	AIR UNIVERSITY
	DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
	EXPERIMENT NO 8

Lab Title: Inheritance

Student Name: _____ **Reg. No:** _____

Objective: _____

_____

LAB ASSESSMENT:

Attributes	Excellent (5)	Good (4)	Average (3)	Satisfactory (2)	Unsatisfactory (1)
Ability to Conduct Experiment					
Ability to assimilate the results					
Effective use of lab equipment and follows the lab safety rules					

Total Marks: _____ Obtained Marks: _____

LAB REPORT ASSESSMENT:

Attributes	Excellent (5)	Good (4)	Average (3)	Satisfactory (2)	Unsatisfactory (1)
Data presentation					
Experimental results					
Conclusion					

Total Marks: _____ Obtained Marks: _____

Date: _____ Signature: _____

EXPERIMENT NO 8

Inheritance

Objective:

- To understand oop concept of inheritance between classes

Equipment Required:

- Visual Studio/ Dev C++

Description:

Inheritance is an" **is-a"** relationship. We use inheritance only if an **is-a** relationship is present between the two classes.

Here are some examples:

- A car is a vehicle.
- Orange is a fruit.
- A surgeon is a doctor.
- A dog is an animal.

Inheritance enables reusability and helps in enhancing the functionality of any class. Inheritance is one of the major pillars of Object-Oriented programming and aids in code maintainability and avoids redundancy.

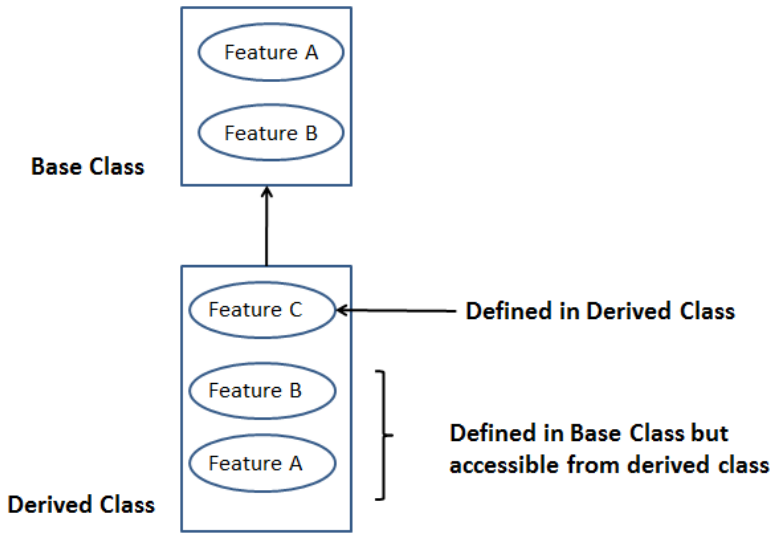

Figure 1: Derived & Base class relationship

Inheritance

Inheritance is a form of software reuse in which you create a class that absorbs an existing class's data and behaviour and enhances them with new capabilities. In inheritance, a class derives the behaviour and structure of another (existing) class.

- Advantages
    - Saves time
    - Reuse of proven, debugged, high quality software

Data members in the base class are part of the derived class. Behaviours defined in the base class are part of the derived class.

Example

```cpp
#include <iostream>
using namespace std;

class Animal {   // base class

   public:
    void eat() {
        cout << "I can eat!" << endl;
    }

    void sleep() {
        cout << "I can sleep!" << endl;
    }
};

class Dog : public Animal {   // derived class

   public:
    void bark() {
        cout << "I can bark! Woof woof!!" << endl;
    }
};

int main() {
    Dog dog1;      // Create object of the Dog class
    dog1.eat();    // Calling members of the base class
    dog1.sleep();
    dog1.bark();   // Calling member of the derived class
    return 0;
}
```

Output

```
I can eat!
I can sleep!
I can bark! Woof woof!!
```

Note that private aspects of the base class are part of the child, but are not (directly) accessible within the derived class.

class DerivedClass : **kind** BaseClass

Where **kind** is one of public, private or protected.

Base class member access specifier	Kind of Inheritance		
	public inheritance	**protected inheritance**	**private inheritance**
public	public in derived class Can be accessed directly by member functions and non-member functions	protected in derived class Can be accessed directly by member functions	private in derived class Can be accessed directly by member functions
protected	protected in derived class Can be accessed directly by member functions	protected in derived class Can be accessed directly by member functions	private in derived class Can be accessed directly by member functions
private	Hidden in derived class Can be accessed by member functions through public or protected member functions of the base class	Hidden in derived class Can be accessed by member functions through public or protected member functions of the base class	Hidden in derived class Can be accessed by member functions through public or protected member functions of the base class

Example

```
// C++ Implementation to show that a derived class
// doesn't inherit access to private data members.
// However, it does inherit a full parent object
class A{
public:
    int x;
protected:
    int y;
private:
    int z;
};

class B : public A{
    // x is public
    // y is protected
    // z is not accessible from B
};

class C : protected A{
    // x is protected
    // y is protected
    // z is not accessible from C
```

```
};

class D : private A{  // 'private' is default for classes
    // x is private
    // y is private
    // z is not accessible from D
};
```

Order of Constructors & Destructors

When a program creates a derived-class object, the derived-class constructor immediately calls the base-class constructor; the base-class constructor's body executes, then the derived-class's member initializers execute and finally the derived-class constructor's body executes. This process cascades up the hierarchy if it contains more than two levels.

When a derived-class object is destroyed, the program calls that object's destructor. This begins a chain (cascade) of destructor calls in which the derived-class destructor and the base destructors execute in reverse of the order in which the constructors executed.

```
#include <iostream>
using namespace std;

class Parent         // base class
{
    public:
    Parent()         // base class constructor
    {
        cout << "Inside base class" << endl;
    }
};

class Child : public Parent   // sub class
{
    public:
    Child()     //sub class constructor
    {
        cout << "Inside sub class" << endl;
    }
};

int main() {
    Child obj;     // creating object of sub class
    return 0;
}
```

Output

```
Inside base class
Inside sub class
```

LAB TASK

i. Write a class **person** which contains the following data members.
   - Id
   - Name
   - address

The class contains the following members functions.

- A default constructor to initialize the value data members.
- An input function to input the value of data members.
- A display function to show the value of data members.

Write a child class **student** which inherits from the **person class.** Define a relationship between two classes using **public** inheritance. The child class contains two additional data members.

- Roll_no
- Marks

The child class **student** contains the following members functions.

- A default constructor to initialize the value data members.
- An input function to input the value of data members.
- A display function to show the value of data member.

```
#include<iostream>
using namespace std;

class person{
        protected:
                string name, address;
        public:
                person(){
                }
                void input(){
                }
                void display(){
                }
};

class student : public person{
        protected:
                int roll_no, marks;
        public:
                student(){
                        }
                void in(){
                }
                void dis(){
                }
```

```
};
int main(){
	student s;
	cout << "There is no record of any student yet ";
	s.display();
	s.dis();
	cout << endl << "Enter record of a student " << endl;
	s.input();
	s.in();
	cout << endl << "Display record of a student " << endl;
	s.display();
	s.dis();
	return 0;
}
```

ii. Write a class **localcontacts** that stores a local contact number. This class contains one data member.

- Number // 4475089

The class contains the following members functions.

- A function to take value of data member from the user.
- A display function to show the value of data member.

Write a child class **nationalcontacts** to store national contact numbers that inherits from the **localcontact** class. Define a relationship between two classes using **public** inheritance. The child class contains an additional data member.

- City code // 051

The child class **nationalcontacts** contains the following members functions.

- A function to take value of data member from the user.
- A display function to show the value of data member.

```
#include<iostream>
using namespace std;

class localcontacts{
	protected:
		long number;
	public:
		void input(){
		}
		void display(){
		}
```

```
};

class nationalcontacts:public localcontacts{
        protected:
                string citycode;
        public:
                void in(){
                }
                void dis(){
                }

};

int main(){
        nationalcontacts n;
        cout << endl << "Enter phone number " << endl;
        n.in();
        n.input();
        cout << endl << "Display phone number " << endl;
        n.dis();
        n.display();
        return 0;
}
```

iii. Write a class **worker** that contains the following data members.

- Id of worker
- Pay of worker

The class **worker** contains the following member functions.

- A default constructor to initialize the data members.
- A function to take values of data members from the user.
- A display function to show the values of data members.

Write a child class **supervisor** that inherits from the worker. Define a relationship between two classes using **protected** inheritance. The child class contains the following additional data members.

- Name of supervisor
- Department of supervisor

The child class **supervisor** contains the following member functions.

- A default constructor to initialize the data members.
- A function to take values of data members from the user.
- A display function to show the values of data members.

```
#include<iostream>
using namespace std;
```

```
class worker{
	protected:
		int id, pay;
	public:
		worker(){
		}
		void input(){
		}
		void display(){
		}

};

class supervisor : protected worker{
	string name, dept;
	public:
		supervisor(){
		}
		void in(){
			worker :: input();    (Hint) or you can use setter and getter functions
		}

		void dis(){
			worker :: display();
		}
};

int main(){
	supervisor s;
	cout << "There is no record of any worker yet ";
	s.dis();
	cout << endl << "Enter record of a worker " << endl;
	s.in();
	cout << endl << "Display record of a worker " << endl;
	s.dis();
	return 0;
}
```

iv. Write a class **computer** that contains the following data members.

- RAM
- Hard_drive
- Core

The class **computer** contains the following member functions.

- A default constructor to initialize the data members.
- A function to take values of data members from the user.
- A display function to show the values of data members.

Write a child class **laptop** that inherits from the **computer class**. Define a relationship between two classes using **protected** inheritance. The child class contains the following additional data members.

- Length of laptop
- height of laptop
- width of laptop
- weight of laptop

The child class **laptop** contains the following member functions.

- A default constructor to initialize the data members.
- A function to take values of data members from the user.
- A display function to show the values of data members.