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

|  |  |
| --- | --- |
| **Course Instructor: Usama Bin Shakeel** | **Dated: 30/03/2022** |
| **Teaching Assistant: Aqsa Khalid** | **Semester: Spring 2022** |
| **Lab Engineer: Nadir Abbas** | **Batch: BSCE2021** |

# **Lab 4A. Defining Classes and Initializing Objects using Constructors**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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: 8]**

Write a C++ code to define an **Account** Class that contains two Data Members **AccountTitle** and Member Functions to Set and Get its value. You will save it as **Account.h**.

The class will have Public Member Functions with the following declarations:

void setAccount (string);

string getAccount (void);

The class will have a Private Data Member **AccountTitle** with the following declaration:

string AccountTitle;

**Task B [Marks: 8]**

For the class definition in Task A, write a main driving code **AccountMain.cpp** to create and manipulate an Account object **myAccount**. You should get the following output:

Initial account title is:

Please enter the account title: Your Name and Roll number

Title in object myAccount is: Your Name and Roll number

Use **getLine()** function to read in the string.

**Task C [Marks: 8]**

Now repeat **Task A** by using the **Account** Constructor to initialize the Name Data Member at the time each Account Object is created.

The constructor function is Account (string);

**Task D [Marks: 8]**

Now repeat **Task B** by defining **Account** Class with a Constructor that Initializes the Account Name. Create three objects as **account1, account2** and **account3.**

**The output must be like:**

account1 name is: Your First name and Roll No.

account2 name is: Your Last name and Roll No.

account3 name is: Your Course Name and Roll No.

**Task E [Marks: 8]**

Now repeat the above performed Task by separating the interface from the implementation.

Create an **Account.h** file for class definition showing only declarations for the member functions.

Create an **Account.cpp** file to define the class member functions. Create an Main.cpp file to contain the driving code.

|  |
| --- |
| **Function.cpp:**  void account::setAccount(string accountTitle ){ cout<<"enter title"<<endl; //to get title cin>>accountTitle; } void account::setAccountNumber (int Number ){  int accountNumber;  cout<<"enter account number "<<endl; //to set number  cin>>Number;  accountNumber = Number; } string account::getAccount( void ){  return AccountTitle; //getting title } int account:: getAccountNumber(void ){ //getting number  return accountNum; } account::account() {  string accountTitle ; } account::account( string Title,int accountNumber) {  cout << "Object has a name : " << Title << endl; //roll no. and name displaying using construct  cout << "Object has a roll no. : " << accountNumber << endl;  } account::account( string Title) {  cout << "Object has a name : " << Title << endl; //displaying name  }  **main.cpp:**  int opt; 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;  cout<<"4.TASK 4"<<endl;  cout<<"5.TASK 5"<<endl;  cout<<"6.EXIT"<<endl;  cin>>opt;  if(opt==1){  cout<<"class has been created"<<endl;  }  if(opt==2){  account myAccount;  string AccountTitle;  int Number;  myAccount.setAccount(AccountTitle); //you have to call this function by commenting other functions  myAccount.setAccountNumber(Number);  myAccount.getAccount();  myAccount.getAccountNumber();  }  if(opt==3){  account acc("ali");  }  if(opt==4){  account Account1("Nimra",1234);  account Account2("Maqbool",3456);  account Account3("oop",12345);  }  if(opt==5){  cout<<"the tasks are done in account.h and account.cpp"<<endl;  }  if(opt==6){  cout<<"YOU CHOOSE TO EXIT"<<endl;  exit(4);  } } while(opt>=1 && opt<=6);  }  **header:**  using namespace std;  class account {  string AccountTitle;  int accountNum; //making private variables public:  account();   void setAccount(string);   void setAccountNumber(int); //functions defination  string getAccount(void);   int getAccountNumber(void);   account(string Title, int accountNumber);   account(string Title); //constructer }; |

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