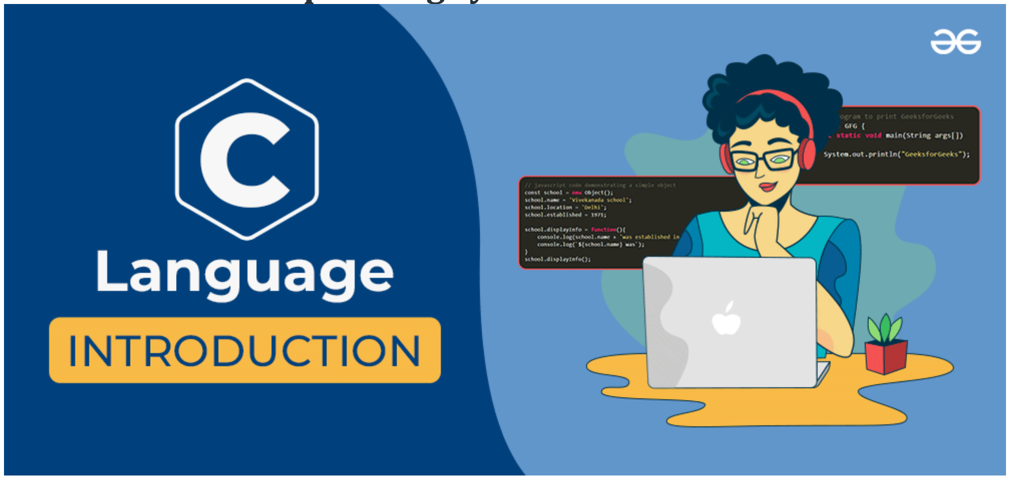
C Language Introduction

C is a procedural programming language initially developed by **Dennis Ritchie** in **1972** at Bell Laboratories of AT&T Labs. It was mainly developed as a system programming language to write the **UNIX operating system**.



The main features of the C language include:

- General Purpose and Portable
- Low-level Memory Access
- Fast Speed
- Clean Syntax

These features make the C language suitable for system programming like an operating system or compiler development.

Why Should We Learn C?

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

So, if a person learns C programming first, it will also help them learn any modern programming language. Also, learning C helps to understand a lot of the underlying architecture of the operating system like pointers, working with memory locations, etc.

Get Started with C *Learn C fundamentals and advanced concepts, then solve practical problems right in your browser window with Educative's interactive skill path [Become a C Programmer.](#) Sign up at Educative.io with the code* ***GEEKS10*** *to save 10% on your subscription.*

Difference Between C and C++

C++ was created to add the OOPs concept into the C language so they both have very similar syntax with a few differences. The following are some of the main differences between C and C++ Programming languages.

- C++ supports OOPs paradigm while C only has the procedural concept of programming.
- C++ has exception handling capabilities. In C, we have to resolve exceptions manually.

- There are no references in C.

There are many more differences between C and C++ which are discussed here: Difference between C and C++

Beginning with C Programming

Writing the First Program in C

The following code is one of the simplest C programs that will help us understand the basic syntax structure of a C program.

Example:

C

```c
#include <stdio.h>

int main() {
  int a = 10;
  printf("%d", a);

  return 0;
}
```

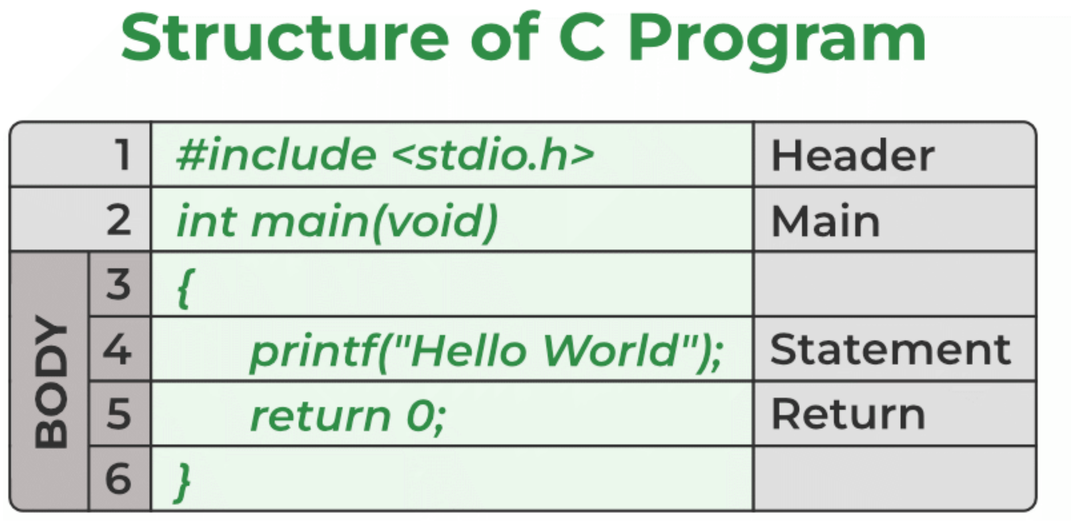
Output

```
10
```

Let us analyze the structure of our program line by line.

Structure of the C program

After the above discussion, we can formally assess the basic 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 lead to a Compilation Error. The structure of a C program is as follows:



Structure of C Program

	1	#include <stdio.h>	Header
	2	int main(void)	Main
BODY	3	{	
	4	printf("Hello World");	Statement
	5	return 0;	Return
	6	}	

Components of a C Program:

1. Header Files Inclusion – Line 1 [#include <stdio.h>]

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. All lines that start with # are processed by a preprocessor which is a program invoked by the compiler. In the above example, the preprocessor copies the preprocessed code of stdio.h to our file. The .h files are called header

files in C.
Some of the 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, and memory allocation
- string.h – Defines string handling functions
- math.h – Defines common mathematical functions.

2. Main Method Declaration – Line 2 [int main()]
The next part of a C program is to declare the main() function. It is the entry point of a C program and the execution typically begins with the first line of the main(). The empty brackets indicate that the main doesn't take any parameter (See this for more details). The int that was written before the main indicates the return type of main(). The value returned by the main indicates the status of program termination. See this post for more details on the return type.

3. Body of Main Method – Line 3 to Line 6 [enclosed in {}]
The body of a function in the C program refers to statements that are a part of that function. It can be anything like manipulations, searching, sorting, printing, etc. A pair of curly brackets define the body of a function. All functions must start and end with curly brackets.

4. Statement – Line 4 [printf("Hello World");]
Statements are the instructions given to the compiler. In C, a statement is always terminated by a **semicolon (;).** In this particular case, we use printf() function to instruct the compiler to display "Hello World" text on the screen.

5. Return Statement – Line 5 [return 0;]
The last part of any C function is the return statement. The return statement refers to the return values from a function. This return statement and return value depend upon the return type of the function. The return statement in our program returns the value from main(). The returned value may be used by an operating system to know the termination status of your program. The value 0 typically means successful termination.

How to Execute the Above Program?

In order to execute the above program, we need to first compile it using a compiler and then we can run the generated executable. There are online IDEs available for free like GeeksforGeeksIDE, that can be used to start development in C without installing a compiler.

1. ***Windows:*** There are many free IDEs available for developing programs in C like Code Blocks and Dev-CPP. IDEs provide us with an environment to develop code, compile it and finally execute it. We strongly recommend Code Blocks.
2. ***Linux:*** GCC compiler comes bundled with Linux which compiles C programs and generates executables for us to run. Code Blocks can also be used with Linux.
3. ***macOS:*** macOS already has a built-in text editor where you can just simply write the code and save it with a ".c" extension.

Application of C

- **Operating systems**: C is widely used for developing operating systems such as Unix, Linux, and Windows.
- **Embedded systems**: C is a popular language for developing embedded systems such as microcontrollers, microprocessors, and other electronic devices.
- **System software**: C is used for developing system software such as device drivers, compilers, and assemblers.
- **Networking**: C is widely used for developing networking applications such as web servers, network protocols, and network drivers.
- **Database systems**: C is used for developing database systems such as Oracle, MySQL, and PostgreSQL.
- **Gaming**: C is often used for developing computer games due to its ability to handle low-level hardware interactions.
- **Artificial Intelligence**: C is used for developing artificial intelligence and machine learning applications such as neural networks and deep learning algorithms.
- **Scientific applications**: C is used for developing scientific applications such as simulation software and numerical analysis tools.
- **Financial applications**: C is used for developing financial applications such as stock market analysis and trading systems.

Features of C Programming Language

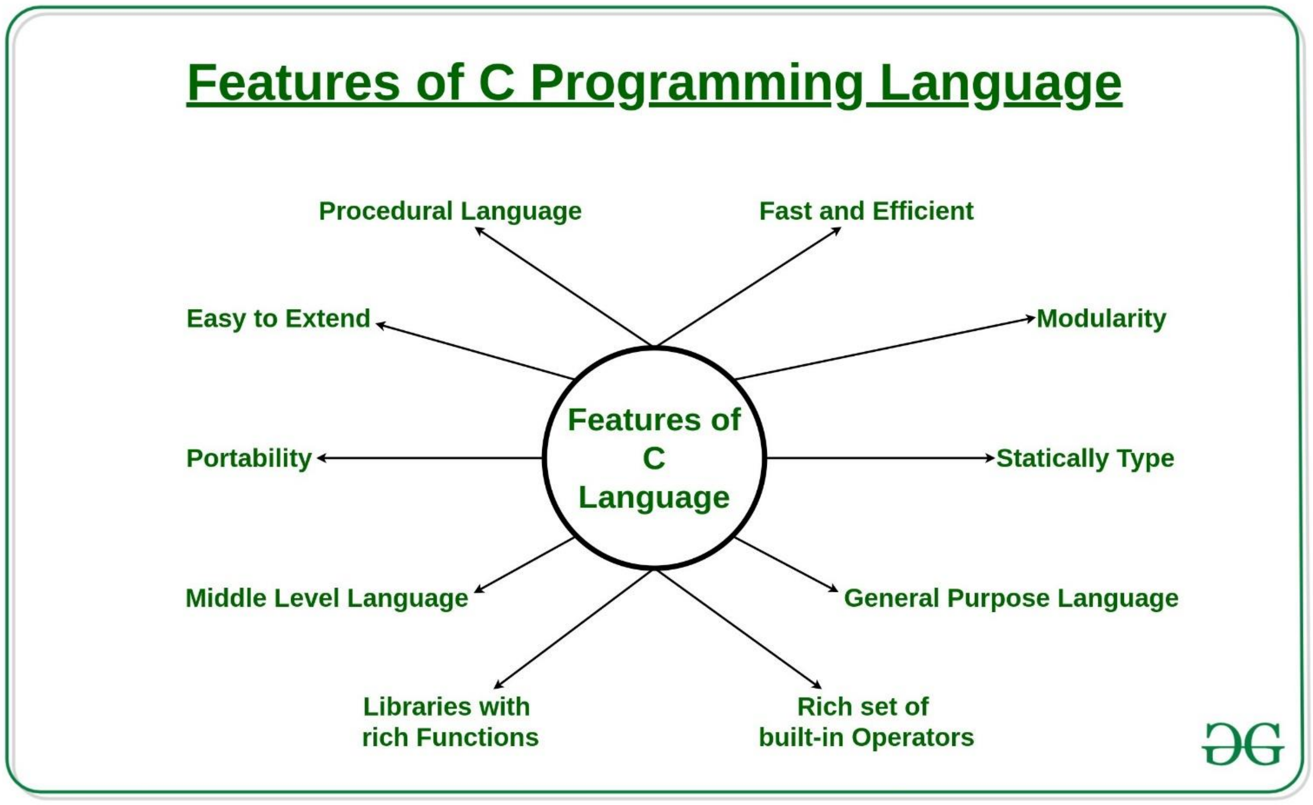
C is a procedural programming language. It was initially developed by Dennis Ritchie in the year 1972. It was mainly developed as a system programming language to write an operating system.

The main features of C language include low-level access to memory, a simple set of keywords, and a clean style, these features make C language suitable for system programming like an operating system or compiler development.

What are the Most Important Features of C Language?

Here are some of the most important features of the C language:

1. Procedural Language
2. Fast and Efficient
3. Modularity
4. Statically Type
5. General-Purpose Language
6. Rich set of built-in Operators
7. Libraries with Rich Functions
8. Middle-Level Language
9. Portability
10. Easy to Extend



Let discuss these features one by one:

1. Procedural Language

In a procedural language like C step by step, predefined instructions are carried out. C program may contain more than one function to perform a particular task. New people to programming will think that this is the only way a particular programming language works. There are other programming paradigms as well in the programming world. Most of the commonly used paradigm is an object-oriented programming language.

2. Fast and Efficient

Newer languages like Java, python offer more features than c programming language but due to additional processing in these languages, their performance rate gets down effectively. C programming language as the middle-level language provides programmers access to direct manipulation with the computer hardware but higher-level languages do not allow this. That’s one of the reasons C language is considered the first choice to start learning programming languages. It’s fast because statically typed languages are faster than dynamically typed languages.

3. Modularity

The concept of storing C programming language code in the form of libraries for further future uses is known as modularity. This programming language can do very little on its own most of its power is held by its libraries. C language has its own library to solve common problems.

4. Statically Type

C programming language is a statically typed language. Meaning the type of variable is checked at the time of compilation but not at run time. This means each time a programmer types a program they have to mention the type of variables used.

5. General-Purpose Language

From system programming to photo editing software, the C programming language is used in various applications. Some of the common applications where it's used are as follows:

- Operating systems: Windows, Linux, iOS, Android, OXS
- Databases: PostgreSQL, Oracle, MySQL, MS SQL Server, etc.

6. Rich set of built-in Operators

It is a diversified language with a rich set of built-in operators which are used in writing complex or simplified C programs.

7. Libraries with Rich Functions

Robust libraries and functions in C help even a beginner coder to code with ease.

8. Middle-Level Language

As it is a middle-level language so it has the combined form of both capabilities of assembly language and features of the high-level language.

9. Portability

C language is lavishly portable as programs that are written in C language can run and compile on any system with either no or small changes.

10. Easy to Extend

Programs written in C language can be extended means when a program is already written in it then some more features and operations can be added to it.

C Hello World Program

The “Hello World” program is the first step towards learning any programming language and also one of the simplest programs you will learn. To print the “Hello World”, we can use the printf function from the stdio.h library that prints the given string on the screen.

C Program to Print “Hello World”

The following C program displays “Hello World” in the output screen:

C

```c
// Simple C program to display "Hello World"

// Header file for input output functions
#include <stdio.h>

// Main function: entry point for execution
int main() {

    // writing print statement to print hello world
    printf("Hello World");

    return 0;
}
```

Output

```
Hello World
```

Explanation:

- **#include <stdio.h>** – This line includes the standard input-output library in the program.
- **int main()** – The main function where the execution of the program begins.
- **printf(“Hello, World!\n”);** – This function call prints “Hello, World!” followed by a new line.
- **return 0;** -This statement indicates that the program ended successfully.

C Comments

The **comments in C** are human-readable explanations or notes in the source code of a C program. A comment makes the program easier to read and understand. These are the statements that are not executed by the compiler or an interpreter.
It is considered to be a good practice to document our code using comments.

When and Why to use Comments in C programming?

1. A person reading a large code will be bemused if comments are not provided about details of the program.
2. C Comments are a way to make a code more readable by providing more descriptions.

3. C Comments can include a description of an algorithm to make code understandable.
4. C Comments can be used to prevent the execution of some parts of the code.

Types of comments in C

In C there are two types of comments in C language:

- **Single-line comment**
- **Multi-line comment**

Comments
//
Single line comment
/*
Multi-line comment
*/

Types of Comments in C

1. Single-line Comment in C

A single-line comment in C starts with (//) double forward slash. It extends till the end of the line and we don't need to specify its end.

Syntax of Single Line C Comment

```
// This is a single line comment
```

Example 1: C Program to illustrate single-line comment

- C

```c
// C program to illustrate
// use of single-line comment
#include <stdio.h>

int main(void)
{
    // This is a single-line comment
    printf("Welcome to GeeksforGeeks");
    return 0;
}
```

Output:

```
Welcome to GeeksforGeeks
```

Comment at End of Code Line

We can also create a comment that displays at the end of a line of code using a single-line comment. But generally, it's better to practice putting the comment before the line of code.

Example:

- C

```
// C program to demonstrate commenting after line of code
#include <stdio.h>

int main() {
    // single line comment here

      printf("Welcome to GeeksforGeeks"); // comment here
    return 0;
}
```

Output

```
Welcome to GeeksforGeeks
```

2. Multi-line Comment in C

The Multi-line comment in C starts with a forward slash and asterisk (**/***) and ends with an asterisk and forward slash (***/**). Any text between **/*** and ***/** is treated as a comment and is ignored by the compiler.
It can apply comments to multiple lines in the program.

Syntax of Multi-Line C Comment

```
/*Comment starts

    continues

    continues

    .

    .

    .

Comment ends*/
```

Example 2: C Program to illustrate the multi-line comment

- C

```
/* C program to illustrate
use of
multi-line comment */
#include <stdio.h>
```

```
int main(void)
{
    /*
    This is a
    multi-line comment
    */

      /*
    This comment contains some code which
    will not be executed.
    printf("Code enclosed in Comment");
    */
    printf("Welcome to GeeksforGeeks");
    return 0;
}
```

Output:

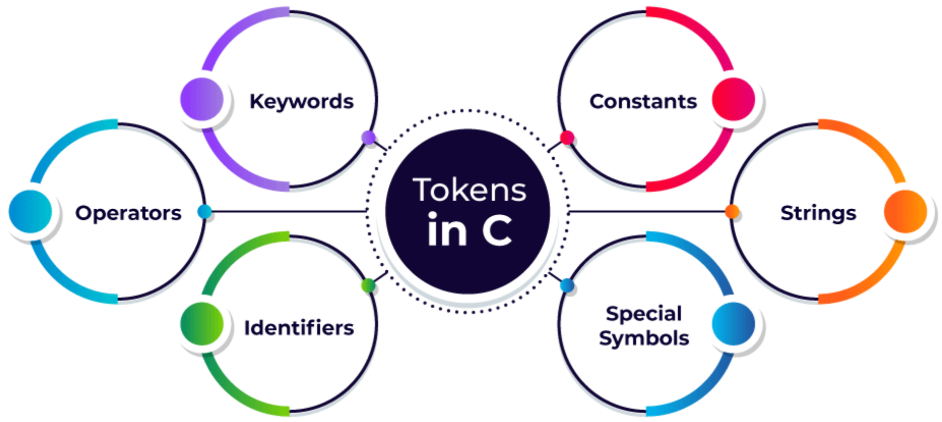
```
Welcome to GeeksforGeeks
```

Tokens in C

A token in C can be defined as the smallest individual element of the C programming language that is meaningful to the compiler. It is the basic component of a C program.

Types of Tokens in C

The tokens of C language can be classified into six types based on the functions they are used to perform. The types of C tokens are as follows:



1. **Keywords**
2. **Identifiers**
3. **Constants**
4. **Strings**
5. **Special Symbols**
6. **Operators**

1. C Token – Keywords

The 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 the text to be substituted for keywords before compilation by using 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
```

Note: *The number of keywords may change depending on the version of C you are using. For example, keywords present in ANSI C are 32 while in C11, it was increased to 44. Moreover, in the latest c23, it is increased to around 54.*

2. C Token – Identifiers

Identifiers are used as the general terminology for the naming of variables, functions, and arrays. These are user-defined names consisting of an 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 identifier called a statement label can be used in goto statements.

Rules for Naming Identifiers

Certain rules should be followed while naming c identifiers which are as follows:

- 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 the first 31 characters are significant.

Note: *Identifiers are case-sensitive so names like variable and Variable will be treated as different.*

For example,

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

3. C Token – Constants

The constants refer to the variables with fixed values. They are like normal variables but with the difference that their values can not be modified in the program once they are defined. Constants may belong to any of the data types.

Examples of Constants in C

```
const int c_var = 20;
const int* const ptr = &c_var;
```

4. C Token – 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++.

Examples of String

```
char string[20] = {‘g’, ’e’, ‘e’, ‘k’, ‘s’, ‘f’, ‘o’, ‘r’, ‘g’, ’e’, ‘e’, ‘k’,
‘s’, ‘\0’};
char string[20] = “geeksforgeeks”;
char string [] = “geeksforgeeks”;
```

5. C Token – Special Symbols

The following special symbols are used in C having some special meaning and thus, cannot be used for some other purpose. Some of these are listed below:

- **Brackets[]:** Opening and closing brackets are used as array element references. 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 mark the start and end of a block of code containing more than one executable statement.
- **Comma (,):** It is used to separate more than one statement like for separating parameters in function calls.
- **Colon(:):** It is an operator that essentially invokes something called an initialization list.
- **Semicolon(;):** It is known as a statement terminator. It indicates the end of one logical entity. That’s why each individual statement must be ended with a semicolon.
- **Asterisk (*):** It is used to create a pointer variable and for the multiplication of variables.
- **Assignment operator(=):** It is used to assign values and for logical operation validation.
- **Pre-processor (#):** The preprocessor is a macro processor that is used automatically by the compiler to transform your program before actual compilation.
- **Period (.):** Used to access members of a structure or union.
- **Tilde(~):** Bitwise One’s Complement Operator.

6. C Token – Operators

Operators are symbols that trigger an action when applied to C variables and other objects. The data items on which operators act 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 a 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 can further are classified into:
    1. Arithmetic operators
    2. Relational Operators

3. Logical Operators
4. Assignment Operators
5. Bitwise Operator

- **Ternary Operator**: The operator that requires three operands to act upon is called the ternary operator. Conditional Operator(?) is also called the ternary operator.

Keywords in C

In C Programming language, there are many rules so to avoid different types of errors. One of such rule is not able to declare variable names with auto, long, etc. This is all because these are keywords. Let us check all keywords in C language.

What are Keywords?

Keywords are predefined or reserved words that have special meanings to the compiler. These are part of the syntax and cannot be used as identifiers in the program. A list of keywords in C or reserved words in the C programming language are mentioned below:

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

auto

auto is the default storage class variable that is declared inside a function or a block. auto variables can only be accessed within the function/block they are declared. By default, auto variables have garbage values assigned to them. Automatic variables are also called local variables as they are local to a function.

```
auto int num;
```

Here num is the variable of the storage class auto and its type is int. Below is the C program to demonstrate the auto keyword:

C

```
// C program to demonstrate
// auto keyword
#include <stdio.h>

int printvalue()
{
  auto int a = 10;
  printf("%d", a);
}

// Driver code
```

```
int main()
{
  printvalue();
  return 0;
}
```

Output

```
10
```

break and continue

The break statement is used to terminate the innermost loop. It generally terminates a loop or a break statement. The continue statement skips to the next iteration of the loop. Below is the C program to demonstrate break and continue in C:

C

```
// C program to show use
// of break and continue
#include <stdio.h>

// Driver code
int main()
{
  for (int i = 1; i <= 10; i++)
  {
    if (i == 2)
    {
      continue;
    }
    if (i == 6)
    {
      break;
    }
    printf("%d ", i);
  }
  return 0;
}
```

Output

```
1 3 4 5
```

switch, case, and default

The switch statement in C is used as an alternate to the if-else ladder statement. For a single variable i.e, switch variable it allows us to execute multiple operations for different possible values of a single variable.

```
switch(Expression)
{
    case '1': // operation 1
            break;
    case:'2': // operation 2
            break;
    default: // default statement to be executed
}
```

Below is the C program to demonstrate the switch case statement:
C
```c
// C program to demonstrate
// switch case statement
#include <stdio.h>

// Driver code
int main() {
  int i = 4;
  switch (i) {
    case 1:
      printf("Case 1\n");break;
    case 2:
      printf("Case 2\n");break;
    case 3:
      printf("Case 3\n");break;
    case 4:
      printf("Case 4\n");break;
    default:
      printf("Default\n");break;
  }
}
```

Output
```
Case 4
```
Note: it is best to add a break statement after every case so that switch statement doesn't continue checking the remaining cases.

Output
```
Case 4

Default
```
char
char keyword in C is used to declare a character variable in the C programming language.
```
char x = 'D';
```
Below is the C program to demonstrate the char keyword:
C
```c
// C program to demonstrate
// char keyword
#include <stdio.h>

// Driver code
int main() {
  char c = 'a';
  printf("%c", c);
  return 0;
}
```

Output
```
a
```

const

The const keyword defines a variable who's value cannot be changed.

```
const int num = 10;
```

Below is the C program to demonstrate the const keyword:

C

```c
// C program to demonstrate
// const keyword
#include <stdio.h>

// Driver code
int main() {
  const int a = 11;
  a = a + 2;
  printf("%d", a);
  return 0;
}
```

This code will produce an error because the integer a was defined as a constant and it's value was later on changed.

Output:

```
error: assignment of read-only variable 'a'
       a = a + 2;
```

do

The do statement is used to declare a do-while loop. A do-while loop is a loop that executes once, and then checks it's condition to see if it should continue through the loop. After the first iteration, it will continue to execute the code while the condition is true.

Below is the C program to demonstrate a do-while loop.

C

```c
// C program to demonstrate
// do-while keyword
#include <stdio.h>

// Driver code
int main()
{
  int i = 1;
  do {
    printf("%d ", i);
    i++;
  } while(i <= 5);

  return 0;
}
```

Output

```
1 2 3 4 5
```

double and float

The doubles and floats are datatypes used to declare decimal type variables. They are similar, but doubles have 15 decimal digits, and floats only have 7.

Example:

```
float marks = 97.5;
double num;
```

Below is the C program to demonstrate double float keyword:

C

```c
// C program to demonstrate
// double float keyword
#include <stdio.h>

// Driver code
int main() {
  float f = 0.3;
  double d = 10.67;
  printf("Float value: %f\n", f);
  printf("Double value: %f\n", d);
  return 0;
}
```

Output

```
Float value: 0.300000

Double value: 10.670000
```

if-else

The if-else statement is used to make decisions, where if a condition is true, then it will execute a block of code; if it isn't true (else), then it will execute a different block of code.

```
if(marks == 97) {
    // if marks are 97 then will execute this block of code
}
else {
    // else it will execute this block of code
}
```

Below is the C program to demonstrate an if-else statement:

C

```c
// C program to demonstrate
// if-else keyword
#include <stdio.h>

// Driver code
int main()
{
  int a = 10;
  if(a < 11)
  {
    printf("A is less than 11");
  }
  else
  {
    printf("A is equal to or "
           "greater than 11");
  }
  return 0;
}
```

Output

```
A is less than 11
```

enum

The enum keyword is used to declare an enum (short for enumeration). An enum is a user-defined datatype, which holds a list of user-defined integer constants. By default, the value of each constant is it's index (starting at zero), though this can be changed. You can declare an object of an enum and can set it's value to one of the constants you declared before. Here is an example of how an enum might be used:

C

```c
// An example program to
// demonstrate working of
// enum in C
#include<stdio.h>

// enum declaration:
enum week{Mon, Tue, Wed, Thur, Fri, Sat, Sun};

// Driver code
int main()
{
//object of the enum (week), called day
  enum week day;
  day = Wed;
  printf("%d", day);
  return 0;
}
```

Output

```
2
```

extern

The extern keyword is used to declare a variable or a function that has an external linkage outside of the file declaration.

C

```c
#include <stdio.h>

extern int a;

int main(){

    printf("%d", a);

      return 0;
}
```

for

The "for" keyword is used to declare a for-loop. A for-loop is a loop that is specified to run a certain amount of times.

Below is the C program to demonstrate a for-loop:

C

```
// C program to demonstrate
// for keyword
#include <stdio.h>

// Driver code
int main()
{
  for (int i = 0; i < 5; i++)
  {
    printf("%d ", i);
  }
  return 0;
}
```

Output

```
0 1 2 3 4
```

goto

The goto statement is used to transfer the control of the program to the given label. It is used to jump from anywhere to anywhere within a function.

Example:

```
goto label;
// code
label:
```

Below is the C program to demonstrate the goto keyword:

C

```
// C program demonstrate
// goto keyword
#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 code
int main(){
    printNumbers();
    return 0;
}
```

Output

```
1 2 3 4 5 6 7 8 9 10
```

int

int keyword is used in a type declaration to give a variable an integer type. In C, the integer variable must have a range of at least -32768 to +32767.

Example:

```
int x = 10;
```

Below is the C program to show the int keyword:

C

```c
// C program to demonstrate
// int keyword
#include <stdio.h>

void sum() {
    int a = 10, b = 20;
    int sum;
    sum = a + b;
    printf("%d", sum);
}

// Driver code
int main() {
    sum();
    return 0;
}
```

Output

```
30
```

short, long, signed, and unsigned

Different data types also have different ranges up to which they can store numbers. These ranges may vary from compiler to compiler. Below is a list of ranges along with the memory requirement and format specifiers on the **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
long int	4	-2,147,483,648 to 2,147,483,647	%ld
unsigned long int	4	0 to 4,294,967,295	%lu
long long int	8	-(2^63) to (2^63)-1	%lld

Data Type	Memory (bytes)	Range	Format Specifier
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
long double	16	3.4E-4932 to 1.1E+4932	%Lf

Below is the C program to demonstrate the short, long, signed, and unsigned keywords:

C

```c
// C program to demonstrate
// short, long, signed,
// and unsigned keyword
#include <stdio.h>

// Driver code
int main() {
  // short integer
  short int a = 12345;

  // signed integer
  signed int b = -34;

  // unsigned integer
  unsigned int c = 12;

  // L or l is used for
  // long int in C.
  long int d = 99998L;

  printf("Integer value with a short int data: %hd", a);
  printf("\nInteger value with a signed int data: %d", b);
  printf("\nInteger value with an unsigned int data: %u", c);
  printf("\nInteger value with a long int data: %ld", d);
  return 0;
}
```

Output

```
Integer value with a short int data: 12345

Integer value with a signed int data: -34

Integer value with an unsigned int data: 12

Integer value with a long int data: 99998
```

return

The return statement returns a value to where the function was called.

Example:

```
return x;
```

Below is the C program to demonstrate the return keyword:

C

```c
// C program to demonstrate
// return keyword
#include <stdio.h>
int sum(int x, int y) {
  int sum;
  sum = x + y;
  return sum;
}

// Driver code
int main() {
  int num1 = 10;
  int num2 = 20;
  printf("Sum: %d",
          sum(num1, num2));
  return 0;
}
```

Output

```
Sum: 30
```

sizeof

sizeof is a keyword that gets the size of an expression, (variables, arrays, pointers, etc.) in bytes.

Example:

```
sizeof(char);
sizeof(int);
sizeof(float); in bytes.
```

Below is the C program to demonstrate sizeof keyword:

C

```c
// C program to demonsstrate
// sizeof keyword
#include <stdio.h>

// Driver code
int main() {
  int x = 10;
  printf("%d", sizeof(x));
  return 0;
}
```

Output

```
4
```

register

Register variables tell the compiler to store variables in the CPU register instead of memory. Frequently used variables are kept in the CPU registers for faster access.

Example:

```
register char c = 's';
```

static

The static keyword is used to create static variables. A static variable is not limited by a scope and can be used throughout the program. It's value is preserved even after it's scope.

For Example:

```
static int num;
```

struct

The struct keyword in C programming language is used to declare a structure. A structure is a list of variables, (they can be of different data types), which are grouped together under one data type.

For Example:

```
struct Geek {
    char name[50];
    int num;
    double var;
};
```

Below is the C program for the struct keyword:

C

```
// C program to demonstrate
// struct keyword
#include <stdio.h>
#include <string.h>

struct Books {
  char  title[50];
  char  author[50];
};

// Driver code
int main( ) {
  // Declare Book1 of type Book
  struct Books book1;

 // book 1 specification
 strcpy(book1.title, "C++ Programming");
 strcpy(book1.author, "Bjarne Stroustrup");

 // Print book details
 printf("Book 1 title : %s\n", book1.title);
 printf("Book 1 author : %s\n", book1.author);
 return 0;
}
```

Output

```
Book 1 title : C++ Programming
Book 1 author : Bjarne Stroustrup
```

typedef

The typedef keyword in C programming language is used to define a data type with a new name in the program. typedef keyword is used to make our code more readable.

For Example:

```
typedef long num
```

In this example we have changed the datatype name of “long” to “num”.

union

The union is a user-defined data type. All data members which are declared under the union keyword share the same memory location.

Example:

```
union GeekforGeeks {
    int x;
    char s;
} obj;
```

Below is the C program for the union keyword:

C

```c
#include <stdio.h>
union student {
  int age;
  char marks;
} s;

// Driver code
int main() {
  s.age = 15;
  s.marks = 56;
  printf("age = %d", s.age);
  printf("\nmarks = %d", s.marks);
}
```

Output

```
age = 56

marks = 56
```

void

The void keyword means nothing i.e, NULL value. When the function return type is used as the void, the keyword void specifies that it has no return value.

Example:

```
void fun() {
    // program
}
```

volatile

The volatile keyword is used to create volatile objects. Objects which are declared volatile are omitted from optimization as their values can be changed by code outside the scope of the current code at any point in time.

For Example:

```
const volatile marks = 98;
```
marks are declared constant so they can’t be changed by the program. But hardware can change it as they are volatile objects.

Conclusion

The points we learned about the keywords are mentioned below:

- Keywords are Reserved words in C with certain meanings.
- We can’t use keywords as any element’s name.
- There are 32 keywords in C all having unique meanings.