**National University of Computer and Emerging Sciences**

Logo

Description automatically generated with medium confidence

**Lab Manual # 04**

Object Oriented Programming

|  |  |
| --- | --- |
| Course Instructor | Mr. Waqas Manzoor |
| Lab Instructor (s) | Miss. Sidra Zafar  Miss. Hira Ilyas |
| Section | BSE-2A |
| Semester | Spring 2022 |

Department of Computer Science

FAST-NU, Lahore, Pakistan

**Important Note:**

* **Define your class in “Date.h” and “Employee.h” file, Implement all the functions of Date class in “Date.cpp” and “Employee.cpp” and Test your class in “Driver.cpp”**

**Problem 1**

**Exercise 1.1:**

* Create a class **Date** having following private data members:

Int Day

Int Month

Int Year

* Create an object of Date “***date1***” and run your program

**Exercise 1.2 [Default Constructor]:**

* Write a default Constructor of Date that initializes the object to 1st January 1926 and prints “*Default Constructor Called*” in start
* Now run your program and test what does date1 prints?

**Exercise 1.3 [Print Function]:**

* Implement a function **Print** in Date class which prints a date in following format:

dd/mm/yyyy (e.g. 1/1/1926 for date1)

* Print object date1 in your main function and run the program.
* What does it print and how can we initialize the data of date1 at the time of creation?

**Exercise 1.4 [Overloaded Constructor with Default Argument]:**

* Write an overloaded Constructor of Date class that initializes the date object to date, month and year provided as parameter and prints “Overloaded Function Called”
* Now create another object ***independanceDay*** in main that is 14/08/2018
* Print ***independanceDay*** by calling Print function of Date class and run your program

**Exercise 1.5 [Destructor]:**

* Write Destructor of Date class that prints “Destructor called”
* Run your program and test it

**Exercise 1.6 [Input Function]:**

* Write a function **Input** in your Date class that takes input from user to populate a Date object
* Call “*date1.Input()*” and “*date1.Print()*” in your driver program and test it

**Exercise 1.7 [Setters]:**

* Create an object ***xmasDay*** using default constructor
* Print xmasDay and see what it prints
* Write Setters i.e. SetDay, SetMonth and SetYear in your class
* Now set xmasDay to 25/12/2018 using Setters in main

**Exercise 1.8 [Getters]:**

* Write Getters i.e. GetDay, GetMonth and GetYear in your date class
* Now print xmasDay using Getters in your Driver program

**Exercise 1.9 [Built-in Assignment Operator]:**

* Create an object **temp** of Date class
* Assign value of xmasDay to temp
* Print temp and test your program

**Exercise 1.10 [Passing object by value]:**

* Write a function **int Compare(Date)** that compares two dates, returns 1 if left hand side object is greater than right hand side object, -1 if lhs is smaller and 0 otherwise
* Test your function

**Exercise 1.11 [Return object by value]:**

* Write a function **Date IncrementMonth()** that returns a newly created Date object with one month next to the current date object. For example, if date1 = 2/01/2016 *date1.IncrementDate()* will return 2/02/2016 without changing date1
* Print both the date1 and newly created date in your driver program to test the result

**Problem 2**

Create a class **Employee** that has private member variables to store each of the following:

class Employee

{

private:

static int totalEmp; // initialize it to zero yourself

int empID;

char name[size\_name];

char address[size\_address];

public:

}

*Note: The sizes of the two c-strings should be declared as* ***named constants*** *before the start of the class declaration.*

Now, carry out the following tasks in the given order:

## **Task 2.1**

Implement a **default constructor** for **Employee** class that assigns empty cstrings to the name and department. Employees Id should be assigned by yourself such that 1st Employee should have ID 1 and 2nd Employee should have ID 2 and so on.

## **Task 2.2**

Implement an **overloaded constructor** for **Employee** class that accepts the following 2 values (in the given order) as arguments and assigns them to the appropriate member variables: employee’s name, and department name. Employees Id should be assigned by yourself such that 1st Employee should have ID 1 and 2nd Employee should have ID 2 and so on.

**Task 2.4**

Implement the **getter** for **totalEmp** and **empID** variables.

**Task 2.5**

Implement the getter and setter functions for the name and address variable.

## **Task 2.6**

Implement a member function **display** of the **Employee** class which should neatly display all attributes of an employee on screen. This function should use the getter functions to get the values of all attributes of an employee.

## **Task 2.7**

In the main function, dynamically allocate an array of **Employee** objects. Your program should ask the user about the size of the array and then allocate the array dynamically. Then, your program should ask the user to enter the values of all attributes (name and address) for each **Employee**. After that, your program should display the total number of employees (by using the **getter** for **totalEmp)** as well as the details of each object on screen (by calling the **display** function for each **Employee**).

**Task 2.7**

Implement the destructor. (Do we really need destructor in this program? Give Reason.)