**National University of Computer and Emerging Sciences**



**Lab Manual 07**

**Object Oriented Programming**

Dated : 30th April,2021

|  |  |
| --- | --- |
| Course Instructor | Miss Abeeda |
| Lab Instructor (s) | Siddiqua Nayyer  Dilawar Shabbir |
| Section | A |
| Semester | Spring 2021 |

Department of Computer Science

FAST-NU, Lahore, Pakistan

## Objectives

After performing this lab, students shall be able to:

* Create overloaded constructors
* Create destructors
* Handle dynamic memory for class data members as well as declaring objects.
* Overload operators

**TASK:**

In this task, we are going to create a small scale **Event Management System**.

First make three files: **event.cpp, event.h** and **driver.cpp** to separate class header and implementation.

1. Create a **class Event** with following variables: **char\* event\_name**, **char\* event\_venue**, **char event\_date[11]** and **char event\_time[9]**.

* Input format for event\_date: dd-mm-yyyy
* Input format for event\_time: hh:mm am/pm

1. Implement default constructor and overloaded constructor. Print “Default Constructor Called” and “Overloaded Constructor Called” in the respective constructors. The declaration for overloaded constructor is as follows:

* **Event(char event\_name[20], char event\_venue[50], char event\_date[11],char event\_time[9]);**

1. Implement all setters and getters for class Event. You can create a helper function **userInput()** to input event details.
2. Implement the destructor **~Event()** for class **Event**. Print “Destructor Called” in the destructor. Deallocate all the dynamically allocated memory of class data members.
3. Maintain an array of three Event objects in the **driver.cpp**. i.e.

* **Event\* events = new Event[3];**

1. Implement **addEventOver()** function. This function will add a new **Event** object to the **events** object array using overloaded constructor.
2. Implement **addEventSet()** function. This function will add a new **Event** object to the **events** object array using setters.
3. Implement **displayEvents()** function. This will display all events currently present in the **events** object array. You can use getters in this function.
4. Implement a menu program in **driver.cpp.**

**Sample Menu Program:**

*Menu Program*

*Add New Event Using Overloaded Constructor: Enter 1*

*Add New Event Using Setters: Enter 2*

*Display All Events: Enter 3*

*Exit: Enter 0*

*Enter your choice: 1*

*Add New Event*

*Enter event name: Programming 101*

*Enter event venue: Rafaqat Ali Auditorium*

*Enter event date: 06-10-2020*

*Enter event time: 11:30 AM*

**Note:**

* Deallocate all dynamically allocated memory.
* Do not use copy constructor.
* Do not use strcpy() function. Copy the character array manually where needed based on ending ‘\0’ character.
* Follow all the code indentation, naming conventions and code commenting guidelines.

**TASK 2:**

Implement a class called **Complex**. The Complex class will have two data members:

* int real; // The real part of complex number
* int imaginary; // Imaginary part of the complex number.

You have to implement default constructor, overloaded constructor, print output, destructor and overload the operators **+, - , << , >>, ==, !=,=, \*, / , ++, --**

**Sample Run:**

|  |  |
| --- | --- |
| **Driver.cpp** | **Output** |
| int main()  {  Complex C1;  Complex C2(5,6);  Complex C3;  cout<<"Input a complex number"<<endl;  //take input here  //print numbers here,using a print output function of the class  if(C1==C2)  cout<<"C1 == C2"<<endl;  else  cout<<"C1 != C2"<<endl;  if(C1!=C3)  cout<<"C1 != C3"<<endl;  else  cout<<"C1==C3"<<endl;  Complex C4= C2 - C3;  C4.printOutput();  C4 = C1++;  C4.printOutput();  C4 = C1--;  C4.printOutput();    C4 = ++C1;  C4.printOutput();  C4 = --C1;  C4.printOutput();  C4 = C1\*C2;  C4.printOutput();  C4 = C1/C2;  C4.printOutput();  system("pause");  return 0;  } |  |
|  |  |

**Note:**

* Deallocate all dynamically allocated memory.
* Make separate complex.h, complex.cpp and driver.cpp files.
* Follow all the code indentation, naming conventions and code commenting guidelines.