St. Francis Institute of Technology
Borivli (West), Mumbai-400103
Department of Information Technology

Experiment – 2

1. Aim: Implementation of Object Oriented Concept of inheritance using C++

Program Code
Write and execute the following program codes in C++ to achieve the given aim and attach it with your own comments and with neat indentation.

1. WAP to implement inheritance using public visibility access.

```
/*

When a base class is publicly inherited by the derived class public
members of the base class become public members of the derived class
hence they are accessible by objects of derived class

By default visibility mode is private

*/



//Public inheritance

#include<iostream>

using namespace std;



class Base

{

    int a; //by default private, hence not inheritable

    public:

    int b;

    void getAB(){

        a=24;

        b=36;
```

```
    }

    int getA(){

        return a;//This will allow us to access private members

    }

    void showA(){

        cout<<"A= "<<a<<endl;

    }

};


class Derived : public Base //Public derivation

{

    int c; //private declaration

    public:

    void add(){

        //We have to use public member functions of base class to
access the private data members of the base class

        c = b+getA();

    }

//In public derivation public members of base class become public
members of derived class hence they can be accessed directly by the
derived class


    void display(){

        cout<<"A= "<<getA()<<endl;

        cout<<"B= "<<b<<endl;

        cout<<"c= "<<c<<endl;
```

```
    }
};


int main(){
    Derived d;
//In public derivation public members of base class become public
members of derived class hence they can be accessed by the object of
the derived class
    d.getAB();
    d.add();
    d.showA();
    d.display();
    d.b = 20;
    d.add();
    d.display();
    return 0;
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
D:\College\PCPF\CPP\Exp2>Exp2-1
A= 24
A= 24
B= 36
c= 60
A= 24
B= 20
c= 44

D:\College\PCPF\CPP\Exp2>
```

Analysis:
If we inherit the base class publicly, then the public member of the base class will become public in the derived class and protected members of the base class will become protected in the derived class.

2. WAP to implement inheritance using private visibility access.

```
/*

When a base class is privately inherited by the derived class public
members of the base class become private members of the derived class
public members of base class can only be accessed by member functions of
derived class

By default visibility mode is private

*/



//Private inheritance

#include<iostream>

using namespace std;



class Base

{

    int a; //by default private, hence not inheritable

    public:

    int b;

    void getAB(){

        a=24;

        b=36;

}

    int getA(){

        return a;//This will allow us to access private members
```

```
    }

    void showA(){

        cout<<"A= "<<a<<endl;

    }

};

class Derived : private Base //Private derivation

{

    int c; //private declaration

    public:

    void add(){

        getAB();  //to access the public functions of base class call
the functions by the member functions of derived class

        c = b+getA();

    }

    void display(){

        showA();

        cout<<"B= "<<b<<endl;

        cout<<"c= "<<c<<endl;

    }

};

int main(){

    Derived d;

//In private derivation public members of base class become private
members of derived class hence they cannot be accessed by the object of
the derived class

    d.add();
```

```
    d.display();

    return 0;

}
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

D:\College\PCPF\CPP\Exp2>Exp2-2
A= 24
B= 36
c= 60

D:\College\PCPF\CPP\Exp2>
```

Analysis:
If we inherit the base class privately, then both public member and protected members of the base class will become Private in derived class.

3. WAP to implement multilevel inheritance

```
//Multi Level inheritance

#include<iostream>

using namespace std;



//Declaring the base class, members and member functions

class Mammal{

    protected:

    string type;

    public:

    string getType(){

        return type;

    }

    void setType(string type){
```

```
        this -> type = type;
    }
};

//Declaration of next class
//Define class Dog from Mammal
class Dog: public Mammal{
    protected:
    string size;
    public:
    void getSize(){
        cout<<"The size of the dog is "<<size<<endl;
    }
    void setSize(string size){
        this -> size = size;
    }
};

//Declaration of the derived class
//Define class Husky from Dog

class Husky: public Dog {
    string name;
    public:
```

```
    void display( string name){
        cout<<"The type of mammal is "<<getType()<<endl;
        getSize();
        cout<<"The name of husky is "<< name<<endl;
    }
};


int main(){
    Husky max;
    max.setType("carnivore");
    max.setSize("medium");
    max.display("Max");
    return 0;
}
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

D:\College\PCPF\CPP\Exp2>Exp2-3
The type of mammal is carnivore
The size of the dog is medium
The name of husky is Max

D:\College\PCPF\CPP\Exp2>
```

Analysis:

Multilevel Inheritance is a feature of C++ where a class is derived from another derived class.

4. WAP to implement multiple inheritances

```
//Multiple inheritance

#include<iostream>

using namespace std;



//Declaring the base class, members and member functions



class Mammal{

    protected:

    string type;

    public:

    string getType(){

        return type;

    }

    void setType(string type){

        this -> type = type;

    }

};



//Declaration of next class

class Dog{

    protected:

    string size;

    public:
```

```
    void getSize(){
        cout<<"The size of the dog is "<<size<<endl;
    }
    void setSize(string size){
        this -> size = size;
    }
};
//Declaration of the derived class
//Define class Result from Mammal and Dog


class Husky: public Dog, public Mammal {
    //the difference in Multilevel and Multiple is that 2 or more
classes are base classes
    string name;
    public:
    void display( string name){
        cout<<"The type of mammal is "<<getType()<<endl;
        getSize();
        cout<<"The name of husky is "<< name<<endl;
    }
};
int main(){
   Husky max;
    max.setType("carnivore");
    max.setSize("medium");
```

```
    max.display("Max");

    return 0;

}
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

D:\College\PCPF\CPP\Exp2>Exp2-4
The type of mammal is carnivore
The size of the dog is medium
The name of husky is Max

D:\College\PCPF\CPP\Exp2>
```

Analysis:
Multiple Inheritance is a feature of C++ where a class can inherit from more than one class. i.e one sub class is inherited from more than one base classes.

8. Post Experimental Exercise

A. Questions:

1. What is inheritance mean in C++? What are the different forms of inheritance?
The capability of a class to derive properties and characteristics from another class is called Inheritance. Inheritance is one of the most important features of Object Oriented Programming. The different types of inheritance are single, multiple, multilevel and hybrid.

2. When do we use the protected visibility specifier to a class member?
Protected variables allow access to the variables only from subclasses and classes within the same package. Protected variables can be useful if you want your data to be read-only, or when you want to abstract your data. Protected access modifier is similar to that of private access modifiers, the difference is that the class member declared as Protected are inaccessible outside the class but they can be accessed by any subclass(derived class) of that class.

3. What is an abstract class?
An *abstract class* is a class that is declared abstract—it may or may not include abstract methods. Abstract classes cannot be instantiated, but they can be subclassed.Abstract classes act as expressions of general concepts from which more specific classes can be derived. A class that contains at least one pure virtual function is considered an abstract class. Classes derived from the abstract class must implement the pure virtual function or they, too, are abstract classes.

4. An educational institute wishes to maintain a database of its employees. The database is divided into a number of classes whose hierarchical relationships are shown in figure. Specify all the classes and define functions to create the database and retrieve individual information as and when required

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

```
// Base Class Staff
class staff
{
private:
    string code;
    string name;
public:
    string getCode(){ //To get employee code
        return code;
    }
    void setCode(){ //To save emp;oyee code
        string code;
        cout<<"Enter the employee code:";
        getline(cin,code);
        cout<<endl;
        this->code=code;
    }
    string getName(){ //To get employee name
        return name;
    }
    void setName(){ //To set Employee name
        string name;
        cout<<"Enter the employee Name:";
        getline(cin,name);
        cout<<endl;
        this->name=name;
    }
};

//Derived class teacher inherits base class staff publically
class Teacher: public staff{

private:
    string subject;
    string publication;

public:
    string getSubject(){//To get subject
        return subject;
    }
    void setSubject(){//To set subject
        string subject;
        cout<<"Enter the employee subject:";
```

```
        getline(cin,subject);
        cout<<endl;
        this->subject=subject;
    }
    string getPublication(){//To get publication
        return publication;
    }
    void setPublication(){//To set publication
        string publication;
        cout<<"Enter the employee publication:";
        getline(cin,publication);
        cout<<endl;
        this->publication=publication;
    }
};

//Derived class typist inherits base class staff publically
class typist: public staff{

private:
    int speed;
public:
    int getSpeed(){ //To get speed
        return speed;
    }
    void setSpeed(){//To set speed
        int speed;
        cout<<"Enter the typist speed:";
        cin>>speed;
        cout<<endl;
        this->speed=speed;
    }
};

//Derived class regular inherits base class staff publically
class regular:public typist{

public:
    regular(){}
};

//Derived class casual inherits base class typist publically
class casual:public typist{
private:
```

```
    float wages;
public:
    float getWages(){
        return wages;//To get wages
    }
    void setWages(){ //To set wages
        float wages;
        cout<<"Enter daily wages:";
        cin>>wages;
        cout<<endl;
        this->wages=wages;
    }
};

//Derived class officer inherits base class staff publically
class officer:public staff{

private:
    int grade;
public:
    int getGrade(){ //To get grade
        return grade;
    }
    void setGrade(){ //To set grade
        int grade;
        cout<<"Enter officer garde:";
        cin>>grade;
        cout<<endl;
        this->grade=grade;
    }
};

int main(){

    Teacher teacher1;
    cout<<"Enter Teacher details"<<endl;
    teacher1.setName();
    teacher1.setCode();
    teacher1.setSubject();
    teacher1.setPublication();
    cout<<"Displaying teacher details:"<<endl;
    cout<<"Teacher name is:"<<teacher1.getName()<<endl;
    cout<<"Teacher code is:"<<teacher1.getCode()<<endl;
    cout<<"Teacher subject is:"<<teacher1.getSubject()<<endl;
```

```
    cout<<"Teacher publication is:"<<teacher1.getPublication()<<endl;

    regular rtypist;
    cout<<"\nEnter regular typist details"<<endl;
    rtypist.setName();
    rtypist.setCode();
    rtypist.setSpeed();
    cout<<"Displaying regular typist details"<<endl;
    cout<<"Typist name is:"<<rtypist.getName()<<endl;
    cout<<"Typist code is:"<<rtypist.getCode()<<endl;
    cout<<"Typist speed is:"<<rtypist.getSpeed()<<endl;

    casual ctypist;
    cout<<"\nEnter casual typist details"<<endl;
    cin.ignore();
    ctypist.setName();
    ctypist.setCode();
    ctypist.setSpeed();
    ctypist.setWages();
    cout<<"Displaying casual typist details"<<endl;
    cout<<"Typist name is:"<<ctypist.getName()<<endl;
    cout<<"Typist code is:"<<ctypist.getCode()<<endl;;
    cout<<"Typist speed is:"<<ctypist.getSpeed()<<endl;
    cout<<"Typist wage is:"<<ctypist.getWages()<<endl;

    officer officer;
    cout<<"Enter Officer Details:"<<endl;
    cin.ignore();
    officer.setName();
    officer.setCode();
    officer.setGrade();
    cout<<"Displaying Officer details:"<<endl;
    cout<<"Officer name is:"<<officer.getName()<<endl;
    cout<<"Officer code is:"<<officer.getCode()<<endl;;
    cout<<"Officer grade is:"<<officer.getGrade()<<endl;

    return 0;
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
D:\College\PCPF\CPP\Exp2>Q4
Enter Teacher details
Enter the employee Name:Jane Doe

Enter the employee code:A191060

Enter the employee subject:Pcpf

Enter the employee publication:Oxford

Displaying teacher details:
Teacher name is:Jane Doe
Teacher code is:A191060
Teacher subject is:Pcpf
Teacher publication is:Oxford

Enter regular typist details
Enter the employee Name:John Doe

Enter the employee code:A191061

Enter the typist speed:80

Displaying regular typist details
Typist name is:John Doe
Typist code is:A191061
Typist speed is:80

Enter casual typist details
Enter the employee Name:Jannet Doe

Enter the employee code:A191062

Enter the typist speed:85

Enter daily wages:1000
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

Displaying casual typist details
Typist name is:Jannet Doe
Typist code is:A191062
Typist speed is:85
Typist wage is:1000
Enter Officer Details:
Enter the employee Name:Josh Doe

Enter the employee code:A191063

Enter officer garde:1

Displaying Officer details:
Officer name is:Josh Doe
Officer code is:A191063
Officer grade is:1

D:\College\PCPF\CPP\Exp2>
```

This program helps us to understand how inheritance works. Inheritance helps us to reuse the same piece of code without having to repeat it over and over. This makes the code more maintainable.

Conclusion:

Inheritance is the mechanism of basing an object or class upon another object or class, retaining similar implementation. In this experiment we have written programs in C++ to implement concepts of single, multiple, multilevel, hybrid inheritance and how to make private members inheritable.

To perform this experiment Visual Studio Code was used as a code editing software and MinGW GCC compiler was used to compile the codes

From this experiment we can infer that an advantage of inheritance is that it helps to minimize the amount of duplicate code in an application by sharing common code amongst several subclasses. This also tends to result in a better organization of code and smaller, simpler compilation units.

D. References:

[1] E Balaguruswamy, “Object Oriented Programming with C++”, second edition Tata McGraw Hill (Chapter 8)