1. What is a v-table?

A v-table, or virtual function table, is a common way for compilers to manage virtual functions in C++. The table keeps a list of the addresses of all the virtual functions and, depending on the runtime type of the object pointed to, invokes the right function.

2. What is a virtual destructor?

A destructor of any class can be declared to be virtual. When the pointer is deleted, the runtime type of the object will be assessed and the correct derived destructor invoked.

3. How do you show the declaration of a virtual constructor?

There are no virtual constructors.

4. How can you create a virtual copy constructor?

By creating a virtual method in your class which itself calls the copy constructor.

5. How do you invoke a base member function from a derived class in which you've overridden that function?

Base::FunctionName();

6. How do you invoke a base member function from a derived class in which you have not overridden that function?

FunctionName();

7. If a base class declares a function to be virtual, and a derived class does not use the term virtual when overriding that class, is it still virtual when inherited by a third-generation class? Yes, the virtuality is inherited and cannot be turned off.

8. What is the protected keyword used for?

protected members are accessible to the member functions of derived objects.

Some More Exercises

1. Show the declaration of a virtual function taking an integer parameter and returning void. virtual void SomeFunction(int);

2. Show the declaration of a class Square, which derives from Rectangle, which in turn derives from Shape. class Square : public Rectangle {};

3. If, in Exercise 2, Shape takes no parameters, Rectangle takes two (length and width), and Square takes only one (length), show the constructor initialization for Square. Square::Square(int length): Rectangle(length, length){}

4. Write a virtual copy constructor for the class Square (from the preceding question).

class Square { public: // ... virtual Square \* clone() const { return new Square(\*this); } // ... };

5. BUG BUSTERS: What is wrong with this code snippet? void SomeFunction (Shape); Shape \* pRect = new Rectangle; SomeFunction(\*pRect);

nothing

6. BUG BUSTERS: What is wrong with this code snippet

? class Shape() { public: Shape(); virtual ~Shape(); virtual Shape(const Shape&); };

You can't declare a copy constructor to be virtual.