Let's get Started

What is C Language?

C programming language, the pioneer of programming languages, is a procedural programming language. Dennis Ritchie created it as a system programming language for writing operating systems. It is one of the most popular programming languages because of its structure, high-level abstraction, machine-independent feature, etc. and is a great starting point for anyone wanting to get into coding.

C is also used a lot in low-level system programming, embedded systems, and hardware. It has also been heavily optimized over the years and is still used to write sophisticated software such as the FreeBSD operating system and the XNU kernel. Low-level memory access, a small collection of keywords, and a clean style are all qualities that make the C language excellent for system programmings, such as **operating system** or **compiler development**.

C is a low-level programming language that can be directly interfaced with the processor. It offers minimal abstraction and maximal control, making it an attractive option for developers who want to write efficient code.

In this article, you will get to know the latest C Interview Questions & Answers you could expect as a fresher, intermediate, and experienced candidate.

C Basic Interview Questions

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

C has characteristics of both assembly-level i.e. low-level and higher-level languages. So as a result, C is commonly called a middle-level language. Using C, a user can write an operating system as well as create a menu-driven consumer billing system.

2. What are the features of the C language?

Some features of the C language are-

1. It is Simple And Efficient.
2. C language is portable or Machine Independent.
3. C is a mid-level Programming Language.
4. It is a structured Programming Language.
5. It has a function-rich library.
6. Dynamic Memory Management.
7. C is super fast.
8. We can use pointers in C.
9. It is extensible.

3. What is a token?

The individual elements of a program are called Tokens. There are following 6 types of tokens are available in C:

- Identifiers
- Keywords
- Constants
- Operators
- Special Characters
- Strings

4. What is the use of printf() and scanf() functions? Also explain format specifiers?

- **printf()** is used to print the output on the display.
- **scanf()** is used to read formatted data from the keyboard.

Some datatype format specifiers for both printing and scanning purposes are as follows:

- **%d**: It's a datatype format specifier for printing and scanning an integer value.
- **%s**: It's a datatype format specifier for printing and scanning a string.
- **%c**: It's a datatype format specifier for displaying and scanning a character value.
- **%f**: The datatype format specifier %f is used to display and scan a float value.

5. What's the value of the expression 5["abxdef"]?

The answer is **'f'.**

Explanation: The string mentioned "abxdef" is an array, and the expression is equal to "abxdef"[5]. Why is the inside-out expression equivalent? Because a[b] is equivalent to *(a + b) which is equivalent to *(b + a) which is equivalent to b[a].

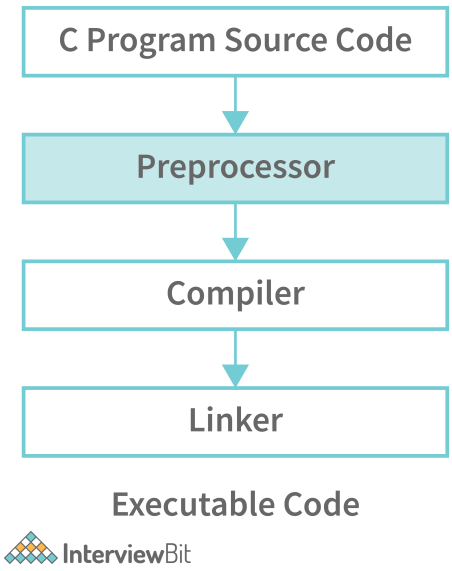
6. What is a built-in function in C?

The most commonly used built-in functions in C are sacnf(), printf(), strcpy, strlwr, strcmp, strlen, strcat, and many more.

Built-function is also known as library functions that are provided by the system to make the life of a developer easy by assisting them to do certain commonly used predefined tasks. For example, if you need to print output or your program into the terminal, we use printf() in C.

7. What is a Preprocessor?

A preprocessor is a software program that processes a source file before sending it to be compiled. Inclusion of header files, macro expansions, conditional compilation, and line control are all possible with the preprocessor.



8. In C, What is the #line used for?

In C, #line is used as a preprocessor to re-set the line number in the code, which takes a parameter as line number. Here is an example for the same.

```
#include <stdio.h>                              /*line 1*/
                                                /*line 2*/
    int main(){                                 /*line 3*/
                                                /*line 4*/
    printf("Hello world\n");                    /*line 5*/
    //print current line                        /*line 6*/
    printf("Line: %d\n",__LINE__);              /*line 7*/
    //reset the line number by 36               /*line 8*/
    #line 36    /*reseting*/
    //print current line                        /*line 36*/
    printf("Line: %d\n",__LINE__);              /*line 37*/
    printf("Bye bye!!!\n");                     /*line 39*/
                                                /*line 40*/
    return 0;                                   /*line 41*/
}                                               /*line 42*/
```

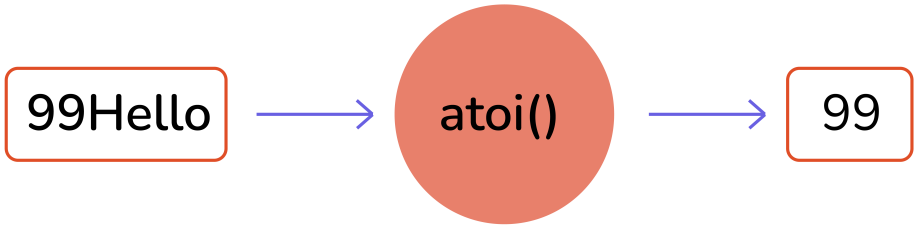
9. How can a string be converted to a number?

The function takes the string as an input that needs to be converted to an integer.

```
int atoi(const char *string)
```

Return Value:

- On successful conversion, it returns the desired integer value
- If the string starts with alpha-numeric char or only contains alpha-num char, 0 is returned.
- In case string starts with numeric character but is followed by alpha-num char, the string is converted to integer till the first occurrence of alphanumeric char.



Converting String to Number

10. How can a number be converted to a string?

The function takes a pointer to an array of char elements that need to be converted, and a format string needs to be written in a buffer as a string

```
int sprintf(char *str, const char *format, ...)
```

The output after running the above code:

Output: Value of Pi = 3.141593

11. What is recursion in C?

When a function in C calls a copy of itself, this is known as recursion. To put it another way, when a function calls itself, this technique is called Recursion. Also, this function is known as recursive function.

Syntax of Recursive Function:

```
void do_recursion()
{
... .. ...
do_recursion();
... .. ...]
}
int main()
{
... .. ...
do_recursion();
... .. ...
}
```

12. Why doesn’t C support function overloading?

After you compile the C source, the symbol names need to be intact in the object code. If we introduce **function overloading** in our source, we should also provide name mangling as a preventive measure to avoid function name clashes. Also, as C is not a strictly typed language many things(ex: data types) are convertible to each other in C. Therefore, the complexity of overload resolution can introduce confusion in a language such as C.

When you compile a C source, symbol names will remain intact. If you introduce function overloading, you should provide a name mangling technique to prevent name clashes. Consequently, like C++, you'll have machine-generated symbol names in the compiled binary.

Additionally, C does not feature strict typing. Many things are implicitly convertible to each other in C. The complexity of overload resolution rules could introduce confusion in such kind of language

13. What is the difference between global int and static int declaration?

The difference between this is in scope. A truly global variable has a global scope and is visible everywhere in your program.

```
#include <stdio.h>

int my_global_var = 0;

int
main(void)

{
  printf("%d\n", my_global_var);
  return 0;
}
```

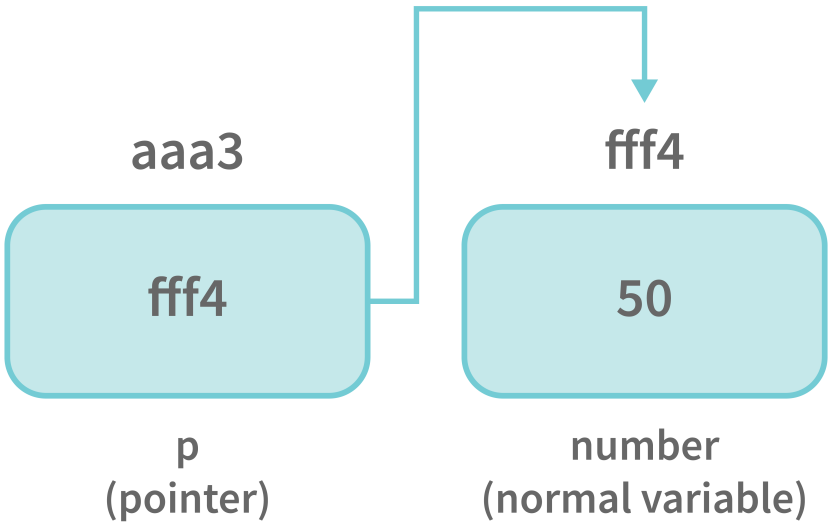
global_temp is a global variable that is visible to everything in your program, although to make it visible in other modules, you'd need an ”extern int global_temp; ” in other source files if you have a multi-file project.

A static variable has a local scope but its variables are not allocated in the stack segment of the memory. It can have less than global scope, although - like global variables - it resides in the .bss segment of your compiled binary.

```c
#include <stdio.h>

int
myfunc(int val)

{
    static int my_static_var = 0;

    my_static_var += val;
    return my_static_var;
}

int
main(void)

{
   int myval;

   myval = myfunc(1);
   printf("first call %d\n", myval);

   myval = myfunc(10);

   printf("second call %d\n", myval);

   return 0;
}
```

14. What is a pointer in C?

A pointer is a variable that stores or points to another variable's address. The value of a variable is stored in a normal variable, whereas the address of a variable is stored in a pointer variable.



15. Difference between const char* p and char const* p?

- const char* p is a pointer to a const char.
- char const* p is a pointer to a char const.

Since const char and char const are the same, it's the same.

16. What is pointer to pointer in C?

In C, a pointer can also be used to store the address of another pointer. A double pointer or pointer to pointer is such a pointer. The address of a variable is stored in the first pointer, whereas the address of the first pointer is stored in the second pointer.

The syntax of declaring a double pointer is given below:

```
int **p; // pointer to a pointer which is pointing to an integer
```

17. Why n++ executes faster than n+1 ?

n++ being a unary operation, it just needs one variable. Whereas, n = n + 1 is a binary operation that adds overhead to take more time (also binary operation: n += 1). However, in modern platforms, it depends on few things such as processor architecture, C compiler, usage in your code, and other factors such as hardware problems.

While in the modern compiler even if you write n = n + 1 it will get converted into n++ when it goes into the optimized binary, and it will be equivalently efficient.

```
$ time perl -le '$n=0; foreach (1..100000000) { $n++ }'
real    0m2.389s
user    0m2.371s
sys 0m0.015s

$ time perl -le '$n=0; foreach (1..100000000) { $n=$n+1 }'
real    0m4.469s
user    0m4.442s
sys 0m0.020s
```

18. What is typecasting in C?

Typecasting is the process to convert a variable from one datatype to another. If we want to store the large type value to an int type, then we will convert the data type into another data type explicitly.

Syntax: `(data_type)expression;`

For Example:

```
int x;
for(x=97; x<=122; x++)
{
   printf("%c", (char)x);   /*Explicit casting from int to char*/
}
```

19. What are the advantages of Macro over function?

Macro on a high-level copy-paste, its definitions to places wherever it is called. Due to which it saves a lot of time, as no time is spent while passing the control to a new function and the control is always with the callee function. However, one downside is the size of the compiled binary is large but once compiled the program comparatively runs faster.

20. What are Enumerations?

Enumeration, also known as Enum in C, is a user-defined data type. It consists of constant integrals or integers that have names assigned to them by the user. Because the integer values are named with enum in C, the whole program is simple to learn, understand, and maintain by the same or even different programmer.

21. When should we use the register storage specifier?

If a variable is used frequently, it should be declared with the register storage specifier, and the compiler may allocate a CPU register for its storage to speed up variable lookup.

C Intermediate Interview Questions

22. Specify different types of decision control statements?

All statements written in a program are executed from top to bottom one by one. Control statements are used to execute/transfer the control from one part of the program to another depending on the condition.

- If-else statement.
  - normal if-else statement.
  - Else-if statement
  - nested if-else statement.
- Switch statement.

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

- The term "r-value" refers to a data value stored in memory at a given address. An r-value is an expression that cannot have a value assigned to it, hence it can only exist on the right side of an assignment operator(=).
- The term "l-value" refers to a memory location that is used to identify an object. The l-value can be found on either the left or right side of an assignment operator(=). l-value is frequently used as an identifier.

24. What is the difference between malloc() and calloc()?

calloc() and malloc() are memory dynamic memory allocating functions. The main difference is that malloc() only takes one argument, which is the number of bytes, but calloc() takes two arguments, which are the number of blocks and the size of each block.

25. What is the difference between struct and union in C?

A struct is a group of complex data structures stored in a block of memory where each member on the block gets a separate memory location to make them accessible at once

Whereas in the union, all the member variables are stored at the same location on the memory as a result to which while assigning a value to a member variable will change the value of all other members.

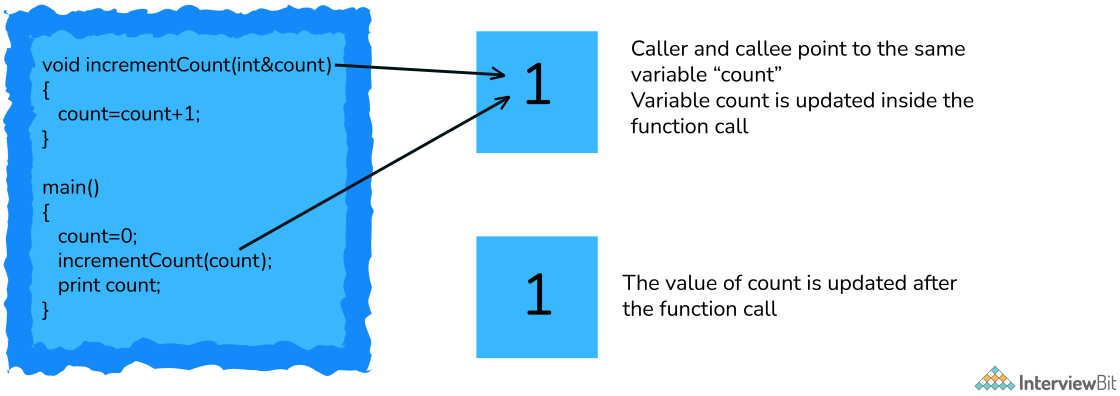
```c
/* struct & union defination*/
struct bar {
    int a;  // we can use a & b both simultaneously
    char b;
}   bar;

union foo {
    int a;  // we can't use both a and b simultaneously
    char b;
}   foo;

/* using struc and union variables*/

struct bar y;
y.a = 3;    // OK to use
y.b = 'c'; // OK to use

union foo x;
x.a = 3; // OK
x.b = 'c'; // NOl this affects the value of x.a!
```

26. What is call by reference in functions?

When we caller function makes a function call bypassing the addresses of actual parameters being passed, then this is called call by reference. In incall by reference, the operation performed on formal parameters affects the value of actual parameters because all the operations performed on the value stored in the address of actual parameters.

27. What is pass by reference in functions?

In Pass by reference, the callee receives the address and makes a copy of the address of an argument into the formal parameter. Callee function uses the address to access the actual argument (to do some manipulation). If the callee function changes the value addressed at the passed address it will be visible to the caller function as well.



Pass By Reference

28. What is a memory leak? How to avoid it?

When we assign a variable it takes space of our RAM (either heap or RAM)dependent on the size of data type, however, if a programmer uses a memory available on the heap and forgets to a delta it, at some point all the memory available on the ram will be occupied with no memory left this can lead to a memory leak.

```
int main()
{
    char * ptr = malloc(sizeof(int));

    /* Do some work */
    /*Not freeing the allocated memory*/
    return 0;
}
```

To avoid memory leaks, you can trace all your memory allocations and think forward, where you want to destroy (in a good sense) that memory and place delete there. Another way is to use C++ smart pointer in C linking it to GNU compilers.

29. What is Dynamic memory allocation in C? Name the dynamic allocation functions.

C is a language known for its low-level control over the memory allocation of variables in DMA there are two major standard library malloc() and free. The malloc() function takes a single input parameter which tells the size of the memory requested It returns a pointer to the allocated memory. If the allocation fails, it returns NULL. The prototype for the standard library function is like this:

void *malloc(size_t size);
The free() function takes the pointer returned by malloc() and de-allocates the memory. No indication of success or failure is returned. The function prototype is like this:

void free(void *pointer);
There are 4 library functions provided by C defined under <stdlib.h> header file to facilitate dynamic memory allocation in C programming. They are:

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

30. What is typedef?

typedef is a C keyword, used to define alias/synonyms for an existing type in C language. In most cases, we use typedef's to simplify the existing type declaration syntax. Or to provide specific descriptive names to a type.

```
typedef <existing-type> <new-type-identifiers>;
```

typedef provides an alias name to the existing complex type definition. With typedef, you can simply create an alias for any type. Whether it is a simple integer to complex function pointer or structure declaration, typedef will shorten your code.

31. Why is it usually a bad idea to use gets()? Suggest a workaround.

The standard input library gets() reads user input till it encounters a new line character. However, it does not check on the size of the variable being provided by the user is under the maximum size of the data type which makes the system vulnerable to buffer overflow and the input being written into memory where it isn't supposed to.

We, therefore, use gets() to achieve the same with a restricted range of input

> Bonus: It remained an official part of the language up to the 1999 ISO C standard, but it was officially removed by the 2011 standard. Most C implementations still support it, but at least GCC issues a warning for any code that uses it.

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

In practice, the difference is in the location where the preprocessor searches for the included file.

For #include <filename> the C preprocessor looks for the filename in the predefined list of system directories first and then to the directories told by the user(we can use -I option to add directories to the mentioned predefined list).

For #include "filename" the preprocessor searches first in the same directory as the file containing the directive, and then follows the search path used for the #include <filename> form. This method is normally used to include programmer-defined header files.

33. What are dangling pointers? How are dangling pointers different from memory leaks?

The dangling pointer points to a memory that has already been freed. The storage is no longer allocated. Trying to access it might cause a Segmentation fault. A common way to end up with a dangling pointer:

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

char *func()
{
    char str[10];
    strcpy(str,"Hello!");
    return(str);
}
```

You are returning an address that was a local variable, which would have gone out of scope by the time control was returned to the calling function. (Undefined behavior)

```c
*c = malloc(5izeof(int));
free(c);
*c = 3; //writing to freed location!
```

In the figure shown above writing to a memory that has been freed is an example of the dangling pointer, which makes the program crash.

A memory leak is something where the memory allocated is not freed which causes the program to use an undefined amount of memory from the ram making it unavailable for every other running program(or daemon) which causes the programs to crash. There are various tools like O profile testing which is useful to detect memory leaks on your programs.

```c
void function(){
    char *leak = malloc (10);   //leak assigned but not freed
}
```

34. What is the difference between ‘g’ and “g” in C?

In C double-quotes variables are identified as a string whereas single-quoted variables are identified as the character. Another major difference being the string (double-quoted) variables end with a null terminator that makes it a 2 character array.

35. What is a near pointer and a far pointer in C?