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

|  |  |
| --- | --- |
| **Course Instructor: Usama Bin Shakeel** | **Dated: 31/03/2022** |
| **Teaching Assistant: Aqsa Khalid** | **Semester: Spring 2022** |
| **Lab Engineer: Nadir Abbas** | **Batch: BSCE2021** |

# **Lab 4B. Solving Problems by Utilization of 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**

A class in C++ is a user-defined type or data structure declared with keyword class that has data and functions as its members whose access is governed by the three access specifiers private, protected, or public. By default, access to members of a C++ class is private.

**Lab Task**

**Task A [Marks: 15]**

Write a program by creating an 'Employee' class having the following functions and print the final salary.

1 - 'SetInfo()' which takes the salary, number of hours of work per day of employee

2 - 'AddSal()' which adds $10 to the salary of the employee if it is less than $500.

3 - 'AddWork()' which adds $5 to the salary of the employee if the number of hours of work per day is more than 6 hours.

|  |
| --- |
| **FUNCTION.CPP:**  void Employee::addWork() //Adding salary if employee works more than 6 hours {  if(hours>6)  {  salary=salary+5;  } } void Employee::setValue(int duration,int cash) //Setting the values in other variables {  salary=cash;  hours=duration; } void Employee::addSalary() //Adding salary if it is less than 500 {  if(salary<500)  {  salary=salary+10;  } } int Employee::getValue() //Getting the salary {  return salary; } Employee::Employee() {}  **MAIN.CPP:**  int x; cout<<"Enter 1 for task 1 or 2 for task 2 : "; cin>>x; if(x==1) {  int out,cash,duration; //initializing other variables  Employee empl; //creating object  cout<<"For how many hours does the employee work :"; //taking input of duration of work  cin>>duration;  cout<<"How much is the salary of the employee :"; //taking input of salary  cin>>cash;  empl.setValue(duration,cash); //Calling all the class fuctions  empl.addSalary();  empl.addWork();  out=empl.getValue();  cout<<"The returned values are:"<<out; //printing calculated value }  **FUNCTION.H:**  class Employee { private:  int salary; //Initializing the variables privately  int hours; public:  int getValue(); //Creating the function prototypes in class  void addWork();  void setValue(int duration,int cash);  void addSalary();  Employee(); }; |

**Task B [Marks: 25]**

Create a class called 'Matrix'. The Matrix class has the following private data members:

1 - number of rows of matrix

2 - number of columns of matrix

3 - elements of matrix (You can use 2D array)

The Matrix class has public member functions for each of the following:

1 - constructor that initializes the number of rows and the number of columns

2 - get the number of rows

2 - get the number of columns

3 - set the elements of the matrix at a given position (i,j)

4 - adding two matrices.

5 - multiplying the two matrices

You can assume that the dimensions are correct for the multiplication and addition.

|  |
| --- |
| **FUNCTION.CPP:**  int Matrix::getRows() //Getting rows {  return rows; } int Matrix::getCols() //Getting columns {  return columns; } Matrix::Matrix(int r,int c) {  columns=c;  rows=r;  int i=0;  while(i<r)  {  matrix[i]=new int[c];  i++;  } } void Matrix::setMatrix(int i,int j,int value) //Setting values {  matrix[i][j]=value; } void Matrix::addMatrix() //Adding the matrices {  int \*\*added=new int\*[rows];  for(int i=0;i<rows;i++)  {  added[i]=new int [columns];  }  for(int i=0;i<rows;i++)  {  for(int j=0;j<columns;j++)  {  added[i][j]=matrix[i][j]+matrix[i][j];  }  }  cout<<"Resultant Matrix is: "<<endl;  for(int i=0;i<rows;i++)  {  for(int j=0;j<columns;j++)  {  cout<<" "<<added[i][j];  }  cout<<endl;  } } void Matrix::multiMatrix() //Multiplicating the matrices {  int \*\*multi=new int\*[rows];  for(int i=0;i<rows;i++)  {  multi[i]=new int [columns];  }  for(int i=0;i<rows;i++)  {  for(int j=0;j<columns;j++)  {  int elements;  for(int k=0;k<rows;k++)  {  elements=elements+(matrix[i][k]\*matrix[k][j]);  }  multi[i][j]=elements;  }  }  cout<<"Resultant matrix is :";  for(int i=0;i<rows;i++)  {  for(int j=0;j<columns;j++)  {  cout<<" "<<multi[i][j];  }  cout<<endl;  } }  **MAIN.CPP:**  {  Matrix mat; //creating object  int row,column,value; //initializing other variables  cout<<"Enter the number of rows:"; //input of rows  cin>>row;  cout<<"Enter the number of columns:"; //input of columns  cin>>column;  mat.Matrix(row,column);  for(int i=0;i<row;i++) //input of elements  {  for(int j=0;j<column;j++)  {  cout<<"Enter value of "<<i+1<<" and "<<j+1<<" of matrix:";  cin>>value;  mat.setMatrix(i,j,value);  }  }  mat.multiMatrix();  mat.addMatrix(); }  **FUNCTION.H:**  class Matrix { private: //Initializing the variables privately  int columns;  int \*\*matrix;  int rows; public:  int getRows(); //Creating the function prototypes in class  Matrix(int r,int c);  int getCols();  void setMatrix(int i,int j,int value);  void addMatrix();  void multiMatrix(); }; |

#### **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: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_