**LAB # 9**

**OBJECT:**

To learn polymorphism and its implementation using virtual functions.

**SOURCE CODE:**

#include <iostream>

using namespace std;

class Base {

public: virtual void show()

{ cout << “Base\n”; }};

class Derv1 : public Base{

public: void show()

{ cout << “Derv1\n”; }

};

class Derv2 : public Base

{

public:

void show()

{ cout << “Derv2\n”; }

};

int main()

{

Derv1 dv1; //object of derived class 1

Derv2 dv2; //object of derived class 2

Base\* ptr; //pointer to base class

ptr = &dv1; //put address of dv1 in pointer

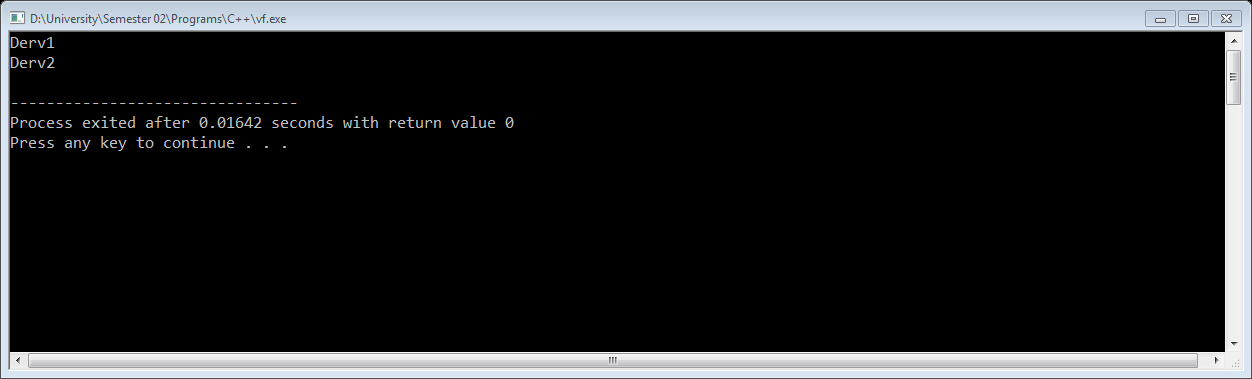
ptr->show(); //execute show()

ptr = &dv2; //put address of dv2 in pointer

ptr->show(); //execute show()

return 0;

}



**EXERCISE:**  Consider the following class definition

Class father

{

protected:

int age;

public:

father (iny x)

{ age = x;}

virtual void iam()

{ cout << “I AM FATHER, my age is …..” << age<<endl;}

};

Derive two classes son and daughter from the above and for each define iam() to write

a similar but appropriate message. Write a main() that creates objects of the three

classes and then calls iam() for them. Declare pointer to father. Successively, assign

addresses of objects of the two derived classes to this pointer and in each case, call

iam() through pointer to demonstrate polymorphism.

**SOURCE CODE:**

#include<iostream>

#include<conio.h>

using namespace std;

class Father{

protected:

int age;

public:

Father(int x): age(x) { }

virtual void iam(){

cout<<"I AM FATHER,my age is "<<age<<endl;

}

};

class Son:public Father{

public:

Son(int x,int y): Father(x){

age=y;

}

virtual void iam(){

cout<<"I AM SON,my age is "<<age<<endl;

}

};

class Daughter:public Father{

public:

Daughter(int x,int y): Father(x){

age=y;

}

virtual void iam(){

cout<<"I AM DAUGHTER,my age is "<<age<<endl;

}

};

int main(){

Father f(50),\*f1;

Son s(50,24);

Daughter d(50,22);

f.iam();

s.iam();

d.iam();

f1=&f;

f1->iam();

f1=&s;

f1->iam();

f1=&d;

f1->iam();

return 0;

}

