http://TechPreparation.com

C Interview Questions

And Answers

2008

C Interview Questions and Answers

What is C language?

The C programming language is a standardized programming language developed in the early 1970s by Ken Thompson and Dennis Ritchie for use on the UNIX operating system. It has since spread to many other operating systems, and is one of the most widely used programming languages. C is prized for its efficiency, and is the most popular programming language for writing system software, though it is also used for writing applications.

printf() Function
What is the output of printf("%d")?

1. When we write printf("%d",x); this means compiler will print the value of x. But as here, there is nothing after %d so compiler will show in output window garbage value.

2. When we use %d the compiler internally uses it to access the argument in the stack (argument stack). Ideally compiler determines the offset of the data variable depending on the format specification string. Now when we write printf("%d",a) then compiler first accesses the top most element in the argument stack of the printf which is %d and depending on the format string it calculated to offset to the actual data variable in the memory which is to be printed. Now when only %d will be present in the printf then compiler will calculate the correct offset (which will be the offset to access the integer variable) but as the actual data object is to be printed is not present at that memory location so it will print what ever will be the contents of that memory location.

3. Some compilers check the format string and will generate an error without the proper number and type of arguments for things like printf(...) and scanf(...).

malloc() Function- What is the difference between "calloc(...)" and "malloc(...)"?

1. calloc(...) allocates a block of memory for an array of elements of a certain size. By default the block is initialized to 0. The total number of memory allocated will be (number_of_elements * size).

malloc(...) takes in only a single argument which is the memory required in bytes. malloc(...) allocated bytes of memory and not blocks of memory like calloc(...).

2. malloc(...) allocates memory blocks and returns a void pointer to the allocated space, or NULL if there is insufficient memory available.

calloc(...) allocates an array in memory with elements initialized to 0 and returns a pointer to the allocated space. calloc(...) calls malloc(...) in order to use the C++ _set_new_mode function to set the new handler mode.

printf() Function- What is the difference between "printf(...)" and "sprintf(...)"?

sprintf(...) writes data to the character array whereas printf(...) writes data to the standard output device.

Compilation How to reduce a final size of executable?

Size of the final executable can be reduced using dynamic linking for libraries.

Linked Lists -- Can you tell me how to check whether a linked list is circular?

Create two pointers, and set both to the start of the list. Update each as follows:

```
while (pointer1) {
pointer1 = pointer1->next;
pointer2 = pointer2->next;
if (pointer2) pointer2=pointer2->next;
if (pointer1 == pointer2) {
print ("circular");
}
}
```

If a list is circular, at some point pointer2 will wrap around and be either at the item just before pointer1, or the item before that. Either way, its either 1 or 2 jumps until they meet.

"union" Data Type What is the output of the following program? Why?

```
#include
main() {
typedef union {
int a;
char b[10];
float c;
}
Union;

Union x,y = {100};
x.a = 50;
strcpy(x.b,"hello");
x.c = 21.50;
printf("Union x : %d %s %f n",x.a,x.b,x.c);
printf("Union y : %d %s %f n",y.a,y.b,y.c);
}
```

What does static variable mean?

there are 3 main uses for the static.
1. If you declare within a function:
It retains the value between function calls

2.If it is declared for a function name:
By default function is extern..so it will be visible from other files if the function declaration is as static..it is invisible for the outer files

3. Static for global variables:
By default we can use the global variables from outside files If it is static global..that variable is limited to with in the file

Advantages of a macro over a function?

Macro gets to see the Compilation environment, so it can expand __ __TIME__ __FILE__ #defines. It is expanded by the preprocessor.

For example, you can't do this without macros
#define PRINT(EXPR) printf(#EXPR "=%d\n", EXPR)

PRINT(5+6*7) // expands into printf("5+6*7=%d", 5+6*7);

You can define your mini language with macros:
#define strequal(A,B) (!strcmp(A,B))

Macros are a necessary evils of life. The purists don't like them, but without it no real work gets done.

What are the differences between malloc() and calloc()?

There are 2 differences.
First, is in the number of arguments. malloc() takes a single argument(memory required in bytes), while calloc() needs 2 arguments(number of variables to allocate memory, size in bytes of a single variable).
Secondly, malloc() does not initialize the memory allocated, while calloc() initializes the allocated memory to ZERO.

What are the different storage classes in C?

C has three types of storage: automatic, static and allocated.

Variable having block scope and without static specifier have automatic storage duration.

Variables with block scope, and with static specifier have static scope. Global variables (i.e, file scope) with or without the static specifier also have static scope.

Memory obtained from calls to malloc(), alloc() or realloc() belongs to allocated storage class.

What is the difference between strings and character arrays?

A major difference is: string will have static storage duration, whereas as a character array will not, unless it is explicity specified by using the static keyword.

Actually, a string is a character array with following properties:

* the multibyte character sequence, to which we generally call string, is used to initialize an array of static storage duration. The size of this array is just sufficient to contain these characters plus the terminating NUL character.

* it not specified what happens if this array, i.e., string, is modified.

* Two strings of same value[1] may share same memory area. For example, in the following declarations:

char *s1 = “Calvin and Hobbes”;
char *s2 = “Calvin and Hobbes”;

the strings pointed by s1 and s2 may reside in the same memory location. But, it is not true for the following:

char ca1[] = “Calvin and Hobbes”;
char ca2[] = “Calvin and Hobbes”;

[1] The value of a string is the sequence of the values of the contained characters, in order.

Difference between const char* p and char const* p

In const char* p, the character pointed by ‘p’ is constant, so u cant change the value of character pointed by p but u can make ‘p’ refer to some other location.

in char const* p, the ptr ‘p’ is constant not the character referenced by it, so u cant make ‘p’ to reference to any other location but u can change the value of the char pointed by ‘p’.

What is hashing?

To hash means to grind up, and that’s essentially what hashing is all about. The heart of a hashing algorithm is a hash function that takes your nice, neat data and grinds it into some random-looking integer.

The idea behind hashing is that some data either has no inherent ordering (such as images) or is expensive to compare (such as images). If the data has no inherent ordering, you can’t perform comparison searches.

If the data is expensive to compare, the number of comparisons used

even by a binary search might be too many. So instead of looking at the data themselves, you'll condense (hash) the data to an integer (its hash value) and keep all the data with the same hash value in the same place. This task is carried out by using the hash value as an index into an array.

To search for an item, you simply hash it and look at all the data whose hash values match that of the data you're looking for. This technique greatly lessens the number of items you have to look at. If the parameters are set up with care and enough storage is available for the hash table, the number of comparisons needed to find an item can be made arbitrarily close to one.

One aspect that affects the efficiency of a hashing implementation is the hash function itself. It should ideally distribute data randomly throughout the entire hash table, to reduce the likelihood of collisions. Collisions occur when two different keys have the same hash value.

There are two ways to resolve this problem. In open addressing, the collision is resolved by the choosing of another position in the hash table for the element inserted later. When the hash table is searched, if the entry is not found at its hashed position in the table, the search continues checking until either the element is found or an empty position in the table is found.

The second method of resolving a hash collision is called chaining. In this method, a bucket or linked list holds all the elements whose keys hash to the same value. When the hash table is searched, the list must be searched linearly.

How can you determine the size of an allocated portion of memory?

You can't, really. free() can , but there's no way for your program to know the trick free() uses. Even if you disassemble the library and discover the trick, there's no guarantee the trick won't change with the next release of the compiler.

Can static variables be declared in a header file?

You can't declare a static variable without defining it as well (this is because the storage class modifiers static and extern are mutually exclusive). A static variable can be defined in a header file, but this would cause each source file that included the header file to have its

own private copy of the variable, which is probably not what was intended.

Can a variable be both const and volatile?

Yes. The const modifier means that this code cannot change the value of the variable, but that does not mean that the value cannot be changed by means outside this code. For instance, in the example in FAQ 8, the timer structure was accessed through a volatile const pointer. The function itself did not change the value of the timer, so it was declared const. However, the value was changed by hardware on the computer, so it was declared volatile. If a variable is both const and volatile, the two modifiers can appear in either order.

Can include files be nested?

Yes. Include files can be nested any number of times. As long as you use precautionary measures , you can avoid including the same file twice. In the past, nesting header files was seen as bad programming practice, because it complicates the dependency tracking function of the MAKE program and thus slows down compilation. Many of today's popular compilers make up for this difficulty by implementing a concept called precompiled headers, in which all headers and associated dependencies are stored in a precompiled state.

Many programmers like to create a custom header file that has #include statements for every header needed for each module. This is perfectly acceptable and can help avoid potential problems relating to #include files, such as accidentally omitting an #include file in a module.

When does the compiler not implicitly generate the address of the first element of an array?

Whenever an array name appears in an expression such as
- array as an operand of the sizeof operator
- array as an operand of & operator
- array as a string literal initializer for a character array
Then the compiler does not implicitly generate the address of the address of the first element of an array.

What is a null pointer?

There are times when it’s necessary to have a pointer that doesn’t point to anything. The macro NULL, defined in , has a value that’s guaranteed to be different from any valid pointer. NULL is a literal zero, possibly cast to void* or char*. Some people, notably C++ programmers, prefer to use 0 rather than NULL.
The null pointer is used in three ways:
1) To stop indirection in a recursive data structure
2) As an error value
3) As a sentinel value

What is the difference between text and binary modes?

Streams can be classified into two types: text streams and binary streams. Text streams are interpreted, with a maximum length of 255 characters. With text streams, carriage return/line feed combinations are translated to the newline n character and vice versa. Binary streams are uninterrupted and are treated one byte at a time with no translation of characters. Typically, a text stream would be used for reading and writing standard text files, printing output to the screen or printer, or receiving input from the keyboard.
A binary text stream would typically be used for reading and writing binary files such as graphics or word processing documents, reading mouse input, or reading and writing to the modem.

What is static memory allocation and dynamic memory allocation?

Static memory allocation: The compiler allocates the required memory space for a declared variable.By using the address of operator,the reserved address is obtained and this address may be assigned to a pointer variable.Since most of the declared variable have static memory,this way of assigning pointer value to a pointer variable is known as static memory allocation. memory is assigned during compilation time.
Dynamic memory allocation: It uses functions such as malloc( ) or calloc( ) to get memory dynamically.If these functions are used to get memory dynamically and the values returned by these functions are assingned to pointer variables, such assignments are known as dynamic memory allocation.memory is assined during run time.

When should a far pointer be used?

Sometimes you can get away with using a small memory model in most of a given program. There might be just a few things that don’t fit in your small data and code segments. When that happens, you can

use explicit far pointers and function declarations to get at the rest of memory. A far function can be outside the 64KB segment most functions are shoehorned into for a small-code model. (Often, libraries are declared explicitly far, so they'll work no matter what code model the program uses.) A far pointer can refer to information outside the 64KB data segment. Typically, such pointers are used with farmalloc() and such, to manage a heap separate from where all the rest of the data lives. If you use a small-data, large-code model, you should explicitly make your function pointers far.

How are pointer variables initialized?

Pointer variable are initialized by one of the following two ways
- Static memory allocation
- Dynamic memory allocation

Difference between arrays and pointers?

- Pointers are used to manipulate data using the address. Pointers use * operator to access the data pointed to by them
- Arrays use subscripted variables to access and manipulate data. Array variables can be equivalently written using pointer expression.

Is using exit() the same as using return?

No. The exit() function is used to exit your program and return control to the operating system. The return statement is used to return from a function and return control to the calling function. If you issue a return from the main() function, you are essentially returning control to the calling function, which is the operating system. In this case, the return statement and exit() function are similar.

What is a method?

Method is a way of doing something, especially a systematic way; implies an orderly logical arrangement (usually in steps).

What is indirection?

If you declare a variable, its name is a direct reference to its value. If you have a pointer to a variable, or any other object in memory, you have an indirect reference to its value.

What is modular programming?

If a program is large, it is subdivided into a number of smaller programs that are called modules or subprograms. If a complex problem is solved using more modules, this approach is known as modular programming.

How many levels deep can include files be nested?

Even though there is no limit to the number of levels of nested include files you can have, your compiler might run out of stack space while trying to include an inordinately high number of files. This number varies according to your hardware configuration and possibly your compiler.

What is the difference between declaring a variable and defining a variable?

Declaring a variable means describing its type to the compiler but not allocating any space for it. Defining a variable means declaring it and also allocating space to hold the variable. You can also initialize a variable at the time it is defined.

What is an lvalue?

An lvalue is an expression to which a value can be assigned. The lvalue expression is located on the left side of an assignment statement, whereas an rvalue is located on the right side of an assignment statement. Each assignment statement must have an lvalue and an rvalue. The lvalue expression must reference a storable variable in memory. It cannot be a constant.

Differentiate between an internal static and external static variable?

An internal static variable is declared inside a block with static storage class whereas an external static variable is declared outside all the blocks in a file.An internal static variable has persistent storage,block scope and no linkage.An external static variable has permanent storage,file scope and internal linkage.

What is the difference between a string and an array?

An array is an array of anything. A string is a specific kind of an array with a well-known convention to determine its length.

There are two kinds of programming languages: those in which a string is just an array of characters, and those in which it's a special type. In C, a string is just an array of characters (type char), with one wrinkle: a C string always ends with a NUL character.

The "value" of an array is the same as the address of (or a pointer to) the first element; so, frequently, a C string and a pointer to char are used to mean the same thing.

An array can be any length. If it's passed to a function, there's no way the function can tell how long the array is supposed to be, unless some convention is used. The convention for strings is NUL termination; the last character is an ASCII NUL (‘’) character.

What is an argument? Differentiate between formal arguments and actual arguments?

An argument is an entity used to pass the data from calling function to the called function. Formal arguments are the arguments available in the function definition. They are preceded by their own data types. Actual arguments are available in the function call.

What are advantages and disadvantages of external storage class?

Advantages of external storage class
1)Persistent storage of a variable retains the latest value
2)The value is globally available

Disadvantages of external storage class
1)The storage for an external variable exists even when the variable is not needed
2)The side effect may produce surprising output
3)Modification of the program is difficult
4)Generality of a program is affected

What is a void pointer?

A void pointer is a C convention for a raw address. The compiler has no

idea what type of object a void Pointer really points to. If you write

int *ip;

ip points to an int. If you write

void *p;

p doesn’t point to a void!

In C and C++, any time you need a void pointer, you can use another pointer type. For example, if you have a char*, you can pass it to a function that expects a void*. You don’t even need to cast it. In C (but not in C++), you can use a void* any time you need any kind of pointer, without casting. (In C++, you need to cast it).

A void pointer is used for working with raw memory or for passing a pointer to an unspecified type.

Some C code operates on raw memory. When C was first invented, character pointers (char *) were used for that. Then people started getting confused about when a character pointer was a string, when it was a character array, and when it was raw memory.

When should a type cast not be used?

A type cast should not be used to override a const or volatile declaration. Overriding these type modifiers can cause the program to fail to run correctly.
A type cast should not be used to turn a pointer to one type of structure or data type into another. In the rare events in which this action is beneficial, using a union to hold the values makes the programmer’s intentions clearer.

When is a switch statement better than multiple if statements?

A switch statement is generally best to use when you have more than two conditional expressions based on a single variable of numeric type.

What is a static function?

A static function is a function whose scope is limited to the current

source file. Scope refers to the visibility of a function or variable. If the function or variable is visible outside of the current source file, it is said to have global, or external, scope. If the function or variable is not visible outside of the current source file, it is said to have local, or static, scope.

What is a pointer variable?

A pointer variable is a variable that may contain the address of another variable or any valid address in the memory.

What is a pointer value and address?

A pointer value is a data object that refers to a memory location. Each memory location is numbered in the memory. The number attached to a memory location is called the address of the location.

What is a modulus operator? What are the restrictions of a modulus operator?

A Modulus operator gives the remainder value. The result of x%y is obtained by (x-(x/y)*y). This operator is applied only to integral operands and cannot be applied to float or double.

Differentiate between a linker and linkage?

A linker converts an object code into an executable code by linking together the necessary build in functions. The form and place of declaration where the variable is declared in a program determine the linkage of variable.

What is a function and built-in function?

A large program is subdivided into a number of smaller programs or subprograms. Each subprogram specifies one or more actions to be performed for a large program. such subprograms are functions. The function supports only static and extern storage classes. By default, function assumes extern storage class. functions have global scope. Only register or auto storage class is allowed in the function

parameters. Built-in functions that predefined and supplied along with the compiler are known as built-in functions. They are also known as library functions.

Why should I prototype a function?

A function prototype tells the compiler what kind of arguments a function is looking to receive and what kind of return value a function is going to give back. This approach helps the compiler ensure that calls to a function are made correctly and that no erroneous type conversions are taking place.

What is Polymorphism ?

'Polymorphism' is an object oriented term. Polymorphism may be defined as the ability of related objects to respond to the same message with different, but appropriate actions. In other words, polymorphism means taking more than one form. Polymorphism leads to two important aspects in Object Oriented terminology - Function Overloading and Function Overriding. Overloading is the practice of supplying more than one definition for a given function name in the same scope. The compiler is left to pick the appropriate version of the function or operator based on the arguments with which it is called. Overriding refers to the modifications made in the sub class to the inherited methods from the base class to change their behavior.

What is Operator overloading ?

When an operator is overloaded, it takes on an additional meaning relative to a certain class. But it can still retain all of its old meanings. Examples:
1) The operators >> and << may be used for I/O operations because in the header, they are overloaded.
2) In a stack class it is possible to overload the + operator so that it appends the contents of one stack to the contents of another. But the + operator still retains its original meaning relative to other types of data.

What is the difference between goto and longjmp() and setjmp()?

A goto statement implements a local jump of program execution, and the longjmp() and setjmp() functions implement a nonlocal, or far,

jump of program execution.
Generally, a jump in execution of any kind should be avoided because it is not considered good programming practice to use such statements as goto and longjmp in your program.

A goto statement simply bypasses code in your program and jumps to a predefined position. To use the goto statement, you give it a labeled position to jump to. This predefined position must be within the same function. You cannot implement gotos between functions.

When your program calls setjmp(), the current state of your program is saved in a structure of type jmp_buf. Later, your program can call the longjmp() function to restore the program's state as it was when you called setjmp().Unlike the goto statement, the longjmp() and setjmp() functions do not need to be implemented in the same function.

However, there is a major drawback to using these functions: your program, when restored to its previously saved state, will lose its references to any dynamically allocated memory between the longjmp() and the setjmp(). This means you will waste memory for every malloc() or calloc() you have implemented between your longjmp() and setjmp(), and your program will be horribly inefficient.

It is highly recommended that you avoid using functions such as longjmp() and setjmp() because they, like the goto statement, are quite often an indication of poor programming practice.