|  |
| --- |
| **Computer Engineering Department - ITU** |
| **CE101L: Object Oriented Programming Lab** |

|  |  |
| --- | --- |
| **Course Instructor: Usama Bin Shakeel** | **Dated: 13/04/2022** |
| **Teaching Assistant: Aqsa Khalid** | **Semester: Spring 2022** |
| **Lab Engineer: Nadir Abbas** | **Batch: BSCE2021** |

# **Lab 6A. Use of Operator Overloading & Inheritance in Classes and Objects**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Roll number** | **Report**  **(out of 100)** | **Scaled to 10** | **Total**  **(out of 10)** |
| NIMRA MAQBOOL | BSCE21012 |  |  |  |

Checked on: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## **Objective**

The objective of this lab is to observe the basic knowledge of programming classes in C++.

## **Equipment and Component**

|  |  |  |
| --- | --- | --- |
| **Component Description** | **Value** | **Quantity** |
| Computer | Available in lab | 1 |

## **Conduct of Lab**

1. Students are required to perform this experiment individually.
2. In case the lab experiment is not understood, the students are advised to seek help from the course instructor, lab engineers, assigned teaching assistants (TA) and lab attendants.

## **Theory and Background**

In C++, we can change the way operators work for user-defined types like objects and structures. This is known as **operator overloading**. For example,

Suppose we have created three objects c1, c2 and result from a class named Complex that represents complex numbers. Since operator overloading allows us to change how operators work, we can redefine how the + operator works and use it to add the complex numbers of c1 and c2 by writing the following code:

**result = c1 + c2;**

instead of something like

result = c1.addNumbers(c2);

This makes our code intuitive and easy to understand.

**Inheritance** is one of the key features of Object-oriented programming in C++. It allows us to create a new class (derived class) from an existing class (base class). The derived class inherits the features from the base class and can have additional features of its own.

**Lab Task**

**Task A: Operator Overloading in Unary Operators (++, --) [Marks: 10]**

In this task, you are required to create a class **NumberIncrement** with the following data members and member functions,

***Private Data Members such as:***

num(int)

***Public Member Functions such as:***

**DefaultConstructor()** – It will initialize num to 10.

**void operator unary symbol()** – It will increment num two times (++num).

**e.g** void operator ++() {…}

**void display()** – It will print num.

1. Create **UML diagram** for Box class with data member and member functions

2. Create object obj of class in main function and call the "**void operator ++ ()**" function e.g ++obj,

3. Call the display function to print incremented num.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | numberIncrement   |  | | --- | | +num:int | | -numberIncrement(int num)  -operator ++()  -print() | |  |  |  |   **Function.h:**  class numberIncrement { private:  int num; //declaring public:  numberIncrement(int number); //defining functions  void operator ++();  void print();  };  **function.cpp:**  using namespace std;  numberIncrement::numberIncrement(int number) {  number = 10; //setting it to 10   num = number; //copying that number in private variable  cout << "the number is set to = " << num << endl; //displaying num }  void numberIncrement::operator++() {  num += 2; //incrementing by 2 } void numberIncrement::print(){  cout<<"the number after incrementing = "<<num<<endl; //displaying the incremented value }  **main.cpp:**  int opt; //initializing do{  cout<<"WHICH TASK DO YOU WANT TO PERFORM?"<<endl;  cout<<"1.TASK 1 "<<endl;  cout<<"2.TASK 2 "<<endl;  cout<<"3.TASK 3 "<<endl; //displaying options  cout<<"4.EXIT"<<endl;  cin>>opt; //taking the option   if(opt==1){  int number; //calling functions  numberIncrement inc(number);  ++inc;  inc.print();  }  **output:**  **Text  Description automatically generated** |

**Task B: Operator Overloading in Binary Operators (+, - etc ) [Marks: 15]**

In this task, you are required to create a class **Addition** with the following data members and member functions,

***Private Data Members such as:***

number1, number2(int)

***Public Member Functions such as:***

**ParametrizedConstructor(int x=0, int y=0)** – It will initialize number1 and number2.

**Addition operator + (Addition &obj) –** It will do the following

Addition operator + (Addition &obj) {

Addition temp;

temp. number1= number1+ obj. number1;

temp. number2= number2+ obj. number2;

return temp;

}

**void print()** – It will print the sum of number1 and number2.

1. Create **UML diagram** for Addition class with data member and member functions

2. Create two objects obj1 and obj2 of class with parameters to call parameterized constructor in main function **e.g** Addition obj1(10, 4), obj2(4, 8)

3. Create another object of class obj3 and add other two objects (obj1, obj2).

**e.g** Addition obj3=obj1+obj2

4. Call print function from obj3 and write your observation about output.

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | Addition   |  | | --- | | -num1:int  -num2:int | | +addition(int x,int y)  +addition()  +operator+(addition &obj)  +print() | |   **addition.h:**  class addition { private:  int num1;  int num2; public:  addition(int x, int y);   addition( ) {  int x=0;  int y=0; // initializing it to zero  num1=x; //copying the value of x and y in num1 and num2  num2=x;  }   addition operator+(addition &obj) {  addition temp; //declaring temp  temp.num1 = num1 + obj.num1;  temp.num2 = num2 + obj.num2; //storing the sum  return temp; //returning temp  }  void print(); //defining  };  **addition.cpp:**  using namespace std; addition::addition(int x, int y) {  cout<<"enter x ";  cin>>x; //taking x from user  cout<<"enter y ";  cin>>y; //taking x from user  num1=x; //copying x and y in num1 and num2  num2=y; }  void addition::print() {  int sum; // declaring sum  sum = num1 + num2; //storing the sum   cout << "SUM = " << sum << endl; //displaying }  **main.cpp:**  if(opt==2){   int x; //initializing  int y;  addition( ); //calling functions  addition obj(x ,y),obj1(x,y);  addition obj3 = obj +obj1;  obj3.print(); }  **output:**  **Text  Description automatically generated** |

**Task C: Inheritance [Marks: 15]**

In this task, you are required to create a class **Human** and class **Gender** with the following data members and member functions,

***Protected Data Members of class Human such as:***

type(string)

***Public Member Functions of class Human such as:***

**void type\_public() –** It will print “Human as Public member in parent class”.

***Public Member Functions of class Gender such as:***

**void setType(string) –** It will access and initialize protected data member of Human class.

**void type\_child()** – It will print “Human as public member in child class”.

**void show() –** It will print “I am a ” with concatenate of protected data member of parent class **e.g** cout << "I am a " << type << endl;

1. Create a **UML diagram**.

2. Create object of child class **Gender** in main function and call member functions such as **void setType(string tp), void show(), type\_child(), type\_public().**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | Human   |  | | --- | | -type:string | | +type\_public() | |  |  |  | | --- | --- | | gender   |  | | --- | | +setString(string tp)  +type\_child()  +show() | |   **humans.h:**  class humans { protected:  string type; //declaring string public:  void type\_public(); //declaring function  }; class gender:public humans{ //child class of gender public:  void setString(string tp); //declaring  void type\_child();  void show(); };  **humans.cpp:**  using namespace std;  void humans::type\_public() {  cout << "HUMANS AS PUBLIC MEMBER IN PARENT CLASS" << endl; //Displaying  }  void gender::setString(string tp) {  cout << "enter a string " << endl; //taking a string   cout<<"without any spaces"<<endl; //string must not contain spaces otherwise only the first word will be printed  cin>>tp; //taking string from user  type=tp; //copying the string }  void gender::type\_child() {  cout << "HUMAN AS PUBLIC MEMBER OF CHILD CLASS" << endl; //displaying }  void gender::show() {  cout << "I AM A " << type << endl; //displaying the type string }  **main.cpp:**  int main() {  int opt; //initializing  do{  cout<<"WHICH TASK DO YOU WANT TO PERFORM?"<<endl;  cout<<"1.TASK 1 "<<endl;  cout<<"2.TASK 2 "<<endl;  cout<<"3.TASK 3 "<<endl; //displaying options  cout<<"4.EXIT"<<endl;  cin>>opt; //taking the option  if(opt==1){  int number; //calling functions  numberIncrement inc(number);  ++inc;  inc.print();  }  if(opt==2){   int x; //initializing  int y;  addition( ); //calling functions  addition obj(x ,y),obj1(x,y);  addition obj3 = obj +obj1;  obj3.print();  }  if(opt==3){   gender g; //creating an obj  g.type\_public(); //calling  string tp;  g.setString("i am studying from ubs");  g.type\_child();  g.show();  }  if(opt==4){  cout<<"YOU CHOOSE TO EXIT.."<<endl;   exit(6); //condition to exit  }  }  while(opt>=1 && opt<=4);  return 0; }  **outPut:**  **Text  Description automatically generated** |

#### **Assessment Rubric for Lab**

**Method for assessment:**

Lab reports and instructor observation during lab sessions. Outcome assessed:

a. Ability to conduct experiments, as well as to analyze and interpret data (P) b. Ability to function on multi-disciplinary teams (A)

c. Ability to use the techniques, skills, and modern engineering tools necessary for engineering practice (P)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Performance metric** | **Task** | **CLO** | **Description** | **Max marks** | **Exceeds expectation** | **Meets expectation** | **Does not meet expectation** | **Obtained marks** |
| 1. Realization of experiment (a) | 1 | 1 | Functionality | 40 | Executes without errors excellent user prompts, good use of symbols, spacing in output. Through testing has been completed (35-40) | Executes without errors, user prompts are understandable, minimum use of symbols or spacing in output. Some testing has been completed (20-34) | Does not execute due to syntax errors, runtime errors, user prompts are misleading or non-existent. No testing has been completed (0-19) |  |
| 2. Teamwork (b) | 1 | 3 | Group Performance | 5 | Actively engages and cooperates with other group member(s) in effective manner (4-5) | Cooperates with other group member(s) in a reasonable manner but conduct can be improved (2-3) | Distracts or discourages other group members from conducting the experiment (0-1) |  |
| 3. Conducting experiment (a, c) | 1 | 1 | On Spot Changes | 10 | Able to make changes (8-10) | Partially able to make changes (5-7) | Unable to make changes (0-4) |  |
| 1 | 1 | Viva | 10 | Answered all questions (8-10) | Few incorrect answers (5-7) | Unable to answer all questions (0-4) |  |
| 4. Laboratory safety and disciplinary rules (a) | 1 | 3 | Code commenting | 5 | Comments are added and does help the reader to understand the code (4-5) | Comments are added and does not help the reader to understand the code (2-3) | Comments are not added (0-1) |  |
| 5. Data collection (c) | 1 | 3 | Code Structure | 5 | Excellent use of white space, creatively organized work, excellent use of variables and constants, correct identifiers for constants, No line-wrap (4-5) | Includes name, and assignment, white space makes the program fairly easy to read. Title, organized work, good use of variables (2-3) | Poor use of white space (indentation, blank lines) making code hard to read, disorganized and messy (0-1) |  |
| 6. Data analysis (a, c) | 1 | 4 | Algorithm | 20 | Solution is efficient, easy to understand, and maintain (15-20) | A logical solution that is easy to follow but it is not the most efficient (6-14) | A difficult and inefficient solution (0-5) |  |
| 7. Computer use (c) | 1 | 2 | Documentation & GitHub Submissions | 5 | Timely (4-5) | Late (2-3) | Not done (0-1) |  |
|  | Max Marks (total): | | | 100 | Obtained Marks (total): | | |  |

Lab Engineer Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_