Top 20 C & C++ Interview Questions

By **BitMan**

1. Define Storage Classes and explain application domain.
register - tell to the compiler for use a CPU register for fast access for that variable. This is a suggestion to the compiler. It may not occur if the register set is limited. Most compilers with optimization on do a fair job of allocating variables to registers.
(In C, you cannot use pointers with register variables, C++ does allow this.)
(In C, global items (.bss in Unix/Linux systems) cannot be declared register.)
(In C, you cannot take the address of a register variable.)
(In C++ you cannot use register for items declared at namespace scope.)
auto – in C it's a variable created and initialized when it is defined. It is not visible outside of the block. In C++ auto is no longer a storage class, it now means automatic type. In C++ automatic types do have their constructor called, but this does not imply initialization.
// As an example; for non-simple types you may run a default constructor that does no initialization.
static - defined inside of the function retain its value between calls. In c it is defined as global in a file is visible on for the functions from that file. In C++, not always initialized to 0. as it is in C.) Strongly discouraged in C++, breaks encapsulation. C++, static member functions do not require instantiation of a class to use, static const data members are not in the instantiated object, they are in the class.
// Only a single copy in memory independent of number of instantiated objects.
extern - the definition of the variable is in another file.
// Strongly discouraged in C++, breaks encapsulation.

2. Define the Storage Qualifiers
const - define a variable that cannot change its value along the program execution.
volatile - define a variable that can be changed indirectly. An example can be a counter register that is updated by hardware.Aalso variables that may be changed by another thread.
mutuable - a member of a structure or object can be changed even if the structure, for example is declared const:
Ex: struct complex {mutuable int x; int y;};
const complex Mycomplex = {1, 2};
Mycomplex.x = 3; /* correct */

3. Give an example for a variable "const" and "volatile". Is it possible?
Yes, a status register for a microcontroller.

4. Detect if a linked list is circular.
Need to use 2 pointers, one incrementing by 1 and another by 2. If the list is circular, then pointer that is incremented by 2 elements will pass over the first pointer.

5. Have you any remarks?
#define res(a) a*a
Using in this form the result will not be like we expect. Thinking that "a" is replaced with (2+3), then we will obtain 2+3*2+3 = 11 instead (2+3)*(2+3) = 25.
Use always #define res(a) (a)*(a)

6. Define a "dangling" pointer
Dangling pointer is obtained by using the address of an object which was freed.

7. Any difference between "const int*ptr" and int *const ptr" ?
Yes, it's a major difference. First define a constant data and second define a constant pointer.
// In the first declaration, ptr is a pointer to an integer who's value cannot be changed.
// In the second declaration, ptr is a pointer who's associated address cannot be changed,
// which points to an integer.
// There is a little known rule for unwinding declarations in C and C++.

8. What is the declaration and definition of a variable?
The definition contains the implementation for a method or variable declaration. This is just an example.
// A declaration states that something exists, but does not result in information on memory requirements.
// A definition states how that something is implemented,
// and as a result gives information about memory requirements.

9. Give a solution for a stack overflow situation.
(Mistake: This is an example of recursion possibly causing stack overflow.)
(The solution in this case would be to use iteration instead of recursion.)
```
func_call() {
funct_call();
}
```
Every time the above function is called the return address is stored onto the stack. Calling in this infinite loop will cause a stack overflow.

10. Give 4 examples for an infinite loop.
a. while (1) {}
b. for (;;) {}
c. do {}while{1};
d. label:
goto label;

11. Define Encapsulation
Part of the information can be hidden about an object. Encapsulation isolates the internal functionality from the rest of the application.

12. Define Inheritance
One class, called derived, can reuse the behavior of another class, known as base class. Methods of the base class can be extended by adding new proprieties or methods.

13. Define Polymorphism
A function having the same name, but different type or number of arguments. The method signatures beyond the name differ.
Ex: The below functions can be members of the same class.
```
void calc(int a, int b);
void calc(float a, float b);
```

14. Define Constructor, Default Constructor and Copy Constructor
Constructor is used for creating and initializing an object.
Default Constructor has no arguments and is created by default if no explicit definition is used.
Copy Constructor is used for initializing the member variables with the values of another object of the same class.

Important: Copy Constructor is by default called when an object is returned from a function or when the object is passed as argument by value.
// The compiler will supply a default copy constructor if none is supplied, but it results in a shallow copy.
// To get a deep copy, you must supply a copy constructor.

15. Define the differences between malloc/free and new/delete.
malloc() - allocate memory in the heap, but don't call the object constructor; unlike new().
Free() = returns malloced memory to the heap. No destructor calls result.
new() and delete() - called for objects: construct and destruct its.

16. Define the standard prototypes for main() function
a. int main (void)
b. int main (int argc, char **argv)
c. int main (int argc, char *argv[])

17. What is destroyed in the following code?
Obj *ptr = new Obj[5];
delete ptr;
Only the first element of the Obj[5] vector is deleted because ptr is pointing to the first element of the array.

18. What is a virtual function?
A derived class may use the redefinition for a function from the base class. To be possible, need to declare that function virtual in the base class.
// This insures that the compiler will force you to supply a definition for the function in derived classes.

19. What is an inline function?
It's a directive for the compiler which says that the code of the function is used every time we have a call. In this way are avoided the jump to the function routine and return. Inline function needs to be small.
// Actually a suggestion to the compiler. May not happen.
// The compiler will insert the definition inline where a function call normally would be used.

20. Explain the public, private and protected access specifies.
public - data and methods can be accessed outside of the object class.
protected - data and methods are available only for derived classes.
(And friend classes.)
private - data and methods cannot be accessed outside of the object class. They can be accessed in any derived class.