CHAPTERFOUR

Inheritance

4.1 Introduction

Inheritance is one of the cornerstones of OOP because it allows the creation of hierarchical classifications. With inheritance, it is possible to create a general classthat defines traits common to a set of related items. This class may then be inherited by other, more specific classes, each adding only those things that are unique to the inheriting class.

In standard C++ terminology, a class that is inherited is referred to as a base class. The class that does the inheriting is called the derived class. Further, a derived class can be used as a base class for another derived class. In this way, a multilayered class hierarchy can be achieved.

4.2 Introducing Inheritance

C++ supports inheritance by allowing one class to incorporate another class into its declaration. Before discussing the theory and details, let's begin with an example of inheritance. The following class, called road_vehicle, very broadly defines vehicles that travel on the road. It stores the number of wheels a vehicle has and the number of passengers it can carry.

```
class road_vehicle
{
   int wheels;
   int passengers;
public:
   void set_wheels(intnum) { wheels = num; }
   intget_wheels() { return wheels; }
   void set_pass(intnum) { passengers = num; }
   intget_pass() { return passengers; }
};
```

You can use this broad definition of a road vehicle to help define specific types of vehicles. For example, the fragment shown here inherits road_vehicle to create a class called truck.

```
class truck : public road_vehicle
{
     int cargo;
  public:
     void set_cargo(int size) { cargo = size; }
     intget_cargo() { return cargo; }
     void show();
};
```

Because truck inherits road_vehicle, truck includes all of road_vehicle. It then adds cargo to it, along with the supporting member functions.Notice how road_vehicle is inherited. The general form for inheritance is shown here:

```
class  derived-class  : access base-class
{
     body of new class
}
```

Here, access is optional. However, if present, it must be either public, private, orprotected. You will learn more about these options later in this chapter. For now,all inherited classes will use public. Using public means that all the public members of the base class will also be public members of the derived class. Therefore, in the preceding example, members of truck have access to the public member functions of road_vehicle, just as if they had been declared inside truck. However, truck does not have access to the private members of road_vehicle.

For example, truck does not have access to wheels.Here is a program that uses inheritance to create two subclasses of road_vehicle. One is truck and the other is automobile.

```
// Demonstrate inheritance.
#include <iostream.h>
// Define a base class for vehicles.
class road_vehicle
{
     int wheels;
```

```
      int passengers;
   public:
      void set_wheels(intnum) { wheels = num; }
      intget_wheels() { return wheels; }
      void set_pass(intnum) { passengers = num; }
      intget_pass() { return passengers; }
};
// Define a truck.
class truck : public road_vehicle
{
      int cargo;
   public:
      void set_cargo(int size) { cargo = size; }
      intget_cargo() { return cargo; }
      void show();
};
enum type {car, van, wagon};
// Define an automoble.
class automobile : public road_vehicle
{
      enum type car_type;
   public:
      void set_type(type t) { car_type = t; }
      enum type get_type() { return car_type; }
      void show();
};
void truck::show()
{
   cout<< "wheels: " <<get_wheels() << "\n";
   cout<< "passengers: " <<get_pass() << "\n";
   cout<< "cargo capacity in cubic feet: "
   << cargo << "\n";
}

void automobile::show()
{
   cout<< "wheels: " <<get_wheels() << "\n";
   cout<< "passengers: " <<get_pass() << "\n";
   cout<< "type: ";
```

```
   switch(get_type())
   {
      case van: cout<< "van\n";break;
      case car: cout<< "car\n";break;
      case wagon: cout<< "wagon\n";
   }
}
intmain()
{
   truck t1, t2;
   automobile c;

   t1.set_wheels(18);
   t1.set_pass(2);
   t1.set_cargo(3200);

   t2.set_wheels(6);
   t2.set_pass(3);
   t2.set_cargo(1200);

   t1.show();
   cout<< "\n"; t2.show();
   cout<< "\n";

   c.set_wheels(4);
   c.set_pass(6);
   c.set_type(van);

   c.show();
   return 0;
}
```

The output from this program is shown here:

```
wheels: 18 passengers: 2
cargo capacity in cubic feet: 3200

wheels: 6 passengers: 3
cargo capacity in cubic feet: 1200

wheels: 4 passengers: 6 type: van
```

When a base class is inherited as public, its public members become public members of the derived class.As this program shows, the major advantage of inheritance is that it lets you create a base class that can be incorporated into more specific classes. In this way, each derived class can be precisely tailored to its own needs while still being part of a general classification.

One other point: Notice that both truck and automobile include a member function called show( ), which displays information about each object. This illustrates another aspect of polymorphism. Since each show( ) is linked with its own class, the compiler can easily tell which one to call for any given object.Now that you have seen the basic procedure by which one class inherits another, let's examine inheritance in detail.