

Q1. What are the basic Datatypes supported in C Programming Language?

Ans: The Datatypes in C Language are broadly classified into 4 categories. They are as follows:

- Basic Datatypes
- Derived Datatypes
- Enumerated Datatypes
- Void Datatypes

The Basic Datatypes supported in C Language are as follows:

Datatype Name	Datatype Size	Datatype Range
short	1 byte	-128 to 127
unsigned short	1 byte	0 to 255
char	1 byte	-128 to 127
unsigned char	1 byte	0 to 255
int	2 bytes	-32,768 to 32,767
unsigned int	2 bytes	0 to 65,535
long	4 bytes	-2,147,483,648 to 2,147,483,647
unsigned long	4 bytes	0 to 4,294,967,295
float	4 bytes	3.4E-38 to 3.4E+38
double	8 bytes	1.7E-308 to 1.7E+308
long double	10 bytes	3.4E-4932 to 1.1E+4932

Q2. What do you mean by Dangling Pointer Variable in C Programming?

Ans: A **Pointer** in C Programming is used to point the memory location of an existing variable. In case if that particular variable is deleted and the Pointer is still pointing to the same memory location, then that particular pointer variable is called as a **Dangling Pointer Variable**.

Q3. What do you mean by the Scope of the variable? What is the scope of the variables in C?

Ans: **Scope** of the variable can be defined as the part of the code area where the variables declared in the program can be accessed directly. In C, all identifiers are lexically (or statically) scoped.

Q4. What are static variables and functions?

Ans: The variables and functions that are declared using the keyword **Static** are considered as Static Variable and Static Functions. The variables declared using Static keyword will have their scope restricted to the function in which they are declared.

Q5. Differentiate between calloc() and malloc()

Ans: **calloc()** and **malloc()** are memory dynamic memory allocating functions. The only difference between them is that **calloc()** will load all the assigned memory locations with value 0 but **malloc()** will not.

Q6. What are the valid places where the programmer can apply Break Control Statement?

Ans: Break Control statement is valid to be used inside a loop and **Switch control statements**.

Q7. How can we store a negative integer?

Ans: To store a negative integer, we need to follow the following steps. Calculate the two's complement of the same positive integer.

Eg: 1011 (-5)

Step-1 – One's complement of 5: 1010

Step-2 – Add 1 to above, giving 1011, which is -5

Q8. Differentiate between Actual Parameters and Formal Parameters.

Ans: The Parameters which are sent from main function to the subdivided function are called as **Actual Parameters** and the parameters which are declared at the Subdivided function end are called as **Formal Parameters**.

Q9. Can a C program be compiled or executed in the absence of a main()?

Ans: The program will be compiled but will not be executed. To execute any C program, main() is required.

Q10. What do you mean by a Nested Structure?

Ans: When a data member of one structure is referred by the data member of another function, then the structure is called a **Nested Structure**.

Q11. What is a C Token?

Ans: *Keywords, Constants, Special Symbols, Strings, Operators, Identifiers* used in C program are referred to as **C Tokens**.

Q12. What is Preprocessor?

Ans: A Preprocessor Directive is considered as a built-in predefined function or macro that acts as a directive to the compiler and it gets executed before the actual C Program is executed.

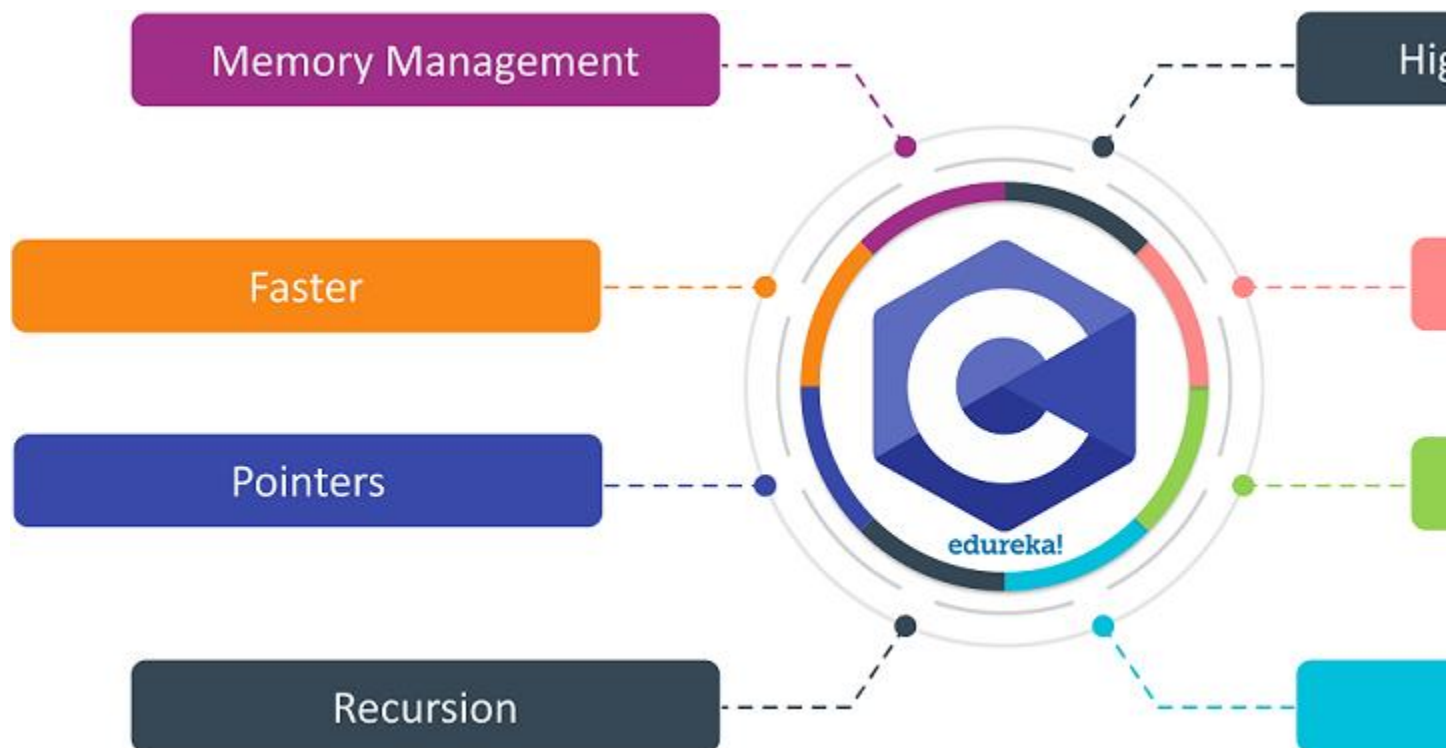
In case you are facing any challenges with these C Programming Interview Questions, please write your problems in the comment section below.

Q13. Why is C called the Mother of all Languages?

Ans: C introduced many core concepts and data structures like **arrays, lists, functions, strings**, etc. Many languages designed after C are designed on the basis of C Language. Hence, it is considered as the mother of all languages.

Q14. Mention the features of C Programming Language.

Ans:



Q15. What is the purpose of printf() and scanf() in C Program?

Ans: **printf()** is used to print the values on the screen. To print certain values, and on the other hand, **scanf()** is used to scan the values. We need an appropriate datatype format specifier for both printing and scanning purposes. For example,

- **%d:** It is a datatype format specifier used to print and scan an **integer** value.
- **%s:** It is a datatype format specifier used to print and **scan** a string.
- **%c:** It is a datatype format specifier used to display and scan a **character** value.
- **%f:** It is a datatype format specifier used to display and scan a **float** value.

Q16. What is an array?

Ans. The array is a simple data structure that stores multiple elements of the same datatype in a reserved and sequential manner. There are three types of arrays, namely,

- One Dimensional Array
- Two Dimensional Array
- Multi-Dimensional Array

Q17. What is /0 character?

Ans: The Symbol mentioned is called a **Null Character**. It is considered as the terminating character used in strings to notify the end of the string to the compiler.

Q18. What is the main difference between the Compiler and the Interpreter?

Ans: Compiler is used in C Language and it translates the complete code into the Machine Code in one shot. On the other hand, Interpreter is used in Java Programming Language and other high-end programming languages. It is designed to compile code in line by line fashion.

Q19. Can I use int datatype to store 32768 value?

Ans: No, Integer datatype will support the range between **-32768 and 32767**. Any value exceeding that will not be stored. We can either use **float** or **long int**.

Intermediate C Programming Interview Questions



Q20. How is a Function declared in C Language?

Ans: A function in C language is declared as follows,

```
1 return_type function_name(formal parameter list)
  {
```

```
2     Function_Body;
3 }
4
```

Q21. What is Dynamic Memory allocation? Mention the syntax.

Ans: Dynamic Memory Allocation is the process of allocating memory to the program and its variables in runtime. Dynamic Memory Allocation process involves three functions for allocating memory and one function to free the used memory.

malloc() – Allocates memory

Syntax:

```
1 ptr = (cast-type*) malloc(byte-size);
```

calloc() – Allocates memory

Syntax:

```
1 ptr = (cast-type*) calloc(n, element-size);
```

realloc() – Allocates memory

Syntax:

```
1 ptr = realloc(ptr, newsize);
```

free() – Deallocates the used memory

Syntax:

```
1 free(ptr);
```

Q22. What do you mean by Dangling Pointer Variable in C Programming?

Ans: A **Pointer** in C Programming is used to point the memory location of an existing variable. In case if that particular variable is deleted and the Pointer is still pointing to the same memory location, then that particular pointer variable is called as a **Dangling Pointer Variable**.

Q23. Where can we not use &(address operator in C)?

Ans: We cannot use **&** on **constants** and on a variable which is declared using the **register storage** class.

Q24. Write a simple example of a structure in C Language

Ans: Structure is defined as a user-defined data type that is designed to store multiple data members of the different data types as a single unit. A structure will consume the memory equal to the summation of all the data members.

```
1 struct employee
2 {
3     char name[10];
4     int age;
5 }e1;
6 int main()
7 {
8     printf("Enter the name");
9     scanf("%s",e1.name);
10    printf("\n");
11    printf("Enter the age");
12    scanf("%d",&e1.age);
13    printf("\n");
14    printf("Name and age of the employee:
15    %s,%d",e1.name,e1.age);
16    return 0;
17 }
```

Q25. Differentiate between call by value and call by reference.

Ans:

Factor	Call by Value	Call by Reference
Safety	Actual arguments cannot be	Operations are performed on

	changed and remain safe	actual arguments, hence not safe
Memory Location	Separate memory locations are created for actual and formal arguments	Actual and Formal arguments share the same memory space.
Arguments	Copy of actual arguments are sent	Actual arguments are passed

//Example of Call by Value method

```

1  #include<stdio.h>;
2  void change(int,int);
3  int main()
4  {
5      int a=25,b=50;
6      change(a,b);
7      printf("The value assigned to a is: %d",a);
8      printf("\n");
9      printf("The value assigned to of b is:
10 %d",b);
11     return 0;
12 }
13 void change(int x,int y)
14 {
15     x=100;
16     y=200;
17 }

```

//Output

The value assigned to of a is: 25
The value assigned to of b is: 50

//Example of Call by Reference method

```

1  #include<stdio.h>;
2  void change(int*,int*);
3  int main()
4  {
5      int a=25,b=50;

```

```

6     change(&a, &b);
7     printf("The value assigned to a is: %d",a);
8     printf("\n");
9     printf("The value assigned to b is: %d",b);
10    return 0;
11 }
12 void change(int *x,int *y)
13 {
14     *x=100;
15     *y=200;
16 }

```

//Output

The value assigned to a is: 100
The value assigned to b is: 200

In case you are facing any challenges with these C Programming Interview Questions, please write your problems in the comment section below.

Q26. Differentiate between getch() and getche().

Ans: Both the functions are designed to read characters from the keyboard and the only difference is that

getch(): reads characters from the keyboard but it does not use any buffers. Hence, data is not displayed on the screen.

getche(): reads characters from the keyboard and it uses a buffer. Hence, data is displayed on the screen.

//Example

```

1  #include<stdio.h>;
2  #include<conio.h>;
3  int main()
4  {
5      char ch;
6      printf("Please enter a character ");
7      ch=getch();
8      printf("\nYour entered character is %c",ch);
9      printf("\nPlease enter another character ");

```

```

10     ch=getche();
11     printf("\nYour new character is %c",ch);
12     return 0;
13 }

```

//Output

Please enter a character
 Your entered character is x
 Please enter another character z
 Your new character is z

Q27. Explain toupper() with an example.

Ans. toupper() is a function designed to convert lowercase words/characters into upper case.

//Example

```

#include<stdio.h>
1  #include<ctype.h>
2  int main()
3  {
4      char c;
5      c=a;
6      printf("%c after conversions  %c", c,
7  toupper(c));
8      c=B;
9      printf("%c after conversions  %c", c,
  toupper(c));

```

//Output:

a after conversions A
 B after conversions B

Q28. Write a code to generate random numbers in C Language.

Ans: Random numbers in C Language can be generated as follows:

```

1  #include<stdio.h>;
2  #include<stdlib.h>;
3  int main()
4  {
5      int a,b;
6      for(a=1;a<=10;a++)
7      {
8          b=rand();
9          printf("%dn",b);
10     }
11     return 0;
12 }

```

//Output

```

1987384758
2057844389
3475398489
2247357398
1435983905

```

Q29. Can I create a customized Head File in C language?

Ans: It is possible to create a new header file. Create a file with function prototypes that need to be used in the program. Include the file in the '#include' section in its name.

Q30. What do you mean by Memory Leak?

Ans: Memory Leak can be defined as a situation where programmer allocates dynamic memory to the program but fails to free or delete the used memory after the completion of the code. This is harmful if daemons and servers are included in the program.

```

1  #include<stdio.h>;
2  #include<stdlib.h>;
3  int main()
4  {

```

```

5     int* ptr;
6     int n, i, sum = 0;
7     n = 5;
8     printf("Enter the number of elements: %dn",
9     n);
10    ptr = (int*)malloc(n * sizeof(int));
11    if (ptr == NULL)
12    {
13        printf("Memory not allocated.n");
14        exit(0);
15    }
16    else
17    {
18        printf("Memory successfully allocated
19 using malloc.n");
20        for (i = 0; i<= n; ++i)
21        {
22            ptr[i] = i + 1;
23        }
24        printf("The elements of the array
25 are: ");
26        for (i = 0; i<=n; ++i)
27        {
28            printf("%d, ", ptr[i]);
29        }
30        return 0;
31    }

```

//Output

Enter the number of elements: 5
Memory successfully allocated using malloc.
The elements of the array are: 1, 2, 3, 4, 5,

In case you are facing any challenges with these C Programming Interview Questions, please write your problems in the comment section below.

Q31. Explain Local Static Variables and what is their use?

Ans: A local static variable is a variable whose life doesn't end with a function call where it is declared. It extends for the lifetime of the complete

program. All calls to the function share the same copy of local static variables.

```
1  #include<stdio.h>
2  void fun()
3  {
4      static int x;
5      printf("%d ", x);
6      x = x + 1;
7  }
8  int main()
9  {
10     fun();
11     fun();
12     return 0;
13 }
```

//Output

0 1

Q32. What is the difference between declaring a header file with < > and " "?

Ans: If the Header File is declared using < > then the compiler searches for the header file within the Built-in Path. If the Header File is declared using " " then the compiler will search for the Header File in the current working directory and if not found then it searches for the file in other locations.

Q33. When should we use the register storage specifier?

Ans: We use Register Storage Specifier if a certain variable is used very frequently. This helps the compiler to locate the variable as the variable will be declared in one of the CPU registers.

Q34. Which statement is efficient and why? x=x+1; or x++; ?

Ans: `x++`; is the most efficient statement as it just a single instruction to the compiler while the other is not.

Q35. Can I declare the same variable name to the variables which have different scopes?

Ans: Yes, Same variable name can be declared to the variables with different variable scopes as the following example.

```
1  int var;  
2  void function()  
3  {  
4      int variable;  
5  }  
6  int main()  
7  {  
8      int variable;  
9  }
```

Q36. Which variable can be used to access Union data members if the Union variable is declared as a pointer variable?

Ans: Arrow Operator(`->`) can be used to access the data members of a Union if the Union Variable is declared as a pointer variable.

Q37. Mention File operations in C Language.

Ans: [Basic File Handling Techniques in C](#), provide the basic functionalities that user can perform against files in the system.

Function	Operation
<code>fopen()</code>	To Open a File
<code>fclose()</code>	To Close a File
<code>fgets()</code>	To Read a File
<code>fprint()</code>	To Write into a File

In case you are facing any challenges with these C Programming Interview Questions, please write your problems in the comment section below.

Q38. What are the different storage class specifiers in C?

Ans: The different storage specifiers available in C Language are as follows:

- **auto**
- **register**
- **static**
- **extern**

Q39. What is typecasting?

Ans: Typecasting is a process of converting one data type into another is known as typecasting. If we want to store the floating type value to an int type, then we will convert the data type into another data type explicitly.

Syntax:

```
1 (type_name) expression;
```

Q40. Write a C program to print hello world without using a semicolon (;).

Ans:

```
1 #include<stdio.h>;
2 voidmain()
3 {
4     if(printf("hello world")){}
5 }
```

//Output:

hello world

Q41. Write a program to swap two numbers without using the third variable.

Ans:

```
#include<stdio.h>
1  #include<conio.h>
2  main()
3  {
4      int a=10, b=20;
5      clrscr();
6      printf("Before swapping a=%d
7  b=%d",a,b);
8      a=a+b;
9      b=a-b;
10     a=a-b;
11     printf("\nAfter swapping a=%d
12 b=%d",a,b);
13     getch();
    }
```

//Output

Before swapping a=10 b=20
After swapping a=20 b=10

Advanced C Programming Interview Questions



Q42. How can you print a string with the symbol % in it?

Ans: There is no escape sequence provided for the symbol % in C. So, to print % we should use '%%' as shown below.

```
1 printf("there are 90%% chances of rain tonight");
```

Q43. Write a code to print the following pattern.

```
1
12
123
1234
12345
```

Ans: To print the above pattern, the following code can be used.

```
1  #include<stdio.h>;
2  int main()
3  {
4      for(i=1;i<=5;i++)
5      {
6          for(j=1;j<=5;j++)
```

```

7         {
8             print("%d",j);
9         }
10        printf("\n");
11    }
12    return 0;
13 }

```

Q44. Explain the # pragma directive.

Ans: The following points explain the Pragma Directive.

- This is a preprocessor directive that can be used to turn on or off certain features.
- It is of two types #pragma startup, #pragma exit and pragma warn.
- #pragma startup allows us to specify functions called upon program startup.
- #pragma exit allows us to specify functions called upon program exit.
- #pragma warn tells the computer to suppress any warning or not.

Q45. How can you remove duplicates in an array?

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

```

1  #include <stdio.h>;
2  int main()
3  {
4      int n, a[100], b[100], calc = 0, i, j, count;
5      printf("Enter no. of elements in array.\n");
6      scanf("%d", &n);
7      printf("Enter %d integers\n", n);
8      for (i = 0; i < n; i++)
9          scanf("%d", &a[i]);
10     for (i = 0; i < n; i++)
11     {
12         for (j = 0; j < calc; j++)
13         {

```

```

14             if(a[i] == b[j])
15                 break;
16             }
17             if (j== calc)
18             {
19                 b[count] = a[i];
20                 calc++;
21             }
22         }
23         printf("Array obtained after removing
24 duplicate elementsn");
25         for (i = 0; i<calc; i++)
26         {
27             printf("%dn", b[i]);
28         }
29         return 0;
    }

```

//Output

Enter no. of elements in array. 5

Enter 5 integers

12

11

11

10

4

Array obtained after removing duplicate elements

12

11

10

4

Q46. What is Bubble Sort Algorithm? Explain with a program.

Ans: Bubble sort is a simple sorting algorithm that repeatedly steps through the list, compares adjacent elements and swaps them if they are in the wrong order. The pass through the list is repeated until the list is sorted.

The following code executes Bubble Sort.

```
1  int main()
2  {
3      int array[100], n, i, j, swap;
4      printf("Enter number of elements\n");
5      scanf("%d", &n);
6      printf("Enter %d Numbers:\n", n);
7      for(i = 0; i<n; i++)
8          scanf("%d", &array[i]);
9      for(i = 0; i<n - 1; i++)
10         {
11             for(j = 0; j < n-i-1; j++) {
12                 if(array[j]>array[j+1])
13                     {
14                         swap=array[j];
15                         array[j]=array[j+1];
16                         array[j+1]=swap;
17                     }
18             }
19         }
20     printf("Sorted Array:\n");
21     for(i = 0; i < n; i++)
22         printf("%d\n", array[i]);
23     return 0;
24 }
```

Q47. What is Round-robin algorithm? Write a code for Round Robin Scheduling.

Ans: Round-robin Algorithm is one of the algorithms employed by process and network schedulers in computing in order to evenly distribute resources in the system.

The following code will execute Round Robin Scheduling

```
1  #include<stdio.h>
2
3  int main()
4  {
```

```

5     inti, limit, total = 0, x, counter = 0, time_quantum;
6     int wait_time = 0, turnaround_time = 0, arrival_time[10]
7     temp[10];
8     float average_wait_time, average_turnaround_time;
9     printf("nEnter Total Number of Processes:t");
10    scanf("%d", &limit);
11    x = limit;
12    for(i = 0; i<limit; i++)
13    {
14        printf("nEnter Details of Process[%d]n", i + 1);
15        printf("Arrival Time:t");
16        scanf("%d", &arrival_time[i]);
17        printf("Burst Time:t");
18        scanf("%d", &burst_time[i]);
19        temp[i] = burst_time[i];
20    }
21
22    printf("nEnter Time Quantum:t");
23    scanf("%d", &time_quantum);
24    printf("nProcess IDttBurst Timet Turnaround Timet Waiti
25    for(total = 0, i = 0; x != 0;)
26    {
27        if(temp[i] <= time_quantum && temp[i]
28        {
29            total = total + temp[i];
30            temp[i] = 0;
31            counter = 1;
32        }
33        else if(temp[i]>0)
34        {
35            temp[i] = temp[i] - time_quantum;
36            total = total + time_quantum;
37        }
38        if(temp[i] == 0 && counter == 1)
39        {
40            x--;
41            printf("nProcess[%d]tt%dt %dt %d", i + 1
42            total - arrival_time[i], total - arrival_time[i] - burst_time
43            wait_time = wait_time + total - arrival_tim
44            burst_time[i];

```

```

45         turnaround_time = turnaround_time + total -
46         counter = 0;
47     }
48     if(i == limit - 1)
49     {
50         i = 0;
51     }
52     else if(arrival_time[i + 1] <= total)
53     {
54         i++;
55     }
56     else
57     {
58         i = 0;
59     }
60 }
61
62 average_wait_time = wait_time * 1.0 / limit;
63 average_turnaround_time = turnaround_time * 1.0 / limit;
64 printf("\nAverage Waiting Time:t%f", average_wait_time)
printf("\nAvg Turnaround Time:t%fn", average_turnaround_
return 0;
}

```

//Output

C:\WINDOWS\SYSTEM32\cmd.exe

```
Enter Total Number of Processes:      4

Enter Details of Process[1]
Arrival Time:    0
Burst Time:      4

Enter Details of Process[2]
Arrival Time:    1
Burst Time:      7

Enter Details of Process[3]
Arrival Time:    2
Burst Time:      5

Enter Details of Process[4]
Arrival Time:    3
Burst Time:      6

Enter Time Quantum:    3

Process ID          Burst Time          Turnaround Time          Waiting Time
Process[1]          4              13              9
Process[3]          5              16             11
Process[4]          6              18             12
Process[2]          7              21             14

Average Waiting Time:  11.500000
Avg Turnaround Time:  17.000000
```

In case you are facing any challenges with these C Programming Interview Questions, please write your problems in the comment section below.

Q48. Which structure is used to link the program and the operating system?

Ans: The answer can be explained through the following points,

- The structure used to link the operating system to a program is **file**.

- The **file** is defined in the header file “**stdio.h**”(standard input/output header file).
- It contains the information about the file being used, its current **size** and its **location** in memory.
- It contains a character **pointer** that points to the **character** that is being opened.
- Opening a file establishes a **link** between the **program** and the **operating system** about which file is to be accessed.

Q49. What are the limitations of scanf() and how can it be avoided?

Ans: The Limitations of scanf() are as follows:

- **scanf()** cannot work with the string of characters.
- It is not possible to enter a **multiword** string into a **single** variable using scanf().
- To avoid this the **gets()** function is used.
- It gets a string from the keyboard and is terminated when **enter** key is pressed.
- Here the **spaces** and **tabs** are acceptable as part of the input string.

Q50. Differentiate between the macros and the functions.

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

- **Macro** call replaces the templates with the expansion in a literal way.
- The **Macro** call makes the program run **faster** but also increases the program size.
- Macro is **simple** and avoids **errors** related to the function calls.
- In a function, call **control** is transferred to the function along with arguments.
- It makes the functions **small** and **compact**.
- Passing **arguments** and getting back the **returned value** takes time and makes the program run at a **slower rate**.

Q #1) What are the key features in the C programming language?

Answer: Features are as follows:

- **Portability:** It is a platform-independent language.
- **Modularity:** Possibility to break down large programs into small modules.
- **Flexibility:** The possibility of a programmer to control the language.
- **Speed:** C comes with support for system programming and hence it compiles and executes with high speed when compared with other high-level languages.
- **Extensibility:** Possibility to add new features by the programmer.

Q #2) What are the basic data types associated with C?

Answer:

- **Int** – Represent the number (integer)
- **Float** – Number with a fraction part.
- **Double** – Double-precision floating-point value
- **Char** – Single character
- **Void** – Special purpose type without any value.

Q #3) What is the description for syntax errors?

Answer: The mistakes/errors that occur while creating a program are called syntax errors. Misspelled commands or incorrect case commands, an incorrect number of parameters in calling method /function, data type mismatches can be identified as common examples for syntax errors.

Q #4) What is the process to create increment and decrement statement in C?

Answer: There are two possible methods to perform this task.

- Use increment (++) and decrement (-) operator.

Example When x=4, x++ returns 5 and x- returns 3.

- Use conventional + or – sign.

Example When x=4, use x+1 to get 5 and x-1 to get 3.

Q #5) What are reserved words with a programming language?

Answer: The words that are a part of the standard C language library are called **reserved words**. Those reserved words have special meaning and it is not possible to use them for any activity other than its intended functionality.

Example: void, return int.

Q #6) What is the explanation for the dangling pointer in C?

Answer: When there is a pointer pointing to a memory address of any variable, but after some time the variable was deleted from the memory location while keeping the pointer pointing to that location is known as a dangling pointer in C.

Q #7) Describe static function with its usage?

Answer: A function, which has a function definition prefixed with a static keyword is defined as a static function. The static function should be called within the same source code.

Q #8) What is the difference between abs() and fabs() functions?

Answer: Both functions are to retrieve absolute value. abs() is for integer values and fabs() is for floating type numbers. Prototype for abs() is under the library file <stdlib.h> and fabs() is under <math.h>.

Q #9) Describe Wild Pointers in C?

Answer: Uninitialized pointers in the C code are known as **Wild Pointers**. They point to some arbitrary memory location and can cause bad program behavior or program crash.

Q #10) What is the difference between ++a and a++?

Answer: '++a' is called prefixed increment and the increment will happen first on a variable. 'a++' is called postfix increment and the increment happens after the value of a variable used for the operations.

Q #11) Describe the difference between = and == symbols in C programming?

Answer: '=' is the comparison operator which is used to compare the value or expression on the left-hand side with the value or expression on the right-hand side.

'=' is the assignment operator which is used to assign the value of the right-hand side to the variable on the left-hand side.

Q #12) What is the explanation for prototype function in C?

Answer: Prototype function is a declaration of a function with the following information to the compiler.

- Name of the function.
- The return type of the function.
- Parameters list of the function.

```
int Sum(int, int);
```

In this example Name of the function is Sum, the return type is the integer data type and it accepts two integer parameters.

Q #13) What is the explanation for the cyclic nature of data types in C?

Answer: Some of the data types in C have special characteristic nature when a developer assigns value beyond the range of the data type. There will be no compiler error and the value changes according to a cyclic order. This is called cyclic nature. Char, int, long int data types have this property. Further float, double and long double data types do not have this property.

Q #14) Describe the header file and its usage in C programming?

Answer: The file containing the definitions and prototypes of the functions being used in the program are called a header file. It is also known as a library file.

Example: The header file contains commands like printf and scanf is from the stdio.h library file.

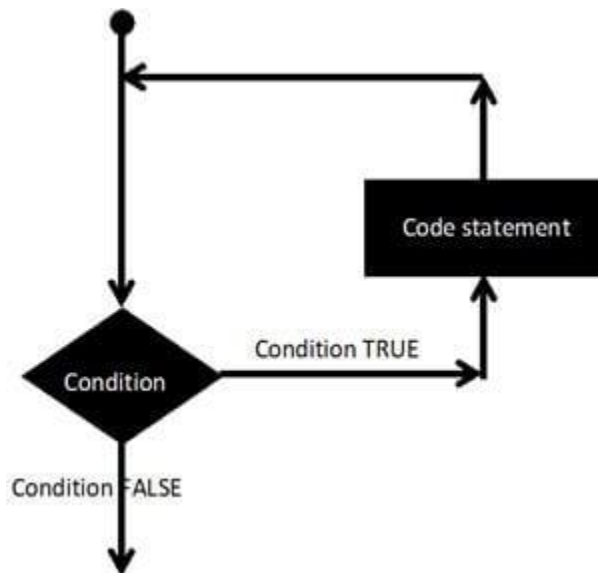
Q #15) There is a practice in coding to keep some code blocks in comment symbols than delete it when debugging. How this affects when debugging?

Answer: This concept is called commenting out and this is the way to isolate some part of the code which scans possible reason for the error. Also, this concept helps to save time because if the code is not the reason for the issue it can simply be removed from comment.

Q #16) What are the general description for loop statements and available loop types in C?

Answer: A statement that allows the execution of statements or groups of statements in a repeated way is defined as a loop.

The following diagram explains a general form of a loop.



There are 4 types of loop statements in C.

- While loop
- For Loop
- Do...While Loop
- Nested Loop
-

Q #17) What is a nested loop?

Answer: A loop that runs within another loop is referred to as a **nested loop**. The first loop is called the Outer Loop and the inside loop is called the Inner Loop. The inner loop executes the number of times defined in an outer loop.

Q #18) What is the general form of function in C?

Answer: The function definition in C contains four main sections.

```

return_type function_name( parameter list )
{
    body of the function
}
  
```

- **Return Type:** Data type of the return value of the function.
- **Function Name:** The name of the function and it is important to have a meaningful name that describes the activity of the function.
- **Parameters:** The input values for the function that are used to perform the required action.
- **Function Body:** Collection of statements that performs the required action.
-

Q #19) What is a pointer on a pointer in C programming language?

Answer: A pointer variable that contains the address of another pointer variable is called pointer on a pointer. This concept de-refers twice to point to the data held by a pointer variable.

```
int a = 5, *x=&a, **y=&x;
```

In this example ****y** returns the value of the variable **a**.

Q #20) What are the valid places to have keyword “Break”?

Answer: The purpose of the Break keyword is to bring the control out of the code block which is executing. It can appear only in looping or switch statements.

Q #21) What is the behavioral difference when the header file is included in double-quotes (“”) and angular braces (<>)?

Answer: When the Header file is included within double quotes (“ ”), compiler search first in the working directory for the particular header file. If not found, then it searches the file in the include path. But when the Header file is included within angular braces (<>), the compiler only searches in the working directory for the particular header file.

Q #22) What is a sequential access file?

Answer: General programs store data into files and retrieve existing data from files. With the sequential access file, such data are saved in a sequential pattern. When retrieving data from such files each data is read one by one until the required information is found.

Q #23) What is the method to save data in a stack data structure type?

Answer: Data is stored in the Stack data structure type using the **First In Last Out (FILO)** mechanism. Only top of the stack is accessible at a given instance. Storing mechanism is referred as a PUSH and retrieve is referred to as a POP.

Q #24) What is the significance of C program algorithms?

Answer: The algorithm is created first and it contains step by step guidelines on how the solution should be. Also, it contains the steps to consider and the required calculations/operations within the program.

Q #25) What is the correct code to have the following output in C using nested for loop?

```
1
12
123
1234
12345
```

Answer:

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

```
int main () {
```

```
    int a;
```

```
    int b;
```

```
    /* for loop execution */
```

```
    for( a = 1; a < 6; a++ )
```

```
    {
```

```
        /* for loop execution */
```

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

```
        {
```

```
            printf("%d",b);
```

```
        }
```

```
        printf("\n");
```

```
    }
```

```
    return 0;
```

```
}
```

```
1 #include <stdio.h>
2
3 int main () {
4
5     int a;
6     int b;
7
8     /* for loop execution */
9     for( a = 1; a < 6; a++ )
10    {
11        /* for loop execution */
12        for( b = 1; b <= a; b++ )
13        {
14            printf("%d",b);
15
16        }
17        printf("\n");
18    }
19
20    return 0;
21 }
```

Q #26) Explain the use of function toupper() with an example code?

Answer: Toupper() function is used to convert the value to uppercase when it used with characters.

Code:

```
#include <stdio.h>
#include <ctype.h>
int main()
{
    char c;

    c = 'a';
    printf("%c -> %c", c, toupper(c));

    c = 'A';
    printf("\n%c -> %c", c, toupper(c));

    c = '9';
    printf("\n%c -> %c", c, toupper(c));
    return 0;
}
```

Result:

1	a -> A
2	A -> A
3	9 -> 9

Q #27) What is the code in a while loop that returns the output of the given code?

```
#include <stdio.h>

int main () {

    int a;

    /* for loop execution */
    for( a = 1; a <= 100; a++ )
    {
        printf("%d\n",a * a);
    }
}
```



```
        return 0;
    }
```

```
1 #include <stdio.h>
2
3 int main () {
4     int a;
5
6     /* for loop execution */
7     for( a = 1; a <= 100; a++ )
8     {
9         printf("%d\n",a * a);
10    }
11
12    return 0;
13 }
14
15 }
```

Answer:

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

```
int main () {
```

```
    int a;
```

```
    while (a<=100)
```

```
    {
        printf ("%d\n", a * a);
        a++;
    }
```

```
    return 0;
```

```
}
```

```
1 #include <stdio.h>
2
3 int main () {
4     int a;
5
6     while (a<=100)
7     {
8         printf ("%d\n", a * a);
9         a++;
10    }
11
12    return 0;
13 }
14 }
```

Q #28) Select the incorrect operator form in the following list(== , <> , >= , <=) and what is the reason for the answer?

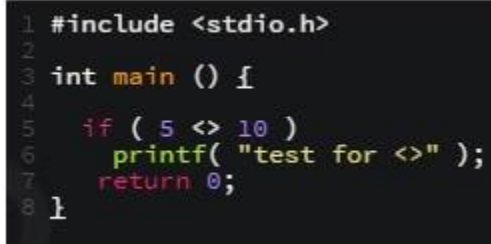
Answer: Incorrect operator is '<>'. This format is correct when writing conditional statements, but it is not the correct operation to indicate not equal in C programming. It gives a compilation error as follows.

Code:

```
#include <stdio.h>

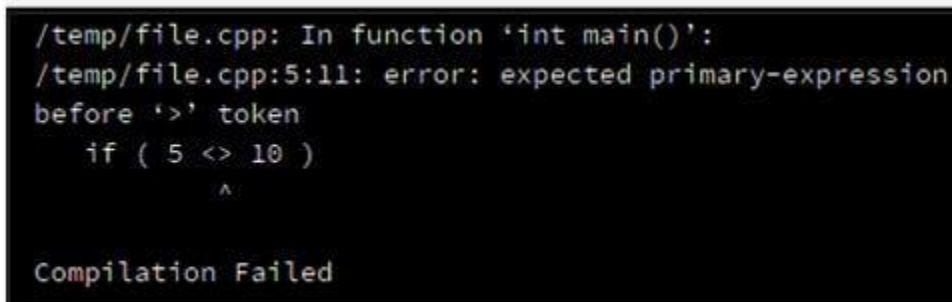
int main () {

    if ( 5 <> 10 )
        printf( "test for <>" );
    return 0;
}
```



```
1 #include <stdio.h>
2
3 int main () {
4
5     if ( 5 <> 10 )
6         printf( "test for <>" );
7     return 0;
8 }
```

Error:



```
/temp/file.cpp: In function 'int main()':
/temp/file.cpp:5:11: error: expected primary-expression
before '>' token
    if ( 5 <> 10 )
           ^
Compilation Failed
```

Q #29) Is it possible to use curly brackets ({}) to enclose a single line code in C program?

Answer: Yes, it works without any error. Some programmers like to use this to organize the code. But the main purpose of curly brackets is to group several lines of codes.

Q #30) Describe the modifier in C?

Answer: Modifier is a prefix to the basic data type which is used to indicate the modification for storage space allocation to a variable.

Example— In a 32-bit processor, storage space for the int data type is 4. When we use it with modifier the storage space change as follows:

- **Long int:** Storage space is 8 bit

- **Short int:** Storage space is 2 bit

•

Q #31) What are the modifiers available in C programming language?

Answer: There are 5 modifiers available in the C programming language as follows:

- Short
- Long
- Signed
- Unsigned
- long long

•

Q #32) What is the process to generate random numbers in C programming language?

Answer: The command rand() is available to use for this purpose. The function returns an integer number beginning from zero(0). The following sample code demonstrates the use of rand().

Code:

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

int main ()
{
    int a;
    int b;

    for(a=1; a<11; a++)
    {
        b = rand();
        printf( "%d\n", b );
    }
    return 0;
}
```

```

1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main ()
5 {
6     int a ;
7     int b ;
8
9
10    for(a=1; a<11; a++)
11    {
12        b = rand();
13        printf( "%d\n", b );
14    }
15    return 0;
16 }

```

Output:

```

1804289383
846930886
1681692777
1714636915
1957747793
424238335
719885386
1649760492
596516649
1189641421

```

Q #33) Describe the newline escape sequence with a sample program?

Answer: The Newline escape sequence is represented by `\n`. This indicates the point that the new line starts to the compiler and the output is created accordingly. The following sample program demonstrates the use of the newline escape sequence.

Code:

```

/*
 * C Program to print string
 */
#include <stdio.h>
#include <string.h>

int main() {
    printf("String 01 ");
    printf("String 02 ");
    printf("String 03 \n");
}

```

```

printf("String 01 \n");
printf("String 02 \n");
return 0;
}

```

Output:

```

1 String 01 String 02 String 03
2 String 01
3 String 02

```

Q #34) Is that possible to store 32768 in an int data type variable?

Answer: Int data type is only capable of storing values between – 32768 to 32767. To store 32768 a modifier needs to be used with the int data type. Long Int can be used and also if there are no negative values, unsigned int is also possible to use.

Q #35) Is there any possibility to create a customized header file with C programming language?

Answer: Yes, it is possible and easy to create a new header file. Create a file with function prototypes that are used inside the program. Include the file in the '#include' section from its name.

Q #36) Describe dynamic data structure in C programming language?

Answer: Dynamic data structure is more efficient to memory. The memory access occurs as needed by the program.

Q #37) Is that possible to add pointers to each other?

Answer: There is no possibility to add pointers together. Since a pointer contains address details there is no way to retrieve the value from this operation.

Q #38) What is indirection?

Answer: If you have defined a pointer to a variable or any memory object, there is no direct reference to the value of the variable. This is called the indirect reference. But when we declare a variable, it has a direct reference to the value.

Q #39) What are the ways to a null pointer that can be used in the C programming language?

Answer: Null pointers are possible to use in three ways.

- As an error value.
- As a sentinel value.
- To terminate indirection in the recursive data structure.
-

Q #40) What is the explanation for modular programming?

Answer: The process of dividing the main program into executable subsection is called module programming. This concept promotes reusability.