C++ General Interview Q & A

By
Karthikh Venkat

For

www.students3k.com

1) What is virtual constructors/destructors?

Virtual destructors: If an object (with a non-virtual destructor) is destroyed explicitly by applying the delete operator to a base-class pointer to the object, the base-class destructor function (matching the pointer type) is called on the object.
There is a simple solution to this problem – declare a virtual base-class destructor. This makes all derived-class destructors virtual even though they don't have the same name as the base-class destructor. Now, if the object in the hierarchy is destroyed explicitly by applying the delete operator to a base-class pointer to a derived-class object, the destructor for the appropriate class is called.

Virtual constructor: Constructors cannot be virtual. Declaring a constructor as a virtual function is a syntax error. Does C++ support multilevel and multiple inheritance? Yes.
What are the advantages of inheritance?
• It permits code reusability.
• Reusability saves time in program development.
• It encourages the reuse of proven and debugged high-quality software, thus reducing problem after a system becomes functional.
What is the difference between declaration and definition?
The declaration tells the compiler that at some later point we plan to present the definition of this declaration.
E.g.: void stars () //function declaration
The definition contains the actual implementation.
E.g.: void stars () // declaration
{
for(int j=10; j>=0; j--) //function body
cout<<"*";
cout<<endl;
}

2. What do you mean by pure virtual functions?

A pure virtual member function is a member function that the base class forces derived classes to provide. Normally these member functions have no implementation. Pure virtual functions are equated to zero.
class Shape { public: virtual void draw() = 0; };

3. What is namespace?

Namespaces allow us to group a set of global classes, objects and/or functions under a name. To say it somehow, they serve to split the global scope in sub-scopes known as *namespaces*.
The form to use *namespaces* is:
namespace *identifier* **{** *namespace-body* **}**
Where *identifier* is any valid identifier and *namespace-body* is the set of classes, objects and functions that are included within the *namespace*. For example:

namespace general { int a, b; } In this case, **a** and **b** are normal variables integrated within the **general** *namespace*. In order to access to these variables from outside the namespace we have to use the scope operator **::**. For example, to access the previous variables we would have to put:
general::a general::b
The functionality of *namespaces* is specially useful in case that there is a possibility that a global object or function can have the same name than another one, causing a redefinition error.

4. What is RTTI?
Runtime type identification (RTTI) lets you find the dynamic type of an object when you have only a pointer or a reference to the base type. RTTI is the official way in standard C++ to discover the type of an object and to convert the type of a pointer or reference (that is, dynamic typing). The need came from practical experience with C++. RTTI replaces many homegrown versions with a solid, consistent approach.

5. What is a template?
Templates allow to create generic functions that admit any data type as parameters and return value without having to overload the function with all the possible data types. Until certain point they fulfil the functionality of a macro. Its prototype is any of the two following ones:
template <class *indetifier***>** *function_declaration*; **template <typename** *indetifier***>** *function_declaration*;
The only difference between both prototypes is the use of keyword **class** or **typename**, its use is indistinct since both expressions have exactly the same meaning and behave exactly the same way.

6. What do you mean by inline function?
The idea behind ***inline*** functions is to insert the code of a called function at the point where the function is called. If done carefully, this can improve the applications performance in exchange for increased compile time and possibly (but not always) an increase in the size of the generated binary executable.

7. What is virtual class and friend class?
Friend classes are used when two or more classes are designed to work together and need access to each other's implementation in ways that the rest of the world shouldn't be allowed to have. In other words, they help keep private things private. For instance, it may be desirable for class DatabaseCursor to have more privilege to the internals of class Database than main() has.

8. What is function overloading and operator overloading?

Function overloading: C++ enables several functions of the same name to be defined, as long as these functions have different sets of parameters (at least as far as their types are concerned). This capability is called function overloading. When an overloaded function is called, the C++ compiler selects the proper function by examining the number, types and order of the arguments in the call.
Function overloading is commonly used to create several functions of the same name that perform similar tasks but on different data types.

Operator overloading allows existing C++ operators to be redefined so that they work on objects of user-defined classes. Overloaded operators are syntactic sugar for equivalent function calls. They form a pleasant facade that doesn't add anything fundamental to the language (but they can improve understandability and reduce maintenance costs).

9. Difference between realloc() and free()?
The free subroutine frees a block of memory previously allocated by the malloc subroutine. Undefined results occur if the Pointer parameter is not a valid pointer. If the Pointer parameter is a null value, no action will occur. The realloc subroutine changes the size of the block of memory pointed to by the Pointer parameter to the number of bytes specified by the Size parameter and returns a new pointer to the block. The pointer specified by the Pointer parameter must have been created with the malloc, calloc, or realloc subroutines and not been deallocated with the free or realloc subroutines. Undefined results occur if the Pointer parameter is not a valid pointer.

10. What do you mean by binding of data and functions?
Encapsulation.

11. What is abstraction?
Abstraction is of the process of hiding unwanted details from the user.

12. What is encapsulation?
Packaging an object's variables within its methods is called encapsulation.

13. What is the difference between an object and a class?
Classes and objects are separate but related concepts. Every object belongs to a class and every class contains one or more related objects.
Ø A Class is static. All of the attributes of a class are fixed before, during, and after the execution of a program. The attributes of a class don't change.
Ø The class to which an object belongs is also (usually) static. If a particular object belongs to a certain class at the time that it is created then it almost certainly will still belong to that class right up until the time that it is destroyed.
Ø An Object on the other hand has a limited lifespan. Objects are created and eventually destroyed. Also during that lifetime, the attributes of the object may undergo significant change.

14. What is polymorphism? Explain with an example?
"Poly" means "many" and "morph" means "form". Polymorphism is the ability of an object (or reference) to assume (be replaced by) or become many different forms of object. Example: function overloading, function overriding, virtual functions. Another example can be a plus '+' sign, used for adding two integers or for using it to concatenate two strings.

15. What do you mean by inheritance?
Inheritance is the process of creating new classes, called derived classes, from existing classes or base classes. The derived class inherits all the capabilities of the base class, but can add embellishments and refinements of its own.

16. What is a scope resolution operator?
A scope resolution operator (**::**), can be used to define the member functions of a class outside the class.

17. What are virtual functions?
A virtual function allows derived classes to replace the implementation provided by the base class. The compiler makes sure the replacement is always called whenever the object in question is actually of the derived class, even if the object is accessed by a base pointer rather than a derived pointer. This allows algorithms in the base class to be replaced in the derived class, even if users don't know about the derived class.

18. What is friend function?
As the name suggests, the function acts as a friend to a class. As a friend of a class, it can access its private and protected members. A friend function is not a member of the class. But it must be listed in the class definition.

19. What is the difference between class and structure?
Structure: Initially (in C) a structure was used to bundle different type of data types together to perform a particular functionality. But C++ extended the structure to contain functions also. The major difference is that all declarations inside a structure are by default public.
Class: Class is a successor of Structure. By default all the members inside the class are private.

20. What is public, protected, private?
Ø Public, protected and private are three access specifiers in C++.
Ø Public data members and member functions are accessible outside the class.
Ø Protected data members and member functions are only available to derived classes.
Ø Private data members and member functions can't be accessed outside the class. However there is an exception can be using friend classes.

21. What is an object?
Object is a software bundle of variables and related methods. Objects have state and behavior.

22. What is a class?
Class is a user-defined data type in C++. It can be created to solve a particular kind of problem. After creation the user need not know the specifics of the working of a class.