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

|  |  |
| --- | --- |
| **Course Instructor: Usama Bin Shakeel** | **Dated:** |
| **Teaching Assistant: Zain** | **Semester: Spring 2023** |
| **Lab Engineer: Rana Hamza Shakil** | **Batch: BSCE2022** |

# **Lab 11 B. Template Classes with Usage of Vectors**

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

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

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

## **Objective**

The objective of this lab is to observe the basic knowledge of programming 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 Backgrou****nd**

Template classes are classes that can be used with different types of data. They are defined with one or more type parameters that specify the types of data that can be used with the class. Vectors are commonly used containers in C++ that can store a collection of elements of the same type.

Vectors are dynamic arrays that can grow and shrink in size as needed. They are defined in the standard library header file <vector>. To create a template class that uses vectors, you can define a class that takes a vector type as a template parameter.

**Lab Task**

**Task A [Marks: 5]**

Please follow the following steps before starting the below tasks:

1. Create a separate header file (**.h file**) for each class declaration of data members and member functions

2. Create a separate source file (**.cpp file**) for the implementation of the class member functions.

3. Create **main.cpp** file for creating objects of the class and other driving codes.

**Task B: Template Classes with Usage of Vectors [Marks: 35]**

**Step 1: Define a template class for a shopping cart**

Create a template class called ShoppingCart that can store any type of product (e.g., int, double, string, or a custom Product class). The shopping cart should have the following member functions:

* void addItem(const T& item): adds an item of type T to the shopping cart. void removeItem(int index): removes the item at the specified index from the shopping cart.
* T getItem(int index) const: returns the item at the specified index from the shopping cart.
* int getSize() const: returns the number of items in the shopping cart.
* double getTotal() const: returns the total cost of all items in the shopping cart.

**Step 2: Implement a real-world scenario using the ShoppingCart template class**

Imagine you are building an e-commerce website that sells various products, such as electronics, clothing, and books. You need to implement a shopping cart feature that allows customers to add and remove items, and view their total cost. Write a program that simulates a customer's shopping experience. The program should use the ShoppingCart template class to store the customer's items. Here are the steps:

* Create a vector of products. You can use a custom Product class with attributes such as name, price, and category.
* Display the available products to the customer and ask them to select one or more items.
* For each selected item, add it to the shopping cart using the addItem function of the ShoppingCart class.
* Display the contents of the shopping cart, along with the total cost, and ask the customer if they want to remove any items.
* If the customer chooses to remove an item, use the removeItem function of the ShoppingCart class to remove the item at the specified index.
* Display the final contents of the shopping cart and the total cost.

**Step 3: Enhance the ShoppingCart template class**

To make the ShoppingCart template class more versatile, add the following member functions:

* void clear(): removes all items from the shopping cart.
* bool isEmpty() const: returns true if the shopping cart is empty, false otherwise.
* void sortItems(): sorts the items in the shopping cart by price, from lowest to highest.

Update your program from Step 2 to use these new member functions. Display the contents of the shopping cart after each operation to show the changes.

|  |
| --- |
| // Paste your code here |

### 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/Quiz | 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\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_