CH. DEVI LAL STATE INSTITUTE OF ENGINEERING & TECHNOLOGY
PANNIWALA MOTA, SIRSA, HARYANA-125077

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

LAB MANUAL

(For the Academic Year 2022 – 2023)

Programme (UG/PG) : **UG (B.Tech-CSE)**

Semester : **III**

Course Code : **PCC-CSE-202T**

Course Title : **OBJECT ORIENTED PROGRAMMING LAB**

Prepared By:

MS.BHARTI SETHI

(Assistant Professor), Department of Computer Science and Engineering)

INDEX OF THE CONTENTS:

1.Introduction to the Lab

In this lab, programming is done on turbo C/Borland C.
Code::Blocks is a free C++ compiler which meets the most demanding needs of its users. It is designedto be very extensible and fully configurable.
It is a cross-platform IDE that supports compiling and running multiple programming languages. It is available for download from:

http://www.codeblocks.org/

Compiling C++ Programs:

C++ source files conventionally use one of the suffixes `.C', `.cc', `.cpp', `.CPP', `.c++', `.cp', or `.cpp'; C++ header files often use `.h' or `.H'; and preprocessed C++ files use the suffix `.ii'. GCC recognizes files with these names and compiles them as C++ programs even if you call the compiler the same way as for compiling C programs (usually with the name gcc).

However, the use of gcc does not add the C++ library. g++ is a program that calls GCC and treats `.c', `.h' and `.i' files as C++ source files instead of C source files unless -x is used, and automatically specifies linking against the C++ library. This program is also useful when precompiling a C header file with a `.h' extension for use in C++ compilations. On many systems, g++ is also installed with the name c++.

When you compile C++ programs, you may specify many of the same command-line options that you use for compiling programs in any language; or command-line options meaningful for Cand related languages; or options that are meaningful only for C++ programs.

2.Lab Objectives and Outcomes

Lab Objectives:

- Introduces the principles of data abstraction, inheritance and polymorphism;
- Introduces the principles of virtual functions and polymorphism
- Introduces handling formatted I/O and unformatted I/O
- Introduces exception handling
- To strengthen problem solving ability by using the characteristics of an object-orientedapproach.
- To design applications using object oriented features.
- To handle Exceptions in programs.
- To teach the students to implement object oriented concepts.

Lab Outcomes:

Upon successful completion of this Lab, the students will be able to:

- Understand the features of C++ supporting object oriented programming.
- Understand the relative merits of C++ as an object oriented programming language
- Understand how to produce object-oriented software using C++.
- Ability to develop applications for a wide range of problems using object oriented programming techniques.

3. List of Experiments:

1. Write a program for multiplication of two matrices using OOP.
2. Write a program to perform addition of two complex numbers using constructor overloading. Here the first constructor which takes no argument is used to create objects which are not initialized, second which takes one argument is used to initialize real and imag parts to equal values and third which takes two Argument is used to initialized real and imag to two different values.
3. Write a program to find the greater of two given numbers in two different classes using friend function.
4. Implement a class string containing the following functions:
   - Overload + operator to carry out the concatenation of strings.
   - Overload = operator to carry out string copy.
   - Overload <= operator to carry out the comparison of strings.
   - Function to display the length of a string.
   - Function to lower ( ) to convert upper case letters to lower case.
   - Function to upper ( ) to convert lower case letters to upper case.
5. Write a program to demonstrate the use of special functions, constructor and destructor in the class template. The program is used to find the bigger of two entered numbers.
6. Write a program to perform the deletion of white spaces such as horizontal tab, vertical tab, space, line feed, new line and carriage return from a text file and store the contents of the file without whitespaces on another file.
7. Write a program to read the class object of student info such as name, age ,sex ,height and weight from the keyboard and to store them on a specified file using read() and write() functions. Again thesame file is opened for reading and displaying the contents of the file on the screen.
8. Create a base class basic info with data members name ,roll no, sex and two member functions getdata and display. Derive a class physical fit from basic info which has data members height and weight and member functions getdata and display. Display all the information using object of derived class.
9. Design three classes STUDENT, EXAM and RESULT. The STUDENT class has data members such as rollno, name. Create a class EXAM by inheriting the STUDENT class. The EXAM class adds data members representing the marks scored in six subjects. Derive the RESULT from the EXAM class and has its own data members such as total marks. Write a program to model this relationship.
10. Write a function power to raise a number m to power n. The function takes a double value for m and int value for Use default value for n to make the function to calculate squares when this argument is omitted.
11. Create a class TIME with members hours, minutes, seconds. Take input, add two time objects passing objects to function and display result.
12. Create a class Student which has data members as name, branch, roll no, age, sex, marks in five subjects. Display the name of the student and his percentage that has more than 70%.Use array of objects.

Reference Books:

1. **The C++ Programming Language, 3rd Edition, B. Stroutstrup, Pearson Education**
2. **OOP in C++, 3rd Edition, T. Gaddis, J. Walters and G. Muganda, Wiley Dream Tech Press.**
3. **Object Oriented Programming in C++, 3rd Edition, R. Lafore, Galigotia Publications Pvt Ltd**

4. INSTRUCTIONS FOR STUDENTS

These are the instructions for the students attending the lab:

- Before entering the lab , student should carry the following things (MANDATORY)

1. Identity card issued by the college.
2. Class notes
3. Lab Manual
4. Practical file

- Student must sign in and sign out in the register provided when attending the lab session without fail.
- Come to the laboratory in time. Students, who are late more than 15 min., will not be allowed to attend the lab.
- Students need to maintain 100% attendance in lab if not a strict action will be taken.
- All bags must be left at the indicated place.
- Refer to the lab staff if you need any help in using the lab.
- Workspace must be kept clean and tidy after experiment is completed.
- Read the Manual carefully before coming to the laboratory and be sure about what you are supposed to do.
- Do the experiments as per the instructions given in the manual.
- Copy all the programs to observation which are taught in class before attending the lab session.

Lab records need to be submitted on or before the date of submission.

5.Algorithms

List of programs to be implemented (for practice)

Exp. No.	Title
1.A.	STATIC MEMBER VARIABLES
1.B.	DEFAULT ARGUMENTS
2.A	CONSTRUCTORS AND DESTRUCTORS
2.B.	OPERATOR OVERLOADING
3.A.	SINGLE INHERTANCE
3.B.	HYBRID INHERITANCE
4.A.	VIRTUAL FUNCTIONS
4.B.	DYNAMIC POLYMORPHISM
5.A.	EXCEPTION HANDLING IN STACKS
5.B.	EXCEPTION HANDLING IN QUEUE

Algorithms to be implemented:

Exp. No: 1.A

STATIC VARIABLE

i. OBJECTIVE:

To write a C++ program to illustrate the static variable functionality using sum of a Fibonacci series as an example.

ii. ALGORITHM:

STEP 1: Create a class with a static variable

STEP 2: This class will contain a method which will increase the value of this static variable.<and this method is a recursive function>.

STEP 3: Use scope resolution operator and increase the scope of the variable.

STEP 4: Create an object of this class in main.
STEP 5: Call this method

Exp. No: 1.B

DEFAULT ARGUMENTS

i. OBJECTIVE:

To write a C++ program to demonstrate default arguments with a simple example.

ii. ALGORITHM:

STEP 1: Include the Header file.

STEP 2: Create a function which takes in three arguments of which one is a preset default argument.

STEP 3: Now call this function and while calling this pass in 2 arguments rather than 3.
STEP 4: The default argument is already set.
STEP 5: The function will execute without showing any error.

iii. SAMPLE INPUTS & OUTPUTS:

Final value=1005.68

Final value=3712.93

RESULT :

The program was successfully compiled & executed, and the output was verified.

Exp. No: 2.A

CONSTRUCTOR AND DESTRUCTOR (Dynamic memory allocation)

i. OBJECTIVE:

To write a C++ program to demonstrate the use of constructors and destructors .

ii. ALGORITHM:

Step 1: Create a class
Step 2: Create a constructor for this class which will set the value of its member variables.
Step 3: Create the other required methods for the given class
Step 4: Define the destructor.
Step 5: Create an object of this class
Step 6: The constructor will be called upon creation and when the program is ending , destructor will be .

iii. SAMPLE INPUTS & OUTPUTS:

Enter the size of matrix needed: 2 2

By help of constructor:

Enter the matrix elements:

1 2

3 4

Matrix Elements

1 2

3 4

By help of copy constructor:

Matrix Elements

1 2

3 4

Enter the matrix elements:

1 2

3 4

Matrix Elements

2 2

3 4

Exp. No: 2.B

OPERATOR OVERLOADING

i. OBJECTIVE:

To write a C++ program to illustrate the operator overloading concept using Matrix addition as an example.

ii. ALGORITHM:

Step 1: Create a class 'matrix' with constructor that sets the value of its member variables.
Step 2: This class also overloads the + and = operator with proper operator overloaded functions.
Step 3: Define the operators.
Step 4: Create the object of this class and call these overloaded functions.

iii. SAMPLE INPUTS & OUTPUTS:

Enter the matrix size of A:

Enter rows & columns 2 2

Enter the matrix A:

1 1 1 1

Enter the matrix size of B:

Enter rows & columns 2 2

Enter the matrix B:

1 1 1 1

Matrix A: 1 1 1 1

Matrix B: 1 1 1 1

Resultant Matrix:

2 2

2 2

Exp. No: 3.A

SINGLE INHERITANCE USING BANKING SYSTEM

i. OBJECTIVE:

To write a C++ program to illustrate the single inheritance using banking system as an example.

ii. ALGORITHM:

STEP 1: Include the Header file.
STEP 2: Create a forward declaration of a class.
STEP 3: Create a new class that has certain member functions and methods.
STEP 4: Create the class earlier defined in step 2 and Create it inherit from the class in step 2 using the resolution: operator
STEP 5: Now create an object of the sun class and Create it call some methods you defined in step 2.

iii. SAMPLE INPUTS & OUTPUTS:

Enter customer name: Ajay

Enter account No: 123

Enter account type: Savings

Choose your choice

1) Deposit

2) withdraw

3) display Balance

4) Display with full details

5) Exit

Enter your Choice: 1

Enter the amount to Deposit 500

Enter your Choice: 1

3

Balance:- 510

Enter your Choice: 5

Exp. No: 3.B

HYBRID INHERITANCE USING STUDENT DATABASE

i. OBJECTIVE:

To write a C++ program to illustrate hybrid inheritance concept using student database creation as an example.

ii. ALGORITHM:

Step 1: Create a class with some methods and variables.
Step 2:Create a new class that extends this class.
Step 3: Create a new class with some new methods.
Step 4: Create a new class again which inherits from classes defined in step 2 and step 3.
Step 5.Create an object of class defined in step 4 and create it call methods defined in step 1 and step 3.

Exp. No: 4.A

VIRTUAL FUNCTION

i. OBJECTIVE:

To write a C++ program to illustrate virtual function implementation.

ii. ALGORITHM:

Step 1:Create a class base with a method marked as virtual.
Step 2:Create a subclass of base called derived and redefine the method defined in step 1.
Step 3:Create an object pointer of type base and Create it point to an object of type base first and call the methods you had marked as virtual using -> operator.
Step 4: repeat the same steps but by making the pointer point to an object of derived type now.

RESULT :

The program was successfully compiled & executed, and the output was verified.

Exp. No: 4.B

DYNAMIC POLYMORPHISM

i. OBJECTIVE:

To write a C++ program to illustrate dynamic polymorphism using different shapes as an example.

ii. ALGORITHM:

Step 1: Make a class area which has a method.
Step 2: Make subclasses to this which have constructors that will take in the value for the requiredfields and will compute the area and put it in one of the member variables.
Step 3: call the method defined in step 1, which is overridden in all the subclasses to show theirrespective areas.

iii. SAMPLE INPUTS & OUTPUTS:

Enter radius of circle: 2
Enter length and breadth of rectangle: 4 6
Enter side of square: 2
Enter height and base of circle: 3 2
Enter parallel sides' length of trapezium: 3, 3

Area of circle=12.56

Area of rectangle=24

Area of square=4

Area of triangle=3

Area of Trapezium=1.5

Exp. No: 5.A

EXCEPTION HANDLING IN STACKS

i. OBJECTIVE:

To write a C++ program to illustrate exception handling concept using stack operation as an example.

ii. ALGORITHM:

STEP 1: Include the Header file.
STEP 2: Create a class Stack which initialize size of the stack
STEP 3: Create function push to do insert operation which in turn throws an exception when it overflow.
STEP 4: Create function pop to do delete operation which in turn throws an exception when it underflow.
STEP 5: Display the operation on console using display function.