>
```

```

int main()
{
    char s1[10], s2[10], s3[10];
    int year;

    // file pointer
    FILE * fp;

    // opening/creation of file
    fp = fopen ("file.txt", "w+");

    // storing string in the file
    fputs("Hello World its 2017", fp);

    // sets the file position to the beginning of the file
    rewind(fp);

    // taking input from file
    fscanf(fp, "%s %s %d", s1, s2, s3, &year);

    printf("String1 |%s|\n", s1 );
    printf("String2 |%s|\n", s2 );
    printf("String3 |%s|\n", s3 );
    printf("Integer |%d|\n", year );

    // close file pointer
    fclose(fp);

    return(0);
}

```

chevron_right

filter_none

Output:

```

String1 |Hello|
String2 |World|
String3 |its|
Integer |2017|

```

4. **scanf_s()** : This function is specific to Microsoft compilers. It is the same as scanf, except it does not cause buffer overload.

Syntax:

```
int scanf_s(const char *format [argument]...);
```

argument(parameter): here you can specify the buffer size **and** actually control the limit of the input so you don't crash the whole application.

On success, the function returns the number of variables filled. In the case of an input failure, before any data could be successfully read, EOF is returned.

Why to use scanf_s()?

scanf just reads whatever input is provided from the console. C does not check whether the user input will fit in the variable that you've designated.

If you have an array called color[3] and you use scanf for "Red", it will work fine but if user enters more than 3 characters scanf starts writing into memory that doesn't belong to colour. C won't catch this or warn you and it might or might not crash the program, depending on if something tries to access and write on that memory slot that doesn't belong to color. This is where scanf_s comes into play. scanf_s checks that the user input will fit in the given memory space.

filter_none

edit

close

play_arrow

link

brightness_4

code

```

// C program to illustrate sscanf_s statement
// scanf_s() will only work in Microsoft Visual Studio.
#include <stdio.h>
#include <stdlib.h>

```

int main()

{

char a[5];

// sizeof(a) is buffer size
scanf_s("%s", a, sizeof(a));

printf("\n%s ", a);

return 0;

}

chevron_right

filter_none

Input:

Red

Output:

Red

Input:

Yellow

Output:

No Output

Illustrating the relation between buffer size and array size.

C

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// C program
// consumes the Enter key
// (newline character) pressed after input
#include<stdio.h>

char ch[100000];
printf("Enter characters: ");
scanf_s("%s", ch, 99999);
getchar();
chevron_right
filter_none
```

C++

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// C++ program
// consumes the Enter key
// (newline character) pressed after input

#include "stdafx.h"

int _tmain(int argc, _TCHAR* argv[])
{
    // example
    char ch[100000];
    printf("Enter characters: ");
    scanf_s("%s", ch, 99999);
    getchar();
    return 0;
}
chevron_right
filter_none
```

1. If the buffer size is equal to or smaller than the size of the array, then inputting bigger than or equal to the buffer size will do nothing.

2. If the buffer size is bigger than the size of an array, then

- a. inputting smaller than buffer size will work out but will give an error

"Run-Time Check Failure #2 - Stack around the variable 'variable_name' was corrupted."

- b. inputting bigger than buffer size will do nothing and give the same error.

5. **fscanf_s()** : Difference between fscanf() and fscanf_s() is same as that of scanf() and scanf_s(). fscanf_s() is secure function and secure functions require the size of each c, C, s, S and [type field to be passed as an argument immediately following the variable.

Syntax:
`int fscanf_s(FILE *stream, const char *format ,[argument]...);`

fscanf_s has an extra argument(parameter) where you can specify the buffer size and actually control the limit of the input.

On success, the function returns the number of variables filled. In the case of an input failure, before any data could be successfully read, EOF is returned.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

//C program to illustrate fscanf_s statement
//This program will run on MS Visual studio
#include <stdio.h>
#include <stdlib.h>

int main()
{
    char s1[10], s2[10], s3[10];
    int year;

    // file pointer
    FILE * fp;

    // Open file securely
    fopen_s(&fp,"file.txt", "w+");

    fputs("Hello World its 2017", fp);
```

```

rewind(fp);

// Using fscanf_s
fscanf_s(fp, "%s", s1, sizeof(s1));
fscanf_s(fp, "%s", s2, sizeof(s2));
fscanf_s(fp, "%s", s3, sizeof(s3));
fscanf_s(fp, "%d", &year, sizeof(year));

printf("String1 |$|\n", s1);
printf("String2 |$|\n", s2);
printf("String3 |$|\n", s3);
printf("Integer |$|\n", year);

fclose(fp);

```

return(0);

}

chevron_right

filter_none

Output:

```

String1 |Hello|
String2 |World|
String3 |its|
Integer |2017|

```

6. **sscanf_s()** : sscanf_s() is secure function of sscanf() and secure functions require the size of each c, C, s, S and [type field to be passed as an argument immediately following the variable.

Syntax:

```
int sscanf_s(const char *restrict buffer, const char *restrict format, ...);
```

sscanf_s has an extra argument(parameter) **where** you can specify the buffer size **and** actually control the limit **of** the input.

On success, the function returns the number of variables filled. In the case of an input failure, before any data could be successfully read, EOF is returned.

filter_none

edit

close

play_arrow

link

brightness_4

code

```
//C program to illustrate sscanf_s statement
//This program will run on MS Visual studio
#include <stdio.h>
```

```

int main()
{
    char s[] = "3 red balls 2 blue balls";
    char str[10], str2[10];
    int i;

    // %*s is used to skip a word
    sscanf_s(s, "%d", &i, sizeof(i));
    sscanf_s(s, "%*d %*s %*s %*s %*s", str, sizeof(str));
    sscanf_s(s, "%*d %*s %*s %*s %*s", str2, sizeof(str2));

    printf("%d %s %s \n", i, str, str2);
}

```

return 0;

}

chevron_right

filter_none

Output:

```
3 blue balls
```

Note: sscanf_s() will only work in Microsoft Visual Studio.

This article is contributed by **Kartik Ahuja**. If you like GeeksforGeeks and would like to contribute, you can also write an article using [contribute.geeksforgeeks.org](#) or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up

Add your personal notes
here! (max 5000 chars)

Save

Recommended Posts:

- [scanf\(\) and fscanf\(\) in C - Simple Yet Powerful](#)
- [C Library math.h functions](#)
- [Problem with scanf\(\) when there is fgets\(\)/gets\(\)/scanf\(\) after it](#)
- [scanf\("%\[^n\]s", str\) Vs gets\(str\) in C with Examples](#)
- [Why to use fgets\(\) over scanf\(\) in C?](#)
- [Difference between scanf\(\) and gets\(\) in C](#)
- [Use of & in scanf\(\) but not in printf\(\)](#)
- [Cin-Cout vs Scanf-Printf](#)
- [Why "&" is not used for strings in scanf\(\) function?](#)
- [Return values of printf\(\) and scanf\(\) in C/C++](#)
- [snprintf\(\) in C library](#)

- SDL library in C/C++ with examples
- isgraph() C library function
- wprintf() and wscanf in C Library
- wcstof function in C library

Improved By : [vaibhavpandeyprayag](#)

Article Tags :

C
C-Input and Output Quiz
c-input-output
C-Library

Practice Tags :

C



1

To-do Done
2.5

Based on 2 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) How to pass an array by value in C ?

Next

[last_page](#) exec family of functions in C

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments



FULL STACK DEVELOPMENT

Geeks Classes Workshop

JAVA APP DEVELOPMENT

COMPETITIVE PROGRAMMING

Java Backend

REACT.JS LIVE

DESIGN PATTERNS

Django WEB DEVELOPMENT



LIVE COURSES
BY
GEEKSFORGEEEKS



Most popular in C
[Print all possible combinations of the string by replacing '\\$' with any other digit from the string](#)
[Predefined Macros in C with Examples](#)
[Format specifiers in different Programming Languages](#)
[C program to print odd line contents of a File followed by even line content](#)
[Introduction to the C99 Programming Language : Part II](#)

More related articles in C
[C program to find square root of a given number](#)
[Problem in comparing Floating point numbers and how to compare them correctly?](#)
[Features of C Programming Language](#)
[Pointer Expressions in C with Examples](#)
[Introduction to the C99 Programming Language : Part III](#)

Interesting Facts in C Programming

Below are some interesting facts about C programming:

1) The case labels of a switch statement can occur inside if-else statements.

```
filter_none
edit
close

play_arrow
link
brightness_4
code

#include <stdio.h>

int main()
{
    int a = 2, b = 2;
    switch(a)
    {
        case 1:
        ;

        if (b==5)
        {
            case 2:
                printf("GeeksforGeeks");
        }
    }
}
```

```
        }
    else case 3:
    {
        }
    }
chevron_right
```

[filter_none](#)

Output :

[GeeksforGeeks](#)

2) arr[index] is same as index[arr]
The reason for this to work is, array elements are accessed using pointer arithmetic.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// C program to demonstrate that arr[0] and
// @arr
#include<stdio.h>
int main()
{
    int arr[10];
    arr[0] = 1;
    printf("%d", @arr);
}

return 0;
}
chevron_right
```

[filter_none](#)

Output :

1

3) We can use '<; :>' in place of '{,}' and '<%, %>' in place of '{,}'

```
filter_none
edit
close
play_arrow
link
brightness_4
code

#include<stdio.h>
int main()
<%
    int arr <:10:>;
    arr<:0:> = 1;
    printf("%d", arr<:0:>);

    return 0;
%>
}
chevron_right
```

[filter_none](#)

Output :

1

4) Using #include in strange places.
Let "a.txt" contains ("GeeksforGeeks");

```
filter_none
edit
close
play_arrow
link
brightness_4
code

#include<stdio.h>
int main()
{
    printf
    #include "a.txt"
    ;
}
chevron_right
```

[filter_none](#)

Output :

[GeeksforGeeks](#)

5) We can ignore input in scanf() by using an '*' after '%' in format specifiers

[filter_none](#)

edit

close

```
play_arrow  
link  
brightness_4  
code  
  
#include<stdio.h>  
int main()  
{  
    int a;  
  
    // Let we input 10 20, we get output as 20  
    // (First input is ignored)  
    // If we remove * from below line, we get 10.  
    scanf("%*d%d", &a);  
  
    printf( "%d ", a);  
  
    return 0;  
}  
chevron_right
```

This article is contributed by **Harsh Agarwal**. If you like GeeksforGeeks and would like to contribute, you can also write an article using [contribute.geeksforgeeks.org](#) or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes 

Add your personal notes here! (max 5000 chars)



Recommended Posts:

- [Py-Facts - 10 interesting facts about Python](#)
- [Interesting Facts about C++](#)
- [Interesting Facts about C#](#)
- [Interesting Facts about Ubuntu](#)
- [Interesting Facts about Linux](#)
- [Interesting Facts About Java](#)
- [C++ bitset interesting facts](#)
- [Interesting facts about C Language](#)
- [Some Interesting facts about default arguments in C++](#)
- [Interesting Facts about Macros and Preprocessors in C](#)
- [Interesting facts about Fibonacci numbers](#)
- [Interesting facts about switch statement in C](#)
- [Interesting facts about strings in Python | Set 1](#)
- [Interesting facts about null in Java](#)
- [Interesting facts about strings in Python | Set 2 \(Slicing\)](#)

Article Tags :

[C](#)

[C-Input and Output Quiz](#)

[c-input-output](#)

[C-Loops & Control Statements](#)

[interesting-facts](#)

Practice Tags :

[C](#)



4

To-do Done
2.2

Based on 15 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

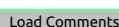
Previous

[first_page Dangling, Void , Null and Wild Pointers](#)

Next

[last_page auto_ptr, unique_ptr, shared_ptr and weak_ptr](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.





Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
Predefined Macros in C with Examples
Format specifiers in different Programming Languages
C program to print odd line contents of a file followed by even line content
C program to find square root of a given number

More related articles in C
Introduction to the C99 Programming Language : Part I
Pointer Comparing Floating point numbers and how to compare them correctly?
Features of C Programming Language
Pointer Expressions in C with Examples
Introduction to the C99 Programming Language : Part III

Database Connectivity using C/C++

SQL (Structured Query Language) is a fourth-generation language (4GL) that is used to define, manipulate, and control an RDBMS (relational database management system).

Before starting the main article, let us get familiar with the used tools.

1. **Compiler:** Code::Blocks IDE with MinGW compiler

Download Link: [Binary Download](#)

Code::Blocks is a cross compiler (it can run on any platform like Windows, Linux and Mac) and it is free to download. This IDE is specially designed for C and C++ and easy to use.

2. **API:** We are going to use SQLAPI++ Library

Download Link: [SQLAPI Download](#)

SQLAPI++ is a C++ library (basically a set of header files) for accessing multiple SQL databases (Oracle, SQL Server, DB2, Sybase, Informix, InterBase, SQLBase, MySQL, PostgreSQL, SQLite, SQL Anywhere and ODBC). It is easy to implement and simple.

3. **OCCI:** Oracle C++ Call Interface

Download Link: [OCCI C++ Download](#)

OCCI is an interface defined by the database company ORACLE that defines a comfortable interface for the C++ programmer to access the Oracle database with classes using parameters that are reminiscent of SQL statements. The interface exists for ORACLE 9i, ORACLE 10 and comes with the Oracle.

We must download and install the above three (if we don't have them). Now we are almost ready to start.

Some settings before starting:

-> Open the code::blocks IDE and go to or click on **settings -> compiler and debugger settings** (You will now see global compiler settings)

-> Now click on "**Linker settings**" in the linker settings click on ADD button and add the following

For **Windows OS :**

Code:

C:\SQLAPI\lib\libsqlapiddll.a

C:\Program Files\CodeBlocks\MinGW\lib\libuser32.a

C:\Program Files\CodeBlocks\MinGW\lib\libversion.a

C:\Program Files\CodeBlocks\MinGW\lib\liboleaut32.a

C:\Program Files\CodeBlocks\MinGW\lib\libole32.a

These will be found in your SQLAPI++ (If you have not extracted in C: drive then select the appropriate location and add the mentioned files to linker settings).

The above code is used to add library files to connect C/C++ program with SQLAPI.

Basically, there are 2 steps:

1. **Connecting to database (and error handling)**

Code:

filter_none

edit

close

play_arrow

link

brightness_4

code

// C++ program for connecting to database (and error handling)

#include<stdio.h>

#include<SQLAPI.h> // main SQLAPI++ header

int main(int argc, char* argv[])

```

{
    // create connection object to connect to database
    SAConnection con;
    try
    {
        // connect to database
        // in this example, it is Oracle,
        // but can also be Sybase, Informix, DB2
        // SQLServer, InterBase, SQLBase and ODBC
        con.Connect ("test",      // database name
                    "tester",   // user name
                    "tester",   // password
                    SA_Oracle_Client); //Oracle Client
        printf("We are connected!\n");

        // Disconnect is optional
        // autodisconnect will occur in destructor if needed
        con.Disconnect();
        printf("We are disconnected!\n");
    }

    catch(SAException & x)
    {
        // SAConnection::Rollback()
        // can also throw an exception
        // (if a network error for example),
        // we will be ready
        try
        {
            // on error rollback changes
            con.Rollback ();
        }
        catch(SAException &)
        {
        }
        // print error message
        printf("%s\n", (const char*)x.ErrText());
    }
    return 0;
}

```

chevron_right

filter_none

Output:

```
We are Connected!
We are Disconnected!
```

1. Executing a simple SQL Command

Now, we will look out to execute a simple SQL query. Firstly, creating a table for the database:

```
create table tbl(id number, name varchar(20));
```

Now, establish the connection to the database then, after your con.connect; method you should use cmd.setCommandText method to pass the query to the database, it is as shown below:

```
con.Connect("test", "tester", "tester", SA_Oracle_Client);
cmd.setCommandText("create table tbl(id number, name varchar(20));");
```

and now, to execute the query we have to use the following command:

```
cmd.Execute();
```

Full Code:

```

filter_none
edit
close

play_arrow
link
brightness_4
code

#include<stdio.h>
#include <SQLAPI.h> // main SQLAPI++ header
int main(int argc, char* argv[])
{
    SAConnection con; // connection object to connect to database
    SACommandcmd;    // create command object
    try
    {
        // connect to database (Oracle in our example)
        con.Connect("test", "tester", "tester", SA_Oracle_Client);

        // associate a command with connection
        // connection can also be specified in SACommand constructor
        cmd.setConnection(&con);

        // create table
        cmd.setCommandText("create table tbl(id number, name varchar(20));");
        cmd.Execute();

        // insert value
        cmd.setCommandText("Insert into tbl(id, name) values (1,'Vinay')");
        cmd.setCommandText("Insert into tbl(id, name) values (2,'Kushal')");
    }
}
```

```

cmd.setCommandText("Insert into tbl(id, name) values (3,'Saransh')");
cmd.ExecuteNonQuery();

// commit changes on success
con.Commit();
printf("Table created, row inserted!\n");
}

catch(SAEException &x)
{
    // SAEConnection::Rollback()
    // can also throw an exception
    // (if a network error for example),
    // we will be ready
    try
    {
        // on error rollback changes
        con.Rollback();
    }
    catch(SAEException &)
    {
    }
    // print error message

    printf("%s\n", (const char*)x.ErrText());
}
return 0;
}
chevron_right
filter_none

```

As we know, Oracle is not auto committed (committing is making permanent reflection of data in the database) so, we have to commit it.

`con.Commit();`

and similarly we can roll back the transactions when an exception occurs, so to do that we use:

`con.Rollback();`

For deleting a row, we use this command.

`cmd.setCommandText("delete from tb1 where id= 2");`

Thus, by the end of this article, we have learned how to connect your C/C++ program to database and perform manipulations.

This article is contributed by **Vinay Garg**. If you like GeeksforGeeks and would like to contribute, you can also write an article and mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes 

Add your personal notes
here! (max 5000 chars)

Recommended Posts:

- [Create a database on Relational Database Service \(RDS\) of Amazon Web Services\(AWS\)](#)
- [How and Why To Create an SQL Database on Azure](#)
- [Apache Cassandra \(NOSQL database\)](#)
- [Top 7 Database You Must Know For Software Development Projects](#)
- [Database Connection and Queries in CodeIgniter](#)
- [PHP program to fetch data from localhost server database using XAMPP](#)
- [What is Edge Computing and Its Importance in the Future?](#)
- [std::basic_istream::ignore in C++ with Examples](#)
- [std::basic_istream::getline in C++ with Examples](#)
- [Top 10 Python Libraries for Data Science in 2020](#)
- [Format specifiers in different Programming Languages](#)
- [C program to find square root of a given number](#)
- [C program to print odd line contents of a File followed by even line content](#)
- [std::is_heap\(\) in C++ with Examples](#)

Article Tags :

C
C++
GBlog
CPP-Library
Practice Tags :
C
CPP

 4

To-do Done
3.5

Based on 7 vote(s)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) C++ string class and its applications

Next

[last_page](#) Secure coding - What is it all about?

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Weekend Geeks Classes Weekdays

JAVA APP DEVELOPMENT SUMMER TRAINING COMPETITIVE PROGRAMMING LIVE Java Backend Development

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
ctype.h<ctype.h> library in C/C++ with Examples
C program to display month by month calendar for a given year
Predefined Macros in C with Examples
Implicit Type Conversion in C with Examples

Most visited in C++
Vector of Vectors in C++ STL with Examples
C++ Tutorial
Which C++ libraries are useful for competitive programming?
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

Function Interposition in C with an example of user defined malloc()

Function interposition is the concept of replacing calls to functions in dynamic libraries with calls to user-defined wrappers.

What are applications?

1. We can count number of calls to function.
2. Store caller's information and arguments passed to function to track usage.
3. Detect memory leak, we can override malloc() and keep track of allocated spaces.
4. We can add our own security policies. For example, we can add a policy that fork cannot be called with more than specified recursion depth.

How to do function interposition?

The task is to write our own malloc() and make sure our own malloc() is called instead of library malloc(). Below is a driver program to test different types of interpositions of malloc().

```
filter_none
edit
close
play_arrow
link
brightness_4
code

// File Name : hello.c

#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>

int main(void)
{
    // Call to user defined malloc
    void *ptr = malloc(4);

    printf("Hello World\n");
    return 0;
}
chevron_right
filter_none
```

1. **Compile time** : Replace library call with our own function when the source code is compiled.

```
filter_none
edit
close
play_arrow
link
brightness_4
code

/* Compile-time interposition of malloc using C preprocessor.
A local malloc.h file defines malloc as wrapper */

// A file that contains our own malloc function
// File Name : mymalloc.c
#include <stdio.h>
#include <malloc.h>
void *mymalloc(size_t s)
```

```

{
    printf("My malloc called");
    return NULL;
}
chevron_right
filter_none
filter_none
edit
close
play_arrow
link
brightness_4
code

// filename : malloc.h
// To replace all calls to malloc with mymalloc
#define malloc(size) mymalloc(size)
void *mymalloc(size_t size);
chevron_right
filter_none

```

Steps to execute above on Linux:

```

// Compile the file containing user defined malloc()
:~$ gcc -c mymalloc.c

// Compile hello.c with output file name as helloc.
// -I. is used to include current folder (.) for header
// files to make sure our malloc.h is becomes available.
:~$ gcc -I. -o helloc hello.c mymalloc.o

// Run the generated executable
:~$ ./helloc
My malloc called
Hello World

```

- Link time : When the relocatable object files are statically linked to form an executable object file.

```

filter_none
edit
close
play_arrow
link
brightness_4
code

// filename : mymalloc.c
/* Link-time interposition of malloc using the
   static linker's (ld) "--wrap symbol" flag. */
#include <stdio.h>

// __real_malloc() is used to called actual library
// malloc()
void *_real_malloc(size_t size);

// User defined wrapper for malloc()
void *_wrap_malloc(size_t size)
{
    printf("My malloc called");
    return NULL;
}
chevron_right
filter_none

```

Steps to execute above on Linux:

```

// Compile the file containing user defined malloc()
:~$ gcc -c mymalloc.c

// Compile hello.c with output name as helloc
// "-Wl,--wrap=malloc" is used tell the linker to use
// malloc() to call __wrap_malloc(). And to use
// __real_malloc() to actual library malloc()
:~$ gcc -Wl,--wrap=malloc -o helloc hello.c mymalloc.o

// Run the generated executable
:~$ ./helloc
My malloc called
Hello World

```

- Load/run time : When an executable object file is loaded into memory, dynamically linked, and then executed.

The environment variable **LD_PRELOAD** gives the loader a list of libraries to load before a command or executable.
We make a dynamic library and make sure it is loaded before our executable for hello.c.

```

filter_none
edit
close
play_arrow
link
brightness_4
code

/* Run-time interposition of malloc based on dynamic linker's
   (ld-linux.so) LD_PRELOAD mechanism */
#define __GNU_SOURCE
#include <stdio.h>

void *malloc(size_t s)
```

```
{  
    printf("My malloc called\n");  
    return NULL;  
}  
chevron_right
```

filter_none

Steps to execute above on Linux:

```
// Compile hello.c with output name as helloc  
:~$ gcc -o hellor hello.c  
  
// Generate a shared library mymalloc.so. Refer  
// https://www.geeksforgeeks.org/working-with-shared-libraries-set-2/  
// for details.  
:~$ gcc -fPIC -shared -o mymalloc.so mymalloc.c  
  
// Make sure shared library is loaded and run before .  
:~$ LD_PRELOAD=./mymalloc.so ./hellor  
My malloc called  
Hello World
```

The code for user defined malloc is kept small for better readability. Ideally, it should allocate memory by making a call to library malloc().

Source:

<https://www.utdallas.edu/~zxl111930/spring2012/public/lec18-handout.pdf>

This article is contributed by **Aditya Chatterjee**. If you like GeeksforGeeks and would like to contribute, you can also write an article and mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up

Add your personal notes
here! (max 5000 chars)

Save

Recommended Posts:

- [User Defined Literals in C++](#)
- [User defined Data Types in C++](#)
- [2D vector in C++ with user defined size](#)
- [How to create an unordered_map of user defined class in C++?](#)
- [How to implement user defined Shared Pointers in C++](#)
- [Multi-set for user defined data type](#)
- [How to create an unordered_set of user defined class or struct in C++?](#)
- [malloc\(\) vs new](#)
- [new vs malloc\(\) and free\(\) vs delete in C++](#)
- [Difference Between malloc\(\) and calloc\(\) with Examples](#)
- [Dynamic Memory Allocation in C using malloc\(\), calloc\(\), free\(\) and realloc\(\)](#)
- [How Linkers Resolve Global Symbols Defined at Multiple Places?](#)
- [C++ map having key as a user define data type](#)
- [C++ set for user define data type](#)
- [Inbuilt library functions for user Input | scanf, fscanf, sscanf, scanf_s, fscanf_s, sscanf_s](#)

Article Tags :

C
C++
CPP-Functions
system-programming
Practice Tags :
C
CPP

thumb_up

1

To-do Done

3

Based on 1 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) Solving f(n) = (1) + (2*3) + (4*5*6) ... n using Recursion

Next

[last_page](#) Lexicographically next permutation in C++

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
How to call function within function in C or C++
Format specifiers in different Programming Languages
Predefined Macros in C with Examples
C program to print odd line contents of a File followed by even line content

Most visited in C++
Vector of Vectors in C++ STL with Examples
C++ STL with Examples
Which C++ libraries are useful for competitive programming?
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

Macros vs Functions

Macros are **pre-processed** which means that all the macros would be processed before your program compiles. However, functions are **not preprocessed but compiled**.

See the following example of Macro:

```
filter_none
edit
close

play_arrow

link
brightness_4
code

#include<stdio.h>
#define NUMBER 10
int main()
{
    printf("%d", NUMBER);
    return 0;
}
chevron_right
filter_none
```

Output:

10

See the following example of Function:

```
filter_none
edit
close

play_arrow

link
brightness_4
code

#include<stdio.h>
int number()
{
    return 10;
}
int main()
{
    printf("%d", number());
    return 0;
}
chevron_right
filter_none
```

Output:

10

Now compile them using the command:

gcc -E file_name.c

This will give you the executable code as shown in the figure:

```

};

# 943 "/usr/include/stdio.h" 3 4
# 2 "example.c" 2
int number(){
    return 10;
}
int main(){
    printf("%d", number());
    return 0;
}
pranjal@ubuntu:~/Desktop$ █

```

This shows that the macros are preprocessed while functions are not.

In macros, no type checking(incompatible operand, etc.) is done and thus use of macros can lead to errors/side-effects in some cases. However, this is not the case with functions. Also, macros do not check for compilation error (if any). Consider the following two codes:

Macros:

```

filter_none
edit
close

play_arrow
link
brightness_4
code

#include<stdio.h>
#define CUBE(b) b*b*b
int main()
{
    printf("%d", CUBE(1+2));
    return 0;
}
chevron_right
filter_none

```

Output: Unexpected output

7

Functions:

```

filter_none
edit
close

play_arrow
link
brightness_4
code

#include<stdio.h>
int cube(int a)
{
    return a*a*a;
}
int main()
{
    printf("%d", cube(1+2));
    return 0;
}
chevron_right
filter_none

```

Output: As expected

27

- Macros are usually one liner. However, they can consist of more than one line. Click [here](#) to see the usage. There are no such constraints in functions.
- The speed at which macros and functions differs. Macros are typically faster than functions as they don't involve actual function call overhead.

Conclusion:

Macros are no longer recommended as they cause following issues. There is a better way in modern compilers that is inline functions and const variable. Below are disadvantages of macros:

- There is no type checking
- Difficult to debug as they cause simple replacement.
- Macro don't have namespace, so a macro in one section of code can affect other section.
- Macros can cause side effects as shown in above CUBE() example.

MACRO

Macro is Preprocessed
 No Type Checking is done in Macro
 Using Macro increases the code length
 Use of macro can lead to side effect at later stages
 Speed of Execution using Macro is Faster
 Before Compilation, macro name is replaced by macro value
 Macros are useful when small code is repeated many times
 Macro does not check any Compile-Time Errors
 See following for more details on macros:
[Interesting facts about Macros and Preprocessors](#)

FUNCTION

Function is Compiled
 Type Checking is Done in Function
 Using Function keeps the code length unaffected
 Functions do not lead to any side effect in any case
 Speed of Execution using Function is Slower
 During function call, transfer of control takes place
 Functions are useful when large code is to be written
 Function checks Compile-Time Errors

This article is contributed by **Pranjal Mathur**. If you like GeeksforGeeks and would like to contribute, you can also write an article and mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

Add your personal notes here! (max 5000 chars)

Save

Recommended Posts:

- X-Macros in C
- Importance of macros in C++
- Multiline macros in C
- Hygienic Macros : An Introduction
- Branch prediction macros in GCC
- Output of C++ programs | Set 25 (Macros)
- Predefined Macros in C with Examples
- Output of the Program | Use Macros Carefully!
- Data Type Ranges and their macros in C++
- Interesting Facts about Macros and Preprocessors in C
- Variable length arguments for Macros
- Functions in C/C++
- Thread functions in C/C++
- Nested functions in C
- C++ Mathematical Functions

Improved By : [RishabhPrabhu](#)

Article Tags :

C
C++
CPP-Functions
cpp-macros

Practice Tags :

C
CPP



10

To-do Done
2.1

Based on 11 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) How to print size of array parameter in C++?

Next

[last_page](#) Solving f(n) = (1 + (2*3) + (4*5*6) ... n using Recursion

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments

GEEKSFORGEEKS

FULL STACK DEVELOPMENT Geeks Classes Weekend Geeks Classes Weekdays

JAVA APP DEVELOPMENT SUMMER TRAINING COMPETITIVE PROGRAMMING DIVE Java Backend Development

React.js LIVE Design Patterns django WEB DEVELOPMENT

LIVE COURSES BY GEEKSFORGEEKS

Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
Predefined Macros in C with Examples
How to call function within function in C or C++
C program to print odd line contents of a File followed by even line content
Introduction to the C99 Programming Language : Part II

Most visited in C++
Vector of Vectors in C++ STL with Examples
C++ Tutorial
Which C++ libraries are useful for competitive programming?
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

Write your own memcpy() and memmove()

The `memcpy` function is used to copy a block of data from a source address to a destination address. Below is its prototype.

```
void * memcpy(void * destination, const void * source, size_t num);
```

The idea is to simply typecast given addresses to `char *`(`char` takes 1 byte). Then one by one copy data from source to destination. Below is implementation of this idea.

filter_none
edit
close

```

play_arrow
link
brightness_4
code

// A C implementation of memcpy()
#include<stdio.h>
#include<string.h>

void myMemcpy(void *dest, void *src, size_t n)
{
    // Typecast src and dest addresses to (char *)
    char *csrc = (char *)src;
    char *cdest = (char *)dest;

    // Copy contents of src[] to dest[]
    for (int i=0; i<n; i++)
        cdest[i] = csrc[i];
}

// Driver program
int main()
{
    char csrc[] = "GeeksforGeeks";
    char cdest[100];
    myMemcpy(cdest, csrc, strlen(csrc)+1);
    printf("Copied string is %s", cdest);

    int isrc[] = {10, 20, 30, 40, 50};
    int n = sizeof(isrc)/sizeof(isrc[0]);
    int idest[n], i;
    myMemcpy(idest, isrc, sizeof(isrc));
    printf("\nCopied array is ");
    for (i=0; i<n; i++)
        printf("%d ", idest[i]);
    return 0;
}
chevron_right
filter_none

```

Output:

```
Copied string is GeeksforGeeks
Copied array is 10 20 30 40 50
```

What is memmove()?

memmove() is similar to memcpy() as it also copies data from a source to destination. memcpy() leads to problems when source and destination addresses overlap as memcpy() simply copies data one by one from one location to another. For example consider below program.

```

filter_none
edit
close

play_arrow
link
brightness_4
code

// Sample program to show that memcpy() can loose data.
#include <stdio.h>
#include <string.h>
int main()
{
    char csrc[100] = "Geeksfor";
    memcpy(csrc+5, csrc, strlen(csrc)+1);
    printf("%s", csrc);
    return 0;
}
chevron_right
filter_none

```

Output:

```
GeeksGeeksfor
```

Since the input addresses are overlapping, the above program overwrites the original string and causes data loss.

```

filter_none
edit
close

play_arrow
link
brightness_4
code

// Sample program to show that memmove() is better than memcpy()
// when addresses overlap.
#include <stdio.h>
#include <string.h>
int main()
{
    char csrc[100] = "Geeksfor";
    memmove(csrc+5, csrc, strlen(csrc)+1);
    printf("%s", csrc);
}
```

```
return 0;
```

```
}
```

```
chevron_right
```

```
filter_none
```

```
Output:
```

```
GeeksGeeksfor
```

How to implement memmove()?

The trick here is to use a temp array instead of directly copying from src to dest. The use of temp array is important to handle cases when source and destination addresses are overlapping.

```
filter_none
```

```
edit
```

```
close
```

```
play_arrow
```

```
link
```

```
brightness_4
```

```
code
```

```
//C++ program to demonstrate implementation of memmove()
```

```
#include<stdio.h>
```

```
#include<string.h>
```

```
// A function to copy block of 'n' bytes from source
```

```
// address 'src' to destination address 'dest'.
```

```
void myMemMove(void *dest, void *src, size_t n)
```

```
{
```

```
    // Typecast src and dest addresses to (char *)
```

```
    char *csrc = (char *)src;
```

```
    char *cdest = (char *)dest;
```

```
    // Create a temporary array to hold data of src
```

```
    char *temp = new char[n];
```

```
    // Copy data from csrc[] to temp[]
```

```
    for (int i=0; i<n; i++)
```

```
        temp[i] = csrc[i];
```

```
    // Copy data from temp[] to cdest[]
```

```
    for (int i=0; i<n; i++)
```

```
        cdest[i] = temp[i];
```

```
    delete [] temp;
```

```
}
```

```
// Driver program
```

```
int main()
```

```
{
```

```
    char csrc[100] = "Geeksfor";
```

```
    myMemMove(csrc+5, csrc, strlen(csrc)+1);
```

```
    printf("%s", csrc);
```

```
    return 0;
```

```
}
```

```
chevron_right
```

```
filter_none
```

```
Output:
```

```
GeeksGeeksfor
```

Optimizations:

The algorithm is inefficient (and honestly double the time if you use a temporary array). Double copies should be avoided unless it is really impossible.

In this case though it is easily possible to avoid double copies by picking a direction of copy. In fact this is what the memmove() library function does.

By comparing the src and the dst addresses you should be able to find if they overlap.

- If they do not overlap, you can copy in any direction
- If they do overlap, find which end of dest overlaps with the source and choose the direction of copy accordingly.
- If the beginning of dest overlaps, copy from end to beginning
- If the end of dest overlaps, copy from beginning to end
- Another optimization would be to copy by word size. Just be careful to handle the boundary conditions.
- A further optimization would be to use vector instructions for the copy since they're contiguous.

This article is contributed by Saurabh Jain. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes 

Add your personal notes
here! (max 5000 chars)

Save

Recommended Posts:

- [memmove\(\) in C/C++](#)
- [memcpy\(\) in C/C++](#)
- [Write a URL in a C++ program](#)
- [How to write your own header file in C?](#)
- [Write a C program that won't compile in C++](#)
- [When should we write our own assignment operator in C++?](#)
- [When should we write our own copy constructor?](#)
- [C program to write an image in PGM format](#)
- [Write a program that produces different results in C and C++](#)

- Write a C macro PRINT(x) which prints x
- How to write a running C code without main()
- Read/Write structure to a file in C
- Does C++ compiler create default constructor when we write our own?
- Write one line functions for strcat() and strcmp()
- Read/Write Class Objects from/to File in C++

Improved By : SwatiGangwar

Article Tags :

C
C++
CPP-Library
cpp-pointer
Practice Tags :
C
CPP

thumb_up
4

To-do Done
3.4

Based on 13 vote(s)

Basic Easy Medium Hard Expert

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

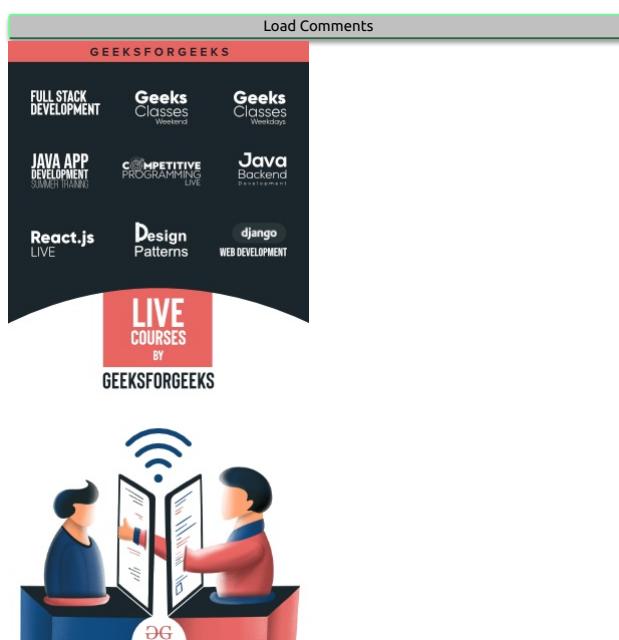
Previous

[first_page](#) Difference between getc(), getchar(), getch() and getche()

Next

[last_page](#) What happens when a virtual function is called inside a non-virtual function in C++

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
How to call function within function in C or C++
Format specifier in different Programming Languages
Predefined Macros in C with Examples
C program to print odd line contents of a file followed by even line content

Most visited in C++
Vector of Vectors in C++ STL with Examples
C++ Tutorial
Which C++ libraries are useful for competitive programming?
Array of Vectors in C++ STL
Map of Vectors in C++ STL with Examples

Commonly Asked C Programming Interview Questions | Set 1

What is the difference between declaration and definition of a variable/function

Ans: Declaration of a variable/function simply declares that the variable/function exists somewhere in the program but the memory is not allocated for them. But the declaration of a variable/function serves an important role. And that is the type of the variable/function. Therefore, when a variable is declared, the program knows the data type of that variable. In case of function declaration, the program knows what are the arguments to that functions, their data types, the order of arguments and the return type of the function. So that's all about declaration. Coming to the definition, when we define a variable/function, apart from the role of declaration, it also allocates memory for that variable/function. Therefore, we can think of definition as a super set of declaration. (or declaration as a subset of definition).

```
// This is only declaration. y is not allocated memory by this statement
extern int y;

// This is both declaration and definition, memory to x is allocated by this statement.
int x;
```

What are different storage class specifiers in C?

Ans: auto, register, static, extern

What is scope of a variable? How are variables scoped in C?

Ans: Scope of a variable is the part of the program where the variable may directly be accessible. In C, all identifiers are lexically (or statically) scoped. See [this](#) for more details.

How will you print "Hello World" without semicolon?

Ans:

[filter_none](#)

[edit](#)
[close](#)
[play_arrow](#)
[link](#)
[brightness_4](#)
[code](#)

```
#include <stdio.h>
int main(void)
{
    if (printf("Hello World")) {
    }
}
```

[chevron_right](#)
[filter_none](#)

See print "Geeks for Geeks" without using a semicolon for answer.

When should we use pointers in a C program?

1. To get address of a variable
2. For achieving pass by reference in C: Pointers allow different functions to share and modify their local variables.
3. To pass large structures so that complete copy of the structure can be avoided.
4. To implement "linked" data structures like linked lists and binary trees.

What is NULL pointer?

Ans: NULL is used to indicate that the pointer doesn't point to a valid location. Ideally, we should initialize pointers as NULL if we don't know their value at the time of declaration. Also, we should make a pointer NULL when memory pointed by it is deallocated in the middle of a program.

What is Dangling pointer?

Ans: Dangling Pointer is a pointer that doesn't point to a valid memory location. Dangling pointers arise when an object is deleted or deallocated, without modifying the value of the pointer, so that the pointer still points to the memory location of the deallocated memory. Following are examples.

[filter_none](#)
[edit](#)
[close](#)

[play_arrow](#)

[link](#)
[brightness_4](#)
[code](#)

```
// EXAMPLE 1
int* ptr = (int*)malloc(sizeof(int));
.....free(ptr);
```

// ptr is a dangling pointer now and operations like following are invalid
*ptr = 10; // or printf("%d", *ptr);

[chevron_right](#)
[filter_none](#)

[filter_none](#)
[edit](#)
[close](#)

[play_arrow](#)

[link](#)
[brightness_4](#)
[code](#)

```
// EXAMPLE 2
int* ptr = NULL;
{
    int x = 10;
    ptr = &x;
}
```

// x goes out of scope and memory allocated to x is free now.

// So ptr is a dangling pointer now.

[chevron_right](#)
[filter_none](#)

What is memory leak? Why it should be avoided

Ans: Memory leak occurs when programmers create a memory in heap and forget to delete it. Memory leaks are particularly serious issues for programs like daemons and servers which by definition never terminate.

[filter_none](#)
[edit](#)
[close](#)

[play_arrow](#)

[link](#)
[brightness_4](#)
[code](#)

```
/* Function with memory leak */
#include <stdlib.h>
```

```
void f()
{
    int* ptr = (int*)malloc(sizeof(int));

    /* Do some work */

    return; /* Return without freeing ptr*/
}
```

[chevron_right](#)
[filter_none](#)

What are local static variables? What is their use?

Ans: A local static variable is a variable whose lifetime doesn't end with a function call where it is declared. It extends for the lifetime of complete program. All calls to the function share the same copy of local static variables. Static variables can be used to count the number of times a function is called. Also, static variables get the default value as 0. For example, the following program prints "0 1"

```

filter_none
edit
close
play_arrow
link
brightness_4
code

#include <stdio.h>
void fun()
{
    // static variables get the default value as 0.
    static int x;
    printf("%d ", x);
    x = x + 1;
}

int main()
{
    fun();
    fun();
    return 0;
}
// Output: 0 1
chevron_right
filter_none

```

What are static functions? What is their use?

Ans:In C, functions are global by default. The "static" keyword before a function name makes it static. Unlike global functions in C, access to static functions is restricted to the file where they are declared. Therefore, when we want to restrict access to functions, we make them static. Another reason for making functions static can be reuse of the same function name in other files. See [this](#) for examples and more details.

- [Commonly Asked C Programming Interview Questions | Set 2](#)
- [Practices Quizzes on C](#)
- [C articles](#)

You may also like:

[Commonly Asked C Programming Interview Questions | Set 2](#)
[Commonly Asked Java Programming Interview Questions | Set 1](#)
[Amazon's most asked interview questions](#)
[Microsoft's most asked interview questions](#)
[Accenture's most asked Interview Questions](#)
[Commonly Asked OOP Interview Questions](#)
[Commonly Asked C++ Interview Questions](#)
[Commonly asked DBMS interview questions | Set 1](#)
[Commonly asked DBMS interview questions | Set 2](#)
[Commonly Asked Operating Systems Interview Questions | Set 1](#)
[Commonly Asked Data Structure Interview Questions.](#)
[Commonly Asked Algorithm Interview Questions](#)
[Commonly asked Computer Networks Interview Questions](#)
[Top 10 algorithms in Interview Questions](#)

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes [arrow_drop_up](#)

Add your personal notes
here! (max 5000 chars)

[Save](#)

Recommended Posts:

- [Commonly Asked C Programming Interview Questions | Set 2](#)
- [Commonly Asked C Programming Interview Questions | Set 3](#)
- [Commonly Asked Java Programming Interview Questions | Set 2](#)
- [Commonly Asked OOP Interview Questions | Set 1](#)
- [Commonly Asked C++ Interview Questions | Set 1](#)
- [Commonly Asked C++ Interview Questions | Set 2](#)
- [Commonly Asked Algorithm Interview Questions | Set 1](#)
- [Commonly asked JavaScript Interview Questions | Set 1](#)
- [Commonly Asked Data Structure Interview Questions | Set 1](#)
- [Commonly Asked Questions in Goldman Sachs Interviews](#)
- [Amazon's most frequently asked interview questions | Set 2](#)
- [10 Most asked Questions from Java Programmers](#)
- [Frequently Asked Questions regarding Placements](#)
- [MAQ Software most Frequently Asked Questions](#)
- [Top 20 Dynamic Programming Interview Questions](#)

Improved By : [moosavi001](#), [balajivanole](#)

Article Tags :

C
[interview-preparation](#)
[placement preparation](#)

Practice Tags :

C

[thumb_up](#)

19

To-do Done
2.2

Based on 27 vote(s)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

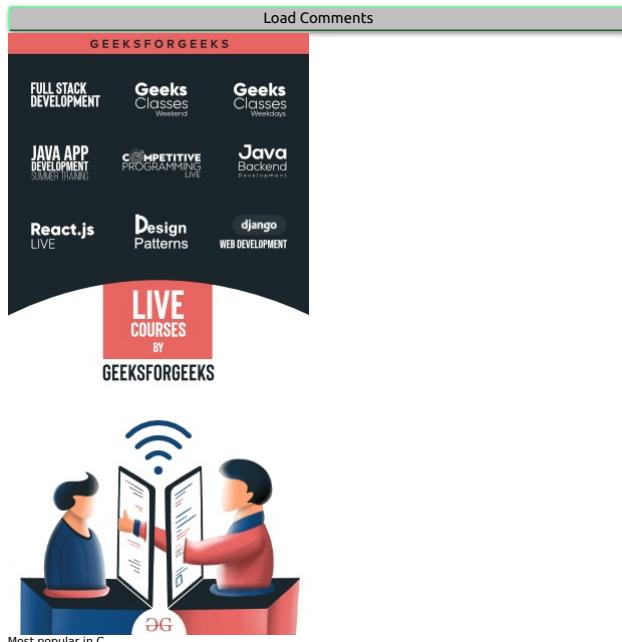
Previous

[first_page](#) C | Variable Declaration and Scope | Question 6

Next

[last_page](#) Write a C program that won't compile in C++

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.



Most popular in C
[Print all possible combinations of the string by replacing 's' with any other digit from the string](#)
[How to print function within function in C or C++](#)
[Format specifiers in different Programming Languages](#)
[Predefined Macros in C with Examples](#)
[C program to print odd line contents of a File followed by even line content](#)

More related articles in C
[C program to find square root of a given number](#)
[Introduction to the C99 Programming Language : Part II](#)
[Problem in comparing Floating point numbers and how to compare them correctly?](#)
[Features of C Programming Language](#)
[<cfloat> float.h in C/C++ with Examples](#)

Commonly Asked C Programming Interview Questions | Set 2

This post is second set of [Commonly Asked C Programming Interview Questions | Set 1](#)

What are main characteristics of C language?

C is a procedural language. The main features of C language include low-level access to memory, simple set of keywords, and clean style. These features make it suitable for system programming like operating system or compiler development.

What is difference between i++ and ++i?

- 1) The expression 'i++' returns the old value and then increments i. The expression ++i increments the value and returns new value.
 - 2) Precedence of postfix ++ is higher than that of prefix ++.
 - 3) Associativity of postfix ++ is left to right and associativity of prefix ++ is right to left.
 - 4) In C++, ++i can be used as l-value, but i++ cannot be. In C, they both cannot be used as l-value.
- See [Difference between ++*p, *p++ and *++p](#) for more details.

What is l-value?

l-value or location value refers to an expression that can be used on left side of assignment operator. For example in expression "a = 3", a is l-value and 3 is r-value.

l-values are of two types:

"nonmodifiable l-value" represent a l-value that can not be modified. const variables are "nonmodifiable l-value".

"modifiable l-value" represent a l-value that can be modified.

Refer [lvalue and rvalue in C language](#) for details.

What is the difference between array and pointer?

See [Array vs Pointer](#)

How to write your own sizeof operator?

filter_none
edit
close

play_arrow
link
brightness_4
code

```
#define my_sizeof(type) (char *)(&type+1)-(char *)(&type)
```

chevron_right

filter_none

See [Implement your own sizeof](#) for more details.

How will you print numbers from 1 to 100 without using loop?

We can use recursion for this purpose.

filter_none
edit
close

play_arrow
link
brightness_4
code

```
/* Prints numbers from 1 to n */
```

```
void printNos(unsigned int n)
```

```
{
```

```
    if(n > 0)
```

```
    {
```

```
        printNos(n-1);
```

```
        printf("%d ", n);
```

```
    }
```

```
}
```

```
chevron_right
```

```
filter_none
```

What is volatile keyword?

The volatile keyword is intended to prevent the compiler from applying any optimizations on objects that can change in ways that cannot be determined by the compiler.

Objects declared as volatile are omitted from optimization because their values can be changed by code outside the scope of current code at any time. See [Understanding "volatile" qualifier in C](#) for more details.

Can a variable be both const and volatile?

Yes, the const means that the variable cannot be assigned a new value. The value can be changed by other code or pointer. For example the following program works fine.

```
filter_none
```

```
edit
```

```
close
```

```
play_arrow
```

```
link
```

```
brightness_4
```

```
code
```

```
int main(void)
```

```
{
```

```
    const volatile int local = 10;
```

```
    int *ptr = (int*) &local;
```

```
    printf("Initial value of local : %d \n", local);
```

```
    *ptr = 100;
```

```
    printf("Modified value of local: %d \n", local);
```

```
    return 0;
```

```
}
```

```
chevron_right
```

```
filter_none
```

- Practices [Quizzes on C](#)
- [C articles](#)

We will soon be publishing more sets of commonly asked C programming questions.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GeeksforGeeks has prepared a complete interview preparation course with premium videos, theory, practice problems, TA support and many more features. Please refer [Placement 100](#) for details

My Personal Notes arrow_drop_up

```
Add your personal notes  
here! (max 5000 chars)
```

```
Save
```

Recommended Posts:

- [Commonly Asked C Programming Interview Questions | Set 3](#)
- [Commonly Asked C Programming Interview Questions | Set 1](#)
- [Commonly Asked Java Programming Interview Questions | Set 2](#)
- [Commonly Asked OOP Interview Questions | Set 1](#)
- [Commonly Asked C++ Interview Questions | Set 1](#)
- [Commonly Asked C++ Interview Questions | Set 2](#)
- [Commonly asked JavaScript Interview Questions | Set 1](#)
- [Commonly Asked Algorithm Interview Questions | Set 1](#)
- [Commonly Asked Data Structure Interview Questions | Set 1](#)
- [Commonly Asked Questions in Goldman Sachs Interviews](#)
- [Amazon's most frequently asked interview questions | Set 2](#)
- [10 Most asked Questions from Java Programmers](#)
- [MAQ Software most Frequently Asked Questions](#)
- [Frequently Asked Questions regarding Placements](#)
- [Top 20 Dynamic Programming Interview Questions](#)

Article Tags :

C
interview-preparation
placement preparation
Practice Tags :
C

thumb_up
13

To-do Done
2.6

Based on 32 vote(s)

[Basic](#) [Easy](#) [Medium](#) [Hard](#) [Expert](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Post navigation

Previous

[first_page](#) C | Advanced Pointer | Question 10

Next

[last_page](#) Difference between pointer and array in C?

Writing code in comment? Please use [ide.geeksforgeeks.org](#), generate link and share the link here.

Load Comments



Most popular in C
Print all possible combinations of the string by replacing '\$' with any other digit from the string
Predict Macro in C with Examples
How to call function with main function in C or C++
C program to print odd line contents of a file followed by even line content
Introduction to the C99 Programming Language : Part II

More related articles in C
C program to find square root of a given number
Format specifiers in different Programming Languages
Problem in comparing Floating point numbers and how to compare them correctly?
Features of C Programming Language
Pointer Expressions in C with Examples