**C Programming Interview Questions and Answers for Freshers –**

**1) How do you construct an increment statement or decrement statement in C?**

There are actually two ways you can do this. One is to use the increment operator ++ and decrement operator –. For example, the statement “x++” means to increment the value of x by 1. Likewise, the statement “x –” means to decrement the value of x by 1. Another way of writing increment statements is to use the conventional + plus sign or – minus sign. In the case of “x++”, another way to write it is “x = x +1

**2) What is the difference between Call by Value and Call by Reference?**

When using Call by Value, you are sending the value of a variable as parameter to a function, whereas Call by Reference sends the address of the variable. Also, under Call by Value, the value in the parameter is not affected by whatever operation that takes place, while in the case of Call by Reference, values can be affected by the process within the function.

**3) Some coders debug their programs by placing comment symbols on some codes instead of deleting it. How does this aid in debugging?**

Placing comment symbols /\* \*/ around a code, also referred to as “commenting out”, is a way of isolating some codes that you think maybe causing errors in the program, without deleting the code. The idea is that if the code is in fact correct, you simply remove the comment symbols and continue on. It also saves you time and effort on having to retype the codes if you have deleted it in the first place.

**4) What is the equivalent code of the following statement in WHILE LOOP format?**

for (a=1; a<=100; a++)

printf ("%d\n", a \* a);

**Answer:**

a=1;

while (a<=100) {

printf ("%d\n", a \* a);

a++;

}

**5) What is a stack?**

A stack is one form of a data structure. Data is stored in stacks using the FILO (First In Last Out) approach. At any particular instance, only the top of the stack is accessible, which means that in order to retrieve data that is stored inside the stack, those on the upper part should be extracted first. Storing data in a stack is also referred to as a PUSH, while data retrieval is referred to as a POP.

**6) What is a sequential access file?**

When writing programs that will store and retrieve data in a file, it is possible to designate that file into different forms. A sequential access file is such that data are saved in sequential order: one data is placed into the file after another. To access a particular data within the sequential access file, data has to be read one data at a time, until the right one is reached.

**7) What is variable initialization and why is it important?**

This refers to the process wherein a variable is assigned an initial value before it is used in the program. Without initialization, a variable would have an unknown value, which can lead to unpredictable outputs when used in computations or other operations.

**8 What is spaghetti programming?**

Spaghetti programming refers to codes that tend to get tangled and overlapped throughout the program. This unstructured approach to coding is usually attributed to lack of experience on the part of the programmer. Spaghetti programing makes a program complex and analyzing the codes difficult, and so must be avoided as much as possible.

**9) Differentiate Source Codes from Object Codes**

Source codes are codes that were written by the programmer. It is made up of the commands and other English-like keywords that are supposed to instruct the computer what to do. However, computers would not be able to understand source codes. Therefore, source codes are compiled using a compiler. The resulting outputs are object codes, which are in a format that can be understood by the computer processor. In [C programming](https://www.guru99.com/c-programming-language.html), source codes are saved with the file extension .C, while object codes are saved with the file extension .OBJ

**10) In C programming, how do you insert quote characters (‘ and “) into the output screen?**

This is a common problem for beginners because quotes are normally part of a printf statement. To insert the quote character as part of the output, use the format specifiers \’ (for single quote), and \” (for double quote).

**11) What is the use of a ‘\0’ character?**

It is referred to as a terminating null character, and is used primarily to show the end of a string value.

**12) What is the difference between the = symbol and == symbol?**

The = symbol is often used in mathematical operations. It is used to assign a value to a given variable. On the other hand, the == symbol, also known as “equal to” or “equivalent to”, is a relational operator that is used to compare two values.

**13) What is the modulus operator?**

The modulus operator outputs the remainder of a division. It makes use of the percentage (%) symbol. For example: 10 % 3 = 1, meaning when you divide 10 by 3, the remainder is 1.

**14) What is a nested loop?**

A nested loop is a loop that runs within another loop. Put it in another sense, you have an inner loop that is inside an outer loop. In this scenario, the inner loop is performed a number of times as specified by the outer loop. For each turn on the outer loop, the inner loop is first performed.

**15) Which of the following operators is incorrect and why? ( >=, <=, <>, ==)**

<> is incorrect. While this operator is correctly interpreted as “not equal to” in writing conditional statements, it is not the proper operator to be used in [C programming](https://www.guru99.com/c-programming-tutorial.html). Instead, the operator != must be used to indicate “not equal to” condition.

**16) Compare and contrast compilers from interpreters.**

Compilers and interpreters often deal with how program codes are executed. Interpreters execute program codes one line at a time, while compilers take the program as a whole and convert it into object code, before executing it. The key difference here is that in the case of interpreters, a program may encounter syntax errors in the middle of execution, and will stop from there. On the other hand, compilers check the syntax of the entire program and will only proceed to execution when no syntax errors are found.

**17) How do you declare a variable that will hold string values?**

The char keyword can only hold 1 character value at a time. By creating an array of characters, you can store string values in it. Example: “char MyName[50]; ” declares a string variable named MyName that can hold a maximum of 50 characters.

**18) Can the curly brackets { } be used to enclose a single line of code?**

While curly brackets are mainly used to group several lines of codes, it will still work without error if you used it for a single line. Some programmers prefer this method as a way of organizing codes to make it look clearer, especially in conditional statements.

**19) What are header files and what are its uses in C programming?**

Header files are also known as library files. They contain two essential things: the definitions and prototypes of functions being used in a program. Simply put, commands that you use in C programming are actually functions that are defined from within each header files. Each header file contains a set of functions. For example: stdio.h is a header file that contains definition and prototypes of commands like printf and scanf.

**20) What is syntax error?**

Syntax errors are associated with mistakes in the use of a programming language. It maybe a command that was misspelled or a command that must was entered in lowercase mode but was instead entered with an upper case character. A misplaced symbol, or lack of symbol, somewhere within a line of code can also lead to syntax error.

**21) What are variables and it what way is it different from constants?**

Variables and constants may at first look similar in a sense that both are identifiers made up of one character or more characters (letters, numbers and a few allowable symbols). Both will also hold a particular value. Values held by a variable can be altered throughout the program, and can be used in most operations and computations. Constants are given values at one time only, placed at the beginning of a program. This value is not altered in the program. For example, you can assigned a constant named PI and give it a value 3.1415 . You can then use it as PI in the program, instead of having to write 3.1415 each time you need it.

**22) How do you access the values within an array?**

Arrays contain a number of elements, depending on the size you gave it during variable declaration. Each element is assigned a number from 0 to number of elements-1. To assign or retrieve the value of a particular element, refer to the element number. For example: if you have a declaration that says “intscores[5];”, then you have 5 accessible elements, namely: scores[0], scores[1], scores[2], scores[3] and scores[4].

**23) Can I use “int” data type to store the value 32768? Why?**

No. “int” data type is capable of storing values from -32768 to 32767. To store 32768, you can use “long int” instead. You can also use “unsigned int”, assuming you don’t intend to store negative values.

**24) Can two or more operators such as \n and \t be combined in a single line of program code?**

Yes, it’s perfectly valid to combine operators, especially if the need arises.  
For example: you can have a code like printf (“Hello\n\n\’World\'”) to output the text “Hello” on the first line and “World” enclosed in single quotes to appear on the next two lines.

**25) Why is it that not all header files are declared in every C program?**

The choice of declaring a header file at the top of each C program would depend on what commands/functions you will be using in that program. Since each header file contains different function definitions and prototype, you would be using only those header files that would contain the functions you will need. Declaring all header files in every program would only increase the overall file size and load of the program, and is not considered a good programming style.

**26) When is the “void” keyword used in a function?**

When declaring functions, you will decide whether that function would be returning a value or not. If that function will not return a value, such as when the purpose of a function is to display some outputs on the screen, then “void” is to be placed at the leftmost part of the function header. When a return value is expected after the function execution, the data type of the return value is placed instead of “void”.

**27) What are compound statements?**

Compound statements are made up of two or more program statements that are executed together. This usually occurs while handling conditions wherein a series of statements are executed when a TRUE or FALSE is evaluated. Compound statements can also be executed within a loop. Curly brackets { } are placed before and after compound statements.

**28) What is the significance of an algorithm to C programming?**

Before a program can be written, an algorithm has to be created first. An algorithm provides a step by step procedure on how a solution can be derived. It also acts as a blueprint on how a program will start and end, including what process and computations are involved.

**29) What is the advantage of an array over individual variables?**

When storing multiple related data, it is a good idea to use arrays. This is because arrays are named using only 1 word followed by an element number. For example: to store the 10 test results of 1 student, one can use 10 different variable names (grade1, grade2, grade3… grade10). With arrays, only 1 name is used, the rest are accessible through the index name (grade[0], grade[1], grade[2]… grade[9]).

**30) Write a loop statement that will show the following output:**

1

12

123

1234

12345

**Answer:**

for (a=1; a<=5; i++) {

for (b=1; b<=a; b++)

printf("%d",b);

printf("\n");

}

**31) What is wrong in this statement? scanf(“%d”,whatnumber);**

An ampersand & symbol must be placed before the variable name whatnumber. Placing & means whatever integer value is entered by the user is stored at the “address” of the variable name. This is a common mistake for programmers, often leading to logical errors.

**32) How do you generate random numbers in C?**

Random numbers are generated in C using the rand() command. For example: anyNum = rand() will generate any integer number beginning from 0, assuming that anyNum is a variable of type integer.

**33) What could possibly be the problem if a valid function name such as tolower() is being reported by the C compiler as undefined?**

The most probable reason behind this error is that the header file for that function was not indicated at the top of the program. Header files contain the definition and prototype for functions and commands used in a C program. In the case of “tolower()”, the code “#include <ctype.h>” must be present at the beginning of the program.

**34) What are comments and how do you insert it in a C program?**

[Comments](https://www.guru99.com/c-comments.html) are a great way to put some remarks or description in a program. It can serves as a reminder on what the program is all about, or a description on why a certain code or function was placed there in the first place. Comments begin with /\* and ended by \*/ characters. Comments can be a single line, or can even span several lines. It can be placed anywhere in the program.

**35) What is debugging?**

Debugging is the process of identifying errors within a program. During program compilation, errors that are found will stop the program from executing completely. At this state, the programmer would look into the possible portions where the error occurred. Debugging ensures the removal of errors, and plays an important role in ensuring that the expected program output is met.

**36) What does the && operator do in a program code?**

The && is also referred to as AND operator. When using this operator, all conditions specified must be TRUE before the next action can be performed. If you have 10 conditions and all but 1 fails to evaluate as TRUE, the entire condition statement is already evaluated as FALSE

**37) In C programming, what command or code can be used to determine if a number of odd or even?**

There is no single command or [function in C](https://www.guru99.com/c-functions.html) that can check if a number is odd or even. However, this can be accomplished by dividing that number by 2, then checking the remainder. If the remainder is 0, then that number is even, otherwise, it is odd. You can write it in code as:

if (num % 2 == 0)

printf("EVEN");

else

printf("ODD");

**38) What does the format %10.2 mean when included in a printf statement?**

This format is used for two things: to set the number of spaces allotted for the output number and to set the number of decimal places. The number before the decimal point is for the allotted space, in this case it would allot 10 spaces for the output number. If the number of space occupied by the output number is less than 10, addition space characters will be inserted before the actual output number. The number after the decimal point sets the number of decimal places, in this case, it’s 2 decimal spaces.

**39) What are logical errors and how does it differ from syntax errors?**

Program that contains logical errors tend to pass the compilation process, but the resulting output may not be the expected one. This happens when a wrong formula was inserted into the code, or a wrong sequence of commands was performed. Syntax errors, on the other hand, deal with incorrect commands that are misspelled or not recognized by the compiler.

**40) What are the different types of control structures in programming?**

There are 3 main control structures in programming: Sequence, Selection and Repetition. Sequential control follows a top to bottom flow in executing a program, such that step 1 is first perform, followed by step 2, all the way until the last step is performed. Selection deals with conditional statements, which mean codes are executed depending on the evaluation of conditions as being TRUE or FALSE. This also means that not all codes may be executed, and there are alternative flows within. Repetitions are also known as loop structures, and will repeat one or two program statements set by a counter.

**41) What is || operator and how does it function in a program?**

The || is also known as the OR operator in C programming. When using || to evaluate logical conditions, any condition that evaluates to TRUE will render the entire condition statement as TRUE.

**42) Can the “if” function be used in comparing strings?**

No. “if” command can only be used to compare numerical values and single character values. For comparing string values, there is another function called strcmp that deals specifically with strings.

**43) What are preprocessor directives?**

Preprocessor directives are placed at the beginning of every C program. This is where library files are specified, which would depend on what functions are to be used in the program. Another use of preprocessor directives is the declaration of constants.Preprocessor directives begin with the # symbol.

**44) What will be the outcome of the following conditional statement if the value of variable s is 10?**

s >=10 && s < 25 && s!=12

The outcome will be TRUE. Since the value of s is 10, s >= 10 evaluates to TRUE because s is not greater than 10 but is still equal to 10. s< 25 is also TRUE since 10 is less then 25. Just the same, s!=12, which means s is not equal to 12, evaluates to TRUE. The && is the AND operator, and follows the rule that if all individual conditions are TRUE, the entire statement is TRUE.

**45) Describe the order of precedence with regards to operators in C.**

Order of precedence determines which operation must first take place in an operation statement or conditional statement. On the top most level of precedence are the unary operators !, +, – and &. It is followed by the regular mathematical operators (\*, / and modulus % first, followed by + and -). Next in line are the relational operators <, <=, >= and >. This is then followed by the two equality operators == and !=. The logical operators && and || are next evaluated. On the last level is the assignment operator =.

**46) What is wrong with this statement? myName = “Robin”;**

You cannot use the = sign to assign values to a string variable. Instead, use the strcpy function. The correct statement would be: strcpy(myName, “Robin”);

**47) How do you determine the length of a string value that was stored in a variable?**

To get the length of a string value, use the function strlen(). For example, if you have a variable named FullName, you can get the length of the stored string value by using this statement: I = strlen(FullName); the variable I will now have the character length of the string value.

**48) Is it possible to initialize a variable at the time it was declared?**

Yes, you don’t have to write a separate assignment statement after the variable declaration, unless you plan to change it later on. For example: char planet[15] = “Earth”; does two things: it declares a string variable named planet, then initializes it with the value “Earth”.

**49) Why is C language being considered a middle level language?**

This is because C language is rich in features that make it behave like a high level language while at the same time can interact with hardware using low level methods. The use of a well structured approach to programming, coupled with English-like words used in functions, makes it act as a high level language. On the other hand, C can directly access memory structures similar to assembly language routines.

**50) What are the different file extensions involved when programming in C?**

Source codes in C are saved with .C file extension. Header files or library files have the .H file extension. Every time a program source code is successfully compiled, it creates an .OBJ object file, and an executable .EXE file.

**51) What are reserved words?**

Reserved words are words that are part of the standard C language library. This means that reserved words have special meaning and therefore cannot be used for purposes other than what it is originally intended for. Examples of reserved words are int, void, and return.

**52) What are linked list?**

A linked list is composed of nodes that are connected with another. In C programming, linked lists are created using pointers. Using linked lists is one efficient way of utilizing memory for storage.

**53) What is FIFO?**

In C programming, there is a data structure known as queue. In this structure, data is stored and accessed using FIFO format, or First-In-First-Out. A queue represents a line wherein the first data that was stored will be the first one that is accessible as well.

**54) What are binary trees?**

Binary trees are actually an extension of the concept of linked lists. A binary tree has two pointers, a left one and a right one. Each side can further branch to form additional nodes, which each node having two pointers as well. Learn more about [Binary Tree in Data Structure](https://www.guru99.com/binary-tree.html) if you are interested.

**55) Not all reserved words are written in lowercase. TRUE or FALSE?**

FALSE. All reserved words must be written in lowercase; otherwise the C compiler would interpret this as unidentified and invalid.

**56) What is the difference between the expression “++a” and “a++”?**

In the first expression, the increment would happen first on variable a, and the resulting value will be the one to be used. This is also known as a prefix increment. In the second expression, the current value of variable a would the one to be used in an operation, before the value of a itself is incremented. This is also known as postfix increment.

**57) What would happen to X in this expression: X += 15; (assuming the value of X is 5)**

X +=15 is a short method of writing X = X + 15, so if the initial value of X is 5, then 5 + 15 = 20.

**58) In C language, the variables NAME, name, and Name are all the same. TRUE or FALSE?**

FALSE. C language is a case sensitive language. Therefore, NAME, name and Name are three uniquely different variables.

**59) What is an endless loop?**

An endless loop can mean two things. One is that it was designed to loop continuously until the condition within the loop is met, after which a break function would cause the program to step out of the loop. Another idea of an endless loop is when an incorrect loop condition was written, causing the loop to run erroneously forever. Endless loops are oftentimes referred to as infinite loops.

**60) What is a program flowchart and how does it help in writing a program?**

A flowchart provides a visual representation of the step by step procedure towards solving a given problem. Flowcharts are made of symbols, with each symbol in the form of different shapes. Each shape may represent a particular entity within the entire program structure, such as a process, a condition, or even an input/output phase.

**61) What is wrong with this program statement? void = 10;**

The word void is a reserved word in C language. You cannot use reserved words as a user-defined variable.

**62) Is this program statement valid? INT = 10.50;**

Assuming that INT is a variable of type float, this statement is valid. One may think that INT is a reserved word and must not be used for other purposes. However, recall that reserved words are express in lowercase, so the C compiler will not interpret this as a reserved word.

**63) What are actual arguments?**

When you create and use functions that need to perform an action on some given values, you need to pass these given values to that function. The values that are being passed into the called function are referred to as actual arguments.

**64) What is a newline escape sequence?**

A newline escape sequence is represented by the \n character. This is used to insert a new line when displaying data in the output screen. More spaces can be added by inserting more \n characters. For example, \n\n would insert two spaces. A newline escape sequence can be placed before the actual output expression or after.

**65) What is output redirection?**

It is the process of transferring data to an alternative output source other than the display screen. Output redirection allows a program to have its output saved to a file. For example, if you have a program named COMPUTE, typing this on the command line as COMPUTE >DATA can accept input from the user, perform certain computations, then have the output redirected to a file named DATA, instead of showing it on the screen.

**66) What are run-time errors?**

These are errors that occur while the program is being executed. One common instance wherein run-time errors can happen is when you are trying to divide a number by zero. When run-time errors occur, program execution will pause, showing which program line caused the error.

**67) What is the difference between functions abs() and fabs()?**

These 2 functions basically perform the same action, which is to get the absolute value of the given value. Abs() is used for integer values, while fabs() is used for floating type numbers. Also, the prototype for abs() is under <stdlib.h>, while fabs() is under <math.h>.

**68) What are formal parameters?**

In using functions in a C program, formal parameters contain the values that were passed by the calling function. The values are substituted in these formal parameters and used in whatever operations as indicated within the main body of the called function.

**69) What are control structures?**

Control structures take charge at which instructions are to be performed in a program. This means that program flow may not necessarily move from one statement to the next one, but rather some alternative portions may need to be pass into or bypassed from, depending on the outcome of the conditional statements.

**70) Write a simple code fragment that will check if a number is positive or negative**

If (num>=0)

printf("number is positive");

else

printf ("number is negative");

**71) When is a “switch” statement preferable over an “if” statement?**

The [switch statement](https://www.guru99.com/c-switch-case-statement.html) is best used when dealing with selections based on a single variable or expression. However, switch statements can only evaluate integer and character data types.

**72) What are global variables and how do you declare them?**

Global variables are variables that can be accessed and manipulated anywhere in the program. To make a variable global, place the variable declaration on the upper portion of the program, just after the preprocessor directives section.

**73) What are enumerated types?**

Enumerated types allow the programmer to use more meaningful words as values to a variable. Each item in the enumerated type variable is actually associated with a numeric code. For example, one can create an enumerated type variable named DAYS whose values are Monday, Tuesday… Sunday.

**74) What does the function toupper() do?**

It is used to convert any letter to its upper case mode. Toupper() function prototype is declared in <ctype.h>. Note that this function will only convert a single character, and not an entire string.

**75) Is it possible to have a function as a parameter in another function?**

Yes, that is allowed in C programming. You just need to include the entire function prototype into the parameter field of the other function where it is to be used.

**76) What are multidimensional arrays?**

Multidimensional arrays are capable of storing data in a two or more dimensional structure. For example, you can use a 2 dimensional array to store the current position of pieces in a chess game, or position of players in a tic-tac-toe program.

**77) Which function in C can be used to append a string to another string?**

The strcat function. It takes two parameters, the source string and the string value to be appended to the source string.

**78) What is the difference between functions getch() and getche()?**

Both functions will accept a character input value from the user. When using getch(), the key that was pressed will not appear on the screen, and is automatically captured and assigned to a variable. When using getche(), the key that was pressed by the user will appear on the screen, while at the same time being assigned to a variable.

**79) Dothese two program statements perform the same output? 1) scanf(“%c”, &letter); 2) letter=getchar()**

Yes, they both do the exact same thing, which is to accept the next key pressed by the user and assign it to variable named letter.

**80) What are structure types in C?**

Structure types are primarily used to store records. A record is made up of related fields. This makes it easier to organize a group of related data.

**81) What does the characters “r” and “w” mean when writing programs that will make use of files?**

“r” means “read” and will open a file as input wherein data is to be retrieved. “w” means “write”, and will open a file for output. Previous data that was stored on that file will be erased.

**82) What is the difference between text files and binary files?**

Text files contain data that can easily be understood by humans. It includes letters, numbers and other characters. On the other hand, binary files contain 1s and 0s that only computers can interpret.

**83) is it possible to create your own header files?**

Yes, it is possible to create a customized header file. Just include in it the function prototypes that you want to use in your program, and use the #include directive followed by the name of your header file.

**84) What is dynamic data structure?**

Dynamic data structure provides a means for storing data more efficiently into memory. Using Using [dynamic memory allocation](https://www.guru99.com/c-dynamic-memory-allocation.html), your program will access memory spaces as needed. This is in contrast to static data structure, wherein the programmer has to indicate a fix number of memory space to be used in the program.

**85) What are the different data types in C?**

The basic [data types in C](https://www.guru99.com/c-variable-datatypes-constants.html) are int, char, and float. Int is used to declare variables that will be storing integer values. Float is used to store real numbers. Char can store individual character values.

**86) What is the general form of a C program?**

A C program begins with the preprocessor directives, in which the programmer would specify which header file and what constants (if any) to be used. This is followed by the main function heading. Within the main function lies the variable declaration and program statement.

**87) What is the advantage of a random access file?**

If the amount of data stored in a file is fairly large, the use of random access will allow you to search through it quicker. If it had been a sequential access file, you would have to go through one record at a time until you reach the target data. A random access file lets you jump directly to the target address where data is located.

**88) In a switch statement, what will happen if a break statement is omitted?**

If a break statement was not placed at the end of a particular case portion? It will move on to the next case portion, possibly causing incorrect output.

**89) Describe how arrays can be passed to a user defined function**

One thing to note is that you cannot pass the entire array to a function. Instead, you pass to it a pointer that will point to the array first element in memory. To do this, you indicate the name of the array without the brackets.

**90) What are pointers?**

[Pointers point](https://www.guru99.com/c-pointers.html) to specific areas in the memory. Pointers contain the address of a variable, which in turn may contain a value or even an address to another memory.

**91) Can you pass an entire structure to functions?**

Yes, it is possible to pass an entire structure to a function in a call by method style. However, some programmers prefer declaring the structure globally, then pass a variable of that structure type to a function. This method helps maintain consistency and uniformity in terms of argument type.

**92) What is gets() function?**

The gets() function allows a full line data entry from the user. When the user presses the enter key to end the input, the entire line of characters is stored to a string variable. Note that the enter key is not included in the variable, but instead a null terminator \0 is placed after the last character.

**93) The % symbol has a special use in a printf statement. How would you place this character as part of the output on the screen?**

You can do this by using %% in the printf statement. For example, you can write printf(“10%%”) to have the output appear as 10% on the screen.

**94) How do you search data in a data file using random access method?**

Use the fseek() function to perform random access input/ouput on a file. After the file was opened by the fopen() function, the fseek would require three parameters to work: a file pointer to the file, the number of bytes to search, and the point of origin in the file.

**95) Are comments included during the compilation stage and placed in the EXE file as well?**

No, comments that were encountered by the compiler are disregarded. Comments are mostly for the guidance of the programmer only and do not have any other significant use in the program functionality.

**96) Is there a built-in function in C that can be used for sorting data?**

Yes, use the qsort() function. It is also possible to create user defined functions for sorting, such as those based on the balloon sort and bubble sort algorithm.

**97) What are the advantages and disadvantages of a heap?**

Storing data on the heap is slower than it would take when using the stack. However, the main advantage of using the heap is its flexibility. That’s because memory in this structure can be allocated and remove in any particular order. Slowness in the heap can be compensated if an algorithm was well designed and implemented.

**98) How do you convert strings to numbers in C?**

You can write you own functions to do string to number conversions, or instead use C’s built in functions. You can use atof to convert to a floating point value, atoi to convert to an integer value, and atol to convert to a long integer value.

**99) Create a simple code fragment that will swap the values of two variables num1 and num2.**

int temp;

temp = num1;

num1 = num2;

num2 = temp;

**100) What is the use of a semicolon (;) at the end of every program statement?**

It has to do with the parsing process and compilation of the code. A semicolon acts as a delimiter, so that the compiler knows where each statement ends, and can proceed to divide the statement into smaller elements for syntax checking.

**101. Why is C called a mid-level programming language?**

C has characteristics of both assembly-level i.e. low-level and higher-level languages. So as a result, C is commonly called a middle-level language. Using C, a user can write an operating system as well as create a menu-driven consumer billing system.

**102. What are the features of the C language?**

Some features of the C language are-

1. It is Simple And Efficient.
2. C language is portable or Machine Independent.
3. C is a mid-level Programming Language.
4. It is a structured Programming Language.
5. It has a function-rich library.
6. Dynamic Memory Management.
7. C is super fast.
8. We can use pointers in C.
9. It is extensible.

**103. What is a token?**

The individual elements of a program are called Tokens. There are following 6 types of tokens are available in C:

* Identifiers
* Keywords
* Constants
* Operators
* Special Characters
* Strings

**104. What is the use of printf() and scanf() functions? Also explain format specifiers?**

* **printf()** is used to print the output on the display.
* **scanf()** is used to read formatted data from the keyboard.

Some datatype format specifiers for both printing and scanning purposes are as follows:

* **%d**: It's a datatype format specifier for printing and scanning an integer value.
* **%s**: It's a datatype format specifier for printing and scanning a string.
* **%c**: It's a datatype format specifier for displaying and scanning a character value.
* **%f**: The datatype format specifier %f is used to display and scan a float value.

**105. What's the value of the expression 5["abxdef"]?**

The answer is **'f'.**

Explanation: The string mentioned "abxdef" is an array, and the  expression is equal to "abxdef"[5]. Why is the inside-out expression equivalent?  Because a[b] is equivalent to \*(a + b) which is equivalent to \*(b + a) which is equivalent to b[a].

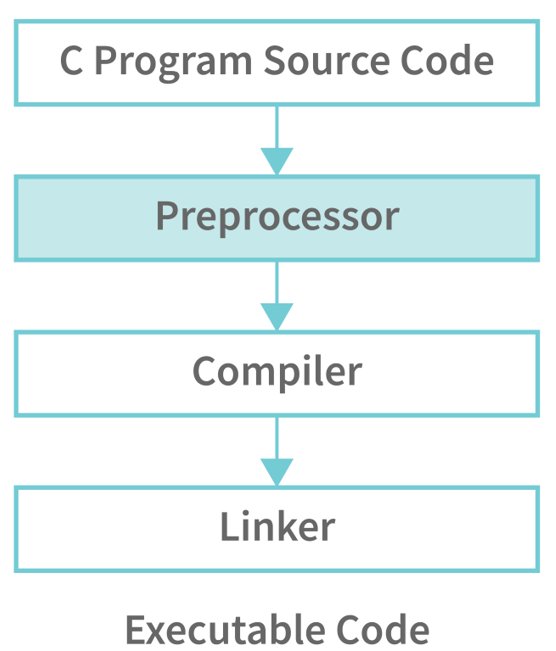
**106. What is a built-in function in C?**

The most commonly used built-in functions in C are scanf(), printf(), strcpy, strlwr, strcmp, strlen, strcat, and many more.

Built-function is also known as library functions that are provided by the system to make the life of a developer easy by assisting them to do certain commonly used predefined tasks. For example, if you need to print output or your program into the terminal, we use printf() in C.

**107. What is a Preprocessor?**

A preprocessor is a software program that processes a source file before sending it to be compiled. Inclusion of header files, macro expansions, conditional compilation, and line control are all possible with the preprocessor.



**108. In C, What is the #line used for?**

In C, #line is used as a preprocessor to re-set the line number in the code, which takes a parameter as line number. Here is an example for the same.

#**include** <stdio.h> /\*line 1\*/

/\*line 2\*/

**int** **main**(){ /\*line 3\*/

/\*line 4\*/

printf("Hello world\n"); /\*line 5\*/

//print current line /\*line 6\*/

printf("Line: %d\n",\_\_LINE\_\_); /\*line 7\*/

//reset the line number by 36 /\*line 8\*/

#**line** 36 /\*reseting\*/

//print current line /\*line 36\*/

printf("Line: %d\n",\_\_LINE\_\_); /\*line 37\*/

printf("Bye bye!!!\n"); /\*line 39\*/

/\*line 40\*/

**return** 0; /\*line 41\*/

} /\*line 42\*/

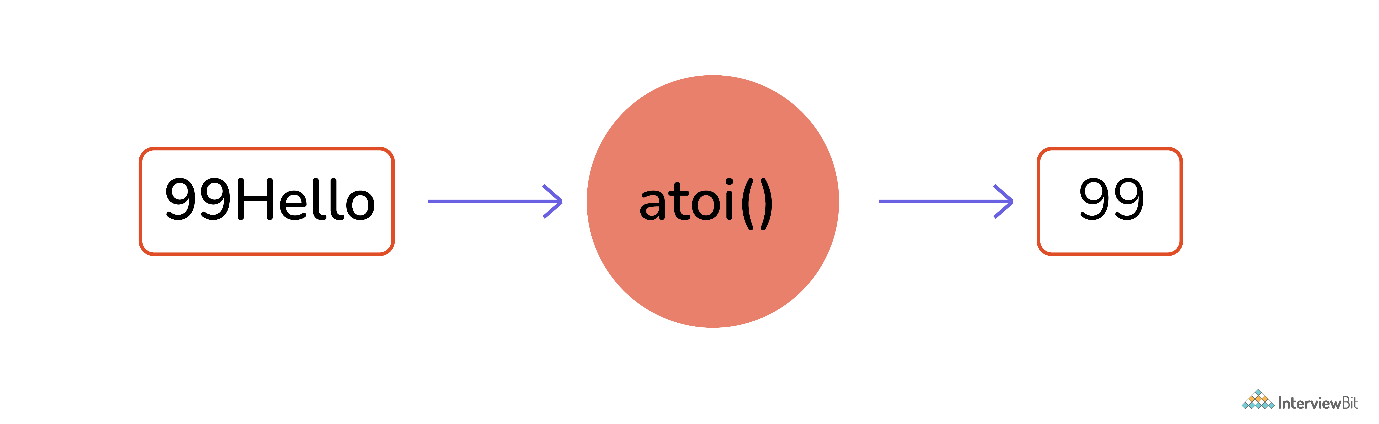
**109. How can a string be converted to a number?**

The function takes the string as an input that needs to be converted to an integer.

**int** **atoi**(**const** **char** \*string)

**Return Value:**

* On successful conversion, it returns the desired integer value
* If the string starts with alpha-numeric char or only contains alpha-num char, 0 is returned.
* In case string starts with numeric character but is followed by alpha-num char, the string is converted to integer till the first occurrence of alphanumeric char.

Converting String to Number

**110. How can a number be converted to a string?**

The function takes a pointer to an array of char elements that need to be converted, and a format string needs to be written in a buffer as a string

int sprintf(char \*str, const char \*format, ...)

The output after running the above code:

Output: Value of Pi = 3.141593

**111. What is recursion in C?**

When a function in C calls a copy of itself, this is known as recursion. To put it another way, when a function calls itself, this technique is called Recursion. Also, this function is known as recursive function.

**Syntax of Recursive Function:**

void do\_recursion()

{

... .. ...

do\_recursion();

... .. ...]

}

int main()

{

... .. ...

do\_recursion();

... .. ...

}

**112. Why doesn’t C support function overloading?**

After you compile the C source, the symbol names need to be intact in the object code. If we introduce **function overloading** in our source, we should also provide name mangling as a preventive measure to avoid function name clashes. Also, as C is not a strictly typed language many things(ex: data types) are convertible to each other in C. Therefore, the complexity of overload resolution can introduce confusion in a language such as C.

When you compile a C source, symbol names will remain intact. If you introduce function overloading, you should provide a name mangling technique to prevent name clashes. Consequently, like C++, you'll have machine-generated symbol names in the compiled binary.

Additionally, C does not feature strict typing. Many things are implicitly convertible to each other in C. The complexity of overload resolution rules could introduce confusion in such kind of language

Get Access to 250+ Guides with Scaler Mobile App!

Experience free learning content on the Scaler Mobile App

4.5

100K+

Play Store

**113. What is the difference between global int and static int declaration?**

The difference between this is in scope. A truly global variable has a global scope and is visible everywhere in your program.

#**include** <stdio.h>

**int** my\_global\_var = 0;

**int**

**main**(**void**)

{

printf("%d\n", my\_global\_var);

**return** 0;

}

global\_temp is a global variable that is visible to everything in your program, although to make it visible in other modules, you'd need an ”extern int global\_temp; ” in other source files if you have a multi-file project.

A static variable has a local scope but its variables are not allocated in the stack segment of the memory. It can have less than global scope, although - like global variables - it resides in the .bss segment of your compiled binary.

#**include** <stdio.h>

**int**

**myfunc**(**int** val)

{

**static** **int** my\_static\_var = 0;

my\_static\_var += val;

**return** my\_static\_var;

}

**int**

**main**(**void**)

{

**int** myval;

myval = myfunc(1);

printf("first call %d\n", myval);

myval = myfunc(10);

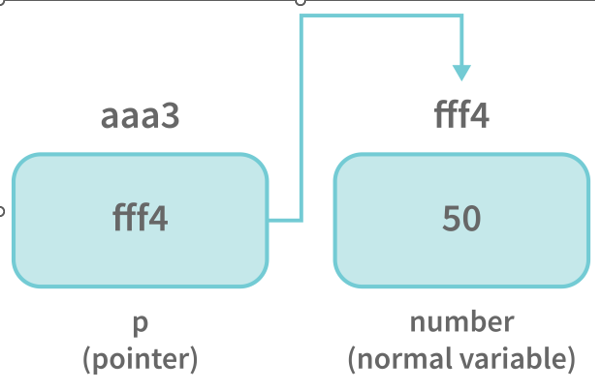
printf("second call %d\n", myval);

**return** 0;

}

**114. What is a pointer in C?**

A pointer is a variable that stores or points to another variable's address. The value of a variable is stored in a normal variable, whereas the address of a variable is stored in a pointer variable.



**115. Difference between const char\* p and char const\* p?**

* const char\* p is a pointer to a const char.
* char const\* p is a pointer to a char const.

Since const char and char const are the same, it's the same.

**116. What is pointer to pointer in C?**

In C, a pointer can also be used to store the address of another pointer. A double pointer or pointer to pointer is such a pointer. The address of a variable is stored in the first pointer, whereas the address of the first pointer is stored in the second pointer.

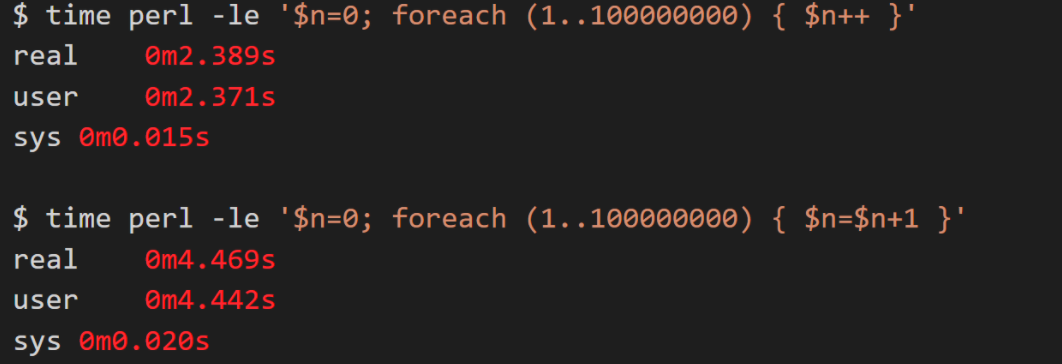
The syntax of declaring a double pointer is given below:

int \*\*p; // pointer to a pointer which is pointing to an integer

**117. Why n++ executes faster than n+1 ?**

n++ being a unary operation, it just needs one variable. Whereas, n = n + 1 is a binary operation that adds overhead to take more time (also binary operation: n += 1). However, in modern platforms, it depends on few things such as processor architecture, C compiler, usage in your code, and other factors such as hardware problems.

While in the modern compiler even if you write n = n + 1 it will get converted into n++ when it goes into the optimized binary, and it will be equivalently efficient.



**118. What is typecasting in C?**

Typecasting is the process to convert a variable from one datatype to another.  If we want to store the large type value to an int type, then we will convert the data type into another data type explicitly.

**Syntax:** (data\_type)expression;

**For Example:**

int x;

for(x=97; x<=122; x++)

{

printf("%c", (char)x); /\*Explicit casting from int to char\*/

}

**119. What are the advantages of Macro over function?**

Macro on a high-level copy-paste, its definitions to places wherever it is called. Due to which it saves a lot of time, as no time is spent while passing the control to a new function and the control is always with the callee function. However, one downside is the size of the compiled binary is large but once compiled the program comparatively runs faster.

**120. What are Enumerations?**

Enumeration, also known as Enum in C, is a user-defined data type. It consists of constant integrals or integers that have names assigned to them by the user. Because the integer values are named with enum in C, the whole program is simple to learn, understand, and maintain by the same or even different programmer.

**121. When should we use the register storage specifier?**

If a variable is used frequently, it should be declared with the register storage specifier, and the compiler may allocate a CPU register for its storage to speed up variable lookup.

**122. Specify different types of decision control statements?**

All statements written in a program are executed from top to bottom one by one. Control statements are used to execute/transfer the control from one part of the program to another depending on the condition.

* If-else statement.
  + normal if-else statement.
  + Else-if statement
  + nested if-else statement.
* Switch statement.

**123. What is an r-value and l-value?**

* The term "r-value" refers to a data value stored in memory at a given address. An r-value is an expression that cannot have a value assigned to it, hence it can only exist on the right side of an assignment operator(=).
* The term "l-value" refers to a memory location that is used to identify an object. The l-value can be found on either the left or right side of an assignment operator(=). l-value is frequently used as an identifier.

**124. What is the difference between malloc() and calloc()?**

[calloc() and malloc()](https://www.scaler.com/topics/difference-between-malloc-and-calloc-in-c/) are memory dynamic memory allocating functions. The main difference is that malloc() only takes one argument, which is the number of bytes, but calloc() takes two arguments, which are the number of blocks and the size of each block.

**125. What is the difference between struct and union in C?**

A struct is a group of complex data structures stored in a block of memory where each member on the block gets a separate memory location to make them accessible at once  
  
Whereas in the union, all the member variables are stored at the same location on the memory as a result to which while assigning a value to a member variable will change the value of all other members.

/\* struct & union definations\*/

**struct** **bar** {

**int** a; // we can use a & b both simultaneously

**char** b;

} bar;

**union** **foo** {

**int** a; // we can't use both a and b simultaneously

**char** b;

} foo;

/\* using struc and union variables\*/

**struct** **bar** **y**;

y.a = 3; // OK to use

y.b = 'c'; // OK to use

**union** **foo** **x**;

x.a = 3; // OK

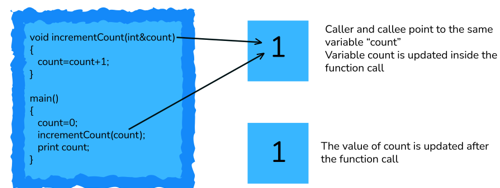
x.b = 'c'; // NOl this affects the value of x.a!

**126. What is call by reference in functions?**

When we caller function makes a function call bypassing the addresses of actual parameters being passed, then this is called call by reference. In incall by reference, the operation performed on formal parameters affects the value of actual parameters because all the operations performed on the value stored in the address of actual parameters.

**127. What is pass by reference in functions?**

In Pass by reference, the callee receives the address and makes a copy of the address of an argument into the formal parameter. Callee function uses the address to access the actual argument (to do some manipulation). If the callee function changes the value addressed at the passed address it will be visible to the caller function as well.



Pass By Reference

**128. What is a memory leak? How to avoid it?**

When we assign a variable it takes space of our RAM (either heap or RAM)dependent on the size of data type, however, if a programmer uses a memory available on the heap and forgets to a delta it, at some point all the memory available on the ram will be occupied with no memory left this can lead to a memory leak.

**int** **main**()

{

**char** \* ptr = malloc(**sizeof**(**int**));

/\* Do some work \*/

/\*Not freeing the allocated memory\*/

**return** 0;

}

To avoid memory leaks, you can trace all your memory allocations and think forward, where you want to destroy (in a good sense) that memory and place delete there. Another way is to use C++ smart pointer in C linking it to GNU compilers.

**129. What is Dynamic memory allocation in C? Name the dynamic allocation functions.**

C is a language known for its low-level control over the memory allocation of variables in DMA there are two major standard library malloc() and free. The malloc() function takes a single input parameter which tells the size of the memory requested It returns a pointer to the allocated memory. If the allocation fails, it returns NULL. The prototype for the standard library function is like this:

void \*malloc(size\_t size);  
The free() function takes the pointer returned by malloc() and de-allocates the memory. No indication of success or failure is returned. The function prototype is like this:

void free(void \*pointer);  
There are 4 library functions provided by C defined under <stdlib.h> header file to facilitate dynamic memory allocation in C programming. They are:

* malloc()
* calloc()
* free()
* realloc()

**130. What is typedef?**

typedef is a C keyword, used to define alias/synonyms for an existing type in C language. In most cases, we use typedef's to simplify the existing type declaration syntax. Or to provide specific descriptive names to a type.

**typedef** <existing-type> <**new**-type-identifiers>;

typedef provides an alias name to the existing complex type definition. With typedef, you can simply create an alias for any type. Whether it is a simple integer to complex function pointer or structure declaration, typedef will shorten your code.

**131. Why is it usually a bad idea to use gets()? Suggest a workaround.**

The standard input library gets() reads user input till it encounters a new line character. However, it does not check on the size of the variable being provided by the user is under the maximum size of the data type which makes the system vulnerable to buffer overflow and the input being written into memory where it isn’t supposed to.

We, therefore, use gets() to achieve the same with a restricted range of input

Bonus: It remained an official part of the language up to the 1999 ISO C standard, but it was officially removed by the 2011 standard. Most C implementations still support it, but at least GCC issues a warning for any code that uses it.

**132. What is the difference between #include "..." and #include <...>?**

In practice, the difference is in the location where the preprocessor searches for the included file.

For #include <filename> the C preprocessor looks for the filename in the predefined list of system directories first and then to the directories told by the user(we can use -I option to add directories to the mentioned predefined list).

For #include "filename" the preprocessor searches first in the same directory as the file containing the directive, and then follows the search path used for the #include <filename> form. This method is normally used to include programmer-defined header files.

**133. What are dangling pointers? How are dangling pointers different from memory leaks?**

The dangling pointer points to a memory that has already been freed. The storage is no longer allocated. Trying to access it might cause a Segmentation fault. A common way to end up with a dangling pointer:

#**include**<stdio.h>

#**include**<string.h>

**char** \***func**()

{

**char** str[10];

strcpy(str,"Hello!");

**return**(str);

}

You are returning an address that was a local variable, which would have gone out of scope by the time control was returned to the calling function. (Undefined behavior)

\*c = malloc(5izeof(**int**));

free(c);

\*c = 3; //writing to freed location!

In the figure shown above writing to a memory that has been freed is an example of the dangling pointer, which makes the program crash.

A memory leak is something where the memory allocated is not freed which causes the program to use an undefined amount of memory from the ram making it unavailable for every other running program(or daemon) which causes the programs to crash. There are various tools like O profile testing which is useful to detect memory leaks on your programs.

**void** **function**(){

**char** \*leak = malloc (10); //leak assigned but not freed

}

**134. What is the difference between ‘g’ and “g” in C?**

In C double-quotes variables are identified as a string whereas single-quoted variables are identified as the character. Another major difference being the string (double-quoted) variables end with a null terminator that makes it a 2 character array.

**135. What is a near pointer and a far pointer in C?**

* **Near Pointer**: In general, the near pointer can be considered because it is used to hold the address, which has a maximum size of just 16 bits. We can't store an address with a size larger than 16 bits using the near pointer. All other smaller addresses that are within the 16-bit limit, on the other hand, can be stored. Because we can only access 64kb of data at a time, you might assume the 16 bits are insufficient. As a result, it is regarded as one of the near-pointer's biggest drawbacks, which is why it is no longer commonly used.
* **Far Pointer:** A far pointer is considered a pointer of size 32 bits. It can, however, use the current segment to access information stored outside the computer's memory. Although, in order to use this type of pointer, we usually need to allocate the sector register to store the data address in the current segment.

**136. Which structure is used to link the program and the operating system?**

The file structure is used to link the operating system and a program. The header file "stdio.h" (standard input/output header file) defines the file. It contains information about the file being used like its current size and its memory location. It contains a character pointer that points to the character which is currently being opened. When you open a file, it establishes a link between the program and the operating system about which file is to be accessed.

**137. Suppose a global variable and local variable have the same name. Is it possible to access a global variable from a block where local variables are defined?**

No. This isn’t possible in C. It’s always the most local variable that gets preference.

**138. Which is better #define or enum?**

* If we let it, the compiler can build enum values automatically. However, each of the defined values must be given separately.
* Because macros are preprocessors, unlike enums, which are compile-time entities, the source code is unaware of these macros. So, if we use a debugger to debug the code, the enum is superior.
* Some compilers will give a warning if we use enum values in a switch and the default case is missing.
* Enum always generates int-type identifiers. The macro, on the other hand, allowed us to pick between various integral types.
* Unlike enum, the macro does not have a defined scope constraint.

**C Interview Questions For Experienced**

**140. How can you remove duplicates in an array?**

The following program will help you to remove duplicates from an array.

#include <stdio.h>

int main() {

int n, a[100], b[100], calc = 0, i, j,count;

printf("Enter no. of elements in array: ");

scanf("%d", &n);

printf("Enter %d integers: ", n);

for (i = 0; i < n; i++)

scanf("%d", &a[i]);

for (i = 0; i<n; i++) {

for (j = 0; j<calc; j++) {

if(a[i] == b[j])

break;

}

if (j== calc) {

b[calc] = a[i];

calc++;

}

}

printf("Array obtained after removing duplicate elements: ");

for (i = 0; i<calc; i++) {

printf("%d ", b[i]);

}

return 0;

}

**141. Can we compile a program without a main() function?**

Yes, we can compile a program without main() function Using Macro.

E.g.

#include<studio.h>

#define abc main

int abc ()

{

printf("Hello World ");

return 0;

}

**142. Write a program to get the higher and lower nibble of a byte without using shift operator?**

#include<stdio.h>

struct full\_byte

{

char first : 4;

char second : 4;

};

union A

{

char x;

struct full\_byte by;

};

main()

{

char c = 100;

union A a;

a.x = c;

printf("the two nibbles are: %d and %d\n", a.by.first, a.by.second);

}

**143. How do you override a defined macro?**

To override a defined macro we can use #ifdef and #undef preprocessors as follows:

* #ifdef A
* #undef A
* #endif
* #define A 10

If macro A is defined, it will be undefined using undef and then defined again using define.

**144. Write a C program to check if it is a palindrome number or not using a recursive method.**

#include <stdio.h>

#include <conio.h>

int reverse(int num);

int isPalindrome(int num);

int main()

{

int num;

printf("Enter a number: ");

scanf("%d", &num);

if(isPalindrome(num) == 1)

{

printf("the given number is a palindrome");

}

else

{

printf("the given number is not a palindrome number");

}

return 0;

}

int isPalindrome(int num)

{

if(num == reverse(num))

{

return 1;

}

return 0;

}

int reverse(int num)

{

int rem;

static int sum=0;

if(num!=0){

rem=num%10;

sum=sum\*10+rem;

reverse(num/10);

}

else

return sum;

return sum;

}

**145. C program to check the given number format is in binary or not.**

#include<stdio.h>

#include<conio.h>

int main() {

int j,num;

printf("Please enter a number :");

scanf("%d",&num);

while(num>0)

{

j=num%10;

if( j!=0 && j!=1 )

{

printf("num is not binary");

break;

}

num=num/10;

if(num==0)

{

printf("num is binary");

}

}

getch();

}

**146. C Program to find a sum of digits of a number using recursion.**

#include<stdio.h>

#include<conio.h>

int sumOfDigits(int num)

{

static int sum = 0;

int rem;

sum = sum + (num%10);

rem = num/10;

if(rem > 0)

{

sumOfDigits(rem);

}

return sum;

}

int main() {

int j,num;

printf("Please enter a number :");

scanf("%d",&num);

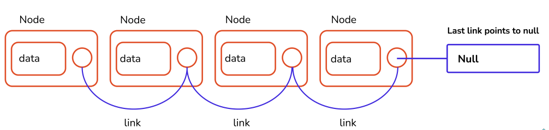
printf("sum of digits of the number = %d ",sumOfDigits(num));

getch();

}

**147. Can you tell me how to check whether a linked list is circular?**

**Single Linked List**



Single Linked List

**Circular Linked List**

Circular linked list is a variation of a linked list where the last node is pointing to the first node's information part. Therefore the last node does not point to null.

**Algorithm to find whether the given linked list is circular**

A very simple way to determine whether the linked list is circular or not

* Traverse the linked list
* Check if the node is pointing to the head.
* If yes then it is circular.

Let's look at the snippet where we code this algorithm.

Create a structure **for** a linked list

Declare

-Variable to store data of the node.

-Pointer variable of **struct** **type** **to** **store** **the** **address** **of** **next** **node**.

**function** **of** **datatype** **tool** **isCircular**(**firstgode**){

-Store the value of first node in temp variable **and** make it traverse all nodes.

-temp-firstgode

-tempenext node pointed by **temp**(temp->next)

-run until temp is at null **or** firstNode

**if** (temp at null)

**not** circular **and** returns false

**if** (temp points first node)

**return** true as its circular.

}

function of datatype node **newNode**(data){

-To insert **new** nodes **and** link each one of them to the previous node by storing the address of the **new** node to the previous one.

-Then make them point to NULL.

}

In **int** main function

-First insert nodes **for** circular linked list **and** check its nature by calling isCircular function.

-Since it is true through **if** statement it prints "yes..

-Second insert a normal linked list and check its nature by calling isCircular function. As its not circular it prints "no",

**148. What is the use of a semicolon (;) at the end of every program statement?**

It is majorly related to how the compiler reads( or parses) the entire code and breaks it into a set of instructions(or statements), to which semicolon in C acts as a boundary between two sets of instructions.

**149. How to call a function before main()?**

To call a function before the main(), pragma startup directive should be used. E.g.-

#pragma startup fun

void fun()

{

printf("In fun\n");

}

main()

{

printf("In main\n");

}

The output of the above program will be -

In fun

In main

This pragma directive, on the other hand, is compiler-dependent. This is not supported by gcc. As a result, it will ignore the startup directive and produce no error. But the output, in that case, will be -

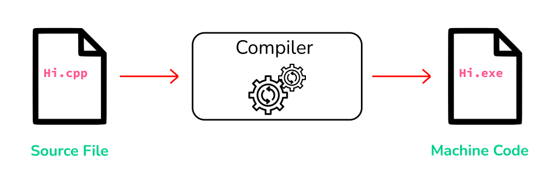
In main

**150. Differentiate between the macros and the functions.**

The differences between macros and functions can be explained as follows:

| **Macros** | **Functions** |
| --- | --- |
| It is preprocessed rather than compiled. | It is compiled not preprocessed. |
| It is preprocessed rather than compiled. | Function checks for compilation errors. |
| Code length is increased. | Code length remains the same. |
| Macros are faster in execution. | Functions are a bit slower in execution. |
| Macros are useful when a small piece of code is used multiple times in a program. | Functions are helpful when a large piece of code is repeated a number of times. |

**151. Differentiate Source Codes from Object Codes**

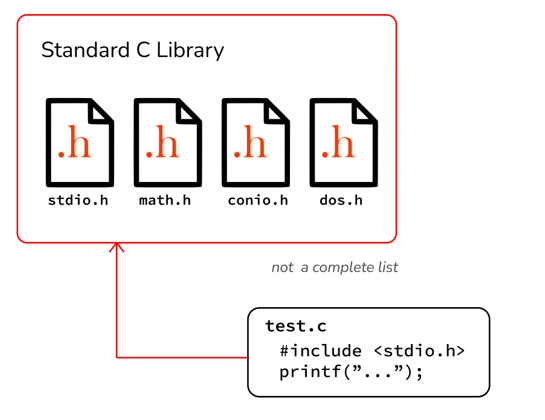
****

Source Code and Object Code Difference

The difference between the Source Code and Object Code is that Source Code is a collection of computer instructions written using a human-readable programming language while Object Code is a sequence of statements in machine language, and is the output after the compiler or an assembler converts the Source Code.

The last point about Object Code is the way the changes are reflected. When the Source Code is modified, each time the Source Code needs to be compiled to reflect the changes in the Object Code.

**152. What are header files and what are its uses in C programming?**



Header Files in C

In C header files must have the extension as .h, which contains function definitions, data type definitions, macro, etc. The header is useful to import the above definitions to the source code using the #include directive. For example, if your source code needs to take input from the user do some manipulation and print the output on the terminal, it should have stdio.h file included as #include <stdio.h>, with which we can take input using scanf() do some manipulation and print using printf().

**153. When is the "void" keyword used in a function**

The keyword “void” is a data type that literally represents no data at all. The most obvious use of this is a function that returns nothing:

**void** **PrintHello**()

{

printf("Hello\n");

**return**; // the function does "return", but no value is returned

}

Here we’ve declared a function, and all functions have a return type. In this case, we’ve said the return type is “void”, and that means, “no data at all” is returned.   
The other use for the void keyword is a void pointer. A void pointer points to the memory location where the data type is undefined at the time of variable definition. Even you can define a function of return type void\* or void pointer meaning “at compile time we don’t know what it will return” Let’s see an example of that.

**void** **MyMemCopy**(**void**\* dst, **const** **void**\* src, **int** numBytes)

{

**char**\* dst\_c = **reinterpret\_cast**<**char**\*>(dst);

**const** **char**\* src\_c = **reinterpret\_cast**<**const** **char**\*>(src);

**for** (**int** i = 0; i < numBytes; ++i)

dst\_c[i] = src\_c[i];

}

**154. What is dynamic data structure?**

A dynamic data structure (DDS) refers to an organization or collection of data in memory that has the flexibility to grow or shrink in size, enabling a programmer to control exactly how much memory is utilized. Dynamic data structures change in size by having unused memory allocated or de-allocated from the heap as needed.

Dynamic data structures play a key role in programming languages like C, C++, and Java because they provide the programmer with the flexibility to adjust the memory consumption of software programs.

**155. Add Two Numbers Without Using the Addition Operator**

For the sum of two numbers, we use the addition (+) operator. In these tricky C programs, we will write a C program to add two numbers without using the addition operator.

#**include**<stdio.h>

#**include**<stdlib.h>

**int** **main**()

{

**int** x, y;

printf("Enter two number: ");

scanf("%d %d",&x,&y);

// method 1

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

// method 2

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

// method 3

printf("%d\n", abs(-x-y));

// method 4

printf("%d", x-(~y)-1);

**return** 0;

}

**155. Subtract Two Number Without Using Subtraction Operator**

#**include**<stdio.h>

#**include**<stdlib.h>

**int** **main**()

{

**int** x, y;

printf("Enter two number: ");

scanf("%d %d",&x,&y);

printf("%d", x+(~y)+1);

**return** 0;

}

The bitwise complement operator is used in this program. The bitwise complement of number ~y=-(y+1). So, expression will become x+(-(y+1))+1=x-y-1+1=x-y

**156. Multiply an Integer Number by 2 Without Using Multiplication Operator**

#**include**<stdio.h>

**int** **main**()

{

**int** x;

printf("Enter a number: ");

scanf("%d",&x);

printf("%d", x<<1);

**return** 0;

}

The left shift operator shifts all bits towards the left by a certain number of specified bits. The expression x<<1 always returns x\*2. Note that the shift operator doesn’t work on floating-point values.

For multiple of x by 4, use x<<2. Similarly x<<3 multiply x by 8. For multiple of the number x by 2^n, use x<<n.

**157. Check whether the number is EVEN or ODD, without using any arithmetic or relational operators**

#**include**<stdio.h>

**int** **main**()

{

**int** x;

printf("Enter a number: ");

scanf("%d", &x);

(x&1)?printf("Odd"):printf("Even");

**return** 0;

}

The bitwise and(&) operator can be used to quickly check the number is odd or even.

**158. Reverse the Linked List. Input: 1->2->3->4->5->NULL Output: 5->4->3->2->1->NULL**

Assume that we have linked list 1 → 2 → 3 → Ø, we would like to change it to Ø ← 1 ← 2 ← 3.

While you travel the linked list, change the current node's next pointer to point to its previous element. reference to the previous nodes should be stored into a temp variable as shown so that we don’t lose track of the swapped node. You also need another pointer to store the next node before changing the reference. Also when we are done return the new head of the reversed list.

/\* Function to reverse the linked list \*/

**static** **void** **reverse**(struct Node\*\* head\_ref)

{

**struct** **Node**\* **prev** = NULL;

**struct** **Node**\* **current** = \*head\_ref;

**struct** **Node**\* **next**;

**while** (current != NULL)

{

// store next

next = current->next;

// reverse curr node pointer

current->next = prev;

// move pointer one position ahead

prev = current;

current = next;

}

\*head\_ref = prev;

}

**159. Check for Balanced Parentheses using Stack**

Given a string s containing just the characters '(', ')', '{', '}', '[' and ']', determine if the input string is valid.

An input string is valid if:

* Open brackets must be closed by the same type of brackets.
* Open brackets must be closed in the correct order.

**Example 1:**  
 Input: s = "()"  
 Output: true  
  
**Example 2:**  
 Input: s = "()[]{}"  
 Output: true  
  
**Example 3:**  
 Input: s = "(]"  
 Output: false

Below is the source code for C Program to Check for Balanced Parentheses using Stack which is successfully compiled and run on Windows System to produce desired output as shown below :

**int** **check**(**char** exp[] )

{

**int** i;

**char** temp;

**for**(i=0;i<strlen(exp);i++)

{

**if**(exp[i]=='(' || exp[i]=='{' || exp[i]=='[')

push(exp[i]);

**if**(exp[i]==')' || exp[i]=='}' || exp[i]==']')

**if**(top==-1) /\*stack empty\*/

{

printf("Right parentheses are more than left parentheses\n");

**return** 0;

}

**else**

{

temp=pop();

**if**(!match(temp, exp[i]))

{

printf("Mismatched parentheses are : ");

printf("%c and %c\n",temp,exp[i]);

**return** 0;

}

}

}

**if**(top==-1) /\*stack empty\*/

{

printf("Balanced Parentheses\n");

**return** 1;

}

**else**

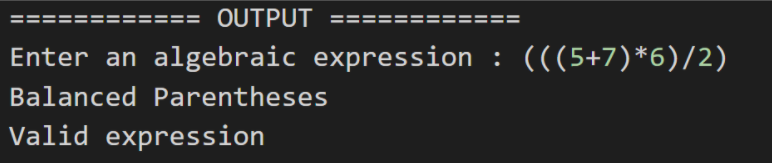
{

printf("Left parentheses more than right parentheses\n");

**return** 0;

}

}



**160. Program to find n’th Fibonacci number**

Fibonacci sequence is characterized by the fact that every number after the first two is the sum of the two preceding ones. For example, consider below sequence

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, . .. and so on

Where in F{n} = F{n-1} + F{n-2} with base values F(0) = 0 and <code>F(1) = 1

Below is naive implementation for finding the nth member of the Fibonacci sequence

// Function to find the nth Fibonacci number

**int** **fib**(**int** n)

{

**if** (n <= 1) {

**return** n;

}

**return** fib(n - 1) + fib(n - 2);

}

**int** **main**()

{

**int** n = 8;

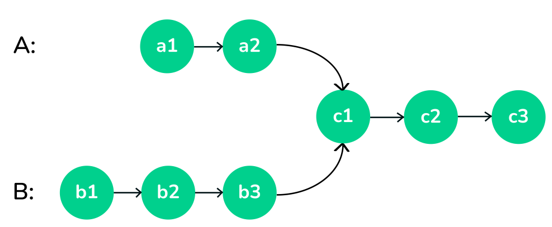
printf("nth Fibonacci number is %d", fib(8));

**return** 0;

}

**161. Write a program to find the node at which the intersection of two singly linked lists begins.**

Let's take an example of the following two linked lists which intersect at node c1.



Intersection of Two Linked List

**Solution -**

* Get count of the nodes in the first list, let count be c1.
* Get count of the nodes in the second list, let count be c2.
* Get the difference of counts d = abs(c1 – c2)
* Now traverse the bigger list from the first node till d nodes so that from here onwards both the lists have an equal no of nodes
* Then we can traverse both the lists in parallel till we come across a common node. (Note that getting a common node is done by comparing the address of the nodes)

// Function to get the intersection point

// of the given linked lists

**int** **getIntersectionNode**(Node\* head1, Node\* head2)

{

Node \*curr1 = head1, \*curr2 = head2;

// While both the pointers are not equal

**while** (curr1 != curr2) {

// If the first pointer is null then

// set it to point to the head of

// the second linked list

**if** (curr1 == NULL) {

curr1 = head2;

}

// Else point it to the next node

**else** {

curr1 = curr1->next;

}

// If the second pointer is null then

// set it to point to the head of

// the first linked list

**if** (curr2 == NULL) {

curr2 = head1;

}

// Else point it to the next node

**else** {

curr2 = curr2->next;

}

}

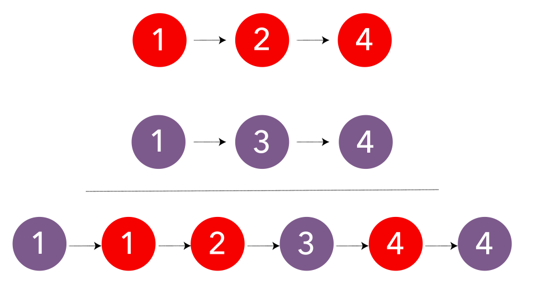
// Return the intersection node

**return** curr1->data;

}

**162. Merge Two sorted Linked List**

Merge two sorted linked lists and return them as a sorted list. The list should be made by splicing together the nodes of the first two lists.



Merging Two Sorted Linked List

NodePtr **merge\_sorted**(NodePtr head1, NodePtr head2) {

// if both lists are empty then merged list is also empty

// if one of the lists is empty then other is the merged list

**if** (head1 == nullptr) {

**return** head2;

} **else** **if** (head2 == nullptr) {

**return** head1;

}

NodePtr mergedHead = nullptr;

**if** (head1->data <= head2->data) {

mergedHead = head1;

head1 = head1->next;

} **else** {

mergedHead = head2;

head2 = head2->next;

}

NodePtr mergedTail = mergedHead;

**while** (head1 != nullptr && head2 != nullptr) {

NodePtr temp = nullptr;

**if** (head1->data <= head2->data) {

temp = head1;

head1 = head1->next;

} **else** {

temp = head2;

head2 = head2->next;

}

mergedTail->next = temp;

mergedTail = temp;

}

**if** (head1 != nullptr) {

mergedTail->next = head1;

} **else** **if** (head2 != nullptr) {

mergedTail->next = head2;

}

**return** mergedHead;

}

Runtime Complexity Linear, O(m + n) where m and n are lengths of both linked lists.

Memory Complexity Constant, O(1)

Maintain a head and a tail pointer on the merged linked list. Then choose the head of the merged linked list by comparing the first node of both linked lists. For all subsequent nodes in both lists, you choose the smaller current node and link it to the tail of the merged list, moving the current pointer of that list one step forward.

You keep doing this while there are some remaining elements in both lists. If there are still some elements in only one of the lists, you link this remaining list to the tail of the merged list.

Initially, the merged linked list is NULL. Compare the value of the first two nodes and make the node with the smaller value the head node of the merged linked list. In this example, it is 4 from head1.

Since it’s the first and only node in the merged list, it will also be the tail. Then move head1 one step forward.

Reference – 3 (163 + 73 = 236 questions)

Q1. What is C programming?

Ans. C is one of the oldest programming languages and was developed in 1972 by Dennis Ritchie. It is a high-level, general-purpose, and structured programming language that is widely used in various tasks, such as developing system applications, desktop applications, operating systems as well as IoT applications. It is the simple and flexible language used for a variety of scripting system applications, which form a significant part of Windows, UNIX, and Linux operating systems.

Q2. Why is C dubbed as a mother language?

Ans. It is known as a mother language because most of the JVMs, compilers, and Kernels are written in C language. If you know C, then you can easily grasp other programming languages.

Q3. Who is the founder of the C language? When was the C language developed?

Ans. C is developed by Dennis M. Ritchie in 1972 at the Bell laboratories of AT & T.

Q4. What are the key features of the C Programming language?

Ans. The following are the major features of C:

Machine Independent

Mid-Level programming language

Memory Management

Structured programming language

Simple and efficient

Function rich libraries

Case Sensitive

Q5. Name some different storage class specifiers in C?

Ans. Storage classes represent the storage of any variable. There are four storage classes in C:

Auto

Register

Extern

Static

Q6. What are the data types supported in the C Programming language?

Ans. This is one of the commonly asked C programming interview questions. The data type specifies the type of data used in programming. C language has some predefined data types with different storage capacities:

Built-in data types: It includes int, char, double, float and void

Derived data types: It includes array, pointers, and references

User-defined data types: Union, structure, and Enumeration

Q7. What do you mean by the scope and lifetime of a variable in C?

Ans. The scope and lifetime of any variable are defined as the section of the code in which the variable is executable or visible. Variables that are described in the block are only accessible in the block. The variable’s lifetime defines the variable’s existence before it is destroyed.

Q8. What do you mean by the pointer in C?

Ans. Pointers are the variables in C that store the address of another variable. It allocates the memory for the variable at the run time. The pointer might be of different data types such as int, char, float, double, etc.

Example:

#include < stdio.h >

int main()

{

int \*ptr, q;

q = 50;

/\* address of q is assigned to ptr \*/

ptr = &q;

/\* display q’s value using ptr variable \*/

printf(“%d”, \*ptr);

return 0;

}

Copy code

Q9. Define Null pointer?

Ans. A null pointer represents the empty location in the computer’s memory. It is used in C for various purposes:

It will assign the pointer variable to the variable with no assigned memory

In pointer related code we can do error handling by checking null pointer before assigning any pointer variable

Pass a null pointer to the variable if we don’t want to pass any valid memory address

Example:

int fun(int \*ptr)

{

/\*Fun specific stuff is done with ptr here\*/

return 10;

}

fun(NULL);

Copy code

Q10. What is Dangling Pointer?

Ans. Dangling pointers are the pointers pointing to the memory location that has been deleted or released. There are three different types of dangling pointers:

Return local variable in a function call

#include < stdio.h >

#include < string.h >

char \*getHello()

{

char str[10]; strcpy(str,”Hello!”);

return(str);

}

int main()

{

//str falls out of scope

//function call char \*getHello() is now a dangling pointer

printf(“%s”, getHello());

}

Copy code

Variable goes out of scope

void main()

{

int \*p1;

…..

{

int ch;

p1 = &ch;

}

…..

// Here ptr is a dangling pointer

}

Copy code

De allocation or free variable memory

#include < stdio.h >

#include < stdlib.h >

int main()

{

char \*\*strPtr;

char \*str = “Hello!”;

strPtr = &str; free(str);

//strPtr is now a dangling pointer

printf(“%s”, \*strPtr);

}

Copy code

Q11. What is the use of function in C?

Ans. Functions are the basic building blocks of any programming language. All C programs are written using the function to maintain reusability and understandability.

Uses of functions in C:

Functions can be used multiple times in a program by calling them whenever required

In C, functions are used to avoid rewriting of code

It is easy to track any C program when it is divided into functions

Syntax of a function

return\_type function\_name( parameter list )

{

//body of the function

}

Copy code

Let’s take a look at some more C programming interview questions.

Q12. What happens when you compile a program in C?

Ans. At the time of compilation, the compiler generates a file with the same name as the C program file with different extensions.

Below is the image to show the compilation process:

2017\_06\_compiation-process.jpg

Q13. What are header files in C?

Ans. Header files are those which contain C function declaration and macro definitions that are to be shared between sourced files. Header files are generated with the extension .h.

There are two types of header files:

Programmer generated a header file

Files that come with your compiler

Syntax: #include <file>

Q14. How many types of operators are there in C Programming?

Ans. An operator is a symbol used to operate the values of variables. There is a wide range of operators used in C programming, including –

Arithmetic Operators:

These are used to perform mathematical calculations such as addition, subtraction, multiplication, division, and modulus.

Example:

#include < stdio.h >

int main()

{

int a = 9,b = 4, c;

c = a+b;

printf(“a+b = %d \n”,c);

c = a-b;

printf(“a-b = %d \n”,c);

c = a\*b;

printf(“a\*b = %d \n”,c);

c = a/b;

printf(“a/b = %d \n”,c);

c = a%b;

printf(“Remainder when a divided by b = %d \n”,c);

return 0;

}

Copy code

Relational Operators:

These are used to check the relation between two operands. If the relation is TRUE, it returns 1; If the relation is FALSE, It returns 0.

Example:

#include < stdio.h >

int main()

{

int a = 2, b = 2, c = 6;

printf(“%d == %d is %d \n”, a, b, a == b);

printf(“%d == %d is %d \n”, a, c, a == c);

printf(“%d > %d is %d \n”, a, b, a > b);

printf(“%d > %d is %d \n”, a, c, a > c);

printf(“%d < %d is %d \n”, a, b, a < b);

printf(“%d < %d is %d \n”, a, c, a < c);

printf(“%d != %d is %d \n”, a, b, a != b);

printf(“%d != %d is %d \n”, a, c, a != c);

printf(“%d > = %d is %d \n”, a, b, a > = b);

printf(“%d > = %d is %d \n”, a, c, a > = c);

printf(“%d < = %d is %d \n”, a, b, a < = b);

printf(“%d < = %d is %d \n”, a, c, a < = c);

return 0;

}

Copy code

Logical Operators:

These are used in decision-making operations. If the expression is TRUE, it returns 1; if the expression is FALSE, it returns 0.

Example:

#include < stdio.h >

int main()

{

int a = 2, b = 2, c = 6, result;

result = (a == b) && (c > b);

printf(“(a == b) && (c > b) is %d \n”, result);

result = (a == b) && (c < b);

printf(“(a == b) && (c < b) is %d \n”, result);

result = (a == b) || (c < b);

printf(“(a == b) || (c < b) is %d \n”, result);

result = (a != b) || (c < b);

printf(“(a != b) || (c < b) is %d \n”, result);

result = !(a != b);

printf(“!(a == b) is %d \n”, result);

result = !(a == b);

printf(“!(a == b) is %d \n”, result);

return 0;

}

Copy code

Assignment Operators:

These are used to assign value to a variable. The most used assignment operator is ‘=’.

Example:

#include < stdio.h >

int main()

{

int a = 5, c;

c = a; // c is 5

printf(“c = %d\n”, c);

c += a; // c is 10

printf(“c = %d\n”, c);

c -= a; // c is 5

printf(“c = %d\n”, c);

c \*= a; // c is 25

printf(“c = %d\n”, c);

c /= a; // c is 5

printf(“c = %d\n”, c);

c %= a; // c = 0

printf(“c = %d\n”, c);

return 0;

}

Copy code

Increment and Decrement Operators:

These are used to change the value of an operand (constant or variable) by 1.

Example:

#include < stdio.h >

int main()

{

int a = 10, b = 100;

float c = 10.5, d = 100.5;

printf(“++a = %d \n”, ++a);

printf(“–b = %d \n”, –b);

printf(“++c = %f \n”, ++c);

printf(“–d = %f \n”, –d);

return 0;

}

Copy code

Bitwise Operators:

These are used to perform bit-level operations between two variables.

Example: Bitwise OR, Bitwise AND

Conditional Operator:

These are used in conditional expressions.

Example: ‘?:’ Conditional operator

Special Operators:

There are some special operators in C used for:

sizeof(): Returns the size of the memory location

&: Returns the address of a memory location

\*: Pointer of a variable

Q15. Explain the process of creating increment and decrement operators in C.

Ans. This is one of the most important C programming interview questions. If you want to perform an increment operation, then use ‘i++,’ which will increase the value by 1. If you want to perform a decrement operation, then use ‘i–,’ it will decrease the value by 1.

Example:

#include < stdio.h >

int main()

{

int x = 10, y = 1;

printf(“Initial value of x = %d\n”, x);

printf(“Initial value of y = %d\n\n”, y);

y = ++x;

printf(“After incrementing by 1: x = %d\n”, x);

printf(“y = %d\n\n”, y);

y = –x;

printf(“After decrementing by 1: x = %d\n”, x);

printf(“y = %d\n\n”, y);

return 0;

}

Copy code

Also Read: 5 Best Programming Languages To Learn For Cybersecurity Professionals

Q16. Mention the difference between local variables and global variables in C?

Ans. Global variables are declared outside the function. That variable can be used anywhere in the program, whereas local variables are declared inside the function, and their scope is only inside that function.

// Global variable

float x = 1;

void my\_test() {

// Local variable called y.

// This variable can’t be accessed in other functions

float y = 77;

println(x);

println(y);

}

void setup() {

// Local variable called y.

// This variable can’t be accessed in other functions.

float y = 2;

println(x);

println(y);

my\_test();

println(y);

}

Copy code

Q17. What is static memory allocation in C?

Ans. Static memory allocation is defined as the allocation of a fixed amount of memory at the compile time, and the operating system uses the data structure called stacks to manage memory allocation.

Example:

void demo

{

int x;

}

int main()

{

int y;

int c[10];

return 1;

}

Copy code

Q18. What is dynamic memory allocation in C?

Ans. Dynamic memory allocation is the process of memory allocation at the run time. There are a group of functions in C used to dynamic memory management i.e. calloc(), malloc(), realloc() and free().

Q19. Explain the difference between calloc() and malloc().

Ans. The main difference between calloc() and malloc() is that calloc() takes two arguments while malloc() takes one argument. Secondly, calloc() initializes allocated memory to ZERO while malloc() does not initialize allocated memory.

Syntax of calloc()

void \*calloc(size\_t n, size\_t size);

Syntax of malloc()

void \*malloc(size\_t n);

Copy code

Q20. What is the output of the following C code?

#include < stdlib.h >

#include < stdio.h >

enum {false, true};

int main()

{

int i = 1;

do

{

printf(“%d\n”, i);

i++;

if (i < 15)

continue;

}

while (false);

getchar();

return 0;

}

Copy code

Ans. Output: 1

The do-while loop executes at every iteration. After the continue statement, it will come to the while (false) statement, and the condition shows false, and ‘i’ is printed only once.

Q21. How can a negative integer be stored in C?

Ans. If the number is with a negative sign, then at the time of memory allocation, the number (ignoring the minus sign) is converted into the binary equivalent. Then the two’s complement of the number is calculated.

Example:

#include < stdio.h >

int main()

{

int a = -4;

int b = -3;

unsigned int c = -4;

unsigned int d = -3;

printf(“%f\n%f\n%f\n%f\n”, 1.0 \* a/b, 1.0 \* c/d, 1.0\*a/d, 1.\*c/b);

}

Copy code

Output:

1.333333

1.000000

-0.000000

-1431655764.000000

Q22. What is the use of nested structure in C?

Ans. A nested structure is used to make the complicated code easy. If we want to add the address of employees with other more details, then we have to create a nested structure for it.

Example:

#include < stdio.h >

struct address

{

char city[20];

int pin;

char phone[14];

};

struct employee

{

char name[20];

struct address add;

};

void main ()

{

struct employee emp;

printf(“Enter employee information?\n”);

scanf(“%s %s %d %s”,emp.name,emp.add.city, &emp.add.pin, emp.add.phone);

printf(“Printing employee information…\n”);

printf(“name:%s\nCity:%s\nPincode:%d\nPhone: %s”,emp.name,emp.add.city,emp.add.pin,emp.add.phone);

}

Copy code

Output:

Enter employee information?

Joe

Delhi

110001

1234567890

Printing employee information…

Name: Joe

City: Delhi

Pincode: 110001

Phone: 123456789

Copy code

Also Read: Most Popular Programming Languages for Data Science

Q23. How to write a program in C for swapping two numbers without the use of the third variable?

Ans.

#include < stdio.h >

int main()

{

int x, y;

printf(“Input two integers (x & y) to swap\n”);

scanf(“%d%d”, &x, &y);

x = x + y;

y = x – y;

x = x – y;

printf(“x = %d\ny = %d\n”,x,y);

return 0;

}

Copy code

Q24. Write a C program for Fibonacci series.

Ans.

#include < stdio.h >

int main()

{

int n1=0,n2=1,n3,i,number;

printf(“Enter the number of elements:”);

scanf(“%d”,&number);

printf(“\n%d %d”,n1,n2);//printing 0 and 1

for(i=2;i < number;++i)//loop starts from 2 because 0 and 1 are already printed

{

n3=n1+n2;

printf(” %d”,n3);

n1=n2;

n2=n3;

}

return 0;

}

Copy code

Q25. Explain the difference between = and == symbols in C programming?

Ans. The assignment operator (=): It is a binary operator used to operate two operands. It is used to assign the value to the variable.

Example: x=(a+b);

y=x;

Equal to operator (==): It is also a binary operator used to compare the left-hand side and right-hand side value, if it is the same, it returns 1 else 0.

int x,y;

x=10;

y=10;

if(x==y)

printf(“True”);

else

printf(“False”);

Copy code

When the expression x==y evaluates, it will return 1

Let’s move forward and take a look at some more frequently asked C programming interview questions.

Q26. What is the use of an extern storage specifier?

Ans. It enables you to declare a variable without bringing it into existence. The value is assigned to it in a different block, and it can be changed in the various blocks as well. So extern storage specifier is a global variable that can be used anywhere in the code.

Q27. What is the difference between rvalue and Ivalue in C?

Ans. The term rvalue refers to objects that appear on the right side, while an Ivalue is an expression that appears on the left side.

Q28. Can a program be compiled with the main function?

Ans. Yes.

Q29. Define stack.

Ans. It is a data structure that is used to store data in a particular order in which operations are performed. There are two types of storing orders, i.e. LIFO (last in first out) and FIFO (first in last out).

Basic operations performed in the stack:

Push

Pop

Peek or Top

isEmpty

Q30. When is the arrow operator used?

Ans. The Arrow operator is used to access elements in structure and union. It is used with a pointer variable. Arrow operator is formed by using a minus sign followed by a greater than a symbol.

Syntax:

(pointer\_name)->(variable\_name)

Q31. Can two operators be combined in a single line of program code?

Ans. Yes.

Q32. Write a program to find the factorial of a number using functions?

Ans: Program to find the factorial of a number

#include < stdio.h >

int factorial\_of\_a\_number(int n)

{

int fact = 1, i;

if(n == 0)

return 1;

else

for(i = 1; i < = n; i++)

{

fact = fact \* i;

}

return fact;

}

int main()

{

int n;

printf(“Enter the number : “);

scanf(“%d”,&n);

if(n < 0)

printf(“Invalid output”);

else

printf(“Factorial of the number %d is %d” ,n, factorial\_of\_a\_number(n));

return 0;

}

Copy code

Q33. What is a token in C?

Ans. The smallest individual unit in a C program is known as a token. Tokens can be classified as:

Keywords

Constants

Identifiers

Strings

Operators

Special symbols

Also Read: Best Software Development Frameworks & Tools To Learn

Q34. Name the keyword used to perform unconditional branching.

Ans. A go-to statement is used to perform unconditional branching.

Q35. What is the use of the comma operator?

Ans. It is used to separate two or more expressions.

E.g. printf (“hello”);

Q36. Write a program to find a sum of first N natural numbers.

Ans.

#include < stdio.h >

void main()

{

int i, num, sum = 0;

printf(“Enter an integer number \n”);

scanf (“%d”, &num);

for (i = 1; i < = num; i++)

{

sum = sum + i;

}

printf (“Sum of first %d natural numbers = %d\n”, num, sum);

}

Copy code

Q37. What is the length of an identifier?

Ans. Its length is 32 characters in C.

Q38. What is typecasting in C?

Ans. It is a way to convert constant from one type to another type. If there is a value of float data type then you can typecast it into other data types.

There are two types of typecasting in C:

Implicit conversion

Explicit conversion

Example:

#include < stdio.h >

main() {

int sum = 17, count = 5;

double mean;

mean = (double) sum / count;

printf(“Value of mean : %f\n”, mean );

}

Copy code

Q39. How are random numbers generated?

Ans. Random numbers are generated by using the rand () command.

Example:

#include <time.h>

#include <stdlib.h>

Srand (time (NULL));

Int r = rand ();

Also Read: Top Universities Offering Free Online Courses For Programmers

Q40. What are the disadvantages of void pointer?

Ans. Disadvantages of a void pointer:

Pointer arithmetic is not defined for void pointer

Void pointers can’t be dereferenced

Q41. Define compound statements.

Ans. These are made up of two or more program statements that are executed together.

Q42. Write a program to print numbers from 1 to 100 without using a loop.

Ans. Program to print numbers from 1 to 100

/\* Prints numbers from 1 to n \*/

void printNos(unsigned int n)

{

if(n > 0)

{

printNos(n-1);

printf(“%d “, n);

}

}

Copy code

Q43. What is FIFO?

Ans. FIFO means first in first out. It is a cost flow assumption, which is used to remove costs from the inventory account.

Q44. What is the use of a built-in stricmp() function?

Ans. It takes two strings and returns an integer.

Q45. What is the name of the function used to close the file stream?

Ans. Fclose().

Q46. What is the structure?

Ans. A user-defined data type that enables the combination of different data types to store a particular type of record, is known as a structure.

Example:

struct Point

{

int x, y;

} p1; // The variable p1 is declared with ‘Point’

struct Point

{

int x, y;

};

int main()

{

struct Point p1; // The variable p1 is declared like a normal variable

}

Copy code

Now, let’s take a look at some program-based C programming interview questions.

Q47. Write a program to reverse a number.

Ans. Below is the program to reverse a number in C:

#include < stdio.h >

int main()

{

int n, rev = 0, rem;

printf(“\nEnter a number : “);

scanf(“%d”, &n);

printf(“\nReversed Number : “);

while(n != 0)

{

rem = n;

rev = rev\*10 + rem;

n /= 10;

}

printf(“%d\n”, rev);

return 0;

}

Copy code

Q48. Write a program to check the prime number in C.

Ans. Below is the program to check the prime number in C:

#include < stdio.h >

#include < conio.h >

void main()

{

int n,i,m=0,flag=0; //declaration of variables.

clrscr(); //It clears the screen.

printf(“Enter the number to check prime:”);

scanf(“%d”,&n);

m=n/2;

for(i=2;i < =m;i++)

{

if(n%i==0)

{

printf(“Number is not prime”);

flag=1;

break; //break keyword used to terminate from the loop.

}

}

if(flag==0)

printf(“Number is prime”);

getch(); //It reads a character from the keyword.

}

Copy code

Q49. Write a program to check Armstrong number in C.

Ans. The program to check Armstrong number in C is as follows:

#include < stdio.h >

#include < conio.h >

main()

{

int n,r,sum=0,temp; //declaration of variables.

clrscr(); //It clears the screen.

printf(“enter the number=”);

scanf(“%d”,&n);

temp=n;

while(n > 0)

{

r=n;

sum=sum+(r\*r\*r);

n=n/10;

}

if(temp==sum)

printf(“armstrong number “);

else

printf(“not armstrong number”);

getch(); //It reads a character from the keyword.

}

Copy code

Q50. Write a program to check the palindrome number in C.

Ans. The program to check palindrome number in C is as follows:

#include < stdio.h >

#include < conio.h >

main()

{

int n,r,sum=0,temp;

clrscr();

printf(“enter the number=”);

scanf(“%d”,&n);

temp=n;

while(n > 0)

{

r=n;

sum=(sum\*10)+r;

n=n/10;

}

if(temp==sum)

printf(“palindrome number “);

else

printf(“not palindrome”);

getch();

}

Copy code

Q51. What are reserved keywords? How many reserved keywords are there in C?

Ans. Reserved keywords are those keywords that have predefined meanings and cannot be used as a variable name. Such keywords are restricted for general use while writing a program. There are 32 reserved keywords in C programming language:

Reserved Keywords

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

voidvolatilewhile

Q52. What is a union?

Ans. A union is a user-defined data type that enables you to store multiple types of data in a single unit or the same memory location. While you can define a union with different members, only one member can hold a value at any given time. Also, it does not hold the sum of the memory of all members and holds the memory of the largest member only.

Q53. What happens if a header file is included twice?

Ans. If a header file is included twice, the compiler will process its contents twice, resulting in an error. You can use a guard(#) to prevent a header file from being included multiple times during the compilation process. Thus, even if a header file with proper syntax is included twice, the second one will get ignored.

Q54. Can a C program compile without the main() function?

Ans. Yes, a C program can be compiled without the main() function. However, it will not execute without the main() function.

Q55. What is the difference between static memory allocation and dynamic memory allocation?

Ans. The differences between static memory allocation and dynamic memory allocation are:

Static Memory Allocation Dynamic Memory Allocation

It is done before the program execution. It takes place during program execution.

Variables get allocated permanently. Variables get allocated when the program unit gets active.

Less efficient. More efficient.

It uses a stack for managing the static allocation of memory. It uses heap for managing the dynamic allocation of memory.

No memory re-usability. It allows memory re-usability.

Execution is faster. Execution is slower.

Memory is allocated at compile time. The allocation of memory is done at run time.

Memory size can not be modified while execution. Example: Array Memory size can be modified while execution. Example: Linked List

Q56. What do you mean by a memory leak in C?

Ans. A memory leak is a kind of resource leak that happens when programmers create a memory in heap and forget to delete it. Thus, the memory which is no longer needed remains undeleted. It may also occur when an object is inaccessible by running code but it is still stored in memory. A memory leak can result in additional memory usage and can affect the performance of a program.

Q57. What do you understand by while(0) and while(1)?

Ans. In while(1) and while(0), 1 means TRUE or ON and 0 means FALSE or OFF.

while(0) means that the looping conditions will always be false and the code inside the while loop will not be executed. On the other hand, while(1) is an infinite loop. It runs continuously until it comes across a break statement mentioned explicitly.

Q58. Explain the difference between prefix increment and postfix increment.

Ans. Both prefix increment and postfix increment do what their syntax implies, i.e. increment the variable by 1.

The prefix increment increments the value of the variable before the program execution and returns the value of a variable after it has been incremented. The postfix increment, on the other hand, increments the value of the variable after the program execution and returns the value of a variable before it has been incremented.

++a <- Prefix increment

a++ <- Postfix increment

Q59. What is the difference between a null pointer and a void pointer?

Ans. A null pointer is one that does not point to any valid location and its value isn’t known at the time of declaration.

Syntax: <data type> \*<variable name> = NULL;

On the other hand, void pointers are generic pointers that do not have any data type associated with them. They can contain the address of any type of variable. Void pointers can point to any data type.

Syntax: void \*<data type>;

Q60. Explain the difference between Call by Value and Call by Reference?

Ans. Call by Value sends the value of a variable as a parameter to a function. Moreover, the value in the parameter is not affected by the operation that takes place.

Call by Reference, on the other hand, passes the address of the variable, instead of passing the values of variables. In this, values can be affected by the process within the function.

Q61. How can the scope of a global symbol be resolved in C?

Ans. The scope of a global symbol can be resolved by using the extern storage, which extends the visibility or scope of variables and functions. Since C functions are visible throughout the program by default, you don’t need to use it with function declaration or definition.

Q62. What is the difference between actual parameters and formal parameters?

Ans. The differences between actual parameters and formal parameters are:

Actual Parameters Formal Parameters

They are included at the time of function call. They are included at the time of the definition of the function.

Actual Parameters do not require data type but the data type should match with the corresponding data type of formal parameters. Data types need to be mentioned.

These can be variables, expressions, and constant without data types. These are variables with their data type.

Q63. Explain modular programming.

Ans. Modular programming is an approach that focuses on dividing an entire program into independent and interchangeable modules or sub-programs, such as functions and modules for achieving the desired functionality. It separates the functionality in such a manner that each sub-program contains everything necessary to execute just one aspect of the desired functionality.

Q64. What is a sequential access file?

Ans. As the name suggests, a sequential access file is used to store files sequentially, i.e. one data is placed into one file after another. It means that if you have to access files you will have to check each file sequentially (reading one data at a time) until your desired file is reached. It is more limitations as access time will be very high and storage cost is high.

Q65. What is the difference between const char\* p and char const\* p?

Ans. const char\* p is a pointer to a constant character whereas char const\* p is a constant pointer to a non-constant character. In const char\* p, we cannot change the value pointed by ptr. However, we can the pointer. In char \*const ptr, we cannot change the pointer p. However, the value pointed by ptr can be changed.

Q66. Can we use the ‘if’ function to compare strings?

Ans. No, we cannot use the ‘if’ function to compare two strings. The ‘if’ function compares numerical and single character values. We can use the ‘strcmp’ function to compare two strings character by character.

Q67. What is the use of toupper() function in C?

Ans. The toupper() function in C converts the lowercase alphabet to uppercase. We define the toupper() function using the ctype.h header file.

Syntax:

int toupper(int ch);

Q68. Explain the newline escape sequence.

Ans. the newline escape sequence denoted by \n is used to insert a newline character into a string.

Q69. Explain the # pragma directive in C.

Ans. # pragma is a special purpose directive that is used to turn on or off certain features. Some of the #pragma directives are:

#pragma startup: It is used to specify the functions that are needed to run before the program starts.

#pragma exit: It is used to specify the functions that are needed to run just before program exit.

#pragma warn: It is used to hide the warning messages which are displayed during compilation.

#pragma GCC poison: This directive is used to remove an identifier completely from the program.

#pragma GCC dependency: This directive allows you to check the relative dates of the current file and another file.

#pragma once: It allows the current source file to be included only once in a single compilation.

Q70. What is variable initialization in C Programming?

Ans. Variable initialization refers to the process of assigning a value to the variable before it is used in the program. Using variables without initialization can result in unexpected outputs. There are two types of variable initialization:

Static Initialization: variable is assigned a value in advance and it acts as a constant.

Dynamic Initialization: variable is assigned a value at run time and can be altered every time the program is being run.

Syntax of the variable initialization:

data\_type variable\_name=constant/literal/expression;

or

variable\_name=constant/literal/expression;

Q71. What is the difference between Source Code and Object Code?

Ans. The differences between source code and object code are:

Source Code Object Code

It is created by a programmer. Object code refers to the output, which is produced when the Source Code is compiled with a C compiler.

Source code is high-level code. It is a low-level code.

Source code is written in plain text by using some high-level programming language like C, C++, Java, or Python. It is written in machine language.

It is understandable by humans but not directly understandable by machines. Object code is machine-understandable but not human-understandable.

Source code is the input to the compiler or any other translator. It is the output of the compiler or any other translator.

It can be easily modified. Object code can not be modified.

Q72. In C programming, how can you determine if a number is odd or even?

Ans. In C programming, we can determine if a number is even or odd by dividing it by 2. If the number is exactly divisible by 2 (the remainder is 0) then it is an even number else it is odd.

Program to check if a number is even or odd using Bitwise Operator:

#include < stdio.h >

int main() {

int num;

printf(“Enter an integer: “);

scanf(“%d”, &num);

// true if num is perfectly divisible by 2

if(num % 2 == 0)

printf(“%d is even.”, num);

else

printf(“%d is odd.”, num);

return 0;

}

Copy code

Output:

Enter an integer: 77

77 is odd.

Check Out the Best Online Courses

Q73. Write a program to swap two numbers without the use of the third variable.

Ans. We can swap two numbers without the use of the third variable by two methods:

Method 1: Using sum and subtraction

#include < stdio.h >

int main()

{

int a=10, b=20;

printf(“Before swap a=%d b=%d”,a,b);

a=a+b;//a=30 (10+20)

b=a-b;//b=10 (30-20)

a=a-b;//a=20 (30-10)

printf(“\nAfter swap a=%d b=%d”,a,b);

return 0;

}

Copy code

Method 2: Using multiplication and division

#include < stdio.h >

#include < stdlib.h >

int main()

{

int a=10, b=20;

printf(“Before swap a=%d b=%d”,a,b);

a=a\*b;//a=200 (10\*20)

b=a/b;//b=10 (200/20)

a=a/b;//a=20 (200/10)

system