C Programming viva Questions

1. What is C language?

C language Tutorial with programming approach for beginners and professionals, helps you to understand the C language tutorial easily. Our C tutorial explains each topic with programs.

The C Language is developed for creating system applications that directly interact with the hardware devices such as drivers, kernels, etc.

C programming is considered as the base for other programming languages, that is why it is known as mother language.

2. Why C is a procedural language?

A procedure is known as a function, method, routine, subroutine, etc. A procedural language **specifies a series of steps for the program to solve the problem**.

A procedural language breaks the program into functions, data structures, etc.

C is a procedural language. In C, variables and function prototypes must be declared before being used.

3. Why C is a structured programming language?

A structured programming language is a subset of the procedural language. **Structure means to break a program into parts or blocks** so that it may be easy to understand.

In the C language, we break the program into parts using functions. It makes the program easier to understand and modify.

4. What are static variables in C ?

C defines another class of variables called static variables. Static variables are of two types:

- **Static variables** that are declared within a function (function static variables). These variables retain their values from the previous call, i.e., the values which they had before returning from the function.
- **File static variables.** These variables are declared outside any function using the keyword static. File static variables are accessible only in the file in which they are declared. This case arises when multiple files are used to generate one executable code.

```
#include ‹stdio.h›
void PrintCount()
```

```
{
 static int Count = 1;
 //Count is initialized only on the first call
 printf( "Count = %d\n", Count);
 Count = Count + 1;
 //The incremented value of Count is retained
}
int main()
{
 PrintCount();
 PrintCount();
 PrintCount();
 return 0;
}
```

OutPut:

```
Count = 1
Count = 2
Count = 3
```

5. What is a scope resolution operator in C ?

The scope resolution operator **(::)** is useful in C programming when both global and local variables have the same name, a statement using that name will access the local variable (more precisely, it will access the variable of that name in the innermost block containing that statement). The scope resolution operator represented as :: (a double colon) can be used to select the global variable explicitly.

Consider the example below.

```
#include<stdio.h>
int x = 10;
int main()
{
  int x = 20;
  printf("%d\n",x);
  printf("%d\n", ::x);
  return 0;
}
```

OutPut:

```
20
10
```

6. What is register variable in C language?

C language allows the use of the prefix register in primitive variable declarations. Such variables are called register variables and are stored in the

registers of the microprocessor. The number of variables which can be declared register are limited. If more variables are declared register variables, they are treated as auto variables. A program that uses register variables executes faster as compared to a similar program without register variables. It is possible to find out the allocation of register variables only by executing and comparing the performance with respect to the time taken by the program (perceptible in large programs). It is the responsibility of the compiler to allow register variables.

In case the compiler is unable to do so, these variables are treated as auto variables. Loop indices, accessed more frequently, can be declared as register variables. For example, index is a register variable in the program given below.

```
//Illustration of register variables in C language

#include <stdio.h>
#include <string.h>
int main()
{
 char Name[30];
 register int Index;

 printf( "Enter a string: ");
 gets( Name );

 printf( "The reverse of the string is : ");
 for( Index = strlen( Name ) - 1; Index >= 0; Index-- )
        printf( "%c", Name[Index]);
 printf( "\n" );
 return 0;
}
```

OutPut:

```
Enter a string: Apple is Red
The reverse of the string is : deR si elppA
```

7. What is a structure in C Language. How to initialise a structure in C?

A structure is a composite data type declaration that defines a physically grouped list of variables to be placed under one name in a block of memory, allowing the different variables to be accessed via a single pointer.

Defining a structure in C: In C language a structure is defined using struct keyword followed by variable or pointer name below is the basic syntax for declaring a structure in C language.

```
struct tag_name {
  type member1;
  type member2;
```

```
  /* declare as many members as desired, but the entire structure size must
be known to the compiler. */
};
```

8. What is an auto variable in C ?

All variables declared within a function are auto by default. The extent of a variable is defined by its scope. Variables declared auto can only be accessed only within the function or the nested block within which they are declared. They are created when the block is entered into and destroyed when it is exited. i.e., memory is allocated automatically upon entry to a block and freed automatically upon exit from the block and note that default value is a garbage value.

9. What is the use of extern in C?

variables is not recommended, as function independence is one of the basic idea of modular programming. Global variables should be used only when it is inevitable (i.e., when all the functions need a particular variable).

When program spans across different files and we want to have a global variable accessible to all functions in these files, the keyword extern should be used before the data type name in the declarations in all these files where it is accessed except one. The linker requires that only one of these files have the definition of the identifier. Global variables definitions can occur in one file only.

Consider the following program example to understand better the use of keyword extern,

```
/* file sub_file.c Defines the function sub_func */
extern int globalVar;

void inc_global()
{
  globalVar++;
}
```

The code of the second file, mainfile.c is given below,

```
/*file mainfile.c Defines the function main and uses the function
inc_global. Compile with cc mainfile.c sub_file.c */
int globalVar;

void main()
{
 /*Assign something to globalVar*/
 globalVar = 10;

printf( " globalVar before calling = %d\n", globalVar);
```

```
inc_global();
printf( " globalVar after calling = %d\n", globalVar);
}
```

Observe the integer **globalVar** has been declared in the sub_file.c as **extern globalVar;**

10. What is the difference between #include ‹header file› and #include "header file" ?

#include "myheaderfile.h"

#include ‹headerfile.h›

The difference between these two is the location the compiler searches for the header file to be included. If the file name is enclosed in quotes, the compiler searches in the same directory in which the file is included. This method is normally used to include programmer defined headers. If the file name is enclosed in brackets which are used for standard library headers; the search is performed in an implementation dependent manner, normally through pre-designated directories. These both file inclusions one using angle brackets and the other using quotes in an include statement can be used in C and C++.

It does:

"path/myheaderfile.h" is short for ./path/myheaderfile.h

is short for /path/headerfile.h

11. What are Derived data types in C ?

Derived data types are object types which are aggregates of one or more types of basic data types. below are the list of derived datatype in C Language.

- Pointer types
- Array types
- Structure types
- Union types
- Function types

12. Explain C pre-processor ?

The C preprocessor or cpp is the macro preprocessor for the C and C++ computer programming languages. The preprocessor provides the ability for the inclusion of header files, macro expansions, conditional compilation, and line control.

13. What is recursion in C ?

Recursion: A function, which calls itself, recursion is simple to write in program but takes more memory space and time to execute.

Advantages of using recursion:-

- Avoid unnecessary calling of functions
- Substitute for iteration
- Useful when applying the same solution

Factorial program in c using Recursion

```
#include

int factorial(unsigned int i) {

  if(i <= 1) {
    return 1;
  }
  return i * factorial(i - 1);
}

int  main() {
  int i = 15;
  printf("Factorial of %d is %d\n", i, factorial(i));
  return 0;
}
```

14. Explain Enumerated types in C language?

Enumerated types are used to define variables that can only assign certain discrete integer values throughout the program.

Enumeration variables are variables that can assume values symbolically

Declaration and usage of an Enumerated variable.

```
enum boolean{
false;
true;
};
enum boolean
```

15. Differentiate call by value and call by reference?

Call by value:
A process in which the values of the actual parameters sent by the calling function are copied to the formal parameters of the called function.

Call by reference:
A process in which the parameters of a calling function are passed to the parameters of the called function using an address.

16. List some basic data types in C ?

In Programming Languages data types are used to define a variable before its use.

Below are list of some basic data types in C language.

- **Integer:** used to define integer numbers, denoted as int
- **Floating point:** decimal
- **Character:** defines character
- **Void:** void type has no value and only one operation: assignment. Plays a role of generic data type.

17) What is typecasting?

Typecasting is a way to convert a variable/constant from one type to another data type.

18) Explain about block scope in C ?

A block is defined as a sequence of statements enclosed between curly braces, { and }. In other words, a block is a compound statement. For example, a function body is a block, because it is simply a sequence of statements enclosed within curly braces. Blocks of statements in if statements and loops. All these blocks of statements actually follow the same rules.

Consider the example shown below, the variable j declared in both main and the user defined function other func. Accessing j uses the local declaration in the called function.

```
//Illustration of block scope
#include <stdio.h>
void OtherFunc( void )
{
        int i = 10;
        printf(" Inside the function OtherFunc, the value of i is %d\n", i);
}
int main()
{
  int i = 20;
  printf(" Inside the function main, the value of i is %d\n", i);
  OtherFunc();
  return 0;
}
```

Output:

Inside the function main, the value of i is 20
Inside the function OtherFunc, the value of i is 10

19) Explain continue keyword in C.

Continue is a jump statement that transfers control back to the beginning of the loop, bypassing any statements that are not yet executed. It can be used only in an iteration statement.

20) Compare array data type to pointer data type.

Array is a collection of variables of same type that are referred through a common name and a **pointer** is a variable that holds a memory address. pointers can point to array and array to pointers

21) What are bit fields in C ?

Bit fields are used to store multiple, logical, neighbouring bits, where each of the sets of bits and single bits can be addressed.

22) What are different storage class specifiers in C

auto, register, static, extern are storage class specifiers in C

23) What is NULL pointer?

NULL is used to indicate that the pointer doesn't point to a valid location. Ideally, we should initialize pointers as NULL if we don't know their value at the time of declaration. Also, we should make a pointer NULL when memory pointed by it is deallocated in the middle of a program.

24) Explain main function in C ?

Main Function is the function where every C program begins executing. it will usually call other functions to help perform its job, some that you wrote, and others from libraries that are provided for you.

25) What does printf does?

The printf is a library function that prints output. requires the \n newline character, even separately like printf ("\n");. The first argument is the string of characters to be printed, with each % indicating where one of the other arguments is to be substituted, and in what form it is to be printed.

26) List some applications of C programming language?

Application of C Programming Language

- To develop embedded software
- It is to create a computer application

- It is effective to create a compiler for various computer languages to convert them into low-level language that is the machine understandable language.
- It can be used to develop an Operating system and UNIX is one which is developed by the C programming language.
- It is used for creating software for various applications and even hardware.

27. Explain Pointers in C programming?

Pointers are variables that are used to store addresses. The concept of the pointer is considered to be one of the difficult part of learning the C and C++ programming languages. There are several easy ways to write programs without pointers, but in case of dynamic memory allocation, the knowledge of pointers is a must.

Knowing about memory locations and addresses defined will enable you with the ideas of how every variable function in a program.

28. How can you find the exact size of a data type in C?

One can determine the exact size of a data type by using the sizeof operator. The storage size of the data type is obtained in bytes by using the syntax: sizeof**(data_type)**.

29. If the size of int data type is two bytes, what is the range of signed int data type?

The range of signed int data type if from **-32768** to **32767**

30. What do you understand by normalization of pointers?

Normalization is the process by which an address is converted to a form such that if two non-normalized pointers point to the same address, they both are converted to normalized form, thereby having a specific address

31. What is the best way to store flag values in a program?

Flag values are used to make decisions between two or more options during the execution of a program. Generally, flag values are small (often two) and it tries to save space by not storing each flag according to its own data type.

The best way to store flag values is to keep each of the values in their own integer variable. If there are large number of flags, we can create an array of characters or integers. We can also store flag values by using low-order bits more efficiently.

32. Explain bit masking in C?

Bit masking refers to selecting a particular set of bits from the byte(s) having many values of bits. Bit masking is used to examine the bit values and can be done by 'AND' operation of byte, bitwise.

33. Is there any demerits of using pointer?

Yes. As pointers have access to a particular memory location, the security level decreases and restricted memory areas can be accessed. Other demerits include memory holes, process and memory panics, et

34. Is there any data type in C with variable size?

Yes, Struct is one of the data type in C that have variable size. It is because the size of the structure depends on the fields which can be variable as set by the user.

35. What is a void pointer?

Void **pointer** is a generic pointer in programming. If the pointer type is unknown, we make use of the void pointer

36. When can you use a pointer with a function?

A pointer can be used with a function-

- When an address is to be passed to a function
- When an array element are to be accessed through a function. Passing base address will give access to the whole array.

37. When can a far pointer be used?

Sometimes the task we are required to do might not fit in the allocated data and code segments. Far pointers help to access rest of the memory inside a program. Far pointers are the information present outside the data segment (generally 64 kb). Such pointers are used when we need to access an address outside of the current segment.

38. In the case of character array what is the difference b/w strlen() and sizeof() ?

In character array strlen() returns the number of characters, while sizeof() returns the number of occupied bytes by character array.

Example

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

int main()
{
        char text[100]= "Sample test";
        printf("%d,%d", strlen(text), sizeof(text));
        return 0;
}
```

In this example **strlen(text)** will return 11 because total number of characters are 11 and **sizeof(text)** will return 100, because text variable will take 100 bytes in memory.

39. When is a switch statement can be better than an if statement?

If you have more than one condition to check on single variable or a single expression, then switch is better than if. In switch statement, program's execution jumps to the matching value if found. If you use if condition, it checks one by one condition. So it is highly recommended to use switch, if you have to check a variable/condition/expression with multiple values.

Example

```
if( errorCode == 0)
        printf("Error code is 0: reading failed error.\n");
if( errorCode == 1)
        printf ("Error code is 1: writing failed error.\n");
if( errorCode == 2)
        printf("Error code is 2: opening failed error.\n");
if( errorCode == 3)
        printf("Error code is 3: write protected.\n");
```

Here, errorCode variable is checking with values 0,1,2 and 3. In such case you can use switch.

```
switch(errorCode)
{
        case 0:
                printf("Error code is 0: reading failed error.\n");
                break;

        case 1:
                printf ("Error code is 1: writing failed error.\n");
                break;

        case 2:
                printf("Error code is 2: opening failed error.\n");
```

```
                break;

        case 3:
                printf("Error code is 3: write protected.\n");
                break;

        default:
                printf("Undefined error.\n");
}
```

40. The default case is necessary to use in switch statement?

No, it's not necessary to use the default. But using default is a good programming technique, that must be used. If variable or expression does not match with any given case values, default case executed.

41. Can we use default case anywhere in the switch statement block?

Yes, of course. You can use default case anywhere with in the switch statement. It is not necessary to write default case at the end of the cases.

42. What happens if break keyword is not included after the case body?

break is optional in switch statements, if break does not exist, program's execution will move to next case and executes statements written within that next case body.

```
1
2       int a=2;
        switch(a)
3       {
          case 1:
4           printf("one ");
5         case 2:
            printf("two ");
6         case 3:
            printf("three ");
7         default:
            printf("other ");
8       }
43.
```

Here output will be **two three other**.

43. How will you differentiate ++a and a++?

++ is an **increment operator**, which increments the the value of a variable. ++a is **pre increment**, where value will be incremented first and then expression evaluate. a++ is a **post increment**, where expression evaluates first and then value increments.

44. Can you make a variable const and volatile both?

Yes, it is possible that a variable can be const and volatile, the value of that variable will not change in the code, but it can be change outside the code due to hardware or other method. Hence volatile can be applied with const.

45. Can printf() function be used in if condition?

Yes, because printf() returns integer value (*that is total number of printed characters*). And In if condition, a non-zero value evaluated as TRUE, and zero evaluated as FALSE.

```
1    if( printf("Hello, World") )
2    {...}
```

46. Write a statement to check whether any number is EVEN or ODD ?

It can be done using **ternary (? :) operator**, here is a statement.

```
int num=113;

(num%2==0)?printf("EVEN"):printf("ODD");
```

47. What is the limitation of ternary operator, comparing with if statement?

In ternary operator, ? : (In case of TRUE or FALSE) only one C statement can use, but in if statement, we can use more than one statement.

48. What are the difference among **getch(),getche(),getchar()**?

These all functions are used to get a single character from keyboard, but there are following differences.

getch() -It takes input, but not echo the character on console, but requires enter key.

getche() -It takes input, but not echo character and does not requires enter key.

getchar() -It takes input, echo character and requires enter key.

49. Can a C variable/identifier name starts with a digit and what are the maximum length of a variable/identifier name?

No, variable/identifier starts with a character or underscore. There are 32 characters can be used as a variable/identifier name.

50. Difference b/w Entry controlled and Exit controlled loop?

In **Entry Controlled** Loop, loop body is checked after checking the test condition i.e. condition is checked first after that loop body will execute while

in **Exit Controlled** Loop, loop body will be executed first after that loop's test condition is checked.

Entry Controlled Loops are : for, while

Exit Controlled Loop is : do while

Consider the example [while loop]:

```
1       int main()
2       {
3          int loop;
4
5          loop=1;    //initialisation of loop counter
6          while(loop<=10)  // loop condition
7          {
8             printf("%d\n",loop);
9             ++loop; // loop counter increment
10         }
11         return 0;
12      }
```

Consider the example [for loop]:

```
1       int main()
2       {
3          int loop;
4          // initialisation, loop test condition & increment of loop counter
5          // with in for body
6          for(loop=1; loop<=10; ++loop)
7          {
8             printf("%d\n",loop);
9          }
10         return 0;
11      }
```

Consider the example [do...while loop]:

```
1       int main()
2       {
3          int loop;
4          loop=1; // initialisation of loop counter
5          do
6          {
7             printf("%d\n",loop);
8             ++loop; // loop counter increment
9          }while(loop<=10); // loop test condition
10         return 0;
11      }
```

In all above examples, for and while loop checks the condition first, but do while checks the condition after the execution of loop body.

51. Difference b/w sentinel control loop & counter control Loop?

In **Counter Controlled** loop, we know that exactly how many times loop body will be executed while in **Sentinel Controlled** loop we don't know about the loop recurrence, Execution of loop is based on condition not counter.

Consider the example:

```
1       /*****Counter Control loop*****/
2       for(i=0;i<10;i++)
3       {
4          //Body of the loop
5       }
6
7
8       /*****Sentinel Control loop*****/
9       while( n>0)
10      {
11         n=n/10;
12      }
```

52. Is nested loop possible?

Yes, it is possible. We can use loop with in the loop any number of times.

Consider the example:

```
1     for(i=1;i<=2;i++)
2     {
3        for(j=1;j<=10;j++)
4        {
5           printf("%d  ",j);
6        }
7        printf("\n");
8     }
```

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

53. What is fall down property?

In C Programming Language, switch case statement follows the fall down property. It means when case block is executed and break statement is not used after the block statements, then it will execute next case or default statements until break not reached or switch not finished.

Consider the example:

```
1       int i=2;
2       switch(i)
```

```
3       {
4           case 1:
5               printf("\none");
6           break;
7           case 2:
8               printf("\ntwo");
9           case 3:
10              printf("\nthree");
11          case 4:
12              printf("\nfour");
13          case 5:
14              printf("\nFive");
15          break;
16          default:
17              printf("\nWrong Choice");
18      }
```

```
Two
Three
Four
Five
```

Here 3, 4, 5 case are un-necessary are calling during fall down property.

54. Which loop statement is executed at least once even loop test condition if false?

do while loop executes once even loop test condition if false. Since **do while** is an exit controlled loop and in this type of loop, loop body execute once than loop test condition checks.

55. Can we use single statement in loop body without using curly braces "{...}"?

Yes, when loop body has single statement, than we can escape curly braces "{...}".

Consider the example:

```
1       for(i=1;i<=10;i++)
2               printf("\n%d ",i);
```

56. What are the jumping statements in C Language and how these works?

In C Programming, some special statements are used to transfer program control to one location to other location. There are following jumping statements are used in c:

1. goto
2. break
3. continue
4. return

1) goto

goto statement is used to jump program's control from one location to define label.

2) break

break statement is used in switch and loop statements, it is used to break the execution of the switch and loop's body and transfer control after the loop/switch statement.

3) continue

continue is used in looping statement, it transfer program's control in the starting of the loop's body.

4) return

Generally return statement is used in function's body, it transfers program's control from called to calling function.

57. What is infinite loop?

A loop which is never finished is known as **infinite loop**, it means the looping condition is always true, so that loop never terminate. Infinite loop is very useful in embedded systems.

58. Which loop is good for programming?

It depends on the programming need, we can not say which is best to use. Where condition checking is required before body execution, just use for or while. And when condition is not required to check before body execution, just use do while.

59. Can we use continue statement without using loop?

No, continue statement can only be used within the loops only, it can be any loop while, do while or for. If we use continue statement without using loop , there will be a compiler error "misplaced continue".

60. Can goto statement transfer program control from one function to another function ?

No, As we know that goto statement can transfer the program's control to specified label, But it has a limitation, as it can transfer program's control within a function only. Outside the function, control can not be transferred by goto statement.