# C INTERVIEW QUESTIONS

**1. Why is C called a mid-level programming language?**
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 billing systems.

**3. What are basic data types supported in the C Programming Language?**
Each variable in C has an associated data type. Each data type requires different amounts of memory and has some specific operations 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 4 categories:

- **Primitive data types:** Primitive data types can be further classified into integer, and floating data types.
    - **Void Types:** Void data types come under primitive data types. Void data types provide no result to their caller and have no value associated with them.
- **User Defined data types:** These data types are defined by the user to make the program more readable.
- **Derived data types:** Data types that are derived from primitive or built-in data types.

**4. What are tokens in C?**
Tokens are identifiers or the smallest single unit in a program that is meaningful 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.

**5. What do you mean by the scope of the variable?**
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.

**6. What are preprocessor directives in C?**
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, Conditional Compilation, and Other directives like #undef, #pragma, etc.

**7. What is the use of static variables in C?**
Static variables in the C programming language are used to preserve the 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 used again in the program without initializing again. Static variables have an initial value assigned to 0 without initialization.

**8. What is the difference between malloc() and calloc() in the C programming language?**
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 memory allocation. | It is used to indicate contiguous memory allocation. |

**9. What do you mean by dangling pointers and how are dangling pointers different from memory leaks in C programming?**
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 this causes our program to crash.
**11. What is recursion in C?**

Recursion is the process of making the function call itself directly or indirectly. A recursive function solves a particular problem by calling a copy of itself and 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.
*For more information, refer to the article – Recursion*

**12. What is the difference between the local and global variables in C?**

Local variables are declared inside a block or function but global variables are 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 outside the block or a function. |
| By default, variables store a garbage value. | By default value of the global value is zero. |
| The life of the local variables is destroyed after the block or a function. | The life of the global variable exists until 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. |

**13. What are pointers and their uses?**

Pointers are used to store the address of the variable or a memory location. Pointer can also be used to refer to another pointer function. The main purpose of the pointer is to save memory space and increase execution 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

**14. What is typedef in C?**

In C programming, typedef is a keyword that defines an alias for an existing 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.*

**Example:**

typedef long long ll

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

Loops are used to execute a block of statements repeatedly. The statement which is to be repeated will be executed n times inside the loop until the given condition is reached. There are two types of loops Entry controlled and Exit-controlled loops in the C programming language. An Infinite loop is a piece of code that lacks a functional exit. So, it repeats indefinitely. 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 the loop. Another can be wrong or unsatisfied break conditions in the program.

**16. What is the difference between type casting and type conversion?**

| **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 need for a casting operator. |
| Type casting is more efficient and reliable. | Type conversion is less efficient and less reliable than type casting. |
| Type casting takes place during the program design by the programmer. | Type conversion is done at compile time. |
| **Syntax:**<br>destination_data_type = (target_data_type) variable_to_be_converted; | **Syntax:**<br>int a = 20; float b; b = a; // a = 20.0000 |

*For more information, refer to the article – Type Casting and Type Conversion*

**17. What are header files and their uses?**

C language has numerous libraries which contain predefined functions to 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.

*For more information, refer to the article – Header Files in C*

**18. What are the functions and their types?**

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.

*For more information, refer to the article – Functions in C*

**19. What is the difference between macro and functions?**

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

*For more information, refer to the article – Macro vs Functions*

**C Programming Interview Questions – Intermediate Level**

**20. How to convert a string to numbers in C?**

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

- **sscanf():** It reads input from a string rather than standard input.
- **stoi() or atoi():** These functions takes a string literal or a character array as an argument and an integer value is returned.

*For more information, refer to the article – String to Numbers in C*

**21. What are reserved keywords?**

Every keyword is meant to perform a specific task in a program. Their meaning is already defined and cannot be used for purposes other than what 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.

*For more information, refer to the article – Variables and Keywords in C*

**22. What is a structure?**

The structure is a keyword that is used to create user-defined data types. The structure allows storing multiple types of data in a single unit. The structure members can only be accessed through the structure variable.

**23. What is union?**

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 variable at 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;
};
```

*For more information, refer to the article – Union in C*

**24. What is an r-value and value?**

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'.*

*For more information, refer to the article – r-value and l-value in C*

**25. What is the difference between call by value and call by reference?**

| 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. |

*For more information, refer to the article – Call by Value and Call by Reference*

**26. What is the sleep() function?**

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.

*For more information, refer to the article – sleep() Function in C*

**27. What are enumerations?**

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, ... };

**28: What is a volatile keyword?**
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.

**31. How is source code different from object code?**

| 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 and is not system specific. | Object code can be modified and is system specific. |
| Source code is less close to the machine and is input to the compiler or any other translator. | Object 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 it does not require any translation. |

*For more information, refer to the article – Source vs Object Code*

**32. What is static memory allocation and dynamic memory allocation?**

- **Static memory allocation:** Memory allocation which is done at compile time is known as static memory allocation. Static memory allocation saves running time. It is faster than dynamic memory allocation as memory allocation is done from the stack. This memory allocation method is less efficient as compared to dynamic memory allocation. It is mostly preferred in the array.
- **Dynamic memory allocation:** Memory allocation done at execution or run time is known as dynamic memory allocation. Dynamic memory allocation is slower than static memory allocation as memory allocation is done from the heap. This memory allocation method is more efficient as compared to static memory allocation. It is mostly preferred in the linked list.

**33. What is pass-by-reference in functions?**
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.
**34. What is a memory leak and how to avoid it?**
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.
*For more information, refer to the article – Memory Leak*
**35. What are command line arguments?**
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*/}*
*For more information, refer to the article – Command Line Arguments in C*
**36. What is an auto keyword?**
Every local variable of a function is known as an automatic variable in the C language. Auto is the default storage class for all the variables which are declared inside 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.
*For more information, refer to the article – Storage Classes in C*

**40. Explain modifiers.**
Modifiers are keywords that are used to change the meaning of basic data types 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

**44. What is the use of an extern storage specifier?**
The extern keyword is used to extend the visibility of the C variables and functions in the C language. Extern is the short name for external. It is used when a particular file needs to access a variable from any other file. Extern keyword increases the redundancy and variables with extern keyword are only declared not defined. By default functions are visible throughout the program, so there is no need to declare or define extern functions.
**45. What is the use of printf() and scanf() functions in C Programming language? Also, explain format specifiers.**
**printf()** function is used to print the value which is passed as the parameter to it on the console screen.
**Syntax:**
*print("%X",variable_of_X_type);*
**scanf()** method, reads the values from the console as per the data type specified.
**Syntax:**
*scanf("%X",&variable_of_X_type);*
In C format specifiers are used to tell the compiler what type of data will be present 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 integer value.
- %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.

*For more information, refer to the article – Format Specifier in C*

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

- **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 greater than 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 be accessed.

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

- **getc():** The function reads a single character from an input stream and returns an integer value (typically the ASCII value of the character) if it succeeds. 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.

*For more information, refer to the article – Difference between getc(), getchar(), getch(), getche()*