LAB NO. 11

PUBLIC, PROTECTED AND PRIVATE INHERITANCE IN C++

Lab Objectives

Following are the lab objectives:

1. Public, private and protected inheritance in C++

Instructions

- This is individual Lab work/task.
- Complete this lab work within lab timing.
- Discussion with peers is not allowed.
- Copy paste from Internet will give you **negative marks**.
- Lab work is divided into small tasks, complete all tasks sequentially.

Public, Protected and Private Inheritance in C++ Programming

In C++ inheritance, we can derive a child class from the base class in different access modes. For example,

```
class Base {

.... ... ....

};

class Derived : public Base {

.... ... ....

};
```

Notice the keyword public in the code

```
class Derived : public Base
```

This means that we have created a derived class from the base class in public mode. Alternatively, we can also derive classes in protected or private modes.

These 3 keywords (public, protected, and private) are known as access specifiers in C++ inheritance.

Public, protected and private inheritance in C++

Public, protected, and private inheritance have the following features:

- **Public inheritance** makes public members of the base class public in the derived class, and the protected members of the base class remain protected in the derived class.
- **Protected inheritance** makes the public and protected members of the base class protected in the derived class.
- **Private inheritance** makes the public and protected members of the base class private in the derived class.

Note: private members of the base class are inaccessible to the derived class.

```
class Base {

  public:

    int x;

  protected:

    int y;

  private:

    int z;
};
class PublicDerived: public Base {
    // x is public
    // y is protected
    // z is not accessible from PublicDerived
};
class ProtectedDerived: protected Base {
    // x is protected
    // y is protected
    // z is not accessible from ProtectedDerived
};
class PrivateDerived: private Base {
    // x is private
    // y is private
    // z is not accessible from PrivateDerived
}
```

Example 1: C++ public Inheritance

```
// C++ program to demonstrate the working of public inheritance
#include <iostream>
using namespace std;
class Base {
   private:
    int pvt = 1;
   protected:
    int prot = 2;
   public:
    int pub = 3;
    // function to access private member
    int getPVT() {
        return pvt;
    }
};
class PublicDerived : public Base {
   public:
    // function to access protected member from Base
    int getProt() {
        return prot;
    }
};
int main() {
    PublicDerived object1;
    cout << "Private = " << object1.getPVT() << endl;
    cout << "Protected = " << object1.getProt() << endl;
    cout << "Public = " << object1.pub << endl;
    return 0;
}
```

Output

```
Private = 1
Protected = 2
Public = 3
```

Here, we have derived PublicDerived from Base in public mode.

As a result, in PublicDerived:

prot is inherited as protected.

pub and getPVT() are inherited as public.

pvt is inaccessible since it is private in Base.

Since private and protected members are not accessible from main(), we need to create public functions getPVT() and getProt() to access them:

```
// Error: member "Base::pvt" is inaccessible
```

```
cout << "Private = " << object1.pvt;

// Error: member "Base::prot" is inaccessible

cout << "Protected = " << object1.prot;
```

Notice that the getPVT() function has been defined inside Base. But the getProt() function has been defined inside PublicDerived.

This is because pvt, which is private in Base, is inaccessible to PublicDerived.

However, prot is accessible to PublicDerived due to public inheritance. So, getProt() can access the protected variable from within PublicDerived.

Accessibility in public Inheritance

Accessibility	private members	protected members	public members
Base Class	Yes	Yes	Yes
Derived Class	No	Yes	Yes

Example 2: C++ protected Inheritance

```
// C++ program to demonstrate the working of protected inheritance

#include <iostream>

using namespace std;

class Base {

   private:

    int pvt = 1;

   protected:

    int prot = 2;

   public:

    int pub = 3;

    // function to access private member

    int getPVT() {

        return pvt;

    }

};

class ProtectedDerived : protected Base {
   public:
    // function to access protected member from Base
    int getProt() {
        return prot;
    }
    // function to access public member from Base
    int getPub() {
        return pub;
    }
};
int main() {
    ProtectedDerived object1;
    cout << "Private cannot be accessed." << endl;
    cout << "Protected = " << object1.getProt() << endl;
    cout << "Public = " << object1.getPub() << endl;
    return 0;
}
```

Output

```
Private cannot be accessed.
Protected = 2
Public = 3
```

Here, we have derived ProtectedDerived from Base in protected mode.

As a result, in ProtectedDerived:

prot, pub and getPVT() are inherited as protected.

pvt is inaccessible since it is private in Base.

As we know, protected members cannot be directly accessed from outside the class. As a result, we cannot use getPVT() from ProtectedDerived.

That is also why we need to create the getPub() function in ProtectedDerived in order to access the pub variable.

```
// Error: member "Base::getPVT()" is inaccessible
```

```
cout << "Private = " << object1.getPVT();
// Error: member "Base::pub" is inaccessible
cout << "Public = " << object1.pub;
```

Accessibility in protected Inheritance

Accessibility	private members	protected members	public members
Base Class	Yes	Yes	Yes
Derived Class	No	Yes	Yes (inherited as protected variables)

Example 3: C++ private Inheritance

```
// C++ program to demonstrate the working of private inheritance
#include <iostream>
using namespace std;
class Base {
   private:
    int pvt = 1;
   protected:
    int prot = 2;
   public:
    int pub = 3;
    // function to access private member
    int getPVT() {
        return pvt;
    }
};
class PrivateDerived : private Base {
   public:
    // function to access protected member from Base
    int getProt() {
        return prot;
    }
    // function to access private member
    int getPub() {
        return pub;
    }
};
int main() {
    PrivateDerived object1;
    cout << "Private cannot be accessed." << endl;
    cout << "Protected = " << object1.getProt() << endl;
    cout << "Public = " << object1.getPub() << endl;
    return 0;
}
```

Output

```
Private cannot be accessed.
Protected = 2
Public = 3
```

Here, we have derived PrivateDerived from Base in private mode.

As a result, in PrivateDerived:

prot, pub and getPVT() are inherited as private.

pvt is inaccessible since it is private in Base.

As we know, private members cannot be directly accessed from outside the class. As a result, we cannot use getPVT() from PrivateDerived.

That is also why we need to create the getPub() function in PrivateDerived in order to access the pub variable.

```
// Error: member "Base::getPVT()" is inaccessible

cout << "Private = " << object1.getPVT();

// Error: member "Base::pub" is inaccessible

cout << "Public = " << object1.pub;
```

Accessibility in private Inheritance

Accessibility	private members	protected members	public members
Base Class	Yes	Yes	Yes
Derived Class	No	Yes (inherited as private variables)	Yes (inherited as private variables)

Lab Tasks

Q1). Imagine a publishing company that markets both book and audio-cassette versions of its works. Create a class publication that stores the title and price of a publication. From this class derive two **public** classes:

1. book, which adds a page count and
2. tape, which adds a playing time in minutes.
3. each of these three classes should have getdata() function to get its data from the user at the keyboard and a putdata() function to display its data.

Write a main() program to test the book and tape class by creating instances of them, asking the user to fill in their data with getdata() and then displaying the data with putdata().

Q2). Write a class Person that has attributes of id, name and address. It has a constructor to initialize, a member function to input and a member function to display data members. Create another **protected** class Student that inherits Person class. It has additional attributes of rollnumber and marks. It also has member function to input and display its data members.

Q3). Write a base class Computer that contains data members of wordsize(in bits), memorysize (in megabytes), storagesize (in megabytes) and speed (in megahertz). Derive a **private** Laptop class that is a kind of computer but also specifies the object’s length, width, height, and weight. Member functions for both classes should include a default constructor, a constructor to inialize all components and a function to display data members.

Note: In all lab tasks data members and member functions must declare with three specifier **public**, **protected** and **private**.

Home Activity

Q1). Write a program having a base class Student with data members rollno, name and Class define a member functions getdata() to input values and another function putdata() to display all values. A class Test is derived from class Student with data members T1marks, T2marks, T3marks, Sessional1, Sessional2, Assignment and Final. Also make a function getmarks() to enter marks for all variables except Final and also make a function putmarks() to display result. Make a function Finalresult() to calculate value for final variable using other marks. Then display the student result along with student data.

Q2). Write a program that declares two classes. The parent class is called Simple that has two data members num1 and num2 to store two numbers. It also has four member functions.

- The add() function adds two numbers and displays the result.
- The sub() function subtracts two numbers and displays the result.
- The mul() function multiplies two numbers and displays the result.
- The div() function divides two numbers and displays the result.

The child class is called Complex that overrides all four functions. Each function in the child class checks the value of data members. It calls the corresponding member function in the parent class if the values are greater than 0. Otherwise it displays error message.

Note: In all home tasks data members and member functions must declare with three specifier **public**, **protected** and **private**.