

## Object Oriented Programming Lab

Laboratory Manuals

**Hareem Aslam – Associate Lecturer** Department of Computer Sciences Faculty of Information Technology (FOIT) University of the Central Punjab, Lahore

#### Contents

[Course Objectives: 3](#_bookmark0)

[Preferred Tool(s) 3](#_bookmark1)

[Text/Reference Book(S) 3](#_bookmark2)

[General Instructions: 3](#_bookmark3)

[Instructors 3](#_bookmark4)

[Lab Manual 01 4](#_bookmark5)

[Task 01: GPA Calculation 6](#_bookmark6)

[Task 02: Swap two integers 6](#_bookmark7)

[Task 03: Copy string 6](#_bookmark8)

[Task 04: Array rotation 6](#_bookmark9)

[Task 05: File Handling 6](#_bookmark10)

[Lab Manual 02 7](#_bookmark11)

[Task 01: Triangle Class 9](#_bookmark12)

[Task 02: University Management System 9](#_bookmark13)

[Task 03: Car Class 10](#_bookmark14)

[Task 04: Complex Number 10](#_bookmark15)

[Lab Manual 03 12](#_bookmark16)

[Graded Lab 1 - Task 14](#_bookmark17)

[Task 01: Student Extension 15](#_bookmark18)

[Task 02: Circle Class 15](#_bookmark19)

[Task 03: Car Class with Acceleration and Brake 16](#_bookmark20)

[Lab Manual 04 18](#_bookmark21)

[Task: Student Class 20](#_bookmark22)

[Lab Manual 05 22](#_bookmark23)

[Graded Lab 2 - Task 24](#_bookmark24)

[Task 01: (Constant Data member and Destructor) 25](#_bookmark25)

[Task 02: (Shallow Copy of the Object) 25](#_bookmark26)

[Task 03: (Deep Copy of the Object) 26](#_bookmark27)

[Lab Manual 06 27](#_bookmark28)

[Task 01: Person Class 29](#_bookmark29)

[Task 02: Person class extension 30](#_bookmark30)

[Lab Manual 07 31](#_bookmark31)

[Task: Polynomial Problem 33](#_bookmark32)

[Lab Manual 08 36](#_bookmark33)

[Graded Lab 3 - Task 38](#_bookmark34)



**Hareem Aslam – Associate Lecturer**

Department of Computer Sciences

[Task 01: Point 7, 8, 9 is not the part of graded task. Practice these tasks after the graded activity.](#_bookmark35)

[.......................................................................................................................................................... 38](#_bookmark35)

[**Lab Manual 09 39**](#_bookmark36)

[Task 01: Single Class Date 41](#_bookmark37)

[Task 02: Classes Day, Month, Year, and Date 41](#_bookmark38)

[Task 03: Classes Person and Company 43](#_bookmark39)

[Task 04: Classes PersonalData, Address, ContactData, Person, and Company 43](#_bookmark40)

[**Lab Manual 10 46**](#_bookmark41)

[Task: Student Management System 48](#_bookmark42)

[**Lab Manual 11 50**](#_bookmark43)

[Task 01: Price 52](#_bookmark44)

[Task 02: Fruit 52](#_bookmark45)

[Task 03: FruitBasket 52](#_bookmark46)

[Task 04: FruitShop 53](#_bookmark47)

[Task 05: Multi-Level Fruit Management System 53](#_bookmark48)

[**Lab Manual 12 54**](#_bookmark49)

[**Graded Lab 4** 56](#_bookmark50)

[Task: Transportation Management System 57](#_bookmark51)

[**Lab Manual 13 59**](#_bookmark52)

[Task: University Bookstore 61](#_bookmark53)

[**Lab Manual 14 64**](#_bookmark54)

[**Graded Lab 5** 66](#_bookmark55)



**Hareem Aslam – Associate Lecturer**

Department of Computer Sciences

#### Course Objectives:

This module will provide the students with a solid theoretical understanding of, as well as practical skills in, object-oriented programming. Practical skills will be learnt using the C++ programming language. The primary aim of the module is to enable the students to tackle complex programming problems, making good use of the object-oriented programming paradigm to simplify the design and implementation process. Laboratory sessions and tutorials will be provided to encourage acquisition of practical problem-solving skills.

#### Preferred Tool(s):

* Microsoft Visual Studio 2022, C++

#### Text/Reference Book(S)

* ***C++ How to Program by Deitel and Deitel, 8th Edition.***
* C++ Programming Language, 4th Edition by Bjarne Stroustrup.
* Effective C++: 55 Specific Ways to Improve Your Programs and Designs by Scott Mayers, 3rd Edition
* Effective Modern C++ by Scott Mayers

#### General Instructions:

* In this Lab, you are **NOT** allowed to discuss your solution with your colleagues, even not allowed to ask how s/he is doing, this may result in negative marking. You can **ONLY** discuss with your Lab Instructor.
* Viva for each task will be taken and considered as a performance.
* Your Lab Instructor will be available in the Lab for your help. Alternatively, you can visit your Lab Instructor in office hours (shared on portal) or send your queries via email to one of the followings.

|  |  |  |
| --- | --- | --- |
| Instructors: | | |
| Lab Instructor | **Hareem Aslam** | [hareem.aslam@ucp.edu.pk](mailto:hareem.aslam@ucp.edu.pk) |
| Course Instructor | **Dr. Ahmad Shabbar Kazmi** | [ahmad.kazmi@ucp.edu.pk](mailto:ahmad.kazmi@ucp.edu.pk) |
| Course Coordinator | **Dr. Abbas Khalid** | [abbas.khalid@ucp.edu.pk](mailto:abbas.khalid@ucp.edu.pk) |



**Hareem Aslam – Associate Lecturer**

Department of Computer Sciences



# Lab Manual 01





**Hareem Aslam – Associate Lecturer**

Department of Computer Sciences



|  |  |
| --- | --- |
| **Lab 01** | |
| **Topic** | Revision Lab |
| **Objective** | The basic purpose of this laboratory is revision of some preliminary concepts of C++ that has been covered in the course of Introduction to Computing and Programming Fundamentals. Its objective is to:   * Recall students previously learned basic concepts. * Revision of arrays and functions. * Understanding problem statements and designing an appropriate solution. |

###### Instructions:

* Indent your code.
* Comment your code.
* Use meaningful variable names.
* Plan your code carefully on a piece of paper before you implement it.
* Name of the program should be same as the task name. i.e., the first program should be Task\_1.cpp
* **void main() is not allowed. Use int main()**
* **You are not allowed to use system**("**pause**")
* **You are not allowed to use any built-in functions**
* **You are required to follow the naming conventions as follow:**
  + **Variables:** firstName; (no underscores allowed)
  + **Function:** getName(); (no underscores allowed)

**Understanding of the question is part of the evaluation.**

**Students are required to complete the following tasks in lab timings.**



**Hareem Aslam – Associate Lecturer**

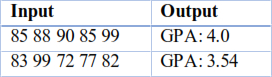
Department of Computer Sciences

**Tasks**

##### Task 01: GPA Calculation

Write a C++ program which:

* Defines an array of size “5”
* Set Values of array by taking Subject Marks as an input from user
* Pass the array to the function
* The function calculates & returns the GPA
* Display the GPA on the Console



##### Task 02: Swap two integers

Create a C++ program that:

* Takes values of two integers as an input from user.
* Swap (interchange) the values of integers using pointer.
* Display the integer values after swapping

##### Task 03: Copy string

Write a non-returning function **strCopy** which takes only two **char\*** as parameters, one is destination and other is source. Your task is to copy all the data of the source into destination.

vo d strCopy(char\*& dest, const char\* src);

**Note: You are not allowed to use any bu lt- n funct on.**

##### Task 04: Array rotation

Write a function that accepts an int array, the array’s size and an integer K as. The function should circularly rotate the array K number of times to the right.

**For example:**

initial array: {1,2,3,4,5,6}

size = 6

K = 2

Modified array: {5,6,1,2,3}

##### Task 05: File Handling

Create a file data.txt with integers as given below. Write a function that reads integers one by one and separates even and odd numbers into two dynamic arrays, say evenArray, oddArray. **Regrow** both the arrays. Stop reading the numbers when **-1** is found in the file. Find maximum number from both the arrays and write them in file output.txt as:

**Data.txt** 1 5 6 2 3 4 5 8 2 0 1 3 9 5 6 -1 5 6 8 3 5 6 4 8

**Note:** Do not read all the numbers in an array. Create dynamic arrays and regrow.



**Hareem Aslam – Associate Lecturer**

Department of Computer Sciences



Lab Manual 02





**Hareem Aslam – Associate Lecturer**

Department of Computer Sciences



|  |  |
| --- | --- |
| **Lab 02** | |
| **Topic** | Classes in C++, access specifiers and methods |
| **Objective** | * Good Code Writing Style and Function Code Templates * Dry Run and use of the Debugger to look at variable addresses and values * Making students familiarize with classes and their implementations in C++ * Class Methods and Access Specifiers |

**Instructions:**

* Indent your code.
* Comment your code.
* Use meaningful variable names.
* Plan your code carefully on a piece of paper before you implement it.
* Name of the program should be same as the task name. i.e. the first program should be Task\_1.cpp
* In each file of your task either it is .cpp file or .h file Your name and roll no must be mention in the start of your file as comment. For example, **//L1F22BSCS1234 – Ali Rehman**
* **void main()** is not allowed. Use **int main()**
* You are not allowed to use any built-in functions
* You are required to follow the naming conventions as follow:
  + **Variables:** firstName; (no underscores allowed)
  + **Function:** getName(); (no underscores allowed)
  + **ClassName:** BankAccount (no underscores allowed)

**Understanding of the question is part of the evaluation.**

**Mobile phone is strictly prohibited during the lab either lab is graded or not.**

**Students are required to complete the following tasks in lab timings. Submission of each task in prescribed timing is mandatory.**



**Hareem Aslam – Associate Lecturer**

Department of Computer Sciences

**Tasks**

##### Task 01: Triangle Class

**Perform this task in single source file.**

Create a class named as **Triangle** having following private attributes:

* **base (double)**
* **height(double)**

Now write the following for the above-mentioned class:

1. The program should include functions to assign user defined values to the above-mentioned variable and a display function to show the values

***Note:*** *Validate the values in the above function. For eg: height can never be negative etc.*

1. Write a non-returning **display** function to print the attributes of the class.
2. Write a function **calculateArea** which calculates the area of the triangle and returns it.
3. Write a program to create **five objects of Triangle** with different data. Display the attributes of the triangle through display function having greatest area (use function written in the class to calculate area).

##### Task 02: University Management System

**Perform this task in single source file.**

**A Student is an object in a university management System. Analyse the concept and identify the data members that a student class should have. Also analyse the behaviour of student in a university management System and identify the methods that should be included in Student class.**

Create a class named as **Student** having the data members which you have identified above.

* The program should include functions to assign user defined values to the above-mentioned variable and a display function to show the values.

***Notes:*** *Validate the values in the above function.*

* Write a non-returning **display** function to print the attributes of the class in the format given below.
* Write a main program to create **five objects** of **Student** with different data. And then display all of the object’s data



**Hareem Aslam – Associate Lecturer**

Department of Computer Sciences

**Perform Task 3 and Task 4 in separate source, header and main driver file.**

##### Task 03: Car Class

Create a class named as **Car** having following private attributes:

* **carModel(intger)**
* **carName(integer)**
* **carPrice(integer)**
* **carOwner(string)**

Now write the following for the above-mentioned class:

* The program should include functions to assign user defined values to the above-mentioned variable and a display function to show the values.

***Notes:*** *Validate the values in the above function.*

* Write a non-returning **display** function to print the attributes of the class in the format given below.
* Write a main program to create **five objects** of **Car** with different data. And then display all of the object’s data

##### Task 04: Complex Number

Create a class named as **ComplexNumber** having following private attributes:

* **realPart(double)**
* **imaginaryPart(double)**

Now write the following for the above-mentioned class:

1. The program should include functions to assign user defined values to the above-mentioned variable and a display function to show the values.

***Notes:*** *Validate the values in the above function.*

1. Write a non-returning **display** function to print the attributes of the class in the format given below.
2. Write a program to create **two objects** of **Complex Number** with different data. And then display all the complex number objects in the following format.
3. Write a function **sum** which sum of two complex numbers which you have created in last step.



**Hareem Aslam – Associate Lecturer**

Department of Computer Sciences

**Sample Outputs:**



Enter 1st Complex Number:

Enter Real part of Complex Number: -6 Enter Imag nary part of Complex Number: 3

Enter 2nd Complex Number:

Enter Real part of Complex Number: 25

Enter Imag nary part of Complex Number: -30

1st Complex Number: -6 + 3 2nd Complex Number: 25 - 30

Sum of (-6 + 3 ) + (25 - 30 ) = 19 + -27



**Hareem Aslam – Associate Lecturer**

Department of Computer Sciences



# Lab Manual 03





**Hareem Aslam – Associate Lecturer**

Department of Computer Sciences



|  |  |
| --- | --- |
| **Lab 03** | |
| **Topic** | * Setters and getters * Constructors and Destructor * **Graded Lab** |
| **Objective** | * Classes in C++with access specifiers and methods * getter(accessors) and setters(mutators) * Default and parameterized constructors * Making students familiarize with the concepts of constructor, overloaded constructor and destructor. * Update Class templates with Good Code Writing Style |

Instructions:

* Strictly follow the Good Code Writing Style which is already shared by your theory teacher.
* In each file of your task either it is .cpp file or .h file Your name and roll no must be mention in the start of your file as comment. For example, **//L1F22BSCS1234 – Ali Rehman**
* **void main()** is not allowed. Use **int main()**
* You are not allowed to use any built-in functions.
* You are required to follow the naming conventions as follow:
  + **Variables:** firstName; (no underscores allowed)
  + **Function:** getName(); (no underscores allowed)
  + **ClassName:** BankAccount (no underscores allowed)
* Students must follow the "Code Writing Guide".
* **You are not allowed to use arrays as class variables**
* **Functions should not have class returned from functions or used as arguments.**
* **Perform all the tasks in separate source, header and main driver file.**

Guidelines for graded lab:

1. **The use of mobile phones is strictly prohibited in this lab.**
2. **Movement from your assigned seat is not allowed.**
3. **It is mandatory to revise your concepts during the lab session.**
4. **Sharing notebooks or any study materials is prohibited in the lab.**
5. **There will be no extensions granted for task submission deadlines.**

**Understanding of the question is part of the evaluation.**

**Mobile phone is strictly prohibited during the lab either lab is graded or not.**



**Hareem Aslam – Associate Lecturer**

Department of Computer Sciences

**Students are required to complete the following tasks in lab timings. Submission of each task in prescribed timing is mandatory.**

### Graded Lab 1 - Task

**Time: 50 minutes**

**Submission Time: 11:59 am**

**Perform this graded Task in 3 different files. .h , .cpp and main.cpp.**

Create a class named as **Student** having the following private data members.

* **rollNo(integer)**
* **name(string)**

###### address(string)

* **cgpa(float)**

###### fee(double)

* **semester(integer)**

Now write the following for the above-mentioned class:

1. Write a non-returning **input()** function to input the attributes of the class.
2. Write a non-returning **display()** function to print the attributes of the class..
3. Write a function **increaseFee()** which accepts the student fee and increase the fee by 7.5%. This function should tell the student about the raise in fee and new fee.
4. Create 5 object of class Student and test your code. All inputs will be taken from user.

***Notes:*** *Validate the values in the above function*



**Hareem Aslam – Associate Lecturer**

Department of Computer Sciences

##### Task 01: Student Extension

**Tasks**

Extend the Graded Task

1. Write separate setter and getter function which should receive parameter and initialize all the attributes with the respective values received in the parameters.
2. Write appropriate **default constructor** for the class Student.
3. Write a non-returning **input()** function to input the attributes of the class by calling appropriate setter functions.
4. Write a non-returning **display()** function to print the attributes of the class by calling appropriate getter functions.
5. Write a function **increaseFee()** which accepts the student fee and increase the fee by 7.5%. This function should tell the student about the raise in fee and new fee.

***Notes:*** *Validate the values in the above function*

##### Task 02: Circle Class

Write a **Circle** class that has the following private member variables:

* **radius**: a double
* **pi**: a double initialized with the value 3.14159

The class should have the following public member functions:

1. Write the default constructor which that sets radius to 0.0.
2. Write the parameterized constructor which Accepts the radius of the circle as an argument.
3. Define **setRadius** a mutator function for the radius variable.
4. Define **getRadius** a accessor function for the radius variable.
5. Define **getArea** returns the area of the circle, which is calculated as area = pi \* radius

\* radius

1. Define **getDiameter r**eturns the diameter of the circle, which is calculated as

diameter = radius \* 2

1. Define **getCircumference.** Returns the circumference of the circle, which is calculated as circumference = 2 \* pi \* radius.

Write a program that demonstrates the Circle class by asking the user for the circle’s radius, creating a Circle object, and then displaying the circle’s area, diameter, and circumference.



**Hareem Aslam – Associate Lecturer**

Department of Computer Sciences

##### Task 03: Car Class with Acceleration and Brake

Write a class named **Car** that has the following private member variables:

* **yearModel:** An int that holds the car’s year model.
* **make:** A string that holds the make of the car.
* **speed:** An int that holds the car’s current speed.

In addition, the class should have the following constructor and other public member functions.

1. Write the default constructor which set the appropriate 0 values to data members.
2. Write the appropriate parameterized constructor should accept the car’s year model and make as arguments. These values should be assigned to the object’s yearModel and make member variables. The constructor should also assign 0 to the speed member variables.
3. Write aappropriate getter and setter functions to get the values stored in an object’s yearModel, make, and speed member variables.
4. Define the **accelarete()** function which should add 5 to the speed member variable each time it is called.
5. Define the **brake()** function should subtract 5 from the speed member variable each time it is called.

***Notes:*** *Validate the values in the above function.*

***Hint:*** *Car speed cannot be negative and above from 150. At the time of brake if car speed is near to zero it’s mean car is already stopped.*

Demonstrate the class in a program that creates a Car objects, and then calls the accelerate function five times. After each call to the accelerate function, get the current speed of the car and display it. Then, call the brake function five times. After each call to the brake function, get the current speed of the car and display it.

**In this task accelerate and brake functions are bonus task.**



**Hareem Aslam – Associate Lecturer**

Department of Computer Sciences

**Sample output:**

Enter Car make year: 2002

Enter Car make Company: Toyota

Car Year Model: 2002 Car Make: Toyota

Accelerating:

Current Speed: 5mph Current Speed: 10mph Current Speed: 15mph Current Speed: 20mph Current Speed: 25mph

Braking:

Current Speed: 20mph Current Speed: 15mph Current Speed: 10mph Current Speed: 5mph Current Speed: 0mph



**Hareem Aslam – Associate Lecturer**

Department of Computer Sciences



# Lab Manual 04





**Hareem Aslam – Associate Lecturer**

Department of Computer Sciences



|  |  |
| --- | --- |
| **Lab 04** | |
| **Topic** | * Arrays of Objects and Pointers to Objects * Object (Pointer) as parameter and Return Type * Constant functions, parameters & attributes |
| **Objective** | * Making students familiarize with the concepts of object passed as parameter and return type. * Deep Copy Vs Shallow Copy   + Making students familiarize with the concepts of constants. |

###### Instructions:

* + Indent your code properly.
  + Use meaningful variable and function names. Use the camelCase notation.
  + Use meaningful prompt lines/labels for all input/output.
  + Strictly follow the Good Code Writing Style which is already shared by your theory teacher.
  + In each file of your task either it is .cpp file or .h file Your name and roll no must be mention in the start of your file as comment. For example, //L1F22BSCS1234 – Ali Rehman
  + Students must follow the "Code Writing Guide".
  + You are not allowed to use string. You can use Cstring only as character arrays.
  + You are not allowed to make array of objects in main().
  + You are not allowed to use array of object.
  + Perform all the tasks in separate source, header and main driver file.
  + Paste your code in the given space below and submit your .cpp and .h file along with your .docx file.

###### Understanding of the question is part of the evaluation.

**Mobile phone is strictly prohibited during the lab either lab is graded or not.**

**Students are required to complete the following tasks in lab timings. Submission of each task in prescribed timing is mandatory.**



**Hareem Aslam – Associate Lecturer**

Department of Computer Sciences

##### Task: Student Class

**Tasks**

Create a class Student that has private member variables to store each of the following six attributes:

* **Roll Number:** An integer variable that holds the student’s roll number
* **Name:** A ctring that holds the student’s name (assume that max length of name is 40)
* **Number of Quizzes:** An integer to store the number of quizzes taken by the student

***(Note: Number of quizzes can be different for each student)***

* **Marks:** An int\* through which you will allocate an array (of appropriate size) to store the marks obtained by the student in different quizzes
* **Total Marks of each Quiz:** An integer to store the total marks of each quiz.

***(Note: You can assume that total marks of all quizzes of a particular student are same)***

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

***Note: cin and cout is not allowed in getter/setter or constructor.***

1. Write appropriate setter functions for each private attributes.
2. Write appropriate setter functions for each private attributes.
3. Implement a **Default Constructor** for Student class in which roll number should be given the value 0, name should be initialized to empty c-string, marks should be initialized as a null pointer, number of quizzes should be initialized to 0, and total marks of each Quiz should be initialized to 10.
4. Implement **Parameterized Constructor** for Student class that accepts 4 arguments: student’s roll number, student’s name, number of quizzes taken by the student, and total marks of each quiz. The values supplied in the arguments should be used to initialize the corresponding member variables, and dynamically allocate memory through the pointer marks. Each element of the marks array should be initialized to 0.
5. Implement a public member function **getInputFromUser** of the student class, which should ask the user to enter the following 4 attributes: Roll No., Name, No. of Quizzes taken by the student, and Total marks of each quiz.

After storing this data in appropriate member variables, this function should allocate the Marks array. If the Marks array has been previously allocated make sure to deallocate it first (see the description of parameterized constructor above in step 3). After that this function should call the following public member function inputMarks to input the marks of each quiz (see next step).



**Hareem Aslam – Associate Lecturer**

Department of Computer Sciences

1. Implement a **public** member function **inputMarks** of the Student class, which should ask the user to enter the marks of all quizzes (one-by-one) and store these marks in the array inside the calling Student object. Also perform input validation on the quiz marks entered by the user. Marks of each quiz should be greater than or equal to 0 and less than or equal to the total marks of the quiz.
2. Implement a **public member function display** of the Student class which should display the roll number, name, marks obtained by the student in different quizzes, and the average marks obtained by the student.

Now, write a driver program (main function) which should demonstrate all of the above functions.

Note: cin and calculations are not allowed in main

1. Write a driver program which should create two objects **student1** and **student2** ask the user to enter all details for each student. Then display details for all the students by using the display function of the class.
2. Create a third object of student says **student3** in such way that parameterized constructor should call.



**Hareem Aslam – Associate Lecturer**

Department of Computer Sciences



# Lab Manual 05





**Hareem Aslam – Associate Lecturer**

Department of Computer Sciences



|  |  |
| --- | --- |
| **Lab 05** | |
| **Topic** | * Arrays of Objects and Pointers to Objects * Object (Pointer) as parameter and Return Type * Constant functions, parameters & attributes * **Graded Lab** |
| **Objective** | * Making students familiarize with the concepts of object passed as parameter and return type. * Deep Copy Vs Shallow Copy * Making students familiarize with the concepts of constants. |

Instructions:

* Strictly follow the Good Code Writing Style which is already shared by your theory teacher.
* In each file of your task either it is .cpp file or .h file Your name and roll no must be mention in the start of your file as comment. For example, **//L1F22BSCS1234 – Ali Rehman**
* **void main()** is not allowed. Use **int main()**
* You are not allowed to use any built-in functions.
* You are required to follow the naming conventions as follow:
  + **Variables:** firstName; (no underscores allowed)
  + **Function:** getName(); (no underscores allowed)
  + **ClassName:** BankAccount (no underscores allowed)
* Students must follow the "Code Writing Guide".
* **You are not allowed to use arrays as class variables**
* **Functions should not have class returned from functions or used as arguments.**
* **Perform all the tasks in separate source, header and main driver file.**

Guidelines for graded lab:

* 1. **The use of mobile phones is strictly prohibited in this lab.**
  2. **Movement from your assigned seat is not allowed.**
  3. **It is mandatory to revise your concepts during the lab session.**
  4. **Sharing notebooks or any study materials is prohibited in the lab.**
  5. **There will be no extensions granted for task submission deadlines.**

**Understanding of the question is part of the evaluation.**

**Mobile phone is strictly prohibited during the lab either lab is graded or not.**

**Students are required to complete the following tasks in lab timings. Submission of each task in prescribed timing is mandatory.**



**Hareem Aslam – Associate Lecturer**

Department of Computer Sciences

**Graded Lab 2 - Task**

Write a class named **Employee** that has the following member variables:

* **name\*:** A c-string that holds the employee s name.
* **idNumber:** An int variable that holds the employee s ID number.

**Time: 40 minutes**

* **department\*:** A c-string that holds the name of the department where the employee works.
* **position\*:** A c-string that holds the employee s job title. ***(Note:*** *Perform the appropriate validation on data members)* The class should have the following constructors:
* Implement a **default constructor** for Employee class in which id number should be given the value 0, name, department, and position should be initialized to empty c-string.
* Implement a **parameterized constructor** for Employee class that accepts the following values as arguments and assigns them to the appropriate member variables by reusing setter functions.

***Note: cin and cout is not allowed in getter/setter or constructor.***

* Write appropriate **setter** functions for each private attribute. when allocating a new dynamic array to class pointers in getters or setter. Previously allocated memory to any call pointer must be deleted before allocation of new memory.
* Write appropriate **getter** functions that return the values in these members variable.
* Implement a public member function **getInputFromUser**, which should ask the user to enter the following attributes: Name, ID number, Departmnet and position.
* Implement a public member function **display** which should display the object details.

***Note: cin and calculations are not allowed in main***



**Hareem Aslam – Associate Lecturer**

Department of Computer Sciences

**Tasks**

Task 01: (Constant Data member and Destructor)

###### Extend the Graded Task.

* Make the data member **idNumber** as constant data member.
* Implement the destructor to deallocate the memory instead of using public function

###### deallocateMemory.

Task 02: (Shallow Copy of the Object)

Write a program in C++ to create a class **Rectangle** with attributes **length and width**.

* Implement a constructor that takes two parameters and initializes the attributes.
* Implement a copy constructor that performs a **shallow copy of the object.**
* Create two Rectangle objects r1 and r2 with different values for length and width, and copy r1 to r2.
* Modify the values of length and width of r1.
* Display the modified values of length and width of r2?

|  |
| --- |
| **Rectangle.h** |
|  |
| **Rectangle.cpp** |
|  |
| **Main.cpp** |
|  |
| **Output:** |
|  |



**Hareem Aslam – Associate Lecturer**

Department of Computer Sciences

Task 03: (Deep Copy of the Object)

Write a program in C++ to create a class called “**Person”** with attributes **name and age**.

* Implement a constructor that takes two parameters and initializes the attributes.
* Implement a copy constructor that performs a **deep copy of the object**.
* Create two Person objects p1 and p2 with different values for name and age, and copy p1 to p2.
* Modify the value of name of p1.
* Display the modified values of name of p2

|  |
| --- |
| **Person.h** |
|  |
| **Person.cpp** |
|  |
| **Main.cpp** |
|  |
| **Output:** |
|  |



**Hareem Aslam – Associate Lecturer**

Department of Computer Sciences



# Lab Manual 06





**Hareem Aslam – Associate Lecturer**

Department of Computer Sciences



|  |  |
| --- | --- |
| **Lab 06** | |
| **Topic** | * Arrays of Objects and Pointers to Objects * Object (Pointer) as parameter and Return Type * Constant functions, parameters & attributes |
| **Objective** | * Making students familiarize with the concepts of object passed as parameter and return type. * Deep Copy Vs Shallow Copy * Making students familiarize with the concepts of constants. |

###### Instructions:

* + Indent your code properly.
  + Use meaningful variable and function names. Use the camelCase notation.
  + Use meaningful prompt lines/labels for all input/output.
  + Strictly follow the Good Code Writing Style which is already shared by your theory teacher.
  + In each file of your task either it is .cpp file or .h file Your name and roll no must be mention in the start of your file as comment. For example, //L1F22BSCS1234 – Ali Rehman
  + Students must follow the "Code Writing Guide".
  + You are not allowed to use string. You can use Cstring only as character arrays.
  + You are not allowed to make array of objects in main().
  + You are not allowed to use array of object.
  + Perform all the tasks in separate source, header and main driver file.
  + Paste your code in the given space below and submit your .cpp and .h file along with your .docx file.

###### Understanding of the question is part of the evaluation.

**Mobile phone is strictly prohibited during the lab either lab is graded or not.**

**Students are required to complete the following tasks in lab timings. Submission of each task in prescribed timing is mandatory.**



**Hareem Aslam – Associate Lecturer**

Department of Computer Sciences

##### Task 01: Person Class

**Task**

Create a C++ program that defines a class called **Person** with attributes

* **Name**(cstring)
* **Age**(integer)
* **Address**(cstring)
* **date of birth** (cstring)

1. Implement appropriate **setters** and **getters** function for all the variables.
2. Implement the **default constructor** of the class to set the values of data members.
3. Implement the **destructor** to deallocate the memory if any.
4. Implements a **getInput** function to take the input from user in appropriate data members.
5. Implement the **display** function to display the person’s details.
6. Implement a function to sort the persons based on their ages in ascending order.
7. In **main()** function implement a dynamic array to store a group of persons. Allow the user to input details for each person and dynamically adjust the size of the array to accommodate all persons.
8. Use this class declaration and define all the required functions and constructors
9. Add a function to sort the persons based on their ages in ascending order.



|  |
| --- |
| Person.h |
| pragma once  fndef flERSON\_H def ne flERSON\_H  class flerson  {  pr vate:  char\* name; nt age;  char\* address;  char\* dateOfB rth; publ c:  // Default constructor flerson();  // flarameter zed constructor  flerson(const char\*, nt a, const char\*, const char\*);  // Destructor  ~flerson();  // Getter and Setter funct ons const char\* getName() const;  vo d setName(const char\*); nt getAge() const;  vo d setAge( nt);  const char\* getAddress() const; vo d setAddress(const char\*);  const char\* getDateOfB rth() const; vo d setDateOfB rth(const char\*);  // Funct on to get nput for a person vo d getInput(); |



**Hareem Aslam – Associate Lecturer**

Department of Computer Sciences



// Funct on to d splay deta ls of a person vo d d splay() const;

//sort ng persons based on age vo d Sort(flerson\*, nt);

};

end f

##### Task 02: Person class extension

1. Extend the Person class with a **static variable** to keep track of the number of objects created.
2. Update the constructor and implement **parameterized constructor** as per the need.
3. Implement a **static function** that returns the total number of persons created.
4. Create instances of the Person class, and display the total count after each object creation.
5. Enhance the Person class to include a private static variable that represents the average age of all created persons
6. Implement a static function to calculate and update this average whenever a new person is added. Create instances of the Person class, and display the average age after each object creation.



**Hareem Aslam – Associate Lecturer**

Department of Computer Sciences



# Lab Manual 07





**Hareem Aslam – Associate Lecturer**

Department of Computer Sciences



|  |  |
| --- | --- |
| **Lab 07** | |
| **Topic** | Operator Overloading |
| **Objective** | * Making students familiarize with the concepts of Operator Overloading * Overloading operators !=, +, -, \*, /, = =, >=, <= etc |

###### Instructions:

* + Indent your code properly.
  + Use meaningful variable and function names. Use the camelCase notation.
  + Use meaningful prompt lines/labels for all input/output.
  + Strictly follow the Good Code Writing Style which is already shared by your theory teacher.
  + In each file of your task either it is .cpp file or .h file Your name and roll no must be mention in the start of your file as comment. For example, //L1F22BSCS1234 – Ali Rehman
  + Students must follow the "Code Writing Guide".
  + You are not allowed to use string. You can use Cstring only as character arrays.
  + You are not allowed to make array of objects in main().
  + You are not allowed to use array of object.
  + Perform all the tasks in separate source, header and main driver file.
  + Paste your code in the given space below and submit your .cpp and .h file along with your .docx file.

###### Understanding of the question is part of the evaluation.

**Mobile phone is strictly prohibited during the lab either lab is graded or not.**

**Students are required to complete the following tasks in lab timings. Submission of each task in prescribed timing is mandatory.**



**Hareem Aslam – Associate Lecturer**

Department of Computer Sciences

##### Task: Polynomial Problem

**Tasks**

A polynomial is an expression consisting of variables and coefficients e.g., 𝑥2 − 4𝑥 + 7

Implement a class of Polynomial with two data members:

1. **dynamic array of coefficients and**
2. **degree.**

***Coefficients*** are the constant values multiplied with variable in a polynomial. E.g., in 𝑥2 − 4𝑥 + 7, coefficient of ‘𝑥’ is 4. If coef is an array of coefficients then:

coef[0] would hold all coefficients of 𝑥0

coef[1] would hold all 𝑥1

coef[n] = 𝑥𝑛.

***Degree*** is largest exponent of that variable. E.g. in 𝑥2 − 4𝑥 + 7 degree is 2.

For polynomial P = 3𝑥4 + 4𝑥2 + 16*,* if someone tries to access P[4] then it should get 3. Similarly if someone tries to access P[3] it should get 0 as there is no term with degree 3.

Write a C++ program to create a class named **“Polynomial”** which must do the following tasks:

* In default constructor, initialize coefficient array with 0s.
* Implement setter method that takes three parameters,
  + coefficient value (int c, e.g.)
  + index of array at which this value (c) should be stored at
  + degree of whole polynomial
* Implement following functions:
  + Additions of polynomials (operator +)
  + Subtraction of polynomials (operator -)
* Implement the default, parameterized, and copy constructors, destructors and accessor methods as needed.

Main function should do the following:

* Declare two polynomials and set their values ONLY by taking input from the user.
* Declare third polynomial and use copy constructor to set it to polynomial 1’s contents. Make sure you perform deep copy otherwise ZERO will be granted for this part!
* Declare fourth polynomial, one by one apply the following functions, and display the resultant fourth object side by side.
* Make sure to have a destructor that deallocates the allocated memory.



**Hareem Aslam – Associate Lecturer**

Department of Computer Sciences



|  |
| --- |
| **Main.cpp** |
| nt ma n() {  nt degree1, degree2;  cout << "Enter the degree of the f rst polynom al: "; c n >> degree1;  flolynom al poly1(degree1);  cout << "Enter coeff c ents for the f rst polynom al\n\t (from h ghest degree to lowest): \n";  poly1. nput(degree1);  cout << "Enter the degree of the second polynom al: "; c n >> degree2;  flolynom al poly2(degree2);  cout << "Enter coeff c ents for the second polynom al\n (from h ghest degree to lowest): \n\n";  poly2. nput(degree2);  cout << "floly 1 : "; poly1.d splay(); cout << endl;  cout << "floly 2 : "; poly2.d splay(); cout << endl;  cout << "Result of Add t on (floly1 + floly2):" << endl; (poly1 + poly2).d splay();  cout << "\nResult of Subtract on (floly1 - floly2):" << endl; (poly1-poly2).d splay();  return 0;  } |



**Hareem Aslam – Associate Lecturer**

Department of Computer Sciences

|  |
| --- |
| **Output:** |
|  |



**Hareem Aslam – Associate Lecturer**

Department of Computer Sciences



# Lab Manual 08





**Hareem Aslam – Associate Lecturer**

Department of Computer Sciences



|  |  |
| --- | --- |
| **Lab 08** | |
| **Topic** | * Operator Overloading * **Graded Lab** * Midterm Revision Exercises |
| **Objective** | * Unary operators post and pre ++, --, +, - * Assignment = * Index [] * Stream Extraction >> and Insertion << |

###### Instructions:

* Indent your code properly.
* Use meaningful variable and function names. Use the camelCase notation.
* Use meaningful prompt lines/labels for all input/output.
* Strictly follow the Good Code Writing Style which is already shared by your theory teacher.
* In each file of your task either it is .cpp file or .h file Your name and roll no must be mention in the start of your file as comment. For example, //L1F22BSCS1234 – Ali Rehman
* Students must follow the "Code Writing Guide".
* You are not allowed to use string. You can use Cstring only as character arrays.
* You are not allowed to make array of objects in main().
* You are not allowed to use array of object.
* Perform all the tasks in separate source, header and main driver file.
* Paste your code in the given space below and submit your .cpp and .h file along with your .docx file.

Guidelines for graded lab:

1. **The use of mobile phones is strictly prohibited in this lab.**
2. **Movement from your assigned seat is not allowed.**
3. **It is mandatory to revise your concepts during the lab session.**
4. **Sharing notebooks or any study materials is prohibited in the lab.**
5. **There will be no extensions granted for task submission deadlines.**

###### Understanding of the question is part of the evaluation.

**Mobile phone is strictly prohibited during the lab either lab is graded or not.**

**Students are required to complete the following tasks in lab timings. Submission of each task in prescribed timing is mandatory.**



**Hareem Aslam – Associate Lecturer**

Department of Computer Sciences

**Graded Lab 3 - Task**

**Point 7, 8, 9 are not the part of graded task.**

Let’s create a class **Date** to store a date having the following attributes:

**int\* day int\* month int\* year**

**Time: 60 minutes**

Provide the following functions:

1. According to the rule of three:
   1. Copy Constructor
   2. operator =
   3. Destructor
2. Overloaded constructor with default values
3. Getter of each attribute (setters are NOT required)
4. Overload these operators
5. operator + //to add certain days to the date
6. operator - //to subtract days from the date
7. operator ==
8. operator >
9. Only using operator == and operator >, provide implementation of:
   1. operator != **ii.** operator < **iii.** operator <= **iv.** operator >=

**Important Instructions:**

Make sure:

1. to check that the time remains valid in all the functions:

**a.** 1 <= day <= 31

**b.** 0 <= month <= 12

**c.** 0 <= year <= n where n is a positive integer

1. Valid days are saved in each month:
2. February has 28 days
3. March, May, July, August, October, and December have 31 days
4. April, June, September, and November have 30 days
5. No need to check leap years (in which February has 29 days)
6. After addition or subtraction, make sure the answers are valid. For example:
7. **30-3-1985** + **6** days will return date as **5-4-1985**
8. **11-12-2021** + **22** days will return date as **2-1-2022**
9. **20-3-1985** - **155** days will return date as **16-10-1984**

##### Task 01: Point 7, 8, 9 the part of graded task.

Practice these tasks after the graded activity.



**Hareem Aslam – Associate Lecturer**

Department of Computer Sciences



# Lab Manual 09





**Hareem Aslam – Associate Lecturer**

Department of Computer Sciences



|  |  |
| --- | --- |
| **Lab 09** | |
| **Topic** | * IS-A or KIND-OF or Inheritance Relationship |
| **Objective** | * Introduction to inheritance * Protected members and different types of inheritance Working on inheritance with special focus on order of construction and destruction or base and child classes * Multi-Level Inheritance * Multiple Inheritance with the Dimond Problem |

###### Instructions:

* Indent your code properly.
* Use meaningful variable and function names. Use the camelCase notation.
* Use meaningful prompt lines/labels for all input/output.
* Strictly follow the Good Code Writing Style which is already shared by your theory teacher.
* In each file of your task either it is .cpp file or .h file Your name and roll no must be mention in the start of your file as comment. For example, //L1F22BSCS1234 – Ali Rehman
* Students must follow the "Code Writing Guide".
* You are not allowed to use string. You can use Cstring only as character arrays.
* You are not allowed to make array of objects in main().
* You are not allowed to use array of object.
* Perform all the tasks in separate source, header and main driver file.
* Paste your code in the given space below and submit your .cpp and .h file along with your .docx file.

###### Understanding of the question is part of the evaluation.

**Mobile phone is strictly prohibited during the lab either lab is graded or not.**

**Students are required to complete the following tasks in lab timings. Submission of each task in prescribed timing is mandatory.**



**Hareem Aslam – Associate Lecturer**

Department of Computer Sciences

Task 01: Single Class Date

**Tasks**

Write the definition of a “**Date”** class representing a calendar date using the following class declaration:

class Date

{

pr vate:

char dayName[10]; nt dayNumber;

char monthName[10]; nt monthNumber;

nt year;

bool leapYear; publ c:

Date();

Date(char \_dayName[], nt \_dayNumber, char \_monthName[], nt \_monthNumber, nt \_year, bool \_leapYear );

Date(const Date& d);

~Date();

vo d ds play();

};

* Implement all required constructors and destructors properly.
* The class should include a display function to print the date in the format DD/MM/YYYY.
* Sample Output of display function.
* **18/12/2023**
* **Day is Monday**
* **Month is December**
* **Year is not Leap Year**
* Create a main function to instantiate a Date object, display it, and test the Date class.

Task 02: Classes Day, Month, Year, and Date

Define and implement four classes:

* **Day,**
* **Month,**
* **Year,**
* **and Date.**
* **Use the following class declaration.**



**Hareem Aslam – Associate Lecturer**

Department of Computer Sciences



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| // Day Class class Day  {  pr vate:  char dayName[10]; nt dayNumber;  publ c:  Day(); Day(char  \_dayName[], nt  \_dayNumber); Day(const Day& d);  ~Day();  vo d ds play();  }; | | // Month Class class Month  {  pr vate:  char monthName[10];  nt monthNumber;  publ c:  Month(); Month(char  \_monthName[], nt  \_monthNumber);  Month(const Month&  d);  ~Month();  vo d ds play();  }; | // Year Class class Year  {  pr vate:  nt year;  bool leapYear;  publ c:  Year();  Year( nt \_year, bool \_leapYear);  Year(const Year&  d);  ~Year();  vo d ds play();  }; | |
|  | // redef ned Data Class class Date  {  pr vate:  Day day; Month month; Year year;  publ c:  Date();  Date(Day d, Month m, Year y); Date(const Date& d);  ~Date();  vo d ds play();  }; | | |  |

* The **Date** class include instances of the **Day, Month, Year** classes as its components.
* Each class should have a display function to print its respective information.
* Implement all required constructors and destructors properly.
* The class should include a display function to print the date in the format DD/MM/YYYY.
  + Sample Output of display function.
    - **18/12/2023**
    - **Day is Monday**
    - **Month is December**
    - **Year is not Leap Year**
  + Create a main function to instantiate a Date object, display it, and test the Date class.
* Ensure the output matches the output from TASK 1.



**Hareem Aslam – Associate Lecturer**

Department of Computer Sciences



##### Task 03: Classes Person and Company

Create two classes, **flerson** and **Company** use the following class definition.

|  |  |
| --- | --- |
| class flerson  {  publ c:  flerson();  flerson(char \_f rstName[], char  \_lastName[], char \_dateOfB rth[],  char \_address[], long  \_cellNo, char \_ema l[]); flerson(const flerson& s);  ~flerson()  vo d d splay(); pr vate:  char f rstName[10]; char lastName[10]; char dateOfB rth[10]; char address[100]; long cellNo;  char ema l[25];  }; | class Company  {  publ c:  Company();  Company(char \_name[], char  \_dateOfStart[],  char \_address[], long  \_phoneNo, char \_ema l[]); Company(const Company& s);  ~Company();  vo d d splay(); pr vate:  char name[10];  char dateOfStart[10]; char address[10]; long phoneNo;  char ema l[25];  }; |

* Create individual main functions to instantiate objects of both classes, display their information, and test the added functionalities.

##### Task 04: Classes PersonalData, Address, ContactData, Person, and Company

* The **PersonalData** class by using this definition.

class flersonalData

{

publ c:

flersonalData();

flersonalData(const char\* \_f rstName, const char\* \_lastName, const Date& \_dateOfB rth);

flersonalData(const flersonalData& s);

~flersonalData() vo d d splay();

pr vate:

char f rstName[10]; char lastName[10]; Date dob;

};



**Hareem Aslam – Associate Lecturer**

Department of Computer Sciences



* The **Address** class by using this definition.

class Address { pr vate:

nt streetNumber; char streetName[50]; char c ty[50];

publ c:

Address();

Address( nt sNumber, const char\* sName, const char\* c); Address(const Address& other);

~Address();

vo d d splay() const;

};

* The **ContactData** class by using this definition.

class Contact

{

publ c:

Contact();

Contact(const char\* \_name[],

Const Address& \_address, const long \_phoneNo, const char\*

\_ema l);

Contact(const Contact& s);

~Contact();

vo d d splay(); pr vate:

char name[10]; Address address; long phoneNo; char ema l[25];

};

* Each class should have a display function to print its information.
* Create individual main functions for **PersonalData**, **Address**, and **ContactData** to instantiate objects, display their information, and test the relationships between them.
* Now all the classes are implemented and tested individually. Let’s implement the classes

**Person** and **Company** using the classes **PersonalData** and **Contact**.

|  |  |
| --- | --- |
| class flerson  {  publ c:  flerson();  flerson(const flersonalData&, const Contact&);  flerson(const flerson& s);  ~flerson()  vo d d splay();  pr vate:  flersonalData data; Contact contact;  }; | class Company  {  publ c:  Company();  Company(const flersonalData&, const Contact&);  Company(const Company& s);  ~Company();  vo d d splay(); pr vate:  flersonalData data; Contact contact;  }; |



**Hareem Aslam – Associate Lecturer**

Department of Computer Sciences

* Finally, implement main functions for **Person** and **Company** classes that use the

**PersonalData**, **Address**, and **ContactData** classes, and test their functionalities.

* Ensure that the outputs match the outputs from TASK 3.



**Hareem Aslam – Associate Lecturer**

Department of Computer Sciences



# Lab Manual 10





**Hareem Aslam – Associate Lecturer**

Department of Computer Sciences



|  |  |
| --- | --- |
| **Lab 10** | |
| **Topic** | * Composition or HAS-A relationship |
| **Objective** | * Introduction to the concept of composition |

###### Instructions:

* + Indent your code properly.
  + Use meaningful variable and function names. Use the camelCase notation.
  + Use meaningful prompt lines/labels for all input/output.
  + Strictly follow the Good Code Writing Style which is already shared by your theory teacher.
  + In each file of your task either it is .cpp file or .h file Your name and roll no must be mention in the start of your file as comment. For example, //L1F22BSCS1234 – Ali Rehman
  + Students must follow the "Code Writing Guide".
  + You are not allowed to use string. You can use Cstring only as character arrays.
  + You are not allowed to make array of objects in main().
  + You are not allowed to use array of object.
  + Perform all the tasks in separate source, header and main driver file.
  + Paste your code in the given space below and submit your .cpp and .h file along with your .docx file.

###### Understanding of the question is part of the evaluation.

**Mobile phone is strictly prohibited during the lab either lab is graded or not.**

**Students are required to complete the following tasks in lab timings. Submission of each task in prescribed timing is mandatory.**



**Hareem Aslam – Associate Lecturer**

Department of Computer Sciences

Task: Student Management System

**Tasks**

In UCP, we have thousands of students who are currently enrolled in any Course. One Course has exactly one *Instructor* associated with it. We need to implement this scenario to store information of *Students*, *Instructors*, and *Courses*.

1. Define a class **Student** having following private data members:

**i.) name (pointer to character array)**

**ii.) registrationNo (pointer to character array) iii.) city (pointer to character array)**

1. Write parameterized constructor with default parameters for student class.
2. Write destructor with no memory leakage.
3. Write copy constructor and assignment operator.
4. Write setter and getter for each data member of student class.
5. Write a display function.
6. Define a class **Instructor** having following private data members:

**i.) name (pointer to character array)**

**ii.) education (pointer to character array)**

1. Write parameterized constructor with default parameters for Instructor class.
2. Write destructor with no memory leakage.
3. Write copy constructor and assignment operator.
4. Write a display function.
5. Write setter and getter for each data member of instructor class.
6. Define a class **Course** having following private data members:

**i.) courseName (pointer to character array)**

**ii.) instructor (??? a course offered has exact one Instructor) iii.) student (??? a Course offered has many Students enrolled)**

1. Write default and parameterized constructor for Course class.
2. Write setter and getter for each data member of Course class.
3. Write destructor with no memory leakage.
4. Write copy constructor and assignment operator.
5. Write a display function.
6. Write a function that return total student count enrolled in class.



**Hareem Aslam – Associate Lecturer**

Department of Computer Sciences

1. Write a display() function in Course class that displays complete Course information like following:

Course Name: OOP LAB

Instructor Name: AASMA ABDUL WAHEED Total Students Enrolled: 2

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Registration No |  | Student Name |
| 1. | L1F1YBSCSWXYZ | Ali |  |
| 2. | L1F1YBSCSWXYZ | Ammar |  |

1. A Course cannot have more than 30 students. Write logic to implement this.
2. Now do following operation in main function:

Create 3 student objects with following data:

|  |  |  |
| --- | --- | --- |
| name: Ali | registrationNo: L1F1YBSCS1ABC | city: Lahore |
| name: Ahmad | registrationNo: L1F1YBSCS2ABC | city: Islamabad |
| name: Ammar | registrationNo: L1F1YBSCS3ABC | city: Lahore |

* 1. Create an object oopLabSecC4 of Course class with default constructor.
  2. Now set course name to ‘OOPFall2023SecC4Lab’.
  3. Display total number of students in oopLabSecC4.

**i.)** Now add all 3 newly created students to oopLabSecC4.

**ii.)** Display total number of students in oopLabSecC4.

**iii.)** Create an Instructor inst1 having name ‘AASMA ABDUL WAHEED’ and education ‘MPHIL’

**iv.)** Add inst1 to oopLabSecC4.

**v.)** Now display oopLabSecC4 all data on screen.

**vi.)** Now create another object oopLabSecC2 of Course class

**vii.)** Now add Ali and Ammar in oopLabSecC2.

**viii.)** Create a new Instructor inst2 having name ‘USAMA PERVAIZ’ and education ‘MPHIL’

**ix.)** Now Add inst2 to oopLabSecC2.

**x.)** Now delete oopLabSecC4.

**xi.)** Display oopLabSecC2 data on screen and see Ali and Ammar are still in oopLabSecC2 or deleted from here too due to deletion of oopLabSecC4.

**xii.)** Now call name of inst1 on screen. Verify that whether inst1 is deleted with oopLabSecC4 or not?



**Hareem Aslam – Associate Lecturer**

Department of Computer Sciences



# Lab Manual 11





**Hareem Aslam – Associate Lecturer**

Department of Computer Sciences



|  |  |
| --- | --- |
| **Lab 11** | |
| **Topic** | * **Dynamic memory allocation** * **Operator overloading** * **File I/O operations** * **Encapsulation** * **Inheritance** * **Composition** * **Multi-level inheritance** |
| **Objective** | * Gain proficiency in operator overloading, including file I/O, assignment, and equality operations in C++. * Master dynamic memory management using constructors, destructors, and copy constructors. * Understand and apply composition to design complex, modular systems using reusable classes. * Develop skills in testing and validating object-oriented designs through comprehensive class testing. |

###### Instructions:

* Indent your code properly.
* Use meaningful variable and function names. Use the camelCase notation.
* Use meaningful prompt lines/labels for all input/output.
* Strictly follow the Good Code Writing Style which is already shared by your theory teacher.
* In each file of your task either it is .cpp file or .h file Your name and roll no must be mention in the start of your file as comment. For example, //L1F22BSCS1234 – Ali Rehman
* Students must follow the "Code Writing Guide".
* You are not allowed to use string. You can use Cstring only as character arrays.
* You are not allowed to make array of objects in main().
* You are not allowed to use array of object.
* Perform all the tasks in separate source, header and main driver file.
* Paste your code in the given space below and submit your .cpp and .h file along with your .docx file.

###### Understanding of the question is part of the evaluation.

**Mobile phone is strictly prohibited during the lab either lab is graded or not.**



**Hareem Aslam – Associate Lecturer**

Department of Computer Sciences

**Students are required to complete the following tasks in lab timings. Submission of each task in prescribed timing is mandatory.**

Task 01: Price

**Tasks**

Define a class **Price** to represent a single variable of type float for storing prices. Write the following for the class:

* Implement default, parameterized, and copy constructors, destructor as well.
* Implement getter and setter methods for accessing and modifying the price.
* Overload the assignment (=) and equality (==) operators.
* Implement global input (>>) and output (<<) operators and it must be able to read and write to a file.

Note: You are not allowed to implement these << and >> operator as friend function.

* Test your class using **main()** function*.*

Task 02: Fruit

Define a class **Fruit** with attributes for the name of the fruit (a string) and the price (an object of class Price).

Write the following for the class:

* Implement default, parameterized, and copy constructors, destructor as well.
* Implement getter and setter methods for accessing and modifying the price.
* Overload the assignment (=) and equality (==) operators.

Implement global input (>>) and output (<<) operators and it must be able to read and write to a file.

Note: You are not allowed to implement these << and >> operator as friend function.

* Test your class using **main()** function*.*

##### Task 03: FruitBasket

Define a class **FruitBasket** to represent a collection of fruits. Include attributes for the quantity (noOfFruits) and a dynamic array of type Fruit.

Write the following for the class:

* Implement default, parameterized, and copy constructors, destructor as well.
* Implement getter and setter methods for accessing and modifying the price.
* Overload the assignment (=) and equality (==) operators.

Implement global input (>>) and output (<<) operators and it must be able to read and write to a file.

Note: You are not allowed to implement these << and >> operator as friend function.

* Test your class using **main()** function*.*



**Hareem Aslam – Associate Lecturer**

Department of Computer Sciences

##### Task 04: FruitShop

Create a class **FruitShop** to represent a shop containing multiple fruit baskets. Include

attributes for the number of baskets (noOfBaskets) and a dynamic array of type FruitBasket. Write the following for the class:

* Implement default, parameterized, and copy constructors, destructor as well.
* Implement getter and setter methods for accessing and modifying the price.
* Overload the assignment (=) and equality (==) operators.

Implement global input (>>) and output (<<) operators and it must be able to read and write to a file.

Note: You are not allowed to implement these << and >> operator as friend function.

* Test your class using **main()** function*.*

##### Task 05: Multi-Level Fruit Management System

Develop a class **FruitBazar** to represent a marketplace with multiple fruit shops. Include attributes for the number of shops (noOfShops) and a dynamic array of type FruitShop. Write the following for the class:

* Implement default, parameterized, and copy constructors, destructor as well.
* Implement getter and setter methods for accessing and modifying the price.
* Overload the assignment (=) and equality (==) operators.

Implement global input (>>) and output (<<) operators and it must be able to read and write to a file.

Note: You are not allowed to implement these << and >> operator as friend function.

* Test your class using **main()** function*.*



**Hareem Aslam – Associate Lecturer**

Department of Computer Sciences



# Lab Manual 12





**Hareem Aslam – Associate Lecturer**

Department of Computer Sciences



|  |  |
| --- | --- |
| **Lab 12** | |
| **Topic** | * **Inheritance** * **Polymorphism** * **Abstraction** * **Composition** |
| **Objective** | * Understanding Inheritance * Utilizing Polymorphism * Abstraction and use of virtual and pure virtual function * Understanding Encapsulation and composition * Error Handling and Robustness |

###### Instructions:

* + Indent your code properly.
  + Use meaningful variable and function names. Use the camelCase notation.
  + Use meaningful prompt lines/labels for all input/output.
  + Strictly follow the Good Code Writing Style which is already shared by your theory teacher.
  + In each file of your task either it is .cpp file or .h file Your name and roll no must be mention in the start of your file as comment. For example, //L1F22BSCS1234 – Ali Rehman
  + Students must follow the "Code Writing Guide".
  + You are not allowed to use string. You can use Cstring only as character arrays.
  + You are not allowed to make array of objects in main().
  + You are not allowed to use array of object.
  + Perform all the tasks in separate source, header and main driver file.
  + Paste your code in the given space below and submit your .cpp and .h file along with your .docx file.

###### Understanding of the question is part of the evaluation.

**Mobile phone is strictly prohibited during the lab either lab is graded or not.**

**Students are required to complete the following tasks in lab timings. Submission of each task in prescribed timing is mandatory.**



**Hareem Aslam – Associate Lecturer**

Department of Computer Sciences

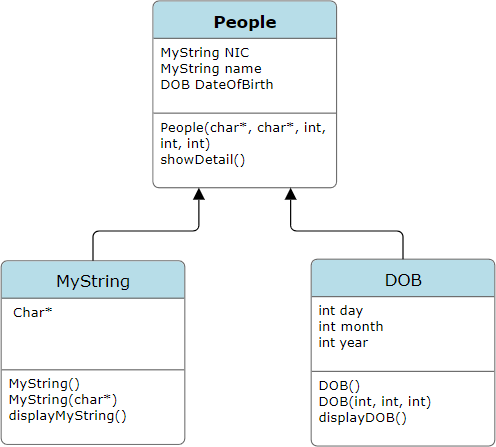
**Graded Lab 4 - Task**

**Time: 30 Minutes**

Create a class People having attributes name, NIC, Address and DOB. DOB object belongs to a class DOB, having day-month-year as its attributes. Mystr ng class has char\* as its attribute.

**Identify the relationship among all** and implement all the required functions in all classes to make them functional.

In main create object of DOB and People to show the full detail of People object.



nt ma n() {

DOB d1(25,02,2003;

d1.d splayDOB(); // Date should be d splayed as **“25th February, 2003”**

fleople p1("35402-112131-8", "Dan yal Farrukh", 4, 7, 2003);

p1.showDeta l(); // Date should be d splayed as **“4th August, 2003”**

system("flause"); return 0;

}



**Hareem Aslam – Associate Lecturer**

Department of Computer Sciences

Task: Transportation Management System

In a transportation management system, the need arises to efficiently model and manage different types of vehicles within a fleet. This scenario involves the development of classes to represent Cars, Trucks, and Buses, each sharing common attributes like "make," "model," and "year."

Classes Structure:

Class: **Vehicle Common Attributes:**

* **make (c-string):** Represents the manufacturer or brand of the vehicle.
* **model (c-string):** Specifies the model’s name of the vehicle.
* **variant (int):** Represents the manufacturing year of the vehicle.

**Default and Parameterized Constructor:**

* Initializes make, model, and variant based on provided parameters.

**Getters and Setters:**

* Allows access and modification of the make, model, and variant attributes.

**Operators Overloaded:**

 **==**: **Compares** two vehicles for equality based on their attributes.

 **=**: **Assigns** one vehicle's attributes to another.

 **++**: **Increments** the manufacturing year of the vehicle.

* + **<< and >>:** Outputs/inputs vehicle attributes to/from a **stream**. **Virtual Functions (to be overridden by child classes):**
  + **calculateFuelEfficiency():** Calculates and returns a default fuel efficiency value. Class: **Car (Derived from Vehicle)**

**Additional Attributes:**

* **numDoors (int):** Represents the number of doors in the car.
* **isConvertible (bool):** Indicates whether the car is convertible or not.

**Constructor:**

* Calls the parent class constructor and initializes car-specific attributes.

**Getters and Setters:**

* Provides access and modification for car-specific attributes.

**Operators Overloaded:**

* Inherits and reuses the operators overloaded in the parent class.

**Overridden Virtual Functions:**

* **calculateFuelEfficiency():** Calculates and returns fuel efficiency specific to a car.



**Hareem Aslam – Associate Lecturer**

Department of Computer Sciences

Class: **Truck (Derived from Vehicle) Additional Attributes:**

* **payloadCapacity (double):** Specifies the payload capacity of the truck in tons.
* **hasTrailer (bool):** Indicates whether the truck is equipped with a trailer or not.

**Constructor:**

* Calls the parent class constructor and initializes truck-specific attributes.

**Getters and Setters:**

* Provides access and modification for truck-specific attributes.

**Operators Overloaded:**

* Inherits and reuses the operators overloaded in the parent class.

**Overridden Virtual Functions:**

* **calculateFuelEfficiency():** Calculates and returns fuel efficiency specific to a truck. Class: **Bus (Derived from Vehicle)**

**Additional Attributes:**

* **seatingCapacity (int):** Represents the maximum number of passengers the bus can accommodate.
* **hasDoubleDecker (bool):** Indicates whether the bus has a double-decker design or not.

**Constructor:**

* Calls the parent class constructor and initializes bus-specific attributes.

**Getters and Setters:**

* Provides access and modification for bus-specific attributes.

**Operators Overloaded:**

* Inherits and reuses the operators overloaded in the parent class.

**Overridden Virtual Functions:**

* **calculateFuelEfficiency():** Calculates and returns fuel efficiency specific to a bus.

Write main to test above classes and relationships. After that in main function Create array of pointers of parent type class and then user polymorphism to access functions and operators of children classes.

The use of inheritance allows for code reuse, and the overridden virtual functions enable polymorphic behaviour, allowing each derived class to provide its specific implementation of fuel efficiency calculation.

Make sure to implement the functions and methods as per the described logic and test the classes to ensure they function correctly. Additionally, consider adding proper error handling to enhance the clarity and robustness of the code.



**Hareem Aslam – Associate Lecturer**

Department of Computer Sciences



# Lab Manual 13





**Hareem Aslam – Associate Lecturer**

Department of Computer Sciences



|  |  |
| --- | --- |
| **Lab 13** | |
| **Topic** | * **Inheritance** * **Polymorphism** * **Abstraction** * **Composition** |
| **Objective** | * Designing and Managing a Simple System * Handling real-world scenarios with C++ OOP principles. * Practicing encapsulation, composition, and proper use of constructors. |

###### Instructions:

* Indent your code properly.
* Use meaningful variable and function names. Use the camelCase notation.
* Use meaningful prompt lines/labels for all input/output.
* Strictly follow the Good Code Writing Style which is already shared by your theory teacher.
* In each file of your task either it is .cpp file or .h file Your name and roll no must be mention in the start of your file as comment. For example, //L1F22BSCS1234 – Ali Rehman
* Students must follow the "Code Writing Guide".
* You are not allowed to use string. You can use Cstring only as character arrays.
* You are not allowed to make array of objects in main().
* You are not allowed to use array of object.
* Perform all the tasks in separate source, header and main driver file.
* Paste your code in the given space below and submit your .cpp and .h file along with your .docx file.

###### Understanding of the question is part of the evaluation.

**Mobile phone is strictly prohibited during the lab either lab is graded or not.**

**Students are required to complete the following tasks in lab timings. Submission of each task in prescribed timing is mandatory.**



**Hareem Aslam – Associate Lecturer**

Department of Computer Sciences

Task: University Bookstore

A public sector university has bookstores throughout the country from where students can buy study materials like handouts, lecture DVDs, books, etc. Each bookstore maintains the stock data of study material. This data is helpful for the university to maintain information about the quantity, price, and other data of each item available at different bookstores.

Considering the above scenario, the system should be designed using C++ and the following classes:

**StockItem**, **Handouts**, and **LectureDVD**.

1. **Inheritance: Base Class - StockItem**

Create an abstract base class, **StockItem**, with common attributes and some member functions such as

|  |  |
| --- | --- |
| **Data Members** | **Member Functions** |
| * Name of Item * Quantity of Item * Price of Item * Course Code * Course Name | * Parameterized constructor with default arguments * Copy constructor * Setter functions to set values of data members of this class. * Getter functions to get the values of data members of this class. * A function to set data of a stock item * A function to display data of a stock item |

1. **Inheritance: Derived Class - Handouts**

Create a derived class, **Handouts**, from the base class **StockItem**. Include additional attributes and some member functions such as

|  |  |
| --- | --- |
| **Data Members** | **Member Functions** |
| * No Pages * Author | * Parameterized constructor with default arguments * Copy constructor * Assignment operator * A function to set data of handouts * A function to display data of handouts. |

1. **Inheritance: Derived Class - LectureDVD**

Create another derived class, **LectureDVD**, from the base class **StockItem**. Include additional attributes and some member functions such as

|  |  |
| --- | --- |
| **Data Members** | **Member Functions** |
| * Total minutes * Total Lectures | * Parameterized constructor with default arguments * A function to set data of DVD. * A function to display data of DVD. |



**Hareem Aslam – Associate Lecturer**

Department of Computer Sciences

1. **Composition: University Bookstore**

Create a class, **UniversityBookstore**, that represents the overall bookstore system. Utilize composition to include instances of **StockItem**, **Handouts**, and **LectureDVD** within the **UniversityBookstore** class. Implement functions to manage and display the study materials available in the bookstore.

|  |  |
| --- | --- |
| **Data Members** | **Member Functions** |
| * StockItem * Handouts * LectureDvd | * Function to add a stock item to the bookstore * Function to display all study materials in the bookstore |

Now, in the main program, you can create instances of **Handouts** and **LectureDVD**, add them to the **UniversityBookstore** using the composition principle, and display the details using polymorphism through the **StockItem** base class.

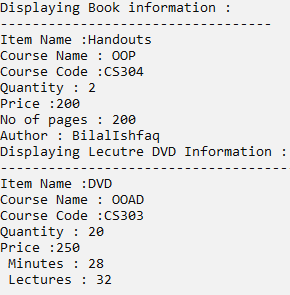
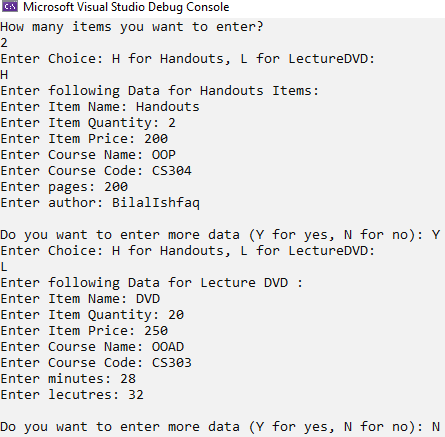
**Solution Guidelines:**

You have to use the concept of Polymorphism to generate the sample output given and submit the screenshot of your output.



**Hareem Aslam – Associate Lecturer**

Department of Computer Sciences



**Hareem Aslam – Associate Lecturer**

Department of Computer Sciences



# Lab Manual 14





**Hareem Aslam – Associate Lecturer**

Department of Computer Sciences



|  |  |
| --- | --- |
| **Lab 14** | |
| **Topic** | * Revision * **Graded Activity** |
| **Objective** | * Hands-on Practice of all concepts of Object-Oriented Programming |

###### Instructions:

* + Indent your code properly.
  + Use meaningful variable and function names. Use the camelCase notation.
  + Use meaningful prompt lines/labels for all input/output.
  + Strictly follow the Good Code Writing Style which is already shared by your theory teacher.
  + In each file of your task either it is .cpp file or .h file Your name and roll no must be mention in the start of your file as comment. For example, //L1F22BSCS1234 – Ali Rehman
  + Students must follow the "Code Writing Guide".
  + You are not allowed to use string. You can use Cstring only as character arrays.
  + You are not allowed to make array of objects in main().
  + You are not allowed to use array of object.
  + Perform all the tasks in separate source, header and main driver file.
  + Paste your code in the given space below and submit your .cpp and .h file along with your .docx file.

###### Understanding of the question is part of the evaluation.

**Mobile phone is strictly prohibited during the lab either lab is graded or not.**

**Students are required to complete the following tasks in lab timings. Submission of each task in prescribed timing is mandatory.**



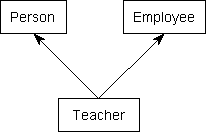
**Hareem Aslam – Associate Lecturer**

Department of Computer Sciences

**Graded Lab 5 - Task**

Provide the C++ implementation of a class hierarchy of the following.

**Time: 40 minutes**



The Person class has name and age as its attributes. It has an overloaded constructor to initialize the values and appropriate accessor and mutator methods. The Employee class has the name of the employer and wage as its attributes with an overloaded constructor and appropriate accessor and mutator methods.

The Teacher class is inherited from the Person and Employee class with an attribute of Pay Scale of type integer. It has an overloaded constructor, appropriate accessor, mutator methods, and a display function to print the Name, Age, Name of Employer, Wage, and Pay Scale of the Teacher.



**Hareem Aslam – Associate Lecturer**

Department of Computer Sciences