



Top 65 C Programming Interview Questions and Answers

If you are preparing for a C programming interview, then you have reached the right place. This article covers the top 65 C programming questions and answers which can be asked in your next C interview.

C is one of the oldest programming languages and was developed in 1972 by Dennis Ritchie. It is a general-purpose and structured programming language that is widely used in various tasks, such as developing system applications, desktop applications, operating systems as well as IoT applications.

Due to its flexibility, C is the building block for many other programming languages, such as Java, Python, and C++, which is why many programming job interviews involve C Interview questions.

Here are the most important C programming interview questions and answers that will help you to improve your chances of getting a job by giving you an edge in the job market where most global and local organizations, big or small, are looking for professionals proficient in C programming.

C Programming Interview Questions and Answers

Following are the top C programming interview questions that you must prepare for:

Q1. What is C programming?

Ans. C is a high-level and general-purpose programming language. It is the simple and flexible language used for a variety of scripting system applications, which form a significant part of windows, UNIX, and Linux operating systems.

Q2. Why is C dubbed as a mother language?

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



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

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

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

Ans. The following are the major features of C:

- Machine Independent
- Mid-Level programming language
- Memory Management
- Structured programming language
- Simple and efficient
- Function rich libraries
- Case Sensitive

Q5. Name some different storage class specifiers in C?

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

- Auto
- Register
- Extern
- Static

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

Ans. The data type specifies the type of data used in programming. C language has some predefined data types with different storage capacity:

- **Built-in data types:** It includes int, char, double, float and void
- **Derived data types:** It includes array, pointers, and references
- **User-defined data types:** Union, structure, and Enumeration

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

Ans. The scope and lifetime of any variable defined as the section of the code in which the variable is executable or visible. Variables that are described in the block are only accessible in



the block. The lifetime of the variable defines the existence of the variable before it is destroyed.

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

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

Example:

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

```
int main()
```

```
{
```

```
    int *ptr, q;
```

```
    q = 50;
```

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

```
    ptr = &q;
```

```
    /* display q's value using ptr variable */
```

```
    printf("%d", *ptr);
```

```
    return 0;
```

```
}
```

[Check Out Top Free & Paid Programming Courses & Certifications](#)

Q9. Define Null pointer?



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

- It will assign the pointer variable to the variable with no assigned memory
- In pointer related code we can do error handling by checking null pointer before assigning any pointer variable
- Pass a null pointer to the variable if we don't want to pass any valid memory address

Example:

```
int fun(int *ptr)
```

```
{
```

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

```
return 10;
```

```
}
```

```
fun(NULL);
```

Q10. What is Dangling Pointer?

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

- **Return local variable in a function call**

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

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

```
char *getHello()
```

```
{
```

```
char str[10]; strcpy(str,"Hello!");
```



```
return(str);  
  
}  
  
int main()  
  
{  
  
//str falls out of scope  
  
//function call char *getHello() is now a dangling pointer  
  
printf("%s", getHello());  
  
}
```

- Variable goes out of scope

```
void main()  
  
{  
  
int *p1;  
  
.....  
  
{  
  
int ch;  
  
p1 = &ch;  
  
}  
  
.....  
  
// Here ptr is dangling pointer
```



```
}
```

- De allocation or free variable memory

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

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

```
int main()
```

```
{
```

```
char **strPtr;
```

```
char *str = "Hello!";
```

```
strPtr = &str; free(str);
```

```
//strPtr is now a dangling pointer
```

```
printf("%s", *strPtr);
```

```
}
```

Q11. What is the use of function in C?

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

Uses of functions in C:

- Functions can be used multiple times in a program by calling them whenever required
- In C, functions are used to avoid rewriting of code
- It is easy to track any C program when it is divided into functions

Syntax of a function

```
return_type function_name( parameter list )
```

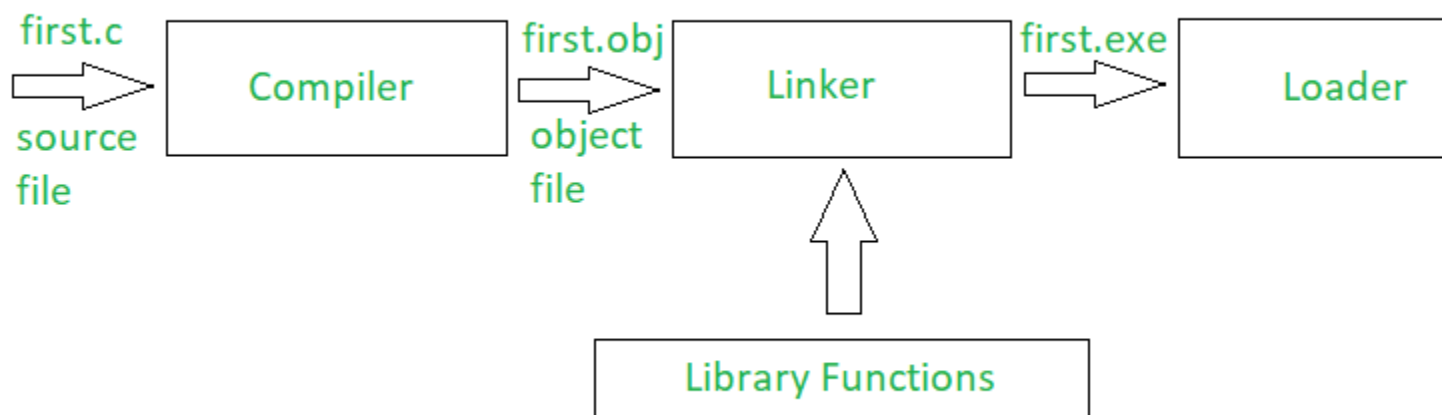


```
{  
  
body of the function  
  
}
```

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

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

Below is the image to show the compilation process:



Also Read>> [Top 10 Most Popular Programming Languages To Learn](#)

Q13. What are header files in C?

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

There are two types of header files:

- Programmer generated a header file



- Files that come with your compiler

Syntax: #include <file>

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

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

- **Arithmetic Operators:** These are used to perform mathematical calculations such as addition, subtraction, multiplication, division, and modulus.

Example:

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

```
int main()
```

```
{
```

```
    int a = 9,b = 4, c;
```

```
    c = a+b;
```

```
    printf("a+b = %d \n",c);
```

```
    c = a-b;
```

```
    printf("a-b = %d \n",c);
```

```
    c = a*b;
```

```
    printf("a*b = %d \n",c);
```

```
    c = a/b;
```




```
printf("a/b = %d \n",c);

c = a%b;

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

return 0;

}
```

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

Example:

```
#include <stdio.h>

int main()

{

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

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

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

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

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

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

printf("%d < %d is %d \n", a, c, a < c);
```



```
printf("%d != %d is %d \n", a, b, a != b);
```

```
printf("%d != %d is %d \n", a, c, a != c);
```

```
printf("%d >= %d is %d \n", a, b, a >= b);
```

```
printf("%d >= %d is %d \n", a, c, a >= c);
```

```
printf("%d <= %d is %d \n", a, b, a <= b);
```

```
printf("%d <= %d is %d \n", a, c, a <= c);
```

```
return 0;
```

```
}
```

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

Example:

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

```
int main()
```

```
{
```

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

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

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

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

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



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

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

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

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

result = !(a != b);

printf("!(a != b) is %d \n", result);

result = !(a == b);

printf("!(a == b) is %d \n", result);

return 0;

}
```

- **Assignment Operators:** These are used to assign value to a variable. The most used assignment operator is '='.

Example:

```
#include <stdio.h>

int main()

{

    int a = 5, c;

    c = a;    // c is 5

    printf("c = %d\n", c);

    c += a;   // c is 10
```



```
printf("c = %d\n", c);

c -= a; // c is 5

printf("c = %d\n", c);

c *= a; // c is 25

printf("c = %d\n", c);

c /= a; // c is 5

printf("c = %d\n", c);

c %= a; // c = 0

printf("c = %d\n", c);

return 0;

}
```

- **Increment and Decrement Operators:** These are used to change the value of an operand (constant or variable) by 1.

Example:

```
#include <stdio.h>

int main()

{

    int a = 10, b = 100;

    float c = 10.5, d = 100.5;

    printf("++a = %d \n", ++a);
```



```
printf("-b = %d \n", -b);

printf(++c = %f \n", ++c);

printf("-d = %f \n", -d);

return 0;

}
```

- **Bitwise Operators:** These are used to perform bit level operations between two variables.

Example: Bitwise OR, Bitwise AND

- **Conditional Operator:** These are used in conditional expressions.

Example: '?' Conditional operator

- **Special Operators: There are some special operators in C used for:**

sizeof(): Returns the size of the memory location

&: Returns the address of a memory location

*****: Pointer of a variable

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

Ans. If you want to perform increment operation, then use 'i++,' which will increase the value by 1. If you want to perform decrement operation, then use 'i-,' it will decrease the value by 1.

Example:

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

```
int main()
```



```
{  
  
    int x = 10, y = 1;  
  
    printf("Initial value of x = %d\n", x);  
  
    printf("Initial value of y = %d\n\n", y);  
  
    y = ++x;  
  
  
    printf("After incrementing by 1: x = %d\n", x);  
  
    printf("y = %d\n\n", y);  
  
    y = -x;  
  
    printf("After decrementing by 1: x = %d\n", x);  
  
    printf("y = %d\n\n", y);  
  
    return 0;  
  
}
```

Also Read >> [5 Best Programming Languages To Learn For Cybersecurity Professionals](#)

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

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

// Global variable

```
float x = 1;
```



```
void my_test() {  
  
    // Local variable called y.  
  
    // This variable can't be accessed in other functions  
  
    float y = 77;  
  
    println(x);  
  
    println(y);  
  
}  
  
void setup() {  
  
    // Local variable called y.  
  
    // This variable can't be accessed in other functions.  
  
    float y = 2;  
  
    println(x);  
  
    println(y);  
  
    my_test();  
  
    println(y);  
  
}
```

Q17. What is static memory allocation in C?

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



Example:

```
void demo
```

```
{
```

```
    int x;
```

```
}
```

```
int main()
```

```
{
```

```
    int y;
```

```
    int c[10];
```

```
    return 1;
```

```
}
```

Q18. What is dynamic memory allocation in C?

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

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

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

Syntax of `calloc()`

```
void *calloc(size_t n, size_t size);
```




Syntax of malloc()

```
void *malloc(size_t n);
```

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

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

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

```
enum {false, true};
```

```
int main()
```

```
{
```

```
    int i = 1;
```

```
    do
```

```
    {
```

```
        printf("%d\n", i);
```

```
        i++;
```

```
        if (i < 15)
```

```
            continue;
```

```
    } while (false);
```

```
    getchar();
```

```
    return 0;
```



```
}
```

Ans. Output: 1

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

Q21. How can a negative integer stored in C?

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

Example:

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

```
int main()
```

```
{
```

```
int a = -4;
```

```
int b = -3;
```

```
unsigned int c = -4;
```

```
unsigned int d = -3;
```

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

```
}
```

Output:

1.333333



1.000000

-0.000000

-1431655764.000000

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

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

Example:

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

```
struct address
```

```
{
```

```
    char city[20];
```

```
    int pin;
```

```
    char phone[14];
```

```
};
```

```
struct employee
```

```
{
```

```
    char name[20];
```

```
    struct address add;
```

```
};
```



```
void main ()  
  
{  
  
    struct employee emp;  
  
    printf("Enter employee information?\n");  
  
    scanf("%s %s %d %s",emp.name,emp.add.city, &emp.add.pin, emp.add.phone);  
  
    printf("Printing employee information...\n");  
  
    printf("name:%s\nCity:%s\nPincode:%d\nPhone:  
%s",emp.name,emp.add.city,emp.add.pin,emp.add.phone);  
  
}
```

Output:

Enter employee information?

Joe

Delhi

110001

1234567890

Printing employee information...

name: Joe

City: Delhi

Pincode: 110001



Phone: 123456789

Also Read>> [Most Popular Programming Languages for Data Science](#)

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

Ans.

```
#include <stdio.h>

int main()

{

    int x, y;

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

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

    x = x + y;

    y = x - y;

    x = x - y;

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

    return 0;

}
```

Q24. Write a C program for Fibonacci series.

Ans.



```
#include<stdio.h>

int main()

{

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

printf("Enter the number of elements:");

scanf("%d",&number);

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

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

{

n3=n1+n2;

printf(" %d",n3);

n1=n2;

n2=n3;

}

return 0;

}
```

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

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



Example: $x=(a+b)$;

```
y=x;
```

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

```
int x,y;
```

```
x=10;
```

```
y=10;
```

```
if(x==y)
```

```
printf("True");
```

```
else
```

```
printf("False");
```

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

Q26. What is the use of extern storage specifier?

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

Q27. Difference between rvalue and lvalue in C?

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

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

Ans. Yes.

**Q29. Define stack?**

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

Basic operations performed in the stack:

- Push
- Pop
- Peek or Top
- isEmpty

Q30. When is the arrow operator used?

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

Syntax:

(pointer_name)->(variable_name)

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

Ans. Yes.

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

Ans: Program to find the factorial of a number

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

```
int factorial_of_a_number(int n)
```

```
{
```




```
int fact = 1, i;

if(n == 0)

return 1;

else

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

{

fact = fact * i;

}

return fact;

}

int main()

{

int n;

printf("Enter the number : ");

scanf("%d",&n);

if(n < 0)

printf("Invalid output");

else

printf("Factorial of the number %d is %d" ,n, factorial_of_a_number(n));
```



```
return 0;
```

```
}
```

Q33. What is a token in C?

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

- Keywords
- Constants
- Identifiers
- Strings
- Operators
- Special symbols

Also Read>> [Best Software Frameworks & Tools To Learn](#)

Q34. Name the keyword used to perform unconditional branching.

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

Q35. What is the use of the comma operator?

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

E.g. `printf ("hello");`

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

Ans.

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

```
void main()
```



```
{  
  
int i, num, sum = 0;  
  
printf("Enter an integer number \n");  
  
scanf ("%d", &num);  
  
for (i = 1; i <= num; i++)  
{  
  
    sum = sum + i;  
  
}  
  
printf ("Sum of first %d natural numbers = %d\n", num, sum);  
  
}
```

Q37. What is the length of an identifier?

Ans. Its length is 32 characters in C.

Q38. What is typecasting in C?

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

There are two types of typecasting in C:

- Implicit conversion
- Explicit conversion

Example:



```
#include <stdio.h>

main() {

    int sum = 17, count = 5;

    double mean;

    mean = (double) sum / count;

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

}
```

Q39. How are random numbers generated?

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

Example:

```
#include <time.h>

#include <stdlib.h>

Srand (time (NULL));

Int r = rand ();
```

Also Read>> [Top Universities Offering Free Online Courses For Programmers](#)

Q40. What are the disadvantages of void pointer?

Ans. Disadvantages of a void pointer:

- Pointer arithmetic is not defined for void pointer
- Void pointers can't be dereferenced



Q41. Define compound statements.

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

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

Ans. Program to print numbers from 1 to 100

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

```
void printNos(unsigned int n)
```

```
{
```

```
    if(n > 0)
```

```
    {
```

```
        printNos(n-1);
```

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

```
    }
```

```
}
```

Q43. What is FIFO?

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

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

Ans. It takes two strings and returns an integer.

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



Ans. Fclose().

Q46. What is the structure?

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

Example:

```
struct Point
```

```
{
```

```
    int x, y;
```

```
} p1; // The variable p1 is declared with 'Point'
```

```
struct Point
```

```
{
```

```
    int x, y;
```

```
};
```

```
int main()
```

```
{
```

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

```
}
```

Q47. Write a program to reverse a number.

Ans.



```
#include <stdio.h>

int main()

{

int n, rev = 0, rem;

printf("\nEnter a number : ");

scanf("%d", &n);

printf("\nReversed Number : ");

while(n != 0)

{

rem = n%10;

rev = rev*10 + rem;

n /= 10;

}

printf("%d\n", rev);

return 0;

}
```

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

Ans.

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



```
#include<conio.h>

void main()

{

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

clrscr(); //It clears the screen.

printf("Enter the number to check prime:");

scanf("%d",&n);

m=n/2;

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

{

if(n%i==0)

{

printf("Number is not prime");

flag=1;

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

}

}

if(flag==0)

printf("Number is prime");
```




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

```
}
```

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

Ans.

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

```
#include<conio.h>
```

```
main()
```

```
{
```

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

```
clrscr(); //It clears the screen.
```

```
printf("enter the number=");
```

```
scanf("%d",&n);
```

```
temp=n;
```

```
while(n>0)
```

```
{
```

```
r=n%10;
```

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

```
n=n/10;
```

```
}
```



```
if(temp==sum)

printf("armstrong number ");

else

printf("not armstrong number");

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

}
```

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

Ans.

```
#include<stdio.h>

#include<conio.h>

main()

{

int n,r,sum=0,temp;

clrscr();

printf("enter the number=");

scanf("%d",&n);

temp=n;

while(n>0)

{
```



```
r=n%10;

sum=(sum*10)+r;

n=n/10;

}

if(temp==sum)

printf("palindrome number ");

else

printf("not palindrome");

getch();

}
```

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

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

Reserved Keywords			
auto	double	int	
break	else	long	struct
case	enum	register	switch
char	extern	return	typedef
const	float	short	union
continue	for	signed	unsigned
default	goto	sizeof	voidvolatilewhile
do	if	static	



Q52. What is a union?

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

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

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

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

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

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

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

Static Memory Allocation	Dynamic Memory Allocation
It is done before the program execution.	It takes place during program execution.
Variables get allocated permanently.	Variables get allocated when the program unit gets active.
Less efficient.	More efficient.



It uses a stack for managing the static allocation of memory.	It uses heap for managing the dynamic allocation of memory.
No memory re-usability.	It allows memory re-usability.
Execution is faster.	Execution is slower.
Memory is allocated at compile time.	The allocation of memory is done at run time.
Memory size can not be modified while execution. Example: Array	Memory size can be modified while execution. Example: Linked List

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

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

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

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

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

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

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

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



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

- ++a <- Prefix increment
- a++ <- Postfix increment

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

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

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

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

Syntax: void *<data type>;

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

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

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

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

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

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

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



Actual Parameters	Formal Parameters
They are included at the time of function call.	They are included at the time of the definition of the function.
Actual Parameters do not require data type but the data type should match with the corresponding data type of formal parameters.	Data types need to be mentioned.
These can be variables, expressions, and constant without data types.	These are variables with their data type.

Q63. Explain modular programming.

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

Q64. What is a sequential access file?

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

Q65. Explain the # pragma directive in C.

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

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



- #pragma warn: It is used to hide the warning messages which are displayed during compilation.
- #pragma GCC poison: This directive is used to remove an identifier completely from the program.
- #pragma GCC dependency: This directive allows you to check the relative dates of the current file and another file.
- #pragma once: It allows the current source file to be included only once in a single compilation.

Visit [Naukri Learning](https://www.naukrilearning.com) website for more