C Programming Technical Interview Questions

Some Questions are repeated to get practice

1. What are the features of the C language?
2. Mention the dynamic memory allocation functions?
3. What is the use of pointer variables in c programming and what do u mean by dangling pointer variable?
4. Local vs Global Variables?
5. What is the use of break control statements?
6. What is a predefined function in c?
7. What are c tokens?
8. What is the use of header files in c?
9. What is a memory leak?
10. Differentiate between call by value and call by reference.
11. What is the difference between a compiler and an interpreter?
12. What is typecasting?
13. Mention four types of storage classes in c.
14. What is the use of the size of an operator in c?
15. Write a c code to swap two numbers without using a third variable?
16. What is a union?
17. What is enumeration?
18. What is a recursion?
19. What are macros in c?
20. Write the difference between macros and functions.
21. Sort an array using a quick sort algorithm.
22. Write a c code to find the Fibonacci series.
23. Palindrome program in C.
24. Implement a C program to display a string in reverse order.
25. What is C programming language?
26. What are the key features of C language?
27. Explain the difference between printf and scanf?
28. What is a pointer in C?
29. What is an array?
30. Explain the difference between ++i and i++?
31. What are storage classes in C?
32. What is a segmentation fault?
33. What is dynamic memory allocation?
34. Explain the difference between malloc and calloc?
35. What is a linked list?
36. How does a doubly linked list differ from a singly linked list?
37. What is a stack?
38. What is a queue?
39. Explain the concept of recursion.
40. How do you declare a pointer to a function in C?
41. What is a null pointer?
42. How do you use pointers with arrays?
43. What is the sizeof operator?

44. Explain the use of the typedef keyword.
45. What is a struct in C?
46. How do you access members of a struct?
47. What is a union in C?
48. What is the main difference between a struct and a union?
49. How do you define a struct in C?
50. Explain malloc and how to use it.
51. What is free and why is it used?
52. How do you use realloc?
53. What is a memory leak?
54. What are the risks of using dynamic memory allocation?
55. How do you open a file in C?
56. How do you read from a file?
57. How do you write to a file?
58. How do you close a file?
59. What is fseek?
60. What is a macro in C?
61. How do you define a macro?
62. What is the difference between #include <filename> and #include "filename"?
63. Explain conditional compilation.
64. What is the purpose of #pragma directive?
65. How do you declare a string in C?
66. What functions are used for string handling in C?
67. How do you concatenate two strings?
68. How do you compare two strings?
69. How do you find the length of a string?
70. What is errno?
71. How do you handle errors in C?
72. What is the assert function?
73. Explain the use of setjmp and longjmp
74. What is exit and how is it used?
75. Why is C called a mid-level programming language?
76. What are the features of the C programming language?
77. What are basic data types supported in the C ProgrammingLanguage?
78. What are tokens in C?
79. What do you mean by the scope of the variable?
80. What are pre processor directives in C?
81. What is the use of static variables in C?
82. What is the difference between malloc() and calloc() in the Cprogramming language?
83. What do you mean by dangling pointers and how are danglingpointers different from memory leaks in C programming?
84. Write a program to convert a number to a string with the helpof sprintf() function in the C library.
85. What is recursion in C?
86. What is the difference between the local and global variablesin C?
87. What are pointers and their uses?
88. What is typedef in C?

89. What are loops and how can we create an infinite loop in C?
90. What is the difference between type casting and typeconversion?
91. What are header files and their uses?
92. What are the functions and their types?
93. What is the difference between macro and functions?
94. How to convert a string to numbers in C?
95. What are reserved keywords?
96. What is a structure?
97. What is a structure?
98. What is an r-value and l -value?
99. What is the difference between call by value and call byreference?
100. What is the sleep() function?
101. What are enumerations?
102. What is a volatile keyword?
103. Write a C program to print the Fibonacci series using recursion and without using recursion.
104. Write a C program to check whether a number is prime or not.
105. How is source code different from object code?
106. What is static memory allocation and dynamic memoryallocation?
107. What is pass-by-reference in functions?
108. What is a memory leak and how to avoid it?
109. What are command line arguments?
110. What is an auto keyword?
111. Write a program to print ‘‘Hello-World’’ without using a semicolon.
112. Write a C program to swap two numbers without using athird variable.
113. Write a program to check whether a string is a palindrome or not.
114. Explain modifiers.
115. Write a program to print the factorial of a given number withthe help of recursion.
116. Write a program to check an Armstrong number.
117. Write a program to reverse a given number.
118. What is the use of an extern storage specifier?
119. What is the use of printf() and scanf() functions in CProgramming language? Also, explain format specifiers.
120. What is near, far, and huge pointers in C?
121. Mention file operations in C.
122. Write a Program to check whether a linked list is circular or not.
123. Write a program to Merge two sorted linked lists.
124. What is the difference between getc(), getchar(), getch() and getche().

Answers

1. What are the features of the c programming language?

- Simple and Efficient.
- Fast.
- Portability.
- Extensibility.
- Function-Rich Libraries.
- Dynamic Memory Management.
- Modularity With Structured Language.
- Mid-Level Programming Language.

2. Mention the dynamic memory allocation functions?

- malloc()
- calloc()
- realloc()
- free()

3. What is the use of pointer variables in c programming and what do u mean by dangling pointer variable?

- Pointers are one of the core components of the C programming language. A pointer can be used to store the memory address of other variables, functions, or even other pointers.

- A dangling pointer in C is a pointer that points to a memory location that has been deallocated or is no longer valid.

4. Local vs Global Variables?

- The variables which are defined within some functions and are accessible to that function only are called Local Variables.

- The variables which are defined outside of function block and

are accessible to entire program are called Global Variables.

5. What is the use of break control statements?
   Ans: The break is used to terminate the loop. So, whenever the control encounters the break statement it will stop the loop iteration and transfer the control to next statement soon after the loop.

6. What is a predefined function in c?
   Ans: Predefined functions are built-in functions. It helps the user to use the already existing code in the program.

7. What are c tokens?

- Keywords

- Identifiers
- Constants
- Special Symbols
- String
- Operators

8. What is the use of header files in c?

- Header files contain the pre-defined library functions.
- Programmers can make use of standard library functions by defining #include in the program. It reduces the number of codes and complexity of the program.

9. What is a memory leak?
   Ans: A memory leak occurs when a program fails to release memory that it has allocated dynamically, leading to a gradual depletion of available memory.

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

Call by value	Call by reference
In call by value, we pass values by copying variables.	In call by reference, we pass the address of the variable.
In call by value, the memory location for formal and actual arguments is created separate.	In case of call by reference, the formal and actual arguments share the same memory location.
Any changes made in a copy of variable will not affect the value of a variable outside the function in call by value.	Whereas, in call by reference any changes made in the variable will also affects the value of a variable outside the function.

11. What is the difference between a compiler and interpreter?

- A compiler compiles the whole code at a time. And displays all the errors.
- The interpreter executes the program line by line and displays the error of every single line.

12. What is typecasting?
    Ans: Typecasting is a process of converting one data type into another data type. The conversation that is done automatically is known as implicit conversion.

13. Mention four types of storage classes in c.

- Automatic
- Register
- Static
- And External

14. What is the use of the size of an operator in c?
    Ans: The size of operator is applied to data types such as int, char, float, etc. to determine the exact size of a variable, in bytes.

15. Write a c code to swap two numbers without using a third variable?

```
#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;
}
```

Output:

```
Before swap: a=10 b=20
After swap: a=20 b=10
```

16. What is a union?

    Ans: Union is a data type in C programming that allows different data types to be stored in the same memory locations.

17. What is enumeration?

    Ans:Enum is a user-defined data type consisting of an integral constant.

18. What is a recursion?

    Ans:Recursion is a process of making a function call itself. In short, the user call function inside the same function.

19. What are macros in c?

    Ans:Macro in c is defined by the #define directive. Macro is a name given to a piece of a code, so whenever the compiler encounters a macro in a program, it will replace it with the macro value.

20. Write the difference between macros and functions.

Macros	Functions
Macros are preprocessed	But Function is compiled
Using macros execution speed is faster	In case of function the speed of execution is slower
Macros name is replaced by macro value, before compilation	Whereas in function, Transfer of control takes place during the function call
Using macros, the length of the code increase	The code length is Unaffected using function.

21. Sort an array using a quick sort algorithm.

```
#include <stdio.h>

// Function to swap two elements
void swap(int* a, int* b) {
    int t = *a;
    *a = *b;
    *b = t;
          }

// Function to partition the array on the basis of pivot
int partition (int arr[], int low, int high) {
    int pivot = arr[high]; // pivot
    int i = (low - 1);  // Index of smaller element

    for (int j = low; j <= high - 1; j++) {
        // If current element is smaller than or equal to pivot
        if (arr[j] <= pivot) {
            i++;    // increment index of smaller element
            swap(&arr[i], &arr[j]);
        }
    }
    swap(&arr[i + 1], &arr[high]);
    return (i + 1);
}

// Function to implement Quick Sort
void quickSort(int arr[], int low, int high) {
    if (low < high) {
        /* pi is partitioning index, arr[pi] is now at right place */
        int pi = partition(arr, low, high);

        // Separately sort elements before partition and after partition
        quickSort(arr, low, pi - 1);
        quickSort(arr, pi + 1, high);
    }
}

// A utility function to print array of size n
void printArray(int arr[], int size) {
    for (int i = 0; i < size; i++)
        printf("%d ", arr[i]);
    printf("\n");
}

int main() {
    int arr[] = {10, 7, 8, 9, 1, 5};
    int n = sizeof(arr) / sizeof(arr[0]);

    printf("Unsorted array: \n");
```

```
    printArray(arr, n);

    quickSort(arr, 0, n - 1);

    printf("Sorted array: \n");
    printArray(arr, n);

    return 0;
}
```

```
Unsorted array:
10 7 8 9 1 5
Sorted array:
1 5 7 8 9 10
```

22. Write a c code to find the Fibonacci series.

```
#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;
 }
```

```
Enter the number of elements: 15
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377
```

23. Palindrome program in C.

```
#include<stdio.h>
int main()
{
int n,r,sum=0,temp;
printf("enter the number=");
scanf("%d",&n);
temp=n;
while(n>0)
{
r=n%10;
sum=(sum*10)+r;
n=n/10;
}
if(temp==sum)
printf("palindrome number ");
else
printf("not palindrome");
return 0;
}
```

Output

```
enter the number=1551
palindrome number


enter the number=1234
not palindrome
```

24. implement a C program to display a string in reverse order.

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

int main()
{
    // string to be reversed.
    char str[100] = "string";

    printf("Original String: %s\n", str);

    // string length
    int len = strlen(str);

    // for loop
    for (int i = 0, j = len - 1; i <= j; i++, j--) {
        // swapping characters
        char c = str[i];
        str[i] = str[j];
        str[j] = c;
    }

    printf("Reversed String: %s", str);

    return 0;
}
```

output

Original String: string
Reversed String: gnirts

25. What is C programming language?

Answer: C is a high-level and general-purpose programming language that is idealfor developing firmware or portable applications. Originally intended for writing system software, C was developed by Dennis Ritchie at Bell Labs in 1972.

26. What are the key features of C language?

Answer: Key features of C include simplicity, speed, low-level memory access, arich set of built-in functions, a robust standard library, and its portability across various platforms.

27. Explain the difference between printf and scanf.

Answer: printf is used for output, displaying text and variables to the console,while scanf is used for input, reading data from the standard input.

28. What is a pointer in C?

Answer: A pointer is a variable that stores the memory address of another variable. Pointers are used for dynamic memory allocation, arrays, structures, andfunctions.

29. What is an array?

Answer: An array is a collection of variables of the same type that are stored incontiguous memory locations and can be accessed using an index.

30. Explain the difference between ++i and i++.

Answer: ++i increments the value of i before it is used in an expression, while i++increments the value of i after it has been used in an expression.

31. What are storage classes in C?

Answer: Storage classes in C define the scope, visibility, and lifetime of variables/functions within a C program. The four storage classes are auto, register,static, and extern.

32. What is a segmentation fault?

Answer: A segmentation fault occurs when a program attempts to access amemory location that it is not allowed to access. This usually results from dereferencing a null or invalid pointer.

33. What is dynamic memory allocation?

Answer: Dynamic memory allocation is the process of allocating memory duringruntime using functions like malloc, calloc, realloc, and free.

34. Explain the difference between malloc and calloc.

Answer: malloc allocates a single block of memory without initializing it, whilecalloc allocates multiple blocks of memory and initializes all bytes to zero.

35. What is a linked list?

Answer: A linked list is a data structure where each element (node) contains a datapart and a reference (or link) to the next element in the sequence. It allows for efficient insertion and deletion of elements.

36. How does a doubly linked list differ from a singly linked list?

Answer: A doubly linked list has nodes with two references: one to the next nodeand one to the previous node, whereas a singly linked list only has a reference to the next node.

37. What is a stack?

Answer: A stack is a linear data structure that follows the Last In, First Out (LIFO) principle. Elements are added and removed from the same end called the"top."

38. What is a queue?

Answer: A queue is a linear data structure that follows the First In, First Out(FIFO) principle. Elements are added at the rear and removed from the front.

39. Explain the concept of recursion.

Answer: Recursion is a programming technique where a function calls itself directly or indirectly to solve a problem by breaking it down into smaller sub-problems.

40. How do you declare a pointer to a function in C?

Answer: You can declare a pointer to a function using the following syntax:return_type (*pointer_name)(parameter_types);.

41. What is a null pointer?

Answer: A null pointer is a pointer that does not point to any valid memory location. It is often used to indicate that the pointer is not intended to point to anobject.

42. How do you use pointers with arrays?

Answer: You can use pointers with arrays to traverse or manipulate array elements. The name of the array acts as a pointer to the first element of the array.

43. What is the sizeof operator?

Answer: The sizeof operator returns the size, in bytes, of a data type or a variable.

44. Explain the use of the typedef keyword.

Answer: The typedef keyword is used to create new data type names or aliases forexisting data types, making code easier to read and maintain.

45. What is a struct in C?

Answer: A struct is a user-defined data type that allows grouping variables ofdifferent types under a single name for easier handling.

46. How do you access members of a struct?

Answer: Members of a struct can be accessed using the dot operator (.) for structvariables and the arrow operator (->) for pointers to structs.

47. What is a union in C?

Answer: A union is a user-defined data type similar to a struct, but with all its members sharing the same memory location, allowing the storage of different datatypes in the same memory space.

48. What is the main difference between a struct and a union?

Answer: The main difference is that in a struct, each member has its own memorylocation, whereas in a union, all members share the same memory location.

49. How do you define a struct in C?

Answer: A struct can be defined as follows:struct

StructName {

dataType member1;

dataType member2;

// More members...

};

50. Explain malloc and how to use it.

Answer: malloc allocates a specified number of bytes of memory and returns apointer to the first byte of the allocated memory. Example:

int *ptr = (int *)malloc(sizeof(int) * 5);

51. What is free and why is it used?

Answer: free deallocates the memory previously allocated by malloc, calloc, orrealloc, freeing up the memory for other uses.

52. How do you use realloc?

Answer: realloc changes the size of previously allocated memory block. It is usedas follows:

ptr = realloc(ptr, new_size);

53. What is a memory leak?

Answer: A memory leak occurs when a program allocates memory by usingmalloc or similar but fails to release it using free, leading to wasted memory.

54. What are the risks of using dynamic memory allocation?

Answer: Risks include memory leaks, segmentation faults, and fragmentation,which can lead to inefficient memory use and program crashes.

55. How do you open a file in C?

Answer: You can open a file using the fopen function, which returns a file pointer.Example:

FILE *file = fopen("filename.txt", "r");

56. How do you read from a file?

Answer: You can read from a file using functions like fscanf, fgets, or fread.Example with fscanf:

fscanf(file, "%d", &variable);

57. How do you write to a file?

Answer: You can write to a file using functions like fprintf, fputs, or fwrite.Example with fprintf:

fprintf(file, "Hello, World!");

58. How do you close a file?

Answer: You close a file using the fclose function. Example:fclose(file);

59. What is fseek?

Answer: fseek is used to move the file pointer to a specific location in a file.Example:

fseek(file, offset, SEEK_SET)

60. What is a macro in C?

Answer: A macro is a fragment of code which has been given a name. Whenever the name is used, it is replaced by the contents of the macro.

61. How do you define a macro?

Answer: Macros are defined using the #define directive. Example:#define PI 3.14

62. What is the difference between #include <filename> and #include"filename"?

Answer: #include <filename> is used for standard library headers, and the compiler searches the standard library directories. #include "filename" is used foruser-defined headers, and the compiler searches the current directory first.

63. Explain conditional compilation.

Answer: Conditional compilation allows parts of code to be compiled or omittedbased on certain conditions using directives like #ifdef, #ifndef, #if, #else, #elif, and #endif.

64. What is the purpose of #pragma directive?

Answer: The #pragma directive is used to provide additional information to thecompiler, often to enable or disable certain features or optimizations.

65. How do you declare a string in C?

Answer: A string in C can be declared as an array of characters. Example:char str[] =

"Hello, World!";

66. What functions are used for string handling in C?

Answer: Common string handling functions include strcpy, strcat, strlen, strcmp,and strstr.

67. How do you concatenate two strings?

Answer: You can concatenate two strings using the strcat function. Example:

strcat(destination, source);

68. How do you compare two strings?

Answer: You can compare two strings using the strcmp function. Example:int result = strcmp(str1, str2);

69. How do you find the length of a string?

Answer: You can find the length of a string using the strlen function. Example:int length = strlen(str);

70. What is errno?

Answer: errno is a macro that expands to a modifiable lvalue (usually an integer)that contains the error code produced by the most recent library function call.

71. How do you handle errors in C?

Answer: Errors in C can be handled using the errno variable, perror function, andby checking the return values of functions to ensure they executed successfully.

72. What is the assert function?

Answer: The assert function is used to perform debugging checks by evaluating acondition. If the condition is false, the program prints an error message and terminates.

73. Explain the use of setjmp and longjmp.

Answer: setjmp saves the current environment for later use by longjmp, whichrestores the saved environment, allowing for non-local jumps in programs, typically for error handling.

74. What is exit and how is it used?

Answer: The exit function terminates the calling process immediately, performingcleanup operations. It is used as follows:

exit(exit_code);

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

Ans :

Due to its ability to support both low-level and high-level features, C is considered a middle-level language.

It is both an assembly-level language, i.e. a low-level language, and a higher-level language.

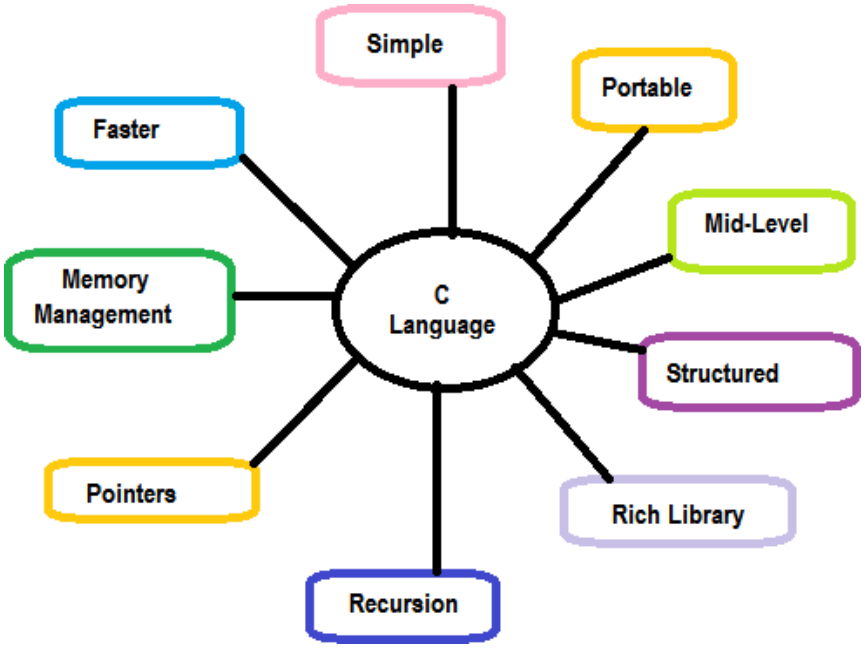
Programs that are written in C are converted into assembly code, and they support pointer arithmetic (low-level) while being machine-independent (high-level).

Therefore, C is often referred to as a middle-level language.

C can be used to write operating systems and menu-driven consumer billingsystems.

76.What are the features of the C programming language?

Ans :



77. What are basic data types supported in the C ProgrammingLanguage?

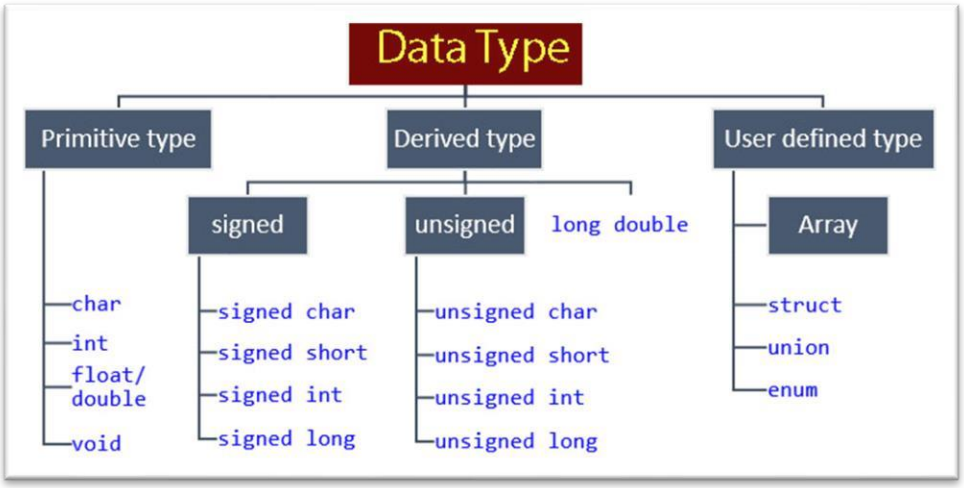
Ans :

Each variable in C has an associated data type.

Each data type requires different amounts of memory and has some specificoperations which can be performed over it.

It specifies the type of data that the variable can store like integer, character,floating, double, etc.

In C data types are broadly classified into 3 categories:



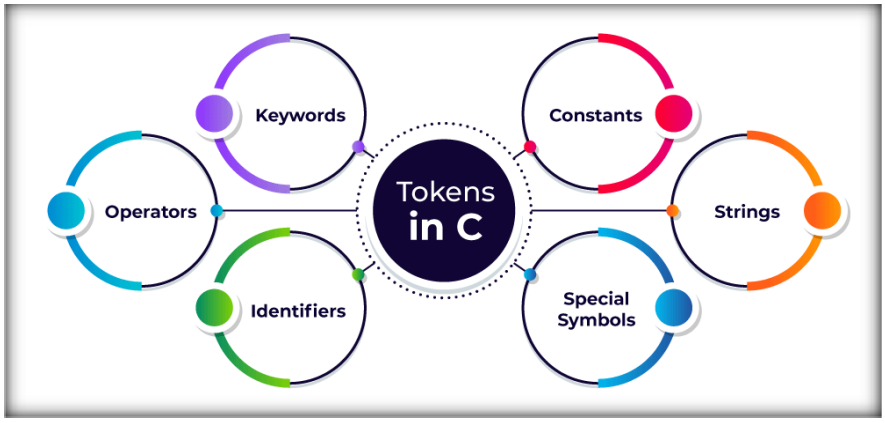
Primitive data types: Primitive data types can be further classified intointeger, and floating data types.

User Defined data types: These data types are defined by the user to makethe program more readable.

Derived data types: Data types that are derived from primitive or built-indata types.

78. What are tokens in C?

Ans : Tokens are identifiers or the smallest single unit in a program that ismeaningful to the compiler. In C we have the following tokens:



- Keywords: Predefined or reserved words in the C programming language. Every keyword is meant to perform a specific task in a program. C Programming language supports 32 keywords.

- Identifiers: Identifiers are user-defined names that consist of an arbitrarily long sequence of digits or letters with either a letter or the underscore (_) as a first Character. Identifier names can ' t be equal to any reserved keywords in the C programming language. There are a set of rules which a programmer must follow in order to name an identifier in C.

- Constants: Constants are normal variables that cannot be modified in the program once they are defined. Constants refer to a fixed value. They are also referred to as literals.

- Strings: Strings in C are an array of characters that end with a null character (‘ \0). Null character indicates the end of the string;

- Special Symbols: Some special symbols in C have some special meaning and thus, they cannot be used for any other purpose in the program. # = {} () , * ; [] are the special symbols in C programming language.

- Operators: Symbols that trigger an action when they are applied to any variable or any other object. Unary, Binary, and ternary operators are used in the C Programming language.

79. What do you mean by the scope of the variable?

Ans :

Scope in a programming language is the block or a region where a defined variable will have its existence and beyond that region, the variable is automatically destroyed. Every variable has its defined scope. In simple terms, the scope of a variable is equal to its life in the program. The variable can be declared in three places These are:

- Local Variables: Inside a given function or a block
- Global Variables: Out of all functions globally inside the program.
- Formal Parameters: In-function parameters only.

80. What are preprocessor directives in C?

Ans : In C preprocessor directives are considered the built-in predefined functions or macros that act as a directive to the compiler and are executed before the program execution.

There are multiple steps involved in writing and executing a program in C.

Main types of Preprocessor Directives are Macros, File Inclusion, ConditionalCompilation, and Other directives like #undef, #pragma, etc.

81. What is the use of static variables in C?

Ans : Static variables in the C programming language are used to preservethe data values between function calls even after they are out of their scope.

Static variables preserve their values in their scope and they can be usedagain in the program without initializing again.

Static variables have an initial value assigned to 0 without initialization.

Example:

```
// C program to print initial
// value of static variable
#include <stdio.h>
int main()
{
        static int var;
        int x;

        printf("Initial value of static variable %d\n", var);
        printf("Initial value of variable without static
%d", x);
        return 0;
}
```

Output:

```
Initial value of static variable 0
Initial value of variable without static 0
```

82. What is the difference between malloc() and calloc() in the C programming language?

Ans :

calloc() and malloc() library functions are used to allocate dynamic memory.

Dynamic memory is the memory that is allocated during the runtime of the program from the heap segment. ‘‘stdlib.h’’ is the header file that is used to facilitate dynamic memory allocation in the C Programming language.

Parameter	Malloc()	Calloc()
Definition	It is a function that creates one block of memory of a fixed size.	It is a function that assigns more than one block of memory to a single variable.
Number of arguments	It only takes one argument.	It takes two arguments.
Speed	malloc() function is faster than calloc().	calloc() is slower than malloc().
Efficiency	It has high time efficiency.	It has low time efficiency.
Usage	It is used to indicate	it is used to indicate contiguous

Parameter	Malloc()	Calloc()
	memory allocation.	memory allocation.

83. What do you mean by dangling pointers and how are danglingpointers different from memory leaks in C programming?

Ans :

- Pointers pointing to deallocated memory blocks in C Programming are known as dangling pointers i.e, whenever a pointer is pointing to a memory location and In case the variable is deleted and the pointer still points to that same memory location then it is known as a dangling pointer variable.
- In C programming memory leak occurs when we allocate memory with the help of the malloc() or calloc() library function, but we forget to free the allocated memory with the help of the free() library function. Memory leak causes the program to use an undefined amount of memory from the RAM which makes it unavailable for other running programs thiscauses our program to crash.

84. Write a program to convert a number to a string with the helpof sprintf() function in the C library.

Ans :

Example:

```
// C program to convert number to
// string using sprintf()
#include <stdio.h>
#include <string.h>

// Driver code
int main()
{
        char res[20];
        float a = 32.23;
        sprintf(res, "%f", a);
        printf("\nThe string for the num is %s", res);
        return 0;
}
```

Output :

```
The string for the num is 32.230000
```

85. What is recursion in C?

Ans : Recursion is the process of making the function call itself directly orindirectly.

A recursive function solves a particular problem by calling a copy of itselfand solving smaller subproblems that sum up the original problems.

Recursion helps to reduce the length of code and make it more understandable.

The recursive function uses a LIFO (Last In First Out) structure like a stack. Every recursive call in the program requires extra space in the stack memory.

86. What is the difference between the local and global variablesin C?

Ans :

Local variables are declared inside a block or function but global variablesare declared outside the block or function to be accessed globally.

Local Variables	Global Variables
Declared inside a block or a function.	Variables that are declared outsidethe block or a function.
By default, variables store a garbage value.	By default value of the global valueis zero.
The life of the local variables is destroyed after the block or a function.	The life of the global variable existsuntil the program is executed.
Variables are stored inside the stack unless they are specified by the programmer.	The storage location of the global variable is decided by the compiler.
To access the local variables in other functions parameter passing is required.	No parameter passing is required. They are globally visible throughout the program.

87. What are pointers and their uses?

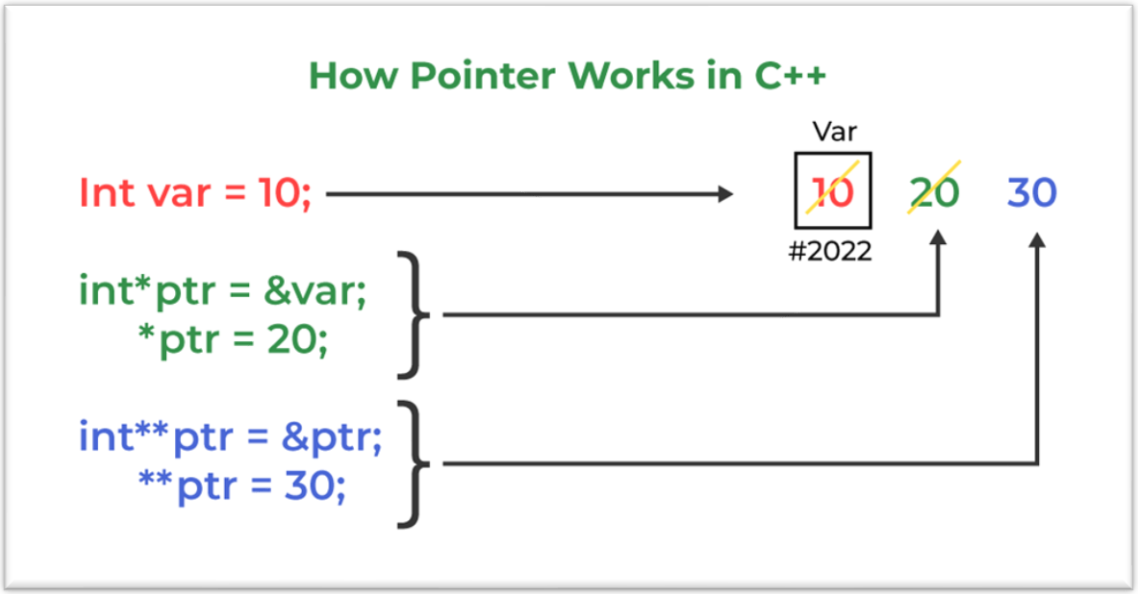
Ans : Pointers are used to store the address of the variable or a memorylocation.

Pointer can also be used to refer to another pointer function.

The main purpose of the pointer is to save memory space and increaseexecution time.

Uses of pointers are:

- To pass arguments by reference
- For accessing array elements
- To return multiple values
- Dynamic memory allocation
- To implement data structures
- To do system-level programming where memory addresses are useful



88. What is typedef in C?

Ans : In C programming, typedef is a keyword that defines an alias for anexisting type.

Whether it is an integer variable, function parameter, or structure declaration, typedef will shorten the name.

Syntax:

```
typedef <existing-type> <alias-name>
```

Here,

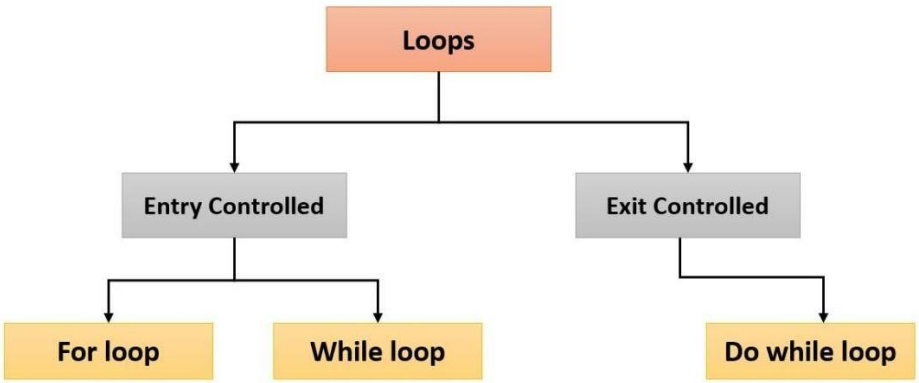
- existing type is already given a name.
- alias name is the new name for the existing variable.

89. What are loops and how can we create an infinite loop in C?

Ans : Loops are used to execute a block of statements repeatedly.

The statement which is to be repeated will be executed n times inside theloop until the given condition is reached.

There are two types of loops Entry controlled and Exit-controlled loops inthe C programming language.



An Infinite loop is a piece of code that lacks a functional exit. So, it repeatsindefinitely.

There can be only two things when there is an infinite loop in the program.

One it was designed to loop endlessly until the condition is met within theloop.

Another can be wrong or unsatisfied break conditions in the program.

Example :

```
// C program for infinite loop
// using for, while, do-while
#include <stdio.h>

// Driver code
int main()
{
	for (;;) {
		printf("Infinite-loop\n");
	}

	while (1) {
		printf("Infinite-loop\n");
	}

	do {
		printf("Infinite-loop\n");
	} while (1);

	return 0;
}
```

90. What is the difference between type casting and typeconversion?

Ans :

Type Casting	Type Conversion
The data type is converted to another data type by a programmer with the help of a casting operator.	The data type is converted to another data by a compiler.
It can be applied to both compatible data types as well as incompatible data types.	Type conversion can only be applied to only compatible data types.
In Type casting in order to cast the data type into another data type, a caste operator is needed	In type conversion, there is no needfor a casting operator.
Type casting is more efficientand reliable.	Type conversion is less efficientand less reliable than type casting.
Type casting takes place during the program design by the programmer.	Type conversion is done at compile time.
Syntax: destination_data_type =(target_data_type)	Syntax: int a = 20; float b; b = a; // a = 20.0000

91. What are header files and their uses?

Ans : C language has numerous libraries which contain predefined functionsto make programming easier.

Header files contain predefined standard library functions.All header

files must have a ‘.h’ extension.

Header files contain function definitions, data type definitions, and macros which can be imported with the help of the preprocessor directive ‘#include’. Preprocessor directives instruct the compiler that these files are needed to be processed before the compilation.

There are two types of header files i.e, User-defined header files and Pre- existing header files.

For example, if our code needs to take input from the user and print desired output to the screen then ‘stdio.h’ header file must be included in the program as #include<stdio.h>.

This header file contains functions like scanf() and printf() which are used to take input from the user and print the content.

92. What are the functions and their types?

Ans : The function is a block of code that is used to perform a task multiple times rather than writing it out multiple times in our program. Functions avoid repetition of code and increase the readability of the program.

Modifying a program becomes easier with the help of function and hence reduces the chances of error.

There are two types of functions:

- User-defined Functions: Functions that are defined by the user to reduce the complexity of big programs. They are built only to satisfy the condition in which the user is facing issues and are commonly known as "tailor-made functions".

- Built-in Functions: Library functions are provided by the compiler package and consist of special functions with special and different meanings. These functions give programmers an edge as we can directly use them without defining them.

93. What is the difference between macro and functions?

Ans :

A macro is a name that is given to a block of C statements as a pre-processordirective.

Macro is defined with the pre-processor directive. Macros are pre-processed which means that all the macros would be preprocessed before the compilation of our program.

However, functions are not preprocessed but compiled.

Macro	Function
Macros are preprocessed.	Functions are compiled.
Code length is increased using macro.	Code length remains unaffected using function.
Execution speed using a macro is faster.	Execution speed using function is

Macro	Function
	slower.
The macro name is replaced by the macro value before compilation.	Transfer of control takes place during the function call.
Macro doesn ’ t check any Compile-Time Errors.	Function check Compile-time errors.

94. How to convert a string to numbers in C?

Ans : In C we have 2 main methods to convert strings to numbers i.e, Usingstring stream, Using stoi() library Function, and using atoi() library function.

- sscanf(): It reads input from a string rather than standard input.
- atoi(): This function takes a string literal or a character array as anargument and an integer value is returned.

95. What are reserved keywords?

Ans : Every keyword is meant to perform a specific task in a program.

Their meaning is already defined and cannot be used for purposes other thanwhat they are originally intended for.

C Programming language supports 32 keywords.

Some examples of reserved keywords are auto, else, if, long, int, switch,typedef, etc.

96. What is a structure?

Ans : The structure is a keyword that is used to create user-defined datatypes.

The structure allows storing multiple types of data in a single unit.

The structure members can only be accessed through the structure variable.

```
struct student
{
  char  name[20];
  int roll_no;
 char address[20];
 char branch[20];
};
```

Below is the C program to implement structure:

Example:

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

// Structure student declared
struct student {
        char name[20];
        int roll_no;
        char address[50];
        char branch[50];
};

int main()
```

```
{
        struct student obj;
strcpy(obj.name, "Kamlesh_Joshi");
        obj.roll_no = 27;
        strcpy(obj.address, "Haldwani");
        strcpy(obj.branch, "Computer Science And
Engineering");
printf("Name: %s\n", obj.name);
        printf("Roll_No: %d \n", obj.roll_no);
        printf("Address: %s\n", obj.address);
        printf("Branch: %s", obj.branch);

        return 0;
}
```

Output :

```
Name: Kamlesh_Joshi
Roll_No: 27
Address: Haldwani
Branch: Computer Science And Engineering
```

97. What is union?

Ans : A union is a user-defined data type that allows users to store multiple types of data in a single unit. However, a union does not occupy the sum of the memory of all members.

It holds the memory of the largest member only. Since the union allocates one common space for all the members we can access only a single variableat a time.

The union can be useful in many situations where we want to use the same memory for two or more members.

Syntax:

```
union name_of_union
{
  data_type name;
  data_type name;
};
```

98.What is an r-value and l -value?

Ans:

- An ‘‘l-value’’ refers to an object with an identifiable location in memory (i.e. having an address).
- An ‘‘l-value’’ will appear either on the right or left side of the assignment operator(=).
- An ‘‘r-value’’ is a data value stored in memory at a given address. An ‘‘r-value’’ refers to an object without an identifiable location in memory (i.e. without an address).

- An ‘‘r-value’’ is an expression that cannot be assigned a value, therefore it can only exist on the right side of an assignment operator(=).

Example:

```
int val = 20;
```

Here, val is the ‘ l-value ’ , and 20 is the ‘ r-value ’ .

99. What is the difference between call by value and call byreference?

Ans :

Call by value	Call by Reference
Values of the variable are passed while function calls.	The address of a variable(location of variable) is passed while the function call.
Dummy variables copy the value of each variable in the function call.	Dummy variables copy the address of actual variables.
Changes made to dummy variables in the called function have no effect on actual variables in the calling function.	We can manipulate the actual variables using addresses.
A simple technique is used to pass the values of variables.	The address values of variables must be stored in pointer variables.

100.What is the sleep() function?

Ans :

- sleep() function in C allows the users to wait for a current thread for a given amount of time.
- sleep() function will sleep the present executable for the given amount of time by the thread but other operations of the CPU will function properly. sleep() function returns 0 if the requested time has elapsed.

101. What are enumerations?

Ans : In C, enumerations (or enums) are user-defined data types. Enumerations allow integral constants to be named, which makes a program easier to read and maintain.

For example, the days of the week can be defined as an enumeration and can be used anywhere in the program.

enum enumeration_name{constant1, constant2, ... };

Example:

```
// An example program to demonstrate working
// of enum in C
#include <stdio.h>
enum week { Mon, Tue, Wed, Thur, Fri, Sat, Sun };
int main()
{
        enum week day;
        day = Wed;
        printf("%d", day);
        return 0;
}
```

Output

```
2
```

- In the above example, we declared ‘‘day’’ as the variable, and the value of ‘‘Wed’’ is allocated to day, which is 2. So as a result, 2 is printed.

102. What is a volatile keyword?

Ans : Volatile keyword is used to prevent the compiler from optimization because their values can ’ t be changed by code that is outside the scope of current code at any time.

The System always reads the current value of a volatile object from the memory location rather than keeping its value in a temporary register at the point it is requested, even if previous instruction is asked for the value from the same object.

103. Write a C program to print the Fibonacci series using recursion and without using recursion.

Ans :

```
#include <stdio.h>

void Fibonacci(int num, int first, int second, int
third)

{

        if (num > 0) {

                third = first + second;

                first = second;

                second = third;

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

```
Fibonacci(num - 1, first, second, third)
        }
}
int main()
{
int num;
        printf("Please Enter number of Elements: ");
        scanf("%d", &num);

        printf(
                "Fibonacci Series with the help of
Recursion:\n");
printf("%d %d ", 0, 1);
        Fibonacci(num - 2, 0, 1, 0);
        printf("\nFibonacci Series without Using
Recursion:\n");
int first = 0, second = 1, third = 0;
        printf("%d %d ", 0, 1);
        for (int i = 2; i < num; i++) {
                third = first + second;

                        printf("%d ", third);

                first = second;
                second = third;
        }
        return 0;
}
```

Output:

```
Please Enter number of Elements: 5
Fibonacci Series with the help of Recursion:
0 1 1 2 3
Fibonacci Series without Using Recursion:
0 1 1 2 3
```

104. Write a C program to check whether a number is prime or not.

Example:

```
#include <math.h>
#include <stdio.h>
int main()
{
        int num;
        int check = 1;
        printf("Enter a number: \n");
        scanf("%d", &num);
        for (int i = 2; i <= sqrt(num); i++) {

                if (num % i == 0) {
                        check = 0;
                        break;
                }
        }
```

```
if (num <= 1) {
        check = 0;
    }
    if (check == 1) {
        printf("%d is a prime number", num);
    }
    else {
        printf("%d is not a prime number", num);
    }
    return 0;
}
```

105. How is source code different from object code?

Ans :

Source Code	Object Code
Source code is generated by the programmer.	object code is generated by a compiler or another translator.
High-level code which is human-understandable.	Low-level code is not human-understandable.
Source code can be easily modified and contains less number of statements than object code.	Object code cannot be modified and contains more statements than source code.
Source code can be changed over time	Object code can be modified and is

Source Code	Object Code
and is not system specific.	system specific.
Source code is less close to the machine and is input to the compiler or any other translator.	Source code is more close to the machine and is the output of the compiler or any other translator.
Language translators like compilers, assemblers, and interpreters are used to translate source code to object code.	Object code is machine code so itdoes not require any translation.

106. What is static memory allocation and dynamic memoryallocation?

Ans :

- Static memory allocation: Memory allocation which is done at compiletime is known as static memory allocation.
  Static memory allocation saves running time. It is faster than dynamicmemory allocation as memory allocation is done from the stack.
  This memory allocation method is less efficient as compared to dynamicmemory allocation. It is mostly preferred in the array.

- Dynamic memory allocation: Memory allocation done at execution or runtime is known as dynamic memory allocation.
  Dynamic memory allocation is slower than static memory allocation asmemory allocation is done from the heap.
  This memory allocation method is more efficient as compared to staticmemory allocation. It is mostly preferred in the linked list.

107. What is pass-by-reference in functions?

Ans :

Pass by reference allows a function to modify a variable without making a copy of the variable.

The Memory location of the passed variable and parameter is the same, so any changes done to the parameter will be reflected by the variables as well.

108. What is a memory leak and how to avoid it?

Ans : Whenever a variable is defined some amount of memory is created in the heap. If the programmer forgets to delete the memory.

This undeleted memory in the heap is called a memory leak.

The Performance of the program is reduced since the amount of available memory was reduced.

To avoid memory leaks, memory allocated on the heap should always be cleared when it is no longer needed.

109. What are command line arguments?

Ans :

Arguments that are passed to the main() function of the program in the command-line shell of the operating system are known as command-line arguments.

Syntax:

```
int main(int argc, char *argv[]){/*code which
is to be executed*/}
```

110. What is an auto keyword?

Ans :

Every local variable of a function is known as an automatic variable in the Clanguage.

Auto is the default storage class for all the variables which are declaredinside a function or a block.

Auto variables can only be accessed within the block/function they have been declared. We can use them outside their scope with the help of pointers.

By default auto keywords consist of a garbage value.

111. Write a program to print ‘‘Hello-World’’ without using a semicolon.

Ans :

```
// C program to print hello-world
// without using semicolon
#include <stdio.h>
// Driver code
int main()
{
        if (printf("Hello - World")) {
        }
        return 0;
}
```

112. Write a C program to swap two numbers without using athird variable.

Ans :

```
#include <stdio.h>


int main()
{
        // Variable declaration
        int var1 = 50;
        int var2 = 60;


        printf(
                "Values before swap are var1 = %d and
var2 = %d\n",
                var1, var2);


        var1 = var1 + var2;
        var2 = var1 - var2;
        var1 = var1 - var2;
        printf("Values after swap are var1 = %d and var2
= %d",
                var1, var2);


        return 0;
}
```

Output

```
Values before swap are var1 = 50 and var2 = 60
Values after swap are var1 = 60  and var2 = 50
```

113. Write a program to check whether a string is a palindrome or not.

Ans :

```
 #include <stdio.h>
#include <string.h>
void Palindrome(char s[])
{
        int start = 0;
        int end = strlen(s) - 1;
        while (end > start) {
                if (s[start++] != s[end--]) {
                        printf("%s is not a Palindrome \n", s);
                        return;
                }
        }
        printf("%s is a Palindrome \n", s);
}
int main()
{
        Palindrome("abba");
        return 0;
}
```

Output

```
abba is a Palindrome
```

114. Explain modifiers.

Ans :

Modifiers are keywords that are used to change the meaning of basic datatypes in C language.

They specify the amount of memory that is to be allocated to the variable.There are five data type modifiers in the C programming language:

- long
- short
- signed
- unsigned
- long long

115. Write a program to print the factorial of a given number withthe help of recursion.

Ans :

```
#include <stdio.h>

unsigned int factorial(unsigned int n)
{
        if (n == 0)
                return 1;
        return n * factorial(n - 1);

}

int main()
{
        int num = 5;
        printf("Factorial of %d is %d", num,
factorial(num));

      return 0;
}
```

Output :

```
Factorial of 5 is 120
```

116. Write a program to check an Armstrong number.

Ans :

```
#include <stdio.h>

int main()
{
	int n;
	printf("Enter Number \n");
	scanf("%d", &n);
	int var = n;
	int sum = 0;
	while (n > 0) {
		int rem = n % 10;
		sum = (sum) + (rem * rem * rem);
		n = n / 10;
	}

	if (var == sum) {
		printf("%d is an Armstrong number \n", var);
	}
	else {
		printf("%d is not an Armstrong number", var);
	}
	return 0;
}
```

Output :

```
Enter Number
0 is an Armstrong number
```

117. Write a program to reverse a given number.

Ans :

```
#include <stdio.h>

// Driver code
int main()
{
        int n, rev = 0;
        printf("Enter Number to be reversed : ");
        scanf("%d", &n);
        int r = 0;
         while (n != 0)
        {
                r = n % 10;
                rev = rev * 10 + r;
                n /= 10;
        }

        printf("Number After reversing digits is: %d", rev);
        return 0;
}
```

Output:

```
Enter Number to be reversed :
Number After reversing digits is: 321
```

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

Ans :

The extern keyword is used to extend the visibility of the C variables andfunctions in the C language.

Extern is the short name for external. It is used when a particular file needsto access a variable from any other file.

Extern keyword increases the redundancy and variables with extern keywordare only declared not defined.

By default functions are visible throughout the program, so there is no needto declare or define extern functions.

119. What is the use of printf() and scanf() functions in CProgramming language? Also, explain format specifiers.

Ans :

printf() function is used to print the value which is passed as the parameterto it on the console screen.

Syntax:

```
print(“%X”,variable_of_X_type);
```

scanf() method, reads the values from the console as per the data typespecified.

Syntax:

```
scanf(“%X”,&variable_of_X_type);
```

In C format specifiers are used to tell the compiler what type of data will bepresent in the variable during input using scanf() or output using print().

- %c: Character format specifier used to display and scan character.
- %d, %i: Signed Integer format specifier used to print or scan an integervalue.
- %f, %e, or %E: Floating-point format specifiers are used for printing or scanning float values.
- %s: This format specifier is used for String printing.
- %p: This format specifier is used for Address Printing.

120. What is near, far, and huge pointers in C?

Ans :

- Near Pointers: Near pointers are used to store 16-bit addresses only. Using the near pointer, we can not store the address with a size greaterthan 16 bits.
- Far Pointers: A far pointer is a pointer of 32 bits size. However, information outside the computer ’ s memory from the current segment can also be accessed.
- Huge Pointers: Huge pointer is typically considered a pointer of 32 bits size. But bits located outside or stored outside the segments can also beaccessed.

121. Mention file operations in C.

Ans :

In C programming Basic File Handling Techniques provide the basicfunctionalities that programmers can perform against the system.

C file operations refer to the different possible operations that we canperform on a file in C such as:

1. Creating a new file —— fopen() with attributes as ‘‘a’’ or ‘‘a+’’ or ‘‘w’’ or ‘‘w+’’
2. Opening an existing file – fopen()
3. Reading from file – fscanf() or fgets()
4. Writing to a file – fprintf() or fputs()
5. Moving to a specific location in a file – fseek(), rewind()
6. Closing a file – fclose()

122. Write a Program to check whether a linked list is circular or not.

Ans :

```
#include  <stdio.h>
#include <stdlib.h>
struct Node {
        int data;
        struct Node* next;
};
int isCircular(struct Node* head)
{

        // If given linked list is null then it is circular
        if (head == NULL) {
return 1;
        }
        struct Node* ptr;
        ptr = head->next;
```

```
        while (ptr != NULL && ptr != head) {
                ptr = ptr->next;
        }
return (ptr == head);
}
struct Node* newnode(int data)
{
        struct Node* first;
        first = (struct Node*)malloc(sizeof(struct Node));
        first->data = data;
        first->next = NULL;
        return first;
}
int main()
{
        struct Node* head = newnode(10);
        head->next = newnode(12);
        head->next->next = newnode(14);
        head->next->next->next = newnode(16);
        head->next->next->next->next = head;
        if (isCircular(head)) {
                printf("Linked List is Circular\n");
        }
        else {
                printf("Linked List is Not Circular\n");
        }
        return 0;
}
```

123. Write a program to Merge two sorted linked lists.

```c
#include  <stdio.h>
#include <stdlib.h>
struct Node {
    int data;
    struct Node* next;
};
struct Node* mergeSortedLists(struct Node* a, struct Node* b) {
    struct Node dummy = {0, NULL}, *tail = &dummy;
    while (a && b)
        if (a->data <= b->data) tail = tail->next = a, a = a->next;
        else tail = tail->next = b, b = b->next;
    tail->next = a ? a : b;
    return dummy.next;
}
void printList(struct Node* head) {
    while (head) printf("%d -> ", head->data), head = head->next;
    printf("NULL\n");
}
int main() {
    struct Node *list1, *list2;
    struct Node* mergedList = mergeSortedLists(list1, list2);
    printList(mergedList);
    return 0;
}
```

124. What is the difference between getc(), getchar(), getch() and getche().

Ans :

- getc(): The function reads a single character from an input stream and returns an integer value (typically the ASCII value of the character) if itsucceeds. On failure, it returns the EOF.
- getchar(): Unlike getc(), gechar() can read from standard input; it is equivalent to getc(stdin).
- getch(): It is a nonstandard function and is present in ‘conio.h’ header file which is mostly used by MS-DOS compilers like Turbo C.
- getche(): It reads a single character from the keyboard and displays it immediately on the output screen without waiting for enter key.

REVESION

C Interview Questions

1) What is **C Language**?

a) C Programming Language is developed by Dennis Ritchie and Ken Thompson for use on the UNIX Operating System.

b) C Language is designed for Procedural Programming.

c) C is a high-level and general-purpose programming language that is an ideal language for developing portable system applications.

2) What are the **advantages** of C Language?

a)Simple

b)Portable

c)Structured Programming

d)Fast Speed

e)Memory Management

f)Extensible

3) What is the **Source Code**?

Source Code is a Code which is written by the Programmer in human Readable form with Proper Programming Syntax.

4) What is the **Executable Code**?

Executable Code is the machine-understandable code, which can be executed by a machine (OS).

5) What is **Object Code**?

Object Code is a Sequence of Statements in machine language, and is the output after the Compiler (or) Assembler Converts the Source Code.

6) What is a **Compiler**?

The compiler is a Software module which translate (convert) the Source Code Of a Program into Executable Code.

7) Difference between **Native Compiler** and **Cross Compiler**?

a) The Compiler used to compile source code for the same type of platform only.

The Compiler used to compile source code for different kinds of platform.

b) A Native Compiler generates codes for the same machine.

A Cross Compiler generates code for another machine.

c) Ex: -Turbo C, GCC

Ex:-C51,ARMCC

8) What are the **Compilation stages** and their **commands**?

a)Preprocessor

1) Responsible for including the header files and removal of comments.

2) Responsible for replacements of macros.

3) Responsible for Conditional Compilation.

Command	cc – e filename.c – o filename.i

b)Translator

1) Responsible for checking syntactical error and for converting source code into assembly code.

2) It will decide the scope (visibility).

Command	cc – s filename.i – o filename.s

c) Assembler

1) Responsible for converting assembly code into opcode.

Command	cc – c filename.s – o filename.o

d) Linker

1) Responsible for linking with libraries and adds operating System information.
2) It will create an executable file.
3) Whenever you get the undefined error it's generated by the linker.

Command	cc filename.o – o filename

9) What are the **types of Errors** occurred in C?

There are two types of errors
1)Compile-time
2)Run time
a)Compile Time Error:
This error is generated by Compiler.
a)Pre-Processor Error
b)Translator Error
c)Linker Error
b) Runtime Error
This error is generated by Operating System.
a)Segmentation fault
b) Bus Error
c) Floating-point Exception Error(Divide with zero)

10) What is **Code-Optimization**?

a) Code Optimization is a program transformation technique which tries to improve the code by making it consumes fewer resources so that faster running machine code will result.
b) Optimization should increase the speed and performance of the program.

11) Write a logic for **Set a Bit**, **Clear a Bit** and **Complement a bit?**

a) Set a Bit:

```
num=num | 1 << pos
```

b) Clear a Bit:

num=num & ~ (1 << pos)

c)Complement a Bit:

num=num ^ 1 << pos

12) What is a **Token**?

a)The Token is an identifier. It can be constant, keyword, string literal, etc.
A token is the smallest individual unit in a program.
b)C has the following tokens:

Identifiers: Identifiers refer to the name of the variables, arrays & functions.

Keywords: Keywords are the predefined words that are explained by the compiler.

Constants: Constants are the fixed values that cannot be changed during the execution of a program.

Operators: An operator is a symbol that performs the particular operation.

Special characters: All the characters except alphabets and digits are treated as special characters.

13) What is a **Variable?**

Variable is a name that can be used to store values, it can take different values but one at a time.

14) What is an **Expression?**

1) An Expression is a combination of operators, constants,

variables, and function calls.

2) An Expression can be arithmetic, logical (or) relational.

15) Difference between **While** and **Do-While**?

1) Condition is Checked first then the statement is executed.

Statement is executed at least once, therefore Condition

is checked.

2) No semicolon at the end of the while.

Semicolon at the end of the while.

3)While loop is an entry controlled loop

Do-While is exit controlled loop

16) D**ifference** between **Declaration** and **Definition?**

1) Declaration tells the Compiler about data type and size of the Variable.
Definition allocates memory to the Variable.
2) A Variable (or) the function can be declared any number of times.
A Variable (or) a function can be defined only once.
3) The memory will not be allocated during declaration
Memory will be allocated

17) **Difference** between **Local Variable** and **Global Variable**?

1) A variable which is declared inside function (or) block is known as local variables.
A Variable which is declared outside the function (or) block is known as global variable.
2) Variables are stored in Stack unless specified.
The Compiler decides the storage location of a Variable.

18) What is **Header File**?

Header File is a File that contains the declaration of a library function, global variables, and macro definitions.
The extension of the header file is ".h"

19) **Difference** between **Little Endian** and **Big Endian**?

1)Little Endian means that the lower-order byte of the number is stored in memory at the lowest address, and the higher order byte is stored at the highest address.
2)Big Endian means that the higher order byte of the number is stored in memory at the lowest address, and the lower order byte is stored at the highest address.

20) What is **Pointer**?

The pointer is one of the derived data the type which is useful for storing the address of another variable through which we can access the data indirectly.

21) What are the **Uses** of **Pointer**?

a) Accessing array elements
b) To get the address of a Variable.
c) Returning more than one value from a function.
d) Accessing Dynamically Allocated Memory.
e) Implementing data structures like Linked Lists, Trees.

22) What is **Void Pointer**?

a) Void Pointer is also known as Generic Pointer.
b) A Void Pointer is a Pointer that has no associated data type with it.
c) A Void Pointer can hold the address of any type and can be typecast to any type.
d) Void pointers cannot be dereferenced directly. If we want to dereference we need to typecast it. Because by typecasting it can be known to the compiler how many bytes it needs to be fetched.
e) The generic pointer can hold any type of pointers like char pointer, struct pointer, an array of pointers.
f) malloc () and calloc () return void*type and this allows these functions to be used to allocate memory of any datatype.
g) Void pointer in C is used to implement generic functions in C.

23) What is **Type-Casting**?

It is a way of Converting a variable from one data-type to another data-type.

Syntax : (type)expression

24) What is **NULL Pointer**?

a) If the pointer is holding zero as an address that pointer is called a NULL pointer.
b) A NULL pointer is a pointer that is pointing to nothing.
c)In case, if we don't have an address to be assigned to

a pointer, then we can simply use NULL.
d)If you try to dereference the NULL pointer, the result will be Segmentation fault.

25) What is **Dangling Pointer**?

a) After freeing the dynamic memory if the pointer is still pointing to the freed memory such pointers are called as Dangling pointer
b) To Avoid Dangling Pointer makes the pointer as NULL Pointer.

26) What is **Wild Pointer**?

a) A Pointer which has not been initialized to anything is known as a wild pointer.

b) The pointer may be initialized to a non-null garbage value that may not be a valid address.

c) Uninitialized pointers are known as wild pointers.

d)These Pointers usually point to some arbitrary memory locations and may cause a program to crash (or) misbehave.

27) How to **Avoid Wild Pointer**?

a) Initialize them with the address of a known variable.
b) Explicitly allocate the memory and put the values in the allocated memory.

28) What are the **advantages** of a **Pointer**?

a) Pointer reduces the code and improving the performance, it is used to retrieving string, trees.
b) We can return multiple values from a function using the pointers.

29) What is an **Array**?

The array is one of the derived data the type which is a collection of similar types of elements and which are in contiguous memory locations.

30) What is a **String**?

a) String is null-terminated character array.
b) Strings are collection of characters ended with null character ('\0').

31) Difference between A**rray** and P**ointer**?

a)A data structure consists of a collection of elements

each identified by the array index.

A Programming language object that stores the

memory address of another value located in the

computer memory.

b)

Syntax: data type variable name[elements];

Syntax : data type *variable name ;

c) Stores the value of the variable of homogeneous data type.

Store the address of the variable of same data type as the

pointer's a data type.

d) An array of pointers can be generated.

A pointer to an array can be generated.

e) Used to allocate fixed-size memory.

Used for dynamic memory allocation.

32) What is **Pointer to an Array**?

a) It is also known as the Array Pointer.
b) We are using the pointer to access the components of the array

Syntax: datatype(*var name) [size of array]

c) Ex: int (*a) [10]
Declares a pointer to an array of int.
d) The pointer to the array must be dereferenced to access the value of each element.

33) What is an **Array of Pointers**?

a) It is an Array of the Pointers variables.
b) It is also known as Pointer Array.

Syntax: data type *varname[Array Size]

c) Ex: int *a [10];
Declares and allocates an array of pointers.
d) Each element must be dereferenced Individually.

34) What are the **Functions** in C?

a) The function blocks where the program code is written.
b) The function is a set of instructions placed together to perform a specific task.
c) A function is a subroutine that is designed to perform one task.

35) What is the **use** of writing a **function in C** Programming Language?

a) C functions are used to avoid the rewriting the same code again and again in our program.
b) C functions can be called any number of times from any place of our program.
c) When a program is divided into functions, then any part of our program can easily be tracked.
d)C functions provide the reusability concept, i.e., it breaks the big task into smaller tasks so that it makes the C program more understandable.

36) What is the A**rgument**?

The calling functions send some values to the called function for communication. These values are called arguments (or) parameters.

37) What are the **Actual Arguments**?

a) The Arguments that are mentioned in the function call is known as actual Arguments.
b) These are the values which are actually sent to the called function.

38) What are the **Formal Arguments**?

a) The name of the arguments which are mentioned in the function definition is called formal (or) dummy arguments.

39) What is the difference between **Call By Value** and **Call By Reference?**

1) When a copy of the value is passed to the function, then the original value is not modified.
When a copy of the value is passed to the function, then the original value is modified.
2) Actual arguments and formal arguments are created in separate memory locations.
Actual arguments and formal arguments are created in the same memory location.
3) In this case, actual arguments remain safe as they cannot be modified.
In this case, actual arguments are not reliable, as they are modified.
4) The copies of the actual arguments are passed to the formal arguments.
The addresses of actual arguments are passed to their respective formal arguments.

40) What is **recursion in C**?

When a function calls itself and this process is known as recursion. The function that calls itself is known as a recursive function.

41) What are the A**dvantages** of **recursion**?

a) The use of recursion makes the code more compact and elegant.
b) It simplifies the logic and hence makes the program easier to understand.

42) What are the **Disadvantages of recursion**?

a) The recursive solution is always logical and

it is very difficult to trace.
b) The code written using recursion is less efficient.
Since recursion is a slow process because of many function calls involve overheads.
c) Recursion uses more processor time.

43) What are **Storage Classes**?

a) In addition to data type, each variable has one more

an attribute is known as a storage class.

b) The proper use of storage classes makes our program

efficient and fast.

c) We can specify a storage class while declaring a variable.

Stax :- storage class data type variable name

d) A Storage Class decides about these four aspects of a

variable

(a) Life Time --- Time between the creation and destruction

of a variable.

(b) Scope --- Locations where the variable is available for

use.

(c) Initial Value --- Default value taken by an uninitialized

variable.

(d) Place of Storage---Place in memory where the storage is

allocated for the variable.

44) What are the different **types** of **Storage Classes**?

a) Automatic
b) External
c)Static
d)Register

45) Explanation of **Storage Classes**?

Storage class	**Linkage**	**Storage**	**Initial Value**	**Scope**	**Life**
Auto	None	Stack	**Garbage**	With In Block	End of Block
Extern	External	Data Segment	**Zero**	Global Multiple Files	Till End of Program
Static	Internal	Data Segment	**Zero**	With In Block	Till End of Program
Register	None	CPU Register	**Garbage**	With In Block	End of Block

46) What are the types of **linkages** in C?

There are three types of linkages in C are -
(a) External Linkage
(b)Internal Linkage
1) Local Variables have no linkage, so their scope is only within the block where they are declared.
2) Global Variables and functions have external linkage, so they can be used in any file of the program.
3) Static Global Variables and Static functions have internal linkage so their scope is only in the file where they are declared.

47) What are the **command line arguments**?

a) The argument passed to the main() function while executing the program is known as command line argument.
b) Command-line arguments are the mechanism where we can supply the input at load time.

48) What is **Dynamic Memory Allocation**?

a) In the case of dynamic memory allocation, memory is allocated at runtime and memory can be increased while executing the program. It is used in the linked list.
b) The malloc() or calloc() the function is required to allocate the memory at the runtime.
c) An allocation or deallocation of memory is done at the execution time of a program.

d) No dynamic pointers are required to access the memory.
e) The dynamic memory is implemented using data segments.
f)Less memory space is required to store the variable.

49) What is **Static Memory Allocation?**

a) In the case of static memory allocation, memory is allocated at compile-time, and memory can't be increased while executing the program. It is used in the array.
b) The lifetime of a variable in static memory is the lifetime of a program.
c) The static memory is allocated using the static keyword.
d) The static memory is implemented using stacks or heap

50) **Difference** between **malloc()** and **calloc()** ?

Prototype: void *malloc(size_t size);
Prototype: void *calloc(size_t nmeb,szie_t size);

a) Assigns single block of demanded memory.
Assigns Multiple blocks of the requested memory.
b)The allocated memory is not initialized to zero.
The allocated memory is Initialized to zero by using calloc()
c) malloc is faster.
calloc is safer.

51) What is a **Memory leak**?

If we lose the base address of dynamically allocated memory that memory becomes a leak. (or) It occurs when programmers create a memory in heap and forget to delete it.

52) Define **Structure**?

Structure is one of the user-defined data types which is a collection of different types of data which are in contiguous memory locations.

53) What is **Self Referential Structure** in C?

In One Structure if one of the member is a pointer of the same structure type then that structure is called a Self Referential Structure.

54) What are **Structure Bit fields**?

a)Bit fields are a mechanism where we can allocate memory for storing the data in the form of "bits".

b) Bitfield variable can declare within a structure (or) with in a union. outside of them it is not possible to declare it.

55) What is **Structure Padding**?

a) Allocating extra bytes for a structure variable than the required one is called as structure Padding.
b) That extra unused byte is called as holes in a structure.
c) Structure padding is a compiler dependent concept.
d) Offset Calculation easy, and processing becomes faster.

56) How to **avoid** Structure Padding?

To Avoid Structure Padding, we can use Pragma Pack as well as an attribute.
Ex: - #Pragma pack (1)
Informing the Compiler whatever the data type it should be multiple of one.

57) What is meant by **union**?

The union is one of the user-defined datatype which is a collection of different types of data that are in the same memory location.

58) Difference between **Structure** and **Union**?

a) The keyword struct is used to define a structure.
The keyword union is used to define a union.
b) When a Variable is associated with structure, the compiler allocates the memory for each member. The size of the structure is greater then (or) equal to the sum of the size of Its members.
When a Variable is associated with the union, the compiler allocates the memory by Considering the size of the largest memory. So, the size of
the union is equal to the size of the largest member.
c) Each member within a structure is assigned a unique storage area of location.
Memory allocated is shared by individual members of the union.
d) An individual member can be accessed at a time.
Only one member can be accessed at a time.

59) What is **Typedef**?

Type def is one of the user-defined data types which is useful for providing another name to the existing data type.so,

those declarations can be simplified.

Syntax: typedef olddatatypename new name

60) What is **enum**?

a) enum is one of the user-defined datatype which is useful for providing a name to a Constants. So that understanding the program becomes an easy.
b) It is a Collection of named integer Constants means each element of enumeration is assigned by an integer value.

61) What is the **need of Enumeration**?

a) Enums can be declared in the local scope.
b)Enum names are automatically initialized by the compiler

62) What is a **file**?

a) A file is a named location which stores data (or) information permanently. A file is always stored inside a storage device using the file name.

b) The file is a collection of data that is present in the secondary memory.

63) What are **Macros**? What are its **advantages** and **disadvantages**?

a) Macros are processor the directive which will be replaced at compile-time by Pre-Processor.
c) The disadvantage with macros are that they just replace the code they do not function calls. So, the length of the code increases. Simultaneously it increases the execution time.
d) Similarly the advantage is they can reduce the time for replacing the same values.

64) Difference between **#define** and **typedef**?

a) It is a pre-processor directive.
It is a user-defined data type.
b) #define action takes place in the pre-processor stage.
Typedef action takes place in the translator stage.
c)Replacement takes place
No Replacement takes place

Syntax: #define macro name macro body

Syntax : typedef olddatatypename new name

65) What is a **debugger**?

A debugger or debugging tool is a a computer program that is used to test and debug other programs.

66) In **header files** whether **functions** are **declared** or **defined**?

Functions are declared within the header file. That is function prototypes exist in a header file, not function bodies. They are defined in the library (lib).

67) How do **signed** and **unsigned** numbers affect memory?

In the case of signed numbers, the the first bit is used to indicate whether positive or negative, this leaves you with one bit short. With unsigned numbers, you have all bits available for that number. The effect is best seen in the number range an unsigned 8-bit the number has range 0-255, while the The 8-bit signed number has a range
-128 to +127.

68) What is **Garbage Collection**?

Periodic Collection of all the free memory space to form Contiguous block of free space by an The operating system is called Garbage Collection.

69) What is **Conditional Compilation**?

a) Conditional Compilation directives help to compile a specific portion of the program (or) let us skip compilation of some specific part of the program based on some conditions.
b) It is used to reduce the size of an executable file.

70) **Difference** between **printf()** , **sprintf()** , **fprintf()**?

a) Print function is used to print character stream of data on stdout console.

Syntax: int printf (const char*,...)

b) String Print function instead of printing on console store it on

char buffer which is specified in sprintf.

Syntax: int sprintf(char*str,const char*string,..)

c)fprintf is used to print the string content in the file but not on stdout console.

Syntax: int fprintf (FILE*fptr, const char*str,...)

71) **Difference** between **scanf()**, **sscanf()**, **fscanf()**?

a) scanf () reads formatted input from stdin.

Syntax: int scanf (const char*,...)

b) sscanf () is used to read formatted input from the string.

Syntax: int sscanf(const char*s,const char * format ,...)

c) fscanf () reads formatted data from file and stores it into variables.

Syntax: int fscanf(FILE*stream,const char*format,...)

72) What is the **advantage** of the **register storage class**?

The main advantage of storing the variable as the register is they are stored in CPU memory which is accessed by very fast compared to RAM. This makes the program to get executed faster. Hence these types of variables are mainly used where quick access to them is required.

73) Which one is **faster n++** (or) **n+1** ?

The Expression n++ requires a single machine instruction. The Expression n+1 requires more instructions to carry out this operation. Hence n++ executes faster.

74) How to write a **running C code without main()**?

We will use Pre-processor directives #define with arguments to give an

impression that the the program runs without main. But in reality, it runs

with a hidden main function.

a) Using a macro that defines main

b) Using Token Pasting Operator

75) How can you **avoid including a header** more than once?

One easy technique to avoid multiple inclusions of the same header are to use the #ifndef, #define and #endif preprocessor directives. When you create a header for your the program, you can #define a symbolic name
that is unique to that header.

76) How to find the **size of a variable** without using the **sizeof() operator**?

```
#include<stdio.h>
#define MY_SIZE(var) ((char*)(&var+1) - (char*) (&var))
int main()
{
   int a;
   printf("Value of my_size(a)=%d\n",MY_SIZE(a));
}
```

77) How to **print a semicolon** without **using a semicolon in the code**?

```
#include<stdio.h>

int main()

{

if(printf("%c",59))

{

}

}
```

78) How to **print 1 to 100** without **using a loop**?

```
#include<stdio.h>
void fun(n)
{
if(n<100)
{
n++;
printf("%d",n);
fun(n);
}
}
main()
{
int n=0;
fun(n);
printf("\n");
}
```

79) How to **deallocate memory** without **using free() in C**?

a) Standard Library function realloc() can be used to

deallocate previously allocated memory.

b) If the size is zero then call to realloc is equivalent to

free (ptr).

Ex: realloc(ptr,0);

80) How to **write a program containing two main() functions?**

```
#include<stdio.h>

void main()

{

printf("In 1st Main \n");

func1();

}

#define main func1

void main()
```

```
{
printf("In 2nd Main\n");
}
```

81) **Difference** between **#include<stdio.h>** and **#include"stdio.h"** ?

Both are same, but #include<stdio.h> will search
in predefined path only whereas #include “stdio.h"
searches in its present working directory.

82) What is a **main()** and **Difference** between **int main() and void main()**?

a) main () is an entry point which is in most programming
languages, when compiler begins to compile the program,
it looks for an entry point, and main () acts as an entry
the point in the C program

b) We can say main is thread/process/function that invokes
automatically by the compiler when the program is being
executed.

a) Every function returns a value to the calling
function, at that time main will be a called
function for compiler/os it will return some
value to the compiler before exit here void and
int defines that main will return a void (nothing)
and int will return an integer values to the compiler.

83) When should we use the **register storage specifier**?

If a variable is used most frequently then it should

be declared using register storage specifier, then

possible the compiler gives CPU for its storage

to speed up the lookup of the variable.

84) Tell some **situations where segmentation error** comes?

a) When try to de-referencing a NULL pointer.

b) Trying to access memory which is already deallocated.

c) Using Wild Pointers.

d) Accessing out of array index bounds

e)Stack Over Flow

f)Improper use of scanf()

85) What is **the Function Pointer**?

Function the pointer is like normal pointers but they

have the capability to point a Function (or)

A function pointer is a variable that stores the

address of a function that can later be called

through that function pointer.

86) What is the **use of a Function Pointer**?

Function Pointers can be useful when you want to create a call back mechanism, and need to pass the address of a function to another function.

87) What are the **advantages of Function Pointer**?

a) Unlike normal pointers, a function pointer points to code,

not data. Typically, a function pointer stores the start of

executable code.

b) Unlike normal pointers, we do not allocate deallocate memory using function pointers.

c) A function's name can also be used to get the function's address.

88) Difference between **sizeof() and strlen() ?**

sizeof()	**strlen()**
sizeof is a unary operator	strlen is a predefined function
It can be used with any data type	It can be used with only strings and character arrays
It gives the size in bytes	It just counts the length of the string
It includes slash zero	It doesn't include slash zero
Char str[]="tie" printf("%d",sizeof(str)) = 4	Char str[]="tie" Printf("%d",strlen(str))=3

89) What is meant by **the Reentrant function**?

a) A function is said to be reentrant if there is a provision to interrupt the function in the course of execution, service the interrupt service routine and then resume the earlier going on function, without hampering its earlier course of action.

b) Reentrant functions are used in applications like hardware interrupt handling, recursion, etc.

90) What is **Linker Error?**

Linker is to link the functions calls with function definitions.

If at this stage function definition not found, linker will generate error.

91) Tell some **situations where Linker error** comes?

1)Due to wrong function prototyping.

2)Incorrect Header Files.

3)One of the linker error is writing Main() instead of main()

4)Another linker error is writing print() instead of print().

92) What is the **L-value error?**

This error occurs when we put constants on the left-hand side of

Equal to Operator and variables on the right-hand side of it.

93) What is a **Bus Error?**

a) A bus occurs when a process is trying to access the memory that the CPU cannot physically address (or) memory tried to access by the program is not a valid memory address.

b) A bus error is a low-level hardware related error. When a program tries to access the memory location whose address is not present in the RAM, bus error generates.

94) What are the **causes** of the **bus error?**

Some common causes of bus errors are invalid file descriptors, unreasonable I/O requests, bad memory allocation, misaligned data structures, compiler bugs, and corrupt boot blocks.

95) What is **a.out** file?

a) a.out remains the default output file name for executables created by certain compilers and linkers
b) This is an abbreviated form of assembler output
c) It may store executable code shared libraries or object code.

96) How does **free() (Or) realloc()** know the size of memory to be **deallocated?**

When memory allocation is done, it allocates extra bytes to hold an integer containing the size of the block. This integer is located at the beginning of the block.

97) What is the **size of the empty structure in c?**

zero

98) Where do **macros get stored in the memory layout in C programming**?

1. Macros are not stored in memory anywhere in the final program but instead, the code for the macro is repeated whenever it occurs.
2. As far as the actual compiler is concerned they don't even exist, they've been replaced by the preprocessor before they get that far.

99) what is the **stack frame?**

The set of values pushed for one function call is termed as a stack frame

100) What are the **memory segments** of a **C Programming?**

1. Stack
2. Text segment
3. Data segment
4. Heap

101) What is meant by the **stack segment?**

1. Stack is of LAST IN FIRST OUT Structure.
2. It is used to store all local variables and auto variables
3. It is used for passing arguments to the functions along with the return address of the instruction which is used to be executed after the function call is over.

102) What is meant by **Text Segment?**

1.Text segment is also called a Code segment

2.It contains executable instructions

3.It is read-only and shared memory

103) What are the **types of data segments?**

There are two types of data segments

1.Initialized Data Segment

2.Uninitialized Data Segment

104) What is meant by **Initialized Data Segment?**

1. It is also called a Data Segment.
2. All the global, static, const, and extern variables initialized by the program are stored in the Data Segment.
3. It is not read-only since it can be changed during run time.
4. It can further be classified as read-only area and read-write area

Example:

```
#include<stdio.h>
int a=15;
// Global Variable stored in Initialized data segment in read-write area //
const char  s[]="techinfoex"
//Global Variable stored in initialized data segment in read-only area//
const char * p="techinfoex"
//Global Variable stored in initialized data segment in read-only area//
int main()
```

```
{
static int b=16;  // Static variable stored in Initialized data segment //
}
```

105) What is meant by **Uninitialized Data Segment**?

1.It is also called as Block Started by Symbol(BSS).
2.Every member of this segment is initialized by the
kernel to zero before the program starts executing.
3.All the global and static which are not initialized by the
program are stored in this segment.

Example:

```
#include<stdio.h>
 char c; //  Uninitialized Global Variable //
int main()
{
static int i; // Uninitialized Static Variable//
}
```

106) What is meant by the **Heap segment?**

1.Heap is the segment where dynamic memory allocation
usually takes place.
2.Using malloc and calloc we can allocate memory in the heap segment.
3.The Heap area is shared by all shared libraries and dynamically
loaded modules in a process.

107)**Difference** between **++(*p) ,++*p,*(++p),*++p,*p++,*(p++) and (*p)++ ?**

1) *p represents the value stored in a pointer.

2) ++ is an increment operator used in prefix and postfix expressions

3) * is a dereference operator

4) Precedence of prefix ++ and * is the same.
Associativity of both is right to left

5) Precedence of postfix ++ is higher than both * and prefix ++.
Associativity of postfix ++ is left to right

a) The expression ++(*p) has three operators. In this
Parentheses is having the highest precedence.
Here, the value is incremented at that particular location.

b) The expression ++*p has two operators of same precedence, so the compiler looks for associativity.
Associativity of operators is right to left.
Therefore, the expression is treated as ++(*p).

c) The expression *(++p) has three operators. In this
Parentheses is having the highest precedence.
Here the pointing location is incremented to next location and then the value is fetched.

d) The expression *++p has two operators of same precedence, so the compiler looks for associativity.
Associativity of operators is right to left.

Therefore, the expression is treated as *(++p).

e) The expression *p++ has two operators. In this

postfix is having higher precedence.

So, pointing location is incremented to next

location and then the value is fetched.

f) The expression *(p++) has three operators. In this

Parentheses is having highest precedence.

Here, the pointing location is incremented to

next location and then the value is fetched.

g) The expression (*p) ++ has three operators. In this

Parentheses is having the highest precedence.

Here, the value is incremented at that particular location.

108) Exlain the following

a)const int *ptr

b)int const*ptr

c) const int * const ptr

d)int *const ptr

e) int const *const ptr

a)const int * ptr

ptr is a pointer to an integer constant

value cannot change

ptr can start pointing to other locations

b)int const *ptr

ptr is a pointer to a constant integer

value cannot change

ptr can start pointing to other locations

c)const int *const ptr

ptr is a constant pointer to an integer constant

value cannot be change

ptr cannot point to other variables once assigned

d)int *const ptr

ptr is a constant pointer to an integer

value can change

ptr cannot point to other variables once assigned

e) int const *const ptr

ptr is a constant pointer to an integer constant

value cannot be changed

ptr cannot point to other variables once assigned

109) What is the **use of return 0 in the main()** function?

return 0 indicates Successful "EXIT_STATUS" of the Program

'0' is the value returned by the main( ) function to the Operating System(OS) after the Program is Executed Successfully

But if during the Program Execution some "ERROR" occurs then instead of '0' some random value gets returned by the main( ) function to the Operating System. This indicates that some "ERROR" had occurred during the Program Execution