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

|  |  |
| --- | --- |
| **Course Instructor: Usama Bin Shakeel** | **Dated: 11/05/2022** |
| **Teaching Assistant: Aqsa Khalid** | **Semester: Spring 2022** |
| **Lab Engineer: Nadir Abbas** | **Batch: BSCE2021** |

# **Lab 9A. Custom Array Class with Templates**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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**

The array of type class contains the objects of the class as its individual elements. Thus, an array of a class type is also known as an array of objects. An array of objects is declared in the same way as an array of any built-in data type.

**Lab Task**

**Task A: Custom Array Class [Marks: 40]**

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

***Private Data Members:***

arraySize, noOfElem, \*arr

***Public Member Functions:***

**//constructor to create and initialize array to zero**

*custArray(int)*

*{*

*}*

***//destructor to delete array***

*~ custArray ()*

*{*

*}*

Do the following operations in main function:

1. Create object of class and pass size of array.

|  |
| --- |
| Function.h:  template<typename T> class customizedArray { //making class  private:  int sizeOfArray; //declaring  int NoOfElement; //declaring  T \*array; public:  customizedArray(int s) {  sizeOfArray = s; //copying  NoOfElement = 0; //placing it to zero   array = new T[sizeOfArray];  for (int i = 0; i < sizeOfArray; i++) {  array[i] = 0; //setting values to zero  }  }  main.cpp:  int s ;  customizedArray<int> c(5); //calling object |

**b)** Extend task A, with the following information:

***Public Member Functions:***

**//add function to assign values to the array**

void add(int)

{

}

**//expand function to increase size of the array**

void expand(int)

{

}

Do the following operations in main function:

1. Create an object of class, add, and display 5 elements, expand array to 15, add 10 elements, and display them.

|  |
| --- |
| Function.h:  template<typename T> class customizedArray { //making class private:  int sizeOfArray; //declaring  int NoOfElement; //declaring  T \*array; public:  customizedArray(int s) {  sizeOfArray = s; //copying  NoOfElement = 0; //placing it to zero  array = new T[sizeOfArray];  for (int i = 0; i < sizeOfArray; i++) {  array[i] = 0; //setting values to zero  }  }   void add() {  for (int i = NoOfElement; i < sizeOfArray; i++) {  cout << "ENTER DATA = ";  cin >> array[i]; //taking input  NoOfElement++;  }  }  void display() {  for (int i = 0; i < sizeOfArray; i++) {  cout << array[i] <<" "; //displaying array elements  }  cout << endl;  }   void expand() {  sizeOfArray= sizeOfArray +10; //increasing size  T \*array1 = new T[sizeOfArray]; //making a new dynamic array  for (int i = 0; i < sizeOfArray; i++) {  array1[i] = array[i]; //copying  }  delete[]array; //deleting  array = array1; //copying  array1=NULL; //null it   }   ~customizedArray() {  cout << "DESTRUCTOR IS CALLED.." << endl;  delete[]array; //deleting  } };  main.cpp:  int main() {  int s ;  customizedArray<int> c(5); //calling object  c.add();  c.display();  c.expand(); //calling  c.add();  c.display();  return 0; }  output:  A picture containing 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: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_