LAB MANUAL OF

OBJECT ORIENTED PROGRAMMING USING C++

ETCS-258

Maharaja Agrasen Institute of Technology, PSP area,
Sector – 22, Rohini, New Delhi – 110085
(Affiliated to Guru Gobind Singh Indraprastha University, New Delhi)

INDEX OF THE CONTENTS

1. INTRODUCTION TO THE LAB

In this lab programming is done on Red Hat Linux using gcc compiler.

Or

Code::Blocks isa free C++ compiler which meets the most demanding needs of its users. It is designed to 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/

The C++ Programming Environment in Linux

The best way to learn a programming language is to try writing programs and test them on a computer. To do this, we need several pieces of software:

- An editor with which to write and modify the C++ program components or source code,
- A compiler with which to convert the source code into machine instructions which can be executed by the computer directly,
- A linking program with which to link the compiled program components with each other and with a selection of routines from existing libraries of computer code, in order to form the complete machine-executable object program,
- A debugger to help diagnose problems, either in compiling programs in the first place, or if the object program runs but gives unintended results.

There are several editors available for Linux (and Unix systems in general). Two of the most popular editors are emacs and vi. For the compiler and linker, we will be using the GNU g++ compiler/linker, and for the debugger we will be using the GNU debugger gdb. For those that prefer an integrated development environment (IDE) that combines an editor, a compiler, a linking program and a debugger in a single programming environment (in a similar way to Microsoft Developer Studio under Windows NT), there are also IDEs available for Linux (e.g. V IDE, kdevelop etc.)

GNU Compiler Collection (GCC)

GCC stands for "GNU Compiler Collection". GCC is an integrated distribution of compilers for several major programming languages. These languages currently include C, C++, Objective-C, Objective-C++, Java, Fortran , and Ada.

The abbreviation *GCC* has multiple meanings in common use. The current official meaning is "GNU Compiler Collection", which refers generically to the complete suite of tools. The name historically stood for "GNU C Compiler", and this usage is still common when the emphasis is on compiling C programs. Finally, the name is also used when speaking of the *language-independent* component of GCC: code shared among the compilers for all supported languages.

The language-independent component of GCC includes the majority of the optimizers, as well as the "back ends" that generate machine code for various processors.

The part of a compiler that is specific to a particular language is called the "front end". In addition to the front ends that are integrated components of GCC, there are several other front ends that are maintained separately. These support languages such as Pascal, Mercury, and COBOL. To use these, they must be built together with GCC proper.

Most of the compilers for languages other than C have their own names. The C++ compiler is G++, the Ada compiler is GNAT, and so on. When we talk about compiling one of those languages, we might refer to that compiler by its own name, or as GCC. Either is correct.

Historically, compilers for many languages, including C++ and Fortran, have been implemented as "preprocessors" which emit another high level language such as C. None of the compilers included in GCC are implemented this way; they all generate machine code directly. This sort of preprocessor should not be confused with the *C preprocessor*, which is an integral feature of the C, C++, Objective-C and Objective-C++ languages.

GCC Command Options

When you invoke GCC, it normally does preprocessing, compilation, assembly and linking. The "overall options" allow you to stop this process at an intermediate stage. For example, the -c option says not to run the linker. Then the output consists of object files output by the assembler.

Other options are passed on to one stage of processing. Some options control the preprocessor and others the compiler itself. Yet other options control the assembler and linker; most of these are not documented here, since you rarely need to use any of them.

Most of the command line options that you can use with GCC are useful for C programs; when an option is only useful with another language (usually C++), the explanation says so explicitly. If the description for a particular option does not mention a source language, you can use that option with all supported languages.

The gcc program accepts options and file names as operands. Many options have multi-letter names; therefore multiple single-letter options may *not* be grouped: -dr is very different from `-d -r'.

You can mix options and other arguments. For the most part, the order you use doesn't matter. Order does matter when you use several options of the same kind; for example, if you specify -L more than once, the directories are searched in the order specified.

Many options have long names starting with `-f' or with `-W'—for example, -fmove-loop-invariants, -Wformat and so on. Most of these have both positive and negative forms; the negative form of -ffoo would be -fno-foo. This manual documents only one of these two forms, whichever one is not the default.

Compiling C++ Programs

C++ source files conventionally use one of the suffixes `.C', `.cc', `.cpp', `.CPP', `.c++', `.cp', or `.cxx'; C++ header files often use `.hh' 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 C and related languages; or options that are meaningful only for C++ programs.

The Vi Editor

All Linux configuration files are written in plain English, easy to read and to adapt. You use a text-editor to write or make changes to such files. The two most popular, powerful, and unfortunately "difficult" text editors, both of which are found in every Linux are Vi and Emacs.

Most GUI-based editors, such as Kedit, are easier to manage. But don't make the mistake of thinking that a GUI-based editor is all you need. There are situations that crop up with Linux that require a text-mode editor -- in other words, when you don't have the luxury of accessing a GUI desktop at all. Vi and Emacs are the only tools that come with every Linux that work in text mode, so learning one or the other is mandatory.

Getting Started

To start Vi, open a terminal or console and simply type "**vi**" (without the quotation marks) followed by the name of any existing file or a new file you want to create.

Vi works in two main modes, one for editing text and the other for giving commands. To switch between the two modes you use the **I** (**Insert**) and **Esc** keys. The program opens in the Command mode, which is used for cursor movements, delete, cut, copy, paste, and saving changes.

The Insert mode is what you'll work in most of the time. You use it to make changes in an open file. Enter the Insert mode by pressing the I key. Newer Vi versions will display the word "INSERT' on the bottom line while you're in Insert mode.

Press the Esc key to switch Vi back to Command mode. As soon as you hit the Esc key the text "INSERT" on the bottom line disappears.

You save your changes to the open file from the Command mode. Press Shift-ZZ to save.

If you make a mistake when saving a file, such as pressing Ctrl-ZZ or closing Vi before saving the file, you'll end up with a swap file (akin to a DOS/Windows temp file) in addition to the original file. Usually the swap file will have the .swp extension. The original file will not contain the recent changes you made; attempting to reopen it will result in an error message.

The swap file is not readable but can be recovered by typing a command like this at the $ prompt and pressing Enter:

vi -r {your file name}

In some extreme cases, recovery is not possible. But in most cases, such as closing Vi before saving, a system crash, or a power failure, recovery works very well. After you recover, you must manually delete the swap file using a command like this at the $ prompt:

rm .{your file name}.swp

Common Vi Commands

Press Key(s):*	Function:
I	Insert text before the cursor
A	Insert text after the cursor
:	Switch to ex mode
$	Go to last place on the line
^	Go to first place on the line
W	Next word
B	Previous word
Shift-G	Last line of the file
20 Shift-G	Go to line 20
Y	Copy. (Note: Y3W = copy 3 words; Y3J = copy 4 lines.)
P	Paste
D	Cut
X	Delete character under the cursor

Steps for Creation of Program on Linux

1. Make a new file called "test" by opening a console and typing this line after the $ prompt and press Enter:

vi test

2. You'll get an empty console screen since Vi will start with the empty new file. Remember, Vi always starts Command mode, so press the **I (Insert)** key to enter the Insert mode. If you're ever not sure whether you're in Insert mode, you can always just hit **I** again.

3. Next, type the contents of your program.

4. Press the Esc key to return to the Command mode.

5. Save the file by pressing the "**:wq**" key. This command save the file as well as exit from the file. But if you just want to save without exiting then press "**:w**".

6**.** Vi should close and you should see your $ prompt back in the console.

7. The program can be compiled by writing the following command on the $ prompt:

g++ filename.extension followed by pressing enter.

Eg: **g++ test.cpp**

8. If there are any errors in the program it will be displayed on the prompt with line numbers.

9. To remove the errors reopen the file you created. At the $ prompt, type this and press Enter:

vi test

This time Vi opens up to the text in your file, not a blank screen.

10. Now save the file again by pressing Esc to enter the Command mode and recompile it.

11. If noerrors then now run the program to get the output by typing this on $ prompt and press enter:

./a.out

12. Wheneveryou make any changes in your program, you need to save and recompile it before running it for the output.

13.Repeat the entire process again, for each program.

SAMPLE PROGRAM

/*Create class first with data members book no, book name and member function getdata() and putdata(). Create a class second with data members author name, publisher and members getdata() and showdata(). Derive a class third from first and second with data member no of pages and year of publication. Display all these information using array of objects of third class.
*/

```
#include<iostream.h>
class first                                          //this is the base class I
{
        char name[20];
        intbookno;
public:
        void getdata()
        {
                cout<<"\nEnter the name of book : ";
                cin>>name;
                cout<<"\nEnter Book No. : ";
                cin>>bookno;
        }
        void putdata()
        {
                cout<<"\nName of Book : "<<name;
                cout<<"\nBookNo. :"<<bookno;
        }
};
class second                                         //this is the base class II
{
        char author[20],publisher[20];
public:
        void getdata()
        {
                cout<<"\nEnter Author's Name : ";
                cin>>author;
                cout<<"\nEnterPublisher : ";
                cin>>publisher;
        }
        void showdata()
        {
                cout<<"\nAuthor'sName : "<<author;
                cout<<"\nPublisher : "<<publisher;
        }
};


class third: public first, public second             //this is the derived class
```

```
{
        intno_pages, year;
public:
        void get()
        {
                first::getdata();
                second::getdata();
                cout<<"\nEnter No. of Pages : ";
                cin>>no_pages;
                cout<<"\nEnter the year of publication : ";
                cin>>year;
        }
        void display()
        {
                putdata();
                showdata();
                cout<<"\nNo. of Pages : "<<no_pages;
                cout<<"\nYear of Publication : "<<year;
        }
};
intmain()
{
        third book[5];
        intnum;
        cout<<"\nEnter the number of books : ";
        cin>>num;
        for(inti=0;i<num;i++)
        {
                book[i].get();
                cout<<endl;
        }
        for(inti=0;i<num;i++)
        {
                book[i].display();
                cout<<endl;
        }
}
```

OUTPUT:

```
Enter the number of books : 1

Enter the name of book : OOPs

Enter Book No. : 128

Enter Author's Name :Sumeet

Enter Publisher : MAIT

Enter No. of Pages : 148

Enter the year of publication : 2005

Name of Book : OOPs

Book No. :128

Author's Name :Sumeet

Publisher : MAIT

No. of Pages : 148

Year of Publication : 2005
```

2. LAB REQUIREMENTS

Software Requirements:

For C++ Programming:
Linux Operating System
VI Editor or any other Text Editor
GCC Compiler

Hardware Requirements:

Computer nodes with proper configuration

OBJECT ORIENTED PROGRAMMING LAB

Paper Code: ETCS-258
Paper: Object Oriented Programming Lab

List of Experiment:
(As prescribed by G.G.S.I.P.U)

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. 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 greatest 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 tolower( ) to convert upper case letters to lower case.

- Function toupper( ) to convert lower case letters to upper case.

5. Create a class called LIST with two pure virtual function store() and retrieve().To store a value call store

and to retrieve call retrieve function. Derive two classes stack and queue from it and override store and

retrieve.

6. Write a program to define the function template for calculating the square of given numbers with different

data types.

7. 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.

8. 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 the white

spaces on another file.

9. 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 the same file

is opened for reading and displaying the contents of the file on the screen.

10. Write a program to raise an exception if any attempt is made to refer to an element whose index is beyond

the array size.

NOTE:- At least 8 Experiments out of the list must be done in the semester.

3.LIST OF EXPERIMENTS
(As prescribed by G.G.S.I.P.U)

Paper Code: ETCS 258	**P**	**C**
Paper: OOP using C++	**2**	**1**

List of Programs:

1. Write a program to take name, address as character array, age as int , salary as float and contains inline functions to set the values and display it.
2. Using the concept of function overloading Write function for calculating the area of triangle ,circle and rectangle.
3. Write a function power to raise a number m to power n. The function takes a double value for m and int value for n. Use default value for n to make the function to calculate squares when this argument is omitted.
4. Create a class TIME with members hours, minutes, seconds. Take input, add two time objects passing objects to function and display result.
5. Write a program for multiplication of two matrices using OOP.
6. 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 who has more than 70%.Use array of objects.
7. Write a program to enter any number and find its factorial using constructor.
8. Write a program to perform addition of two complex numbers using constructor overloading. 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.
9. Write a program to generate a Fibonacci series using copy constructor.
10. Write a program to find the biggest of three numbers using friend function.
11. Write a program to demonstrate the use of friend function with Inline assignment.
12. Write a program to find the greatest of two given numbers in two different classes using friend function.
13. Write a program to find the sum of two numbers declared in a class and display the numbers and sum using friend class.
14. Write a program to overload unary increment (++) operator .
15. Write a program to overload binary + operator.
16. Write a program to overload less than (<) operator.
17. Write a program to overload assignment (=) operator.
18. Write a program to overload new and delete operators.
19. 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.

20. Create class first with data members book no, book name and member function getdata and putdata. Create a class second with data members author name ,publisher and members getdata and showdata.

Derive a class third from first and second with data member no of pages and year of publication. Display all these information using array of objects of third class.

21. 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 totalmarks. Write a program to model this relationship.
22. Create a base class called SHAPE. Use this class to store two double type values. Derive two specific classes called TRIANGLE and RECTANGLE from the base class. Add to the base class, a member function getdata to initialize base class data members and another member function display to compute and display the area of figures. Make display a virtual function and redefine this function in the derived classes to suit their requirements. Using these three classes design a program that will accept driven of a TRINGLE or RECTANGLE interactively and display the area.
23. Write a program to define the function template for swapping two items of the various data types such as integer ,float,and characters.
24. Write a program to define the function template for calculating the square of given numbers with different data types.
25. Write a program to illustrate how template functions can be overloaded.
26. Write a program to illustrate how to define and declare a class template for reading two data items from the keyboard and to find their sum.
27. Write a program to demonstrate the use of special functions, constructor and destructor in the class template. The program is used to find the biggest of two entered numbers.
28. Write a program to read a set of lines from the keyboard and to store it on a specified file.
29. Write a program to read a text file and display its contents on the screen.
30. Write a program to copy the contents of a file into another.
31. 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 the same file is opened for reading and displaying the contents of the file on the screen.

4. FORMAT OF THE LAB RECORD TO BE PREPARED BY THE STUDENTS

1. The front page of the lab record prepared by the students should have a cover page as displayed below.

NAME OF THE LAB

Font should be(Size 20", italics bold, Times New Roman)

Faculty name

Font should be (12", Times Roman)

Student name

Roll No.:

Semester:

Group:

Font should be (12", Times Roman)

Maharaja Agrasen Institute of Technology, PSP Area,

Sector – 22, Rohini, New Delhi – 110085

Font should be (18", Times Roman)

2. The second page in the record should be the index as displayed below.

INTRODUCTION TO COMPUTERS

PRACTICAL RECORD

PAPER CODE : ETCS-258

Name of the student :

University Roll No. :

Branch :

Section/ Group :

PRACTICAL DETAILS

Experiments according to ITC lab syllabus prescribed by GGSIPU

Exp. no	Experiment Name	Date of performance	Date of checking	Remarks	Marks

5. MARKING SCHEME FOR THE PRACTICAL EXAMS

There will be two practical exams in each semester.

- Internal Practical Exam
- External Practical Exam

INTERNAL PRACTICAL EXAM

It is taken by the concerned lecturer of the batch.

MARKING SCHEME FOR THIS EXAM IS:

Total Marks: 40

Division of 40 marks is as follows

1. Regularity: 25
   - Performing program in each turn of the lab
   - Attendance of the lab
2. Viva Voice: 10
3. File 5

NOTE: For the regularity, marks are awarded to the student out of 10 for each experiment performed in the lab and at the end the average marks are giving out of 25.

EXTERNAL PRACTICAL EXAM

It is taken by the concerned lecturer of the batch and by an external examiner. In this exam student needs to perform the experiment allotted at the time of the examination, a sheet will be given to the student in which some details asked by the examiner needs to be written and at the last viva will be taken by the external examiner.

MARKING SCHEME FOR THIS EXAM IS:

Total Marks: 60

Division of 60 marks is as follows

1. Sheet filled by the student: 20
2. Viva Voice: 15
3. Experiment performance: 15
4. Answer Sheet submitted: 10

NOTE:

- Internal marks + External marks = Total marks given to the students
  (40 marks) (60 marks) (100 marks)
- Experiments given to perform can be from any section of the lab.

6. DETAILS OF THE EACH SECTION

ALONGWITH

EXAMPLES, EXERCISES

&

EXPECTED VIVA QUESTIONS

Section I:
Classes and Objects

Exersises:

1. Write a program to take name, address as character array, age as int , salary as float and contains inline functions to set the values and display it.
2. Using the concept of function overloading Write function for calculating the area of triangle ,circle and rectangle.
3. Write a function power to raise a number m to power n. The function takes a double value for m and int value for n. Use default value for n to make the function to calculate squares when this argument is omitted.
4. Create a class TIME with members hours, minutes, seconds. Take input, add two time objects passing objects to function and display result.
5. Write a program for multiplication of two matrices using OOP.
6. 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 who has more than 70%.Use array of objects.
7. Write a program to enter any number and find its factorial using constructor.
8. Write a program to perform addition of two complex numbers using constructor overloading. 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.
9. Write a program to generate a Fibonacci series using copy constructor.
10. Write a program to find the biggest of three numbers using friend function.
11. Write a program to demonstrate the use of friend function with Inline assignment.
12. Write a program to find the greatest of two given numbers in two different classes using friend function.
13. Write a program to find the sum of two numbers declared in a class and display the numbers and sum using friend class.

Section II:

Inheritance and Polymorphism

Exersises:

1. Write a program to overload unary increment (++) operator .
2. Write a program to overload binary + operator.
3. Write a program to overload less than (<) operator.
4. Write a program to overload assignment (=) operator.
5. Write a program to overload new and delete operators.
6. 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.

7. Create class first with data members book no, book name and member function getdata and putdata. Create a class second with data members author name ,publisher and members getdata and showdata. Derive a class third from first and second with data member no of pages and year of publication. Display all these information using array of objects of third class.
8. 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 totalmarks. Write a program to model this relationship.
9. Create a base class called SHAPE. Use this class to store two double type values. Derive two specific classes called TRIANGLE and RECTANGLE from the base class. Add to the base class, a member function getdata to initialize base class data members and another member function display to compute and display the area of figures. Make display a virtual function and redefine this function in the derived classes to suit their requirements. Using these three classes design a program that will accept driven of a TRINGLE or RECTANGLE interactively and display the area.

Section III:

Templates and Generic Functions

Exersises:

1. Write a program to define the function template for swapping two items of the various data types such as integer ,float,and characters.
2. Write a program to define the function template for calculating the square of given numbers with different data types.
3. Write a program to illustrate how template functions can be overloaded.
4. Write a program to illustrate how to define and declare a class template for reading two data items from the keyboard and to find their sum.
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 biggest of two entered numbers.

Section IV:
File Handling

Exersises:

1. Write a program to read a set of lines from the keyboard and to store it on a specified file.
2. Write a program to read a text file and display its contents on the screen.
3. Write a program to copy the contents of a file into another.
4. 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 the same file is opened for reading and displaying the contents of the file on the screen.

Viva- voce questions

What is the output of printf("%d")
Difference between "C structure" and "C++ structure".
Diffrence between a "assignment operator" and a "copy constructor"
What is the difference between "overloading" and "overridding"?
Explain the need for "Virtual Destructor".
Can we have "Virtual Constructors"?
What are the different types of polymorphism?
What are Virtual Functions? How to implement virtual functions in "C"
What are the different types of Storage classes?
What is Namespace?
What are the types of STL containers?.
Difference between "vector" and "array"?
Can we generate a C++ source code from the binary file?
What are inline functions?
Explain "passing by value", "passing by pointer" and "passing by reference"
What for "mutable" keyword is used?
What is the difference between "calloc" and "malloc"?
Difference between "printf" and "sprintf".
What is "map" in STL?
When shall I use Multiple Inheritance?
Why preincrement operator is faster than postincrement?
What is the difference between an ARRAY and a LIST?
What is faster : access the element in an ARRAY or in a LIST?
Define a constructor - what it is and how it might be called
Describe PRIVATE, PROTECTED and PUBLIC – the differences and give examples.
What is a COPY CONSTRUCTOR and when is it called ?
Explain term POLIMORPHISM and give an example using eg. SHAPE object: If I have a base class SHAPE, how would I define DRAW methods for two objects CIRCLE and SQUARE.
What is the word you will use when defining a function in base class to allow this function to be a polymorphic function?
Can we access virtual functions through objects using dot operator.
#Can we make friend function virtual.
Can we overload friend function.
Can we inherite friend functions.
#what are the benifits of operator overloading.
what is the difference between containership and inheritance?
#what are pure virtual functions?
why is it important to redefine the pure virtual function in drived class?