11 Virtual functions and virtual base classes

This chapter shows how the rules governing inheritance between a base class and derived classes are extended in C++ to enable the object-oriented programming concept of polymorphism. Languages implementing this concept permit objects of different classes to respond to a message in different ways.

In C++, such a message is a virtual function call. A virtual function, defined in a base class, can be redefined in derived classes so that objects of these derived classes will respond in different ways to what is, apparently, the same function call. A virtual function definition is bound to the program code, using a pointer, at run-time; this is known as late binding or dynamic binding.

An abstract base class definition holds one or more pure virtual function definitions. No objects of an abstract base class can be declared.

This chapter also shows how the use of a virtual base class can resolve a problem that may otherwise occur with multiple inheritance.

11.1 Selecting virtual functions

The keyword virtual is used in the prototype of a member function given within a base class definition. The virtual function definition can then be overridden by another definition in a derived class that has the same identifier and signature.

The overriding is accomplished in a program through the use of a pointer whose referenced type is that of the base class. If this pointer is then assigned the address of an object of a derived class, where the virtual function has been redefined, it can be used to indirectly select the derived class virtual function definition at run-time.

This contrasts to the indirect selection of an ordinary member function using such a pointer when, as would be expected, the base class definition corresponding to the function identifier and signature is selected.

Virtual function calls are implemented using a table of pointers to such function definitions. Although this is transparent to the user, it imposes a small run-time overhead.

11.2 Example program illustrating virtual functions

The program illustrates the difference in inherited behaviour between ordinary member functions and virtual member functions when they are indirectly selected with a base class pointer.

The program uses the line library in which a simple base class crosses is defined that includes a virtual function write1() and an ordinary member function write2(). The library also holds the definitions for two derived classes stars, and queries in which both write1() and write2() are redefined.
Virtual functions and virtual base classes

In the program file, objects of each of these classes are declared, together with a pointer \( p \) whose referenced class is \( \text{crosses} \). This pointer is assigned, and reassigned, at run-time, to hold the address of a base class object or a derived class object. This pointer is used to indirectly select one of the virtual function \( \text{write1}() \) definitions, at run-time, when the following statement is executed:

\[
p \rightarrow \text{write1}();
\]

If \( p \), through the use of the switch default, holds the address of a base class \( \text{crosses} \) object \( \text{cross} \), then the \( \text{write1}() \) definition in the base class is used which writes out \( \text{cross} \) once.

If \( p \) is assigned the address of a derived class \( \text{stars} \) object \( \text{star} \), then the \( \text{write1}() \) definition in the \( \text{stars} \) class is used which writes out \( \text{star} \) twice.

If \( p \) is assigned the address of a derived class \( \text{queries} \) object \( \text{query} \), then the \( \text{write1}() \) definition in the \( \text{queries} \) class is used which writes out \( \text{query} \) three times.

These alternative actions performed on an object by the virtual function \( \text{write1}() \) can be contrasted by the single action performed by the ordinary member function \( \text{write2}() \) if the statement above is replaced by

\[
p \rightarrow \text{write2}();
\]

In all cases here, when the program is run, the definition for \( \text{write2}() \) given in the base class is used which writes out a \( \text{cross} \), \( \text{star} \), or \( \text{query} \) object once depending on the last address assigned to \( p \).

The indirect selection of a derived class ordinary member function redefinition of an inherited member function can only be achieved through a pointer whose referenced type is the derived class. For example, the \( \text{stars} \) definition of \( \text{write2}() \) would be used if the declaration

\[
\text{stars* } q;
\]

is made before the statement

\[
q \rightarrow \text{write2}();
\]

In this case, the compiler would bind the \( \text{stars} \) definition to the code.