FAST NATIONAL UNIVERSTIY OF COMPUTER AND EMERGING SCIENCES, PESHAWAR

DEPARTMENT OF COMPUTER SCIENCE

CL217 – OBJECT ORIENTED PROGRAMMING LAB

INHERITANCE IN C++

LAB MANUAL # 09

Instructor: Fariba LAIQ

SEMESTER SPRING 2023

OBJECT ORIENTED PROGRAMMING LANGUAGE

Table of Contents

Types of Inheritance w.r.t Access Control

There are three kinds of inheritance w.r.t access control

1) Public Inheritance
2) Private Inheritance
3) Protected Inheritance

Base class member access specifier	Type of Inheritence		
	Public	Protected	Private
Public	Public	Protected	Private
Protected	Protected	Protected	Private
Private	Not accessible (Hidden)	Not accessible (Hidden)	Not accessible (Hidden)

1) Public Inheritance

- ❖ In public inheritance, the public members of the base class become the public members of the derived class.
- ❖ Thus the objects of the derived class can access public members (both data and functions) of the base class.
- ❖ Similarly, the protected data members of the base class also become the protected members of derived class.
- ❖ **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.
- ❖ If a derived class is declared in public mode, then the members of the base class are inherited by the derived class just as they are.

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

Accessibility in public Inheritance

Example

```
#include<iostream>
using namespace std;
class A
{
    private:
    int a1, a2;
    protected:
    int pa1, pa2;
    public:
    void ppp(void)
    {
        cout<<"Value of pa1 of class A: "<<pa1<<endl;
        cout<<"Value of pa2 of class A: "<<pa2<<endl;
    }
}; // end of base class A

//derived class
class B : public A
{
    public:
    void get(void)
    {
        cout<<"Enter value of pa1: "; cin>>pa1;
        cout<<"Enter value of pa2: "; cin>>pa2;
    }
};  // end of derived class B

int main()
{
    B obj;
    obj.get();
    obj.ppp();
} // end of main() function


/*
Output
Enter value of pa1: 23
Enter value of pa2: 33
Value of pa1 of class A: 23
Value of pa2 of class A: 33

*/
```

In the above program, the class B is publicly derived from class A.

The objects of the class B:

- Cannot access the private data members a1 and a2 of base class A.
- Can access the public member function ppp() of base class A.
- Can access the protected data members pa1 and pa2 of base class A.

Test public inheritance

```
// tests publicly
#include <iostream>
#include <conio.h>
using namespace std;
class A //base class
{
private:
    int privdataA;
protected:
    int protdataA;
public:
    int pubdataA;
};
class B : public A //publicly-derived class
{
public:
    void funct()
    {
        int a;
        a = privdataA; //error: not accessible
        a = protdataA; //OK
        a = pubdataA; //OK
    }
};

void main()
{
    int a;
    B objB;
    a = objB.privdataA; //error: not accessible
    a = objB.protdataA; //error: not accessible
    a = objB.pubdataA; //OK (A public to B)
}
```

```
/*
Output

*/
```

2) Private Inheritance

- In Private Inheritance, the objects of the derived class cannot access the public members of the base class.
- Its objects can only access the protected data members of the base class.
- In this case, all the members of the base class become private members in the derived class.
- private inheritance makes the public and protected members of the base class private in the derived class.
- The private members of the base class are always private in the derived class.

Examples

```
#include<iostream>
using namespace std;
class A
{
    private:
    int a1, a2;
    protected:
    int pa1, pa2;
    public:
    void ppp(void)
    {
        cout<<"Value of pa1 of class A: "<<pa1<<endl;
        cout<<"Value of pa2 of class A: "<<pa2<<endl;
    }
}; // end of base class A
//derived class
class B : private A   //privately-derived class
{
    public:
    void get(void)
    {
        cout<<"Enter value of pa1: "; cin>>pa1;
        cout<<"Enter value of pa2: "; cin>>pa2;
        cout<<"Value of pa1 of class A: "<<pa1<<endl;
        cout<<"Value of pa2 of class A: "<<pa2<<endl;
    }
};  // end of derived class B

int main()
{
    B obj;
    obj.get();
    //obj.ppp();
} // end of main() function


/*
Output
Enter value of pa1: 12
Enter value of pa2: 33
Value of pa1 of class A: 12
Value of pa2 of class A: 33
*/
```

In the above program, the class B is derived as private from the base class A.

The objects of the class B:

- Cannot access the private data members a1 and a2 of base class A.
- Cannot access the public member function ppp() of base class A.
- Can only access the protected data members pa1 and pa2 of base class A.

If access specifier is private—that is, the inheritance is **private**—then:

a) The public members of A are private members of B. They can be accessed by the member functions (and friend functions) of B.

b) The protected members of A are private members of B. They can be accessed by the member functions (and friend functions) of B.

c) The private members of A are hidden in B. They cannot be directly accessed in B. They can be accessed by the member functions (and friend functions) of B through the public or protected members of A.

```
// tests publicly- and privately-derived classes
#include <iostream>
#include <conio.h>
using namespace std;
class A //base class
```

```
{
private:
    int privdataA;
protected:
    int protdataA;
public:
    int pubdataA;
};

class B : public A //publicly-derived class
{
public:
    void funct()
        {
            int a;
            a = privdataA; //error: not accessible
            a = protdataA; //OK
            a = pubdataA; //OK
        }
};

class C : private A //privately-derived class
{
public:
    void funct()
        {
            int a;
            a = privdataA; //error: not accessible
            a = protdataA; //OK
            a = pubdataA; //OK
        }
};

void main()
{
    int a;
    B objB;
    a = objB.privdataA; //error: not accessible
    a = objB.protdataA; //error: not accessible
    a = objB.pubdataA; //OK (A public to B)

    C objC;

    a = objC.privdataA; //error: not accessible
    a = objC.protdataA; //error: not accessible
    a = objC.pubdataA; //error: not accessible (A private to C)
}
```

Note: If you don’t supply any access specifier when creating a class, private is assumed.

3) Protected Inheritance

- ❖ The object of the class that is derived as protected can only access the protected member of the base class.
- ❖ The public members of the base class become protected members in the derived class.
- ❖ protected inheritance makes the public and protected members of the base class protected in the derived class.

Examples

```
#include<iostream>
using namespace std;
class A
{
    private:
    int a1, a2;
    protected:
    int pa1, pa2;
    public:
    void ppp(void)
    {
        cout<<"Value of pa1 of class A: "<<pa1<<endl;
        cout<<"Value of pa2 of class A: "<<pa2<<endl;
    }
}; // end of base class A

//derived class
class B : protected A   // protectedly-derived class
{
    public:
    void get(void)
    {
        cout<<"Enter value of pa1: "; cin>>pa1;
        cout<<"Enter value of pa2: "; cin>>pa2;
        cout<<"Value of pa1 of class A: "<<pa1<<endl;
        cout<<"Value of pa2 of class A: "<<pa2<<endl;
    }
};  // end of derived class B

int main()
{
    B obj;
    obj.get();
    //obj.ppp();
} // end of main() function

/*
Output
Enter value of pa1: 12
Enter value of pa2: 33
Value of pa1 of class A: 12
Value of pa2 of class A: 33
*/
```

❖ In the above program, the class B is derived as protected from the base class A. The object of class B:

- Can only access the protected data members pa1 and pa2 of the base class A.

❖ If Access Specifier is protected—that is, the inheritance is protected—then:

a) The public members of A are protected members of B. They can be accessed by the member functions (and friend functions) of B.

b) The protected members of A are protected members of B. They can be accessed by the member functions (and friend functions) of B.

c) The private members of A are hidden in B. They cannot be directly accessed in B. They can be accessed by the member functions (and friend functions) of B through the public or protected members of A.

All in one code:

```
class A
{
        public:
        int pubA;
        private:
        int privA;
        protected:
        int protA;
};

class B: public A
{
        public:
                void inputData()
                {
                        cout<<"Enter value of pubA: "<<endl;
                        cin>>pubA;
                        cout<<"Enter value of protA: "<<endl;
                        cin>>protA;
                //      cout<<"Enter value of privA: "<<endl;
                //      cin>>privA; cannot access priv member of A
                }
                void display()
                {
                        cout<<"Displaying members of class A in class B (public inheritence)"<<endl;
                        cout<<"pub A: "<<pubA<<endl;
                //      cout<<"priv A: "<<privA<<endl;  cannot access private member of A
                        cout<<"prot A: "<<protA<<endl;
                }
};
```

```
class C: private A
{
        public:
                void inputData()
                {
                        cout<<"Enter value of pubA: "<<endl;
                        cin>>pubA;
                        cout<<"Enter value of protA: "<<endl;
                        cin>>protA;
                //      cout<<"Enter value of privA: "<<endl;
                //      cin>>privA; cannot access private member of A
                }
                void display()
                {
                        cout<<"Displaying members of class A in class C (private
                        inheritence)"<<endl;
                        cout<<"pub A: "<<pubA<<endl;
                //      cout<<"priv A: "<<privA<<endl; cannot access private member of A
                        cout<<"prot A: "<<protA<<endl;
                }
};
class D: protected A
{
        public:
                void inputData()
                {
                        cout<<"Enter value of pubA: "<<endl;
                        cin>>pubA;
                        cout<<"Enter value of protA: "<<endl;
                        cin>>protA;
                //      cout<<"Enter value of privA: "<<endl;
                //      cin>>privA; cannot access private member of A
                }
                void display()
                {
                        cout<<"Displaying members of class A in class D (protected
                        inheritence)"<<endl;
                        cout<<"pub A: "<<pubA<<endl;
                //      cout<<"priv A: "<<privA<<endl; cannot access private member of A
                        cout<<"prot A: "<<protA<<endl;
                }
};
```

```
int main()
{
        B b;
        cout<<"Object B created with public inheritence"<<endl;
        b.inputData();
        b.display();
        cout<<"Accessing members of class A from the object of class b in main"<<endl;
        cout<<"PubA: "<<b.pubA<<endl;
//      b.privA;  invalid
//      b.protA;  invalid

        C c;
        cout<<"Object C created with private inheritence"<<endl;
        c.inputData();
        c.display();
        cout<<"Accessing members of class A from the object of class C in main"<<endl;
//      c.pubA; invalid
//      c.privA; invalid
//      c.protA;  invalid

        cout<<"Object D created with protected inheritence"<<endl;
        D d;
        d.inputData();
        d.display();
        cout<<"Accessing members of class A from the object of class D in main"<<endl;
//      d.pubA;
//      d.privA; invalid
//      d.protA; invalid
}
```

- ❖ A new class can be derived from one or more existing classes. Based upon the member of the base classes from which a class is derived, the inheritance is divided into two categories.
    - Single Inheritance
    - Multiple Inheritance

Multiple Inheritance

Single Inheritance: In single inheritance, the new class is derived from only one base class.

Multiple Inheritance: In multiple inheritance, the new class is derived from more than one base classes.

Multiple Inheritance is a feature of C++ where a class can inherit from more than one classes. The constructors of inherited classes are called in the same order in which they are inherited. For example, in the following program, B’s constructor is called before A’s constructor.

A class can be derived from more than one base class.

Eg:

(i) A CHILD class is derived from FATHER and MOTHER class
(ii) A PETROL class is derived from LIQUID and FUEL class.

```
#include<iostream>
using namespace std;
class student
{
	protected:
	string name;
	int marks;

	public:
		student(string name, int marks)
		{
			this->name=name;
			this->marks=marks;
		}

};
class teacher
{
	protected:
	string name;
	int salary;
	public:
	teacher(string name, int salary)
	{
		this->name=name;
		this->salary=salary;
	}

};

class TA:protected student, protected teacher
{
	public:
	TA(string name, int marks, int salary): student(name, marks), teacher(name, salary)
	{

	}


	void display()
	{
		cout<<"Name: "<<student::name<<" Marks: "<<marks<<" Salary: "<<salary<<endl;
	}


};
```

```
int main()
{
		TA t("Ali", 90, 10000);
		t.display();
}
```

More about Public, Protected and Private Inheritance in C++ Programming

https://www.programiz.com/cpp-programming/public-protected-private-inheritance

References

https://beginnersbook.com/2017/08/cpp-data-types/

http://www.cplusplus.com/doc/tutorial/basic_io/

https://www.w3schools.com/cpp/default.asp

https://www.javatpoint.com/cpp-tutorial

https://www.geeksforgeeks.org/object-oriented-programming-in-cpp/?ref=lbp

https://www.programiz.com/

https://ecomputernotes.com/cpp/