UNIT 4 INHERITANCE - EXTENDING CLASSES

Structure **Page No.**

4.0 Introduction 52
4.1 Objectives 52
4.2 Concept of Inheritance 52
4.3 Base Class and Derived Class 53
4.4 Visibility Modes 54
4.5 Single Inheritance 54
4.5.1 Private Inheritance
4.5.2 Public Inheritance
4.5.3 Protected Inheritance
4.6 Multiple Inheritance 56
4.7 Nested Classes 57
4.8 Virtual Functions 57
4.9 Summary 59
4.10 Solutions/Answers 59

4.0 INTRODUCTION

In this Unit, you will go through the concept of inheritance using C++. Inheritance allows a class to include the members of other classes without repetition of members. There were three ways to inheritance means, "public parts of super class remain public and protected parts of super class remain protected." Private Inheritance means "Public and Protected Parts of Super Class remain Private in Sub-Class". Protected Inheritance means "Public and Protected Parts of Superclass remain protected in Subclass. We shall also deal with nested classes in this Unit.

4.1 OBJECTIVES

After going through this Unit, you should be able to :

- describe the concepts of inheritance;
- apply inheritance concepts to real-life programs, and
- define different types of inheritance.

4.2 CONCEPT OF INHERITANCE

Inheritance is a concept which is the result of commonality between classes. Due to this mechanism, we need not repeat the declaration as well as member functions in a class if they are already present in another class.

For example, consider the classes namely "minister" and "prime minister". Whatever information is present in minister, the same will be present in prime minister also. Apart from that there will be some extra information in class prime minister due to the extra privileges enjoyed by him. Now, due to the mechanism of inheritance, it is enough only to indicate that information which is a specific to prime minister in its class. In addition, the class prime minister will inherit the information of class minister.

4.3 BASE CLASS AND DERIVED CLASS

Let us take the classes, Employee and Manager. A Manager is an Employee with some additional information.

Now, when we are declaring the classes Employee and Manager without applying the concept of inheritance, they will look as follows:

```
class  Employee
{       public:
        char* name;
        int age;
        char* address;
        int salary;
        char*department;
        int id;
};
```

Now, the class Manager is as follows:

```
Class Manager
{       public:
        char* name;
        int age;
        char* address;
        int salary;
        char*department;
        int id;
        employee* team_members; //He heads a group of employees
        int level;                  // his position in hierarchy of the organisation
        .
        .
        .
        .
};
```

Now, without repeating the entire information of class Employee in class Manager, we can declare the Manager class as follows:

```
class Manager: Public Employee
{       public:
        Employee*Team_members;
        int level;
        .
        .
        .
        .
};
```

The latest declaration of class Manager is the same as that of its previous one, with the exception that we did not repeat the information of class Employee explicitly. This is what is meant by the Application of inheritance mechanism.

Please note that in the above example, Employee is called Base Class and Manager is called Derived Class.

4.4 VISIBILITY MODES

There were a total of three visibility modes. They are private, public, and protected. In the previous Units, we have already learnt about private and public.

If a member of a class is declared as "protected", then only member functions and friends of the class in which it is declared and by member functions and friends of classes derived from this class can use its member.

4.5 SINGLE INHERITANCE

In this Section, you will learn the ways of deriving a class from single class. So, there will be only one base class for the derived class.

4.5.1 Private Inheritance

Consider the following classes:

```
class A { /*.....*/);
class C: private A
{       /*
        .
        .
        .
        .
        */
}
```

All the public parts of class A and all the protected parts of class A, become private members/parts of the derived class C in class C. No private member of class A can be accessed by class C. To do so, you need to write public or private functions in the Base class. A public function can be accessed by any object, however, private function can be used only within the class hierarchy that is class A and class C and friends of these classes in the above cases.

4.5.2 Public Inheritance

Consider the following classes:

```
class A{/*.......*/};
class E: public A
{/*
  :
  :
  :
};
```

Now, all the public parts of class A become public in class E and protected part of A become protected in E

4.5.3 Protected Inheritance

Consider the following classes:

```
class A{/*.....*/};
class E: protected A
{       /*
        .
        .
        .
        */
};
```

Now, all the public and protected parts of class A become protected in class E.

No private member of class A can be accessed by class E. Let us take a single example to demonstrate the inheritance of public and private type in more details. Let us assume a class close_shape as follows:

```
class closed_shape
{
        public:
        .
        .
        .
}
class circle: public closed_shape
        // circle is derived in public access mode from class
        // closed-shape
{
        float x, y;         // Co-ordinates of the centre of the circle
        float radius;
        public:
        .
        .
        .
        .
}

class semi-circle : public circle
{   private:
        .
        .
        .
   public:
        .
        .
        .
        .
}

class rectangle: private closed_shape
{
        float x1, y1;
        float x2,y2;
        public:
        .
        .
        .
        .
};
```

```
class rounded_rectangle : public rectangle
{private:
 public :
            .
            .
            .
}
```

Figure 1 shows the access control for these inherited classes.

Figure 1: Access control

Please note the following in the above diagram:

- Class rectangle is a privately derived class, thus, all the public and protected members of class closed_shape will become private in rectangle class and, therefore, rounded-rectangle will not be able to access as they for this inherited class are private. On the other hand, circle & semi-circle both are derived as public classes and will allow access except for private members.

4.6 MULTIPLE INHERITANCE

A class can have more than one direct base classes.

Consider the following classes:

```
Class A {/* .....*/};
Class B {/* .....*/};
Class C : public A, public B
```

```
{       /*
        .
        .
        .
        .

        */
        };
```

This is called Multiple Inheritance. If a class is having only one base class, then it is known as single inheritance.

In the case of Class C, other than the operations specified in it, the union of operations of classes A and B can also be applied.

4.7 NESTESD CLASSES

A class may be declared as a member of another class. Consider the following:

```
Class M1
{
        int n;
        public:
                int m;
};

class M2
{
        int n;
        public:
                int m;
};

class M3
{       M1   N1;
        public:
                M2   N2;
};
```

Now, N1 and N2 are nested classes of M3. M3 can access only public members of N1 and N2. A nested class is hidden in the lexically enclosing class.

4.8 VIRTUAL FUNCTIONS

Polymorphism is a mechanism that enables same interface functions to work with the whole class hierarchy. Polymorphism mechanism is supported in C++ by the use of virtual functions. The concept of virtual function is related to the concept of dynamic binding. The term Binding refers to binding of actual code to a function call. Dynamic binding also called late binding is a binding mechanism in which the actual function call is bound at run-time and it is dependent on the contents of function pointer at run time. It meant that by altering the content of function pointers, we may be able to call different functions having a same name but different code, that is demonstrating polymorphic behaviour.

Let us look into an example for the above concept:

```
#include <iostream.h>
class employee
{
public:
        char *name;
        char *department;

        employee (char *n, char *d)
        {
                name = n;
                department = d;
        }
        virtual void print ( );
};

void employee:: print ( )
{
        cout << "name:"<<name;
        cout << "department:"   << department;
}

class manager : public employee
{
public:
        short position;
        manager (char *n, char *d, short p) : employee (n, d)
        {
                name = n;
                department = d;
                position = p;
        }
void print ( )
{
        cout << name << "\n" << department << "\n" << position;
}
};
void main ( )
{
        employee* e ("john", "sales");
        manager* m ("james", "marketing", 3);
        e → print ( )
        m →print ( ) ;
}
```

The output will be:
John
Sales
James
marketing
3

☞ Check Your Progress

1) None of the subclass can access _____members of a base class.

2) _____members of a class can only be accessed by another class for which it is a member.

3) Both public and protected members of a class become _____when this class is privately derived.

4) Both public and protected members of a class become _____when this class is protectedly derived.

4.9 SUMMARY

In this Unit, you have been exposed to the concepts of base class and derived classes. A derived class is a class which includes the member of another class. This concept is also known as inheritance. When a derived class has more than one direct base class, then it is called Multiple Inheritance. There were three types of inheritance. We can also declare classes as members of another class. We have also touched on the concept of polymorphism.

4.10 SOLUTIONS/ANSWERS

Check Your Progress

1) Private
2) Public
3) Private
4) Protected.