|  |
| --- |
|  |
| Inheritance |
|  |

Lab Sections:

1. Objectives
2. Introduction
3. Definition
4. Declaration Syntax
5. Experiment

|  |
| --- |
|  |

**Inheritance**

1. **Objectives**

**After this experiment you will be able to implement a class that inherits from another class.**

1. **Introduction**

A class is composed of its state (private data) and behavior (public member functions). The public member functions operate on the private data. Symbolically, a class represents a physical object, like a person; the data of a class represents a description (characteristics) of that person; and the functions represent what the person will be able to do (its behavior). Inheritance implies there exists an “**is a**” relationship between two objects. For example, an employee “is a” person. Therefore, all the characteristics that apply to a person also apply to an employee, including any additional characteristics that are unique to an employee.

1. **Definitions/Important Terms**

We will define several terms you need to know to understand classes. They are as follows:

1. A **base class or parent class** is the class that is being inherited from.
2. The **derived class or child class** is the class that inherits from the base class. All the data and functions defined in the base class are accessible to the derived class.
3. **Protected** class members act as private members for both the base and inherited classes.
4. A **virtual** functionis a function in the base class that may be replaced in the derived class. If replaced, the virtual function will provide its own implementation (behavior) for that function.
5. **Virtual functions allows 3 things**: a function in the base class can be replaced in the derived class with the exact same name and signature, it allows the new function in the derived class to have precedence when called, and it allows you to change the new function to suit your needs.

**Syntax for a Class Inheritance**

class Derived\_Class : Base\_Class

{

Derived\_Class members

};

1. **Experiment**

**Step 1: In this experiment you will investigate the implementation of inheritance. Enter, save, compile and execute the following program in MSVS. Call the new directory “InheritanceExp1” and the program “Inheritance1.cpp”. Answer the questions below:**

#include <string>

#include <iostream>

using namespace std;

class Person

{

protected:

string firstName;

string lastName;

public:

Person(void)

{

cout<<"In Person's constructor -- A Derived Class"<<endl;

cout<<"Enter firstname: ";

cin>>firstName;

cout<<"Enter lastname: ";

cin>>lastName;

}

string getFirstName(void)

{

return firstName;

}

string getLastName(void)

{

return lastName;

}

};

class Employee : public Person

{

protected:

float salary;

public:

Employee()

{

cout<<"In Employee's constructor -- A Derived Class"<<endl;

cout<<"Enter Salary: ";

cin>>salary;

}

float getSalary(void)

{

return salary;

}

};

int main (void)

{

Employee Number\_one; //calls Constructor of Person, then Constructor of employee

Employee Number\_two; //calls Constructor of Person, then Constructor of employee

cout<<endl<<endl<<endl;

cout<<"Retrieving Number\_one's first name and last name from class Person\n";

cout<<" "<<Number\_one.getFirstName()<<" "<< Number\_one.getLastName()<<endl;

cout<<"Retrieving Number\_one's salary from class Employee\n";

cout<<" "<<Number\_one.getSalary()<<endl;

cout<<endl<<endl<<endl;

cout<<"Retrieving Number\_two's first name and last name from class Person\n";

cout<<" "<<Number\_two.getFirstName()<<" "<< Number\_two.getLastName()<<endl;

cout<<"Retrieving Number\_two's salary from class Employee\n";

cout<<" "<<Number\_two.getSalary()<<endl;

system("PAUSE");

return 0;

}

**Question 1:** What are the names of the base and derived classes?

**Question 2:** Describe how the derived class accesses the properties (data and functions) of the base class.

**Question 3:** Based on the output of the code, which constructor is called first: the constructor in the derived class or the constructor in the base class?

**Question 4:** What is the purpose of making members of a class protected, as opposed to public or private? (This is a possible job interview question! Get it right and memorize it for life!)

**Step 2: In this experiment you will investigate the implementation of inheritance. Enter, save, compile and execute the following program in MSVS. Call the new directory “InheritanceExp2” and the program “Inheritance2.cpp”. Answer the questions below:**

#include <string>

#include <iostream>

using namespace std;

class Person

{

protected:

string firstName;

string lastName;

public:

Person(void)

{

cout<<"In Person's constructor -- A Derived Class"<<endl;

cout<<"Enter firstname: ";

cin>>firstName;

cout<<"Enter lastname: ";

cin>>lastName;

}

string getFirstName(void)

{

return firstName;

}

string getLastName(void)

{

return lastName;

}

virtual void CalculateAndPrintPayrollInformation(void)

{

cout<<"This is Payroll Information for a Base Object"<<endl;

cout<<"Gross Pay is 0"<<endl;

cout<<"Income Tax is 0"<<endl;

cout<<"Net Pay is 0"<<endl<<endl;

}

};

class Employee : public Person

{

protected:

float annual\_salary;

double gross\_pay, hours\_worked, hourly\_rate, income\_tax, net\_pay;

public:

Employee()

{

cout<<"In Employee's constructor -- A Derived Class"<<endl;

cout<<"Enter Annual Salary: ";

cin>>annual\_salary;

}

float getSalary(void)

{

return annual\_salary;

}

void CalculateAndPrintPayrollInformation(void)

{

cout<<"In Taxpayer's constructor -- A derived class of employe"<<endl;

cout<<"Enter hours worked: ";

cin>>hours\_worked;

cout<<"Enter hourly rate: ";

cin>>hourly\_rate;

gross\_pay = hours\_worked \* hourly\_rate;

income\_tax = gross\_pay \* 0.25;

net\_pay = gross\_pay - income\_tax;

cout<<"Gross Pay = "<<gross\_pay<<endl;

cout<<"Income Tax = "<<income\_tax<<endl;

cout<<"Net Pay = "<<net\_pay<<endl;

}

};

int main (void)

{

Person John;

Employee Mary;

cout<<endl<<endl<<endl;

cout<<"Enter John's information\n";

cout<<endl<<John.getFirstName()<<" "<<John.getLastName()<<endl;

John.CalculateAndPrintPayrollInformation();

cout<<endl<<endl<<endl;

cout<<"Enter Mary's information\n";

cout<<endl<<Mary.getFirstName()<<" "<<Mary.getLastName()<<" salary is "

<< Mary.getSalary()<<endl;

Mary.CalculateAndPrintPayrollInformation();

system("PAUSE");

return 0;

}

**Question 5:** What are the code differences between the programs presented in Step 1 and Step 2? (Hint: it’s a new virtual function)

**Question 6:** Referring to the output of the program in Step 2, how is the virtual function used in the Person and Employee classes?

**Question 7:** Change the type of John from Person to Employee in the main function of the program in Step 2. Execute this modified program and compare the output with the output of the original

program in Step 2. Based on the 2 different outputs, do you think the original **base** class implementation of the virtual function is executed in this modified program? Why or why not?

**Question 8:** What is the purpose of the “virtual” keyword?