Savitribai Phule Pune University

S. Y. B. C. A. (Science)

Semester-IV

BCA 244

Lab Course – I

C++ Programming Laboratory

Work Book

Name:____

College Name: ____

Roll No.: ______________________Seat No: _____________________________

Academic Year:___________________________________

From the Chairman's Desk

It gives me a great pleasure to present this workbook prepared by the Board of studies in Computer Applications.

The workbook has been prepared with the objectives of bringing uniformity in implementation of lab assignments across all affiliated colleges, act as a ready reference for both fast and slow learners and facilitate continuous assessment using clearly defined rubrics.

The workbook provides, for each of the assignments, the aims, pre-requisites, related theoretical concepts with suitable examples wherever necessary, guidelines for the faculty/lab administrator, instructions for the students to perform assignments and a set of exercises divided into three sets.

I am thankful to the Chairman of this course and the entire team of editors. I am also thankful to the reviewers and members of BOS, Mr. Rahul Patil and Mr. Arun Gangarde. I thank all members of BOS and everyone who have contributed directly or indirectly for the preparation of the workbook.

Constructive criticism is welcome and to be communicated to the Chairman of the Course and overall coordinator Mr. Rahul Patil. Affiliated colleges are requested to collect feedbacks from the students for the further improvements.

I am thankful to Hon. Vice Chancellor of Savitribai Phule Pune University Prof. Dr. Nitin Karmalkar and the Dean of Faculty of Science and Technology Prof. Dr. M G Chaskar for their support and guidance.

Prof. Dr. S S Sane
Chairman, BOS in Computer Applications
SPPU, Pune

Editors:

Mrs. Pooja S. Pawar	K.R.T. Arts, B.H. Commerce and A.M. Science College, Nashik
Mrs. Deepashree K.Mehendale	Dr. D. Y. Patil Arts Commerce and Science College ,Pimpri
Mr. Sanjay T. Wani	Womens College of Home Science and BCA, Loni
Mrs. Suvarna Pardeshi	Ahmednagar College, Ahmednagar
Mrs. Summaiya Tamboli	Abeda Inamdar Senior College of Arts, Science and Commerce, Pune

Compiled By:

Mrs. Pooja S. Pawar (Chairman, C++ Programming Laboratory)
K.R.T. Arts, B.H. Commerce and A.M. Science College, Nashik

Reviewed By:

1. Prof. Arun Gangarde
   New Arts, Commerce and Science College, Ahmednagar.
   BOS, BCA(Science)

2. Prof. Rahul Patil
   K.R.T. Arts, B.H. Commerce and A.M. Science College, Nashik
   BOS, BCA(Science)

Introduction

1. About the work book:

This workbook is intended to be used by S.Y.B.C.A. (Science) students for the BCA-244 C++ Programming Laboratory Assignments in Semester–IV. This workbook is designed by considering all the practical concepts / topics mentioned in syllabus.

2. The objectives of this workbook are:

1) Defining the scope of the course.
2) To bring the uniformity in the practical conduction and implementation in all colleges affiliated to SPPU.
3) To have continuous assessment of the course and students.
4) Providing ready reference for the students during practical implementation.
5) Provide more options to students so that they can have good practice before facing the examination.
6) Catering to the demand of slow and fast learners and accordingly providing the practice assignments to them.

3. How to use this workbook:

1) The C++ is divided into nine assignments.
2) Each C++ assignment has three SETs.
3) It is mandatory for students to complete the SET A and SET B in given slot.

4. Instructions to the students

1) Students are expected to carry this book every time they come to the lab for practical.
2) Students should prepare oneself beforehand for the Assignment by reading the relevant material.
3) Instructor will specify which problems to solve in the lab during the allotted slot and student should complete them and get verified by the instructor. However student should spend additional hours in Lab and at home to cover as many problems as possible given in this work book.
4) Students will be assessed for each exercise on a scale from 0 to 5.

Not done	0
Incomplete	1
Late Complete	2
Needs improvement	3
Complete	4
Well Done	5

5. Guidelines for Instructors

1) Explain the assignment and related concepts in around ten minutes using white board if required or by demonstrating on Projector.
2) You should evaluate each assignment carried out by a student on a scale of 5 as specified above by ticking appropriate box.
3) The value should also been written on assignment completion page of the respective Lab course.

6. Guidelines for Lab administrator

You have to ensure appropriate hardware and software is made available to each student.

The operating system and software requirements on server side and also client-side areas given below:

1) Server and Client Side - (Operating System) Linux/Windows
2) Turbo C.

Table of Contents

Assignment Completion Sheet

Lab Course I			
C++ Assignments			
Assignment Number	**Assignment Name**	**Marks (out of 5)**	**Teachers Sign**
1	Introduction to C++		
2	Classes and Objects		
3	Constructors		
4	Memory Allocation		
5	Functions		
6	Operator Overloading		
7	Inheritance		
8	Virtual Functions		
9	File Handling		
Total (Out of 45)			
Total (Out of 15)			

Certificate

This is to certify that Mr./Ms._____ has successfully completed BCA244 – C++ Programming Laboratory course in year_____ and his/her seat no. is_____ _. He/she has scored_ Marks out of 15.

Instructor *H.O.D. / Coordinator*

Internal Examiner *External Examiner*

Assignment No.	1	Session:	02
Assignment Name :	Introduction to C++		

Prerequisite:

- Basic Concepts of C Language
- C++ Compiler installed on your Machine

Guidelines for Teachers/Instructors:

- Explain Difference between C & C++
- Explain Basic syntax used in C++
- Demonstration of C++ Installation

Instructions for Students:-

- Remember the difference between C & C++ concepts and Syntax
- To know the different compilers of C++
- Solve Set A, B and C assigned by instructor in allocated slots only.

1. Introduction:

C++, as we all know is an extension to C language which was developed by Bjarne Stroustrup at Bell labs.

What is C++?

C++ is a cross-platform language that can be used to create high-performance applications. C++ gives programmers a high level of control over system resources and memory.
The language was updated 3 major times in 2011, 2014, and 2017 to C++11, C++14, and C++17.

Why Use C++ ?

C++ is one of the world's most popular programming languages.
C++ can be found in today's operating systems, Graphical User Interfaces, and embedded systems.
C++ is an object-oriented programming language which gives a clear structure to programs and allows code to be reused, lowering development costs.
C++ is portable and can be used to develop applications that can be adapted to multiple platforms.
C++ is fun and easy to learn!
As C++ is close to C# and Java, it makes it easy for programmers to switch to C++ or vice versa

C++ Get Started

To start using C++, you need two things:
A text editor, like Notepad, to write C++ code
A compiler, like GCC, to translate the C++ code into a language that the computer will understand
There are many text editors and compilers.

C++ is an **Object-Oriented Programming language**. The most important facilities that C++ adds on to C are classes, inheritance, function overloading and operator overloading. These features enable creating abstract data types, inherit properties from existing data types and support polymorphism, thereby making C++ a truly object-oriented language.

Applications of C++ Programming

As mentioned before, C++ is one of the most widely used programming languages. It has it's presence in

almost every area of software development. list few of them here:

- **Application Software Development -** C++ programming has been used in developing almost all the major Operating Systems like Windows, Mac OSX and Linux. Apart from the operating systems, the core parts of many browsers like Mozilla Firefox and Chrome have been written using C++. C++ also has been used in developing the most popular database system called MySQL.
- **Programming Languages Development -** C++ has been used extensively in developing new programming languages like C#, Java, JavaScript, Perl, UNIX"s C Shell, PHP and Python, and Verilog etc.
- **Computation Programming -** C++ is the best friends of scientists because of fast speed and computational efficiencies.
- **Games Development -** C++ is extremely fast which allows programmers to do procedural programming for CPU intensive functions and provides greater control over hardware, because of which it has been widely used in development of gaming engines.
- **Embedded System -** C++ is being heavily used in developing Medical and Engineering Applications like softwares for MRI machines, high-end CAD/CAM systems etc.

This list goes on, there are various areas where software developers are happily using C++ to provide great softwares.

Structure of C++ Program

Header files are included at the beginning just like in a C program. Here iostream is a header file which provides the user with input & output streams. Header files contain predeclared function libraries, which can be used by users for their ease.

Comments can be included in the program.
For single line comment// is used before mentioningcomment
For Eg: - // This is my first CPP Program.

For multiple line comment, enclose the comment between **/*** and ***/**
For Eg: - /* This is Multilinecomment*/

main function:- Every C++ Program has main function. When your program starts, main ( ) is called automatically. Function begin with an opening brace { and end with a closing brace }. Everything between the opening and closing braces is considered as a part of the function.

C++ Program Template

Following template can be used to write C++ program. Choose a meaningful filename for source file that reflects the purpose of the program with file extension of ".cpp". Write programming statements inside the body of the main ( ) function. C++ Program is a collection of functions. The execution of program begins at main ( ).

```
/* Comment to state the purpose of this program */
#include <iostream.h>
int main()
{
  // Your Programming statements HERE!
return 0;
}
```

The Standard Output Stream (cout)

The predefined object cout is an instance of ostream class. The cout object is said to be "connected to" the standard output device, which usually is the display screen. The cout is used in conjunction with the stream insertion operator, which is written as <<. The use of cout statement causes the string in quotation marks to be

displayed on the screen.

Let us begin by writing our first C++ Program that prints "Hello SYBCA students" message on display console by using standard output stream (cout) statement.

```
/* First C++ program that says Hello SYBCA student message (hello.cpp) */
#include <iostream.h>                  // needed to perform I/O operations
int main ()
{                                      // Program entry point
cout<< "Hello SYBCA students";     // Say Hello
return 0;                              // Terminate main( )
}                                       // End of main( )
```

The Standard Input Stream (cin)

The predefined object **cin** is an instance of **istream** class. The cin object is said to be attached to the standard input device, which usually is the keyboard. The **cin** is used in conjunction with the stream extraction operator, which is written as >>. The use of cin input statement causes the program to wait for the user to type in an input.

Let us consider C++ Program that accepts 2 integer values from user, adds them and prints the result.

```
/*
Program to add two integer values.
*/
#include <iostream.h>                  // needed to perform I/O operations
int main ()                            // Program entry point
{
        int a,b, c;                    // Variable Declaration
        cout<<" Enter 2 values"; // Displays the message
        cin>>a;                        // Accepts value from user
        cin>>b;                         // Accepts value from user
c=a+b;                                 //Calculates value
        cout<<c;                       // Displays the results
        return 0;
}
```

Cascading of I/O Operators

The << and >> operators can be used repeatedly, this is called as cascading of I/O Operators. The statement
cout<< "Sum ="<< sum << "\n";
first sends the string "Sum ="to cout and then sends the value of sum. Finally, it sends the newline character so that the next output will be on the new line.
We can also make use of >> multiple times in one statement called as cascading.
Input operator >> can be cascaded as shown below:

```
cin>> number1 >> number2;
```

The values are assigned from left to right. That is, if we key in two values, say, 10 and 20, then 10 will be assigned to number1 and 20 to number2.

Practice Programs

1. Write a C++ program to find area of circle.
2. Write a C++ program to find maximum of 3 numbers.
3. Write a C++ program to check if number is even or odd.
4. Write a C++ program to find factorial.
5. Write a C++ program to reverse a number.
6. Write a C++ program to check whether number is perfect.

7. Write a C++ program to check whether the number is Armstrong or not?

Set A

1. Write a C++ program to print Floyd's triangle.

```
1
2 3
4 5 6
7 8 9 10
```

2. Write a C++ program to generate multiplication table.

3. Write a C++ program using switch statement which accepts two integers and an operator as (+, -, *, /) and performs the corresponding operation and displays the result.

4. Write C++ program to check whether number is palindrome or not?

Set B

1. Write a C++ program to display factors of a number.

2. Write a C++ program to calculate following series:
(1) + (1+2) + (1+2+3) + (1+2+3+4) + ... +(1+2+3+4+...+n)

3. Write a C++ program to convert decimal number to hexadecimal.

Set C

1. Write a C++ program to print right oriented right angled pyramid.

```
      1
    2 3
  4 5 6
7 8 9 10
```

2. Write a C++ program to check whether a number can be expressed as sum of two prime numbers.

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well Done [ ]

Signature of Instructor

Assignment No.	2	Session :	03
Assignment Name :	Classes and Objects		

Pre-requisite:

- Real time examples of classes and objects
- Attributes of classes

Guidelines for Teachers/Instructors:

- Explain structure of Class
- Demonstration of Class and object declaration

Instructions for Students

- Understand Data and member function
- Solve Set A, B and C assigned by instructor in allocated slots only.

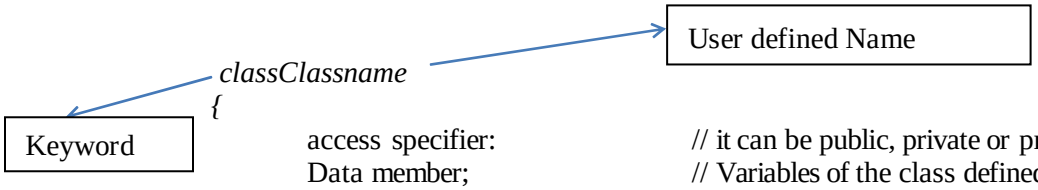
Class: -

It is a user defined data type which holds its own data members and member functions, which can be accessed and used by creating an instance of that class.
For example: - Consider class of students. There may be many students with different names and Qualities but all of them will share some common properties like all of them will have Roll No, Name, Class, Branch etc. So here, student is the class and rollno, name, class, branch is their properties.

An **Object:-**

Is an instance of a Class. When a class is defined, no memory is allocated but when it is instantiated (i.e. an object is created) memory is allocated.

A class is defined in C++ using keyword class followed by the name of class. The body of class is defined inside the curly brackets and terminated by a semicolon at the end.



For the above class student, class will be defined as follows:

```
class student
{
        int rollno;
        char name[20];
        char class[20];
        char branch[20];          //Data Members
public:
       void getdata();            //Member Function
```

```
    void putdata();              // Member Function
};
```

A class can have public, private or protected label sections for access specifiers. The default access for data members is private.

Public: - public member is accessible from anywhere outside the class but within a program. You can set and get the value of public variables without any member function.
Private: -A **private** member variable or function cannot be accessed, or even viewed from outside the class. Only the class and friend functions can access private members.

Protected: - It is member variable or function is very similar to a private member but it provided one additional benefit that they can be accessed in child classes which are calledderivedclasses.

Defining Member Functions:-

Member functions are the functions, which have their declaration inside the class definition and works on the data members of that class. The definition of member functions can be inside or outside the definition of class. If the member function is defined inside the class definition it can be defined directly, but if its defined outside the class, then we have to use the scope resolution operator along with class name along with function name.

Member function defined inside the class:-

Member function inside the class does not require to be declared first here we can directly define the *function.*

```
class student
{
        int rollno;
        char name[20];
        char class[20];
        char branch[20];          //Data Members
public:
        void getdata()            //Member Function defined inside the class
        {
        cout<<"enter rollno";
        cin>>rollno;
        cout<<"enter name";
        cin>>name;
        cout<<"enter branch";
        cin>>branch;
        }
};
```

Member function defined outside the class:-

In this case we must first declare the function inside the class and then define the function outside the class using scope resolution operator.

```
class student
{
        int rollno;
        char name[20];
        char class[20];
        char branch[20];          //Data Members
public:
        void getdata();
        void putdata();
};
```

```
void  student::getdata()                    // Use of scope resolution operator
{
        cout<<"enter rollno";
        cin>>rollno;
        cout<<"enter name";
        cin>>name;
        cout<<"enter branch";
        cin>>branch;
}
```

Array of Objects: -

Like array of other user-defined data types, an array of type class can also be created. The array of type class contains the objects of the class as its individual elements. Thus, an array of a class type is also known as an array of objects. An array of objects is declared in the same way as an array of any built-in data type.

The syntax for declaring an array of objects is

class_name array_name [size];

Simple example illustrating use of array of objects.

```
#include<iostream.h>
class books
{
char tit1e [30];
float price ;
public:
voidgetdata ( );
void putdata ( );
};
 void books :: getdata ( )
{
        cout<<"Title:";
        cin>>title;
        cout<<"Price:";
        cin>>price;
 }
void books :: putdata ( )
{
        cout<<"Title:"<<title<<"\n";
        cout<<"Price:"<<price<< "\n";
}
int main ( )
{
        books book[4] ;                          // Array of books created
        for(int i=0;i<4;i++)
        {
        cout<<"Enter details o£ book "<<(i+1)<<"\n";
        book[i].getdata( );
        }
        for(int i=0;i<4;i++)
        {
        cout<<"\nBook "<<(i+l)<<"\n";
        book[i].putdata( ) ;
        }
```

return 0;}

Manipulators: -

Manipulators are operators used in C++ for formatting output. The data is manipulated by the programmer‟s choice of display. Formatted output is very important in development field so that the data is easily read and understood. To make use of manipulators we need to include header file *iomanip.h*.

Some of the commonly used manipulators are as follows:-

Manipulator	Purpose	Syntax
Endl	**endl** manipulator is used to terminate a line and flushes the buffer.	Endl
Setw	**setw** manipulator sets the width of the field assigned for the output. The field width determines the minimum number of characters to be written in some output representations.	setw(no of characters) for eg:- setw(10) This will set the width of the field to 10 characters
Setfill	This is used after setw manipulator. If a value does not entirely fill a field, then the character specified in the setfill argument of the manipulator is used for filling the fields.	setfill(character) for eg:- setfill(*) This will fill the field with *
Setprecision	It sets the decimal precision for floating point values.	setprecision(number of digits) for eg:- setprecision(2); This will print the floating point output to 2 decimal places.

Set A

1. Write the definition for a class called Cylinder that contains data member's radius and height. The class has the following member functions:
    a. void setradius(float) to set the radius of data member
    b. void setheight(float) to set the height of data member
    c. float volume( ) to calculate and return the volume of the cylinder
    d. float area( ) to calculate and return the area of the cylinder.

   Write a C++ program to create two cylinder objects and display each cylinder and its area and volume.

2. Create a class named „DISTANCE‟ with: - feet and inches as data members. The class has the following member functions:
    a. To input distance
    b. To output distance
    c. To add two distance objects

   Write a C++ program to create objects of DISTANCE class. Input two distances and output the sum.

3. Write a C++ program to create a class District. Having district_code, district_name, area_sqft, population, literacy_rate. For displaying details use appropriate manipulators. The program should contain following menu :
    a. Accept details of n district
    b. Display details of district.
    c. Display details of district having highest literacy rate.
    d. Display details of district having least population.

4. Define a class string to perform different operations:

a. To find length of string.
b. To concatenate two strings.
c. To reverse the string.
d. To compare two strings.

Set B

1. Create a class for different departments in a college containing data members as Dept_Id, Dept_Name, Establishment_year, No_of_Faculty, No_of_students. Write a C++ program with following member functions:
   a. To accept „n‟ Department details
   b. To display department details of a specific Department
   c. To display department details according to a specified establishment year

2. Write a C++ program to define a class Bus with the following specifications :
   - Bus No
   - Bus Name
   - No of Seats
   - Starting point
   - Destination

   Write a menu driven program by using appropriate manipulators to
   a. Accept details of n buses.
   b. Display all bus details.
   c. Display details of bus from specified starting and ending destination by user.

Set C

1. Write a C++ program to create a class employee having Emp_no, Name, Basic, DA, HRA, Allowances. Write necessary member functions to accept and display details of Employee and generate a Pay slip using appropriate manipulators for formatted display.

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well Done [ ]

Signature of Instructor

Assignment No.	3	Session :	3
Assignment Name :	Constructors		

Prerequisite:

- Basic Concepts of Functions
- Concepts of classes and objects

Guidelines for Teacher /Instructors:

- Demonstration of how to create constructor and types of constructor with a simple program is expected

Instructions for Students:

- Students must read theory and syntax for solving programs before his/her practical slot.
- Solve Set A, B and C assigned by instructor in allocated slots only.

Constructor

A constructor is a special member function of a class which initializes objects of a class. In C++, Constructor is automatically called when object of class is created.

How constructors are different from a normal member function?

A constructor is different from normal functions in following ways:

- Constructor has same name as the class itself.
- Constructors don't have return type.
- Constructors are declared as public member function.
- A constructor is automatically called when an object is created.
- If we do not specify a constructor, C++ compiler generates a default constructor.

Constructors can be defined either inside the class definition or outside class definition using class name and scope resolution:: operator.

Example:

```
class Number
{
        int x;
    public:
        Number( )                    //Constructor Defined Inside Theclass
        {
            x=10;
        }
 };
```

Example:

```
class Number
{
        int x;
    public:

        Number( );                  //Constructor Declared
};

  Number : : Number( )              //Constructor Define Outside Class
 {
        x=10;
  }
```

Types of Constructors:-

1. **Default Constructors:** Default constructor is the constructor which doesn't take any argument. It has no parameters.

```
class Number
{
        int x;
    public:
        Number( )                   //Constructor defined inside the class
        {
            x=10;
        }
};
  int main( )
  {
      Number obj1;
      return 0;
  }
```

2. **Parameterized Constructors:** Parameterized constructor is the constructor that takes arguments. These arguments help to initialize an object when it is created.

If there is more than one constructor define in a class, it is called as **Constructor Overloading.**

```
class Number
{
        int x;
    public:
        Number(int a )
        {
            x=a;
        }
};

int main( )
{
    // The constructors can be called explicitly or implicitly.
        Number   obj1=Number(10);   //Explicit   call
        Number obj2(10);            //Implicit   call
        return 0;
}
```

3. **Copy Constructor:** A copy constructor is a constructor which initializes an object using another object of the same class. It takes a reference of object of the same class as its argument. It copies data from one object to other by copying every member of an object with the member of object passed as argument.

```
class Number
{
        int x;
  public:
        Number(int a )
        {

      }
 Number(Number & N)
      {
            x=N.x;
        }

  };
 int main( )
 {
      Number obj1 (10),obj(obj1);
      Return 0;
 }
```

Set A

1. Write a C++ program to create a class Number which contains two integer data members. Create and initialize the object by using default constructor, parameterized constructor. Write a member function to display maximum from given two numbers for all objects.

2. Write a C++ program using class to calculate simple interest amount. (Use parameterized constructor with default value for rate)

3. Write a C++ program to create a class Mobile which contains data members as Mobile_Id, Mobile_Name, Mobile_Price. Create and Initialize all values of Mobile object by using parameterized constructor. Display the values of Mobile object where Mobile_price should be right justified with a precision of two digits.

4. Write a program to find sum of numbers between 1 to n using constructor where value of n will be passed to the constructor.

Set B

1. Write the definition for a class called „point" that has x & y as integer data members. Use copy constructor to copy one object to another. (Use Default and parameterized constructor to initialize the appropriate objects)

2. Write a C++ program to create a class Date which contains three data members as dd, mm, and yyyy. Create and initialize the object by using parameterized constructor and display date in dd-mon-yyyy format. (Input: 19-12-2014 Output: 19-Dec-2014) Perform validation for month.

Set C

1. Write a C++ program to create a class Worker that has data members as Worker_Name, No_of_Days_worked, Pay Rate. Create and initialize the object using default constructor, parameterized constructor and copy constructor. Also write necessary member function to calculate and display the salary of worker.

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well Done [ ]

Signature of Instructor

Assignment No.	4	Session :	02
Assignment Name :	Memory Allocation		

Prerequisite:

- Basic Concepts of Memory Allocation
- Concept of static and dynamic Memory Allocation
- Concept of Constructor
- Concept Of Pointers

Guideines for Teacher /Instructors:

- Demonstration of how to allocate memory using new operator and how to free the allocated memory using delete operator with a simple program is expected.

Instructions for Students:

- Students must read theory and syntax for solving programs before his/her practical slot.
- Solve Set A, B and C assigned by instructor in allocated slots only.

Memory allocation is a process by which computer programs and services are assigned with physical or virtual memory space.

Dynamic Memory Allocation

C++ allows us to allocate the memory of a variable or an array in run time. This is known as dynamic memory allocation.

Dynamic memory allocation Operators

Dynamic memory allocation in C++ used to perform memory allocation manually by programmer. C++ has two operators **new** and **delete** that perform the task of allocating and freeing the memory in a better and easier way.

New Operator: The new operator denotes a request for memory allocation on the Heap. If sufficient memory is available, new operator initializes the memory and returns the address of the newly allocated and initialized memory to the pointer variable.

Syntax to use new operator:
To allocate memory of any data type, the syntax is:

```
Pointer-variable = new data-type;
Pointer-variable = new data-type (value);        //To initialize memory
Pointer-variable = new data-type [size];
```

Here, pointer-variable is the pointer of type data-type. Data-type could be any built-in data type including array or any user defined data types.

Delete Operator: Delete operator is used to de-allocate the memory created by new operator at run-time. Once the memory is no longer needed it should be freed so that the memory becomes available again for other request of dynamic memory.

Syntax for delete operator

delete ptr;
delete[ ] ptr;

Example using new and delete operator

```
#include<iostream.h> int
main()
{
        int size, i;
        int *ptr;
        cout<<"\n\tEnter size of Array : ";
        cin>>size;
        ptr = new int[size]; //Creating memory at run-time and return first byte of address to ptr.
        for(i=0;i<5;i++)
        {
                cout<<"\nEnter any number : ";
                cin>>ptr[i];
        }


        for(i=0;i<5;i++)
        cout<<ptr[i]<<", ";
        delete[ ] ptr;          //deallocating all the memory created by new operator

}
```

Dynamic Constructor: The constructor that can be used to allocate memory while creating Objects. Allocation of memory to objects at the time of their construction is known as dynamic construction of objects.

Destructor

Destructor is a member function which destructs or deletes an object that has been created by constructor.

A destructor function is called automatically when the object goes out of scope like:

a. when the function ends
b. when the program ends
c. a block containing local variables ends
d. a delete operator is called

How destructors are different from a normal member function?

- Destructors have same name as the class preceded by a tilde(~).
- Destructors don‟t take any argument and does not return anything .

Example:

```
class Number
{
        int x:
public :

        Number( )
        {
```

```
            cout << "\ Constructor called :";
        }

        ~Number( )
        {

        cout<<"\Destructor called :";
        }
};
int main( )
{
        Number Obj1;
        {
            Number Obj2;
        }                       //Destructor Ob2 called
        return 0;               //Destructor Ob1 called
}
```

Set A

1. Write a C++ program to create a class which contains single dimensional integer array of given size. Define member function to display median of a given array. (Use Dynamic Constructor to allocate and Destructor to free memory of anobject)
2. Write a program to create memory space using the new keyword and to destroy it using delete keyword
3. Write a C++ Program to store GPA of a number of students and display it where n is thenumber of students entered by the user
4. Write a c++ program that determines a given number is prime or nor .(use Dynamic Constructor to allocate and Destructor to free memory of an object)

Set B

1. Write a C++ program to create a class which contains two dimensional integer array of size mXn. Write a member function to display transpose of entered matrix. (Use Dynamic Constructor for allocating memory and Destructor to free memory of an object)
2. Write a program for combining two strings also show the execution of dynamic constructor.For this declare a class string with data members as name and length.

Set C

1. A book shop maintains the inventory of books that are being sold at the shop. The list includes details such as author_name, title, price, publisher, and stock position.If customer wants to purchase a book he gives details of book along with the number of copies required. If requested copies are available the total cost of requested copies is displayed; otherwise the message "Required copies not in stock" is displayed. Design a system using a class called Bookshop with suitable member functions and constructor. (Use new operator to allocate memory)

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well Done [ ]

Signature of Instructor

Assignment No.	5	Session :	04
Assignment Name :	Functions in C++		

Prerequisites:

- Functions
- Inline function
- Example of inline function
- Friend function
- Friend function using single class
- Friend function using two classes
- Function Overloading
- Ways to overload a function
- Default Argument

Guidelines for teacher/Instructor

- Demonstration of inline functions, friend functions, function overloading and default argument must with proper rules

Instruction to Students

- Student must read the theory, syntax and solved example of inline function, friend function, function overloading and default arguments.
- Solve SET A, SET B or SET C assign by instructor in allocated time slot only.

Functions:-

Functions are the building blocks of C++ programs where all the program activity occurs.
Function is a collection of declarations and statements.
A large single list of instructions becomes difficult to understand. For this reason functions are used. A function has a clearly defined objective (purpose) and a clearly defined interface with other functions in the program. Reduction in program size is another reason for using functions.

Inline Function:

Inline function is one of the important features of C++. To inline a function, place the keyword inline before the function name and define the function before any calls are made to the function. The compiler can ignore the inline qualifier in case defined function is more than a line.

Syntax

```
inline return type function-name(argument list)
{
        //Function Body
}
```

Example of inline function to return max of two numbers −

```
#include <iostream.h>
inline int Max(int x, int y)
{
return (x > y)? x : y;
}
int main()
{
```

```
cout<< "Max (20,10): " << Max(20,10) <<endl; cout<< "Max (0,200): " << Max(0,200) <<endl; cout<<
"Max (100,1010): " << Max(100,1010) <<endl;
return 0;
}
```

Friend Function

A friend function of a class is defined outside that class scope but it has the right to access all private and protected members of the class. Even though the prototypes for friend functions appear in the class definition, friends are not member functions. Since they are non-member functions, Friend functions are called without object of that class.

Syntax

```
class class_name
{
        ... .. ...
        friend return_type function_name(argument/s)
        {
                //Function Body
        }
        ... .. ...
}
```

Example of friend function for single class

```
#include<iostream.h>
class calculate
{
        int x,y;
public:
        void readdata()
        {
                cout<<"\n enter two numbers";
                cin>>x>>y;
        }
        friend void add (calculate c);
};
void add (calculate cal)
{
        cout<<"sum is"<<cal.x+cal.y;
}
main( )
```

```
{
        calculate calc;
        calc.readdata();
        add(calc);
}
```

Creating friend function as a bridge between two classes:-

A friend function can act as a bridge between two classes.

Example of friend function for two classes

```
#include<iostream.h>
#include<conio.h>
class B;                                    //forward declaration
class A
{
  int x;
 Public:
    void getdata( int p)
     {
            x=p;
     }
    void display()
    {
    Cout<<"x="<<x<<endl;

    }
    Friend int sum(A d,B e);
    };
Class B
    {
       int y;
            public:
    void getdata(int q)
    {
      y=q;
    }
    void display ()
```

```
{
 Cout<<"\ny="<<y<<endl;
}
friend int sum(A d, B e)
};
int sum(A d,B e)
{
return( A.x + B.y)
}
int main()
{
        A d1;
        B e1;
        d1.getdata(10);
        e1.getdata(20);
        d1.display();
        e1.display();
        cout<<sum(d1,e1);
}
```

Function Overloading:

If any class has multiple functions with same names but different parameters then they are said to be overloaded. Function overloading allows you to use the same name for different functions, to perform, either same or different functions in the same class. Function overloading is usually used to enhance the readability of the program. If you have to perform one single operation but with different number or types of arguments, then you can simply overload thefunction.

Ways to overload a function

By changing number of Arguments.

By having different types of argument.

By changing number of Arguments:-

In this type of function overloading we define two functions with same names but different number of parameters of the same type.

For example, in the below mentioned program we have made two sum () functions to return sum of two and three integers.

```
int sum (int x, int y)
{
    cout<<x+y;
}
int sum(int x, int y, int z)
{
```

```
    cout<<x+y+z;
}
```

Here sum( ) function is overloaded, to have two and three arguments, which sum() function will be called, depends on the number of arguments.

```
int main()
{
        sum(10,20);          // sum() with 2 parameter will be called
        sum(10,20,30);       //sum() with 3 parameter will be called
}
```

By having different types of argument:-

In this type of overloading we define two or more functions with same name and same number of parameters, but the type of parameter is different. For example in this program, we have two sum() function, first one gets two integer arguments and second one gets two double arguments.

```
int sum(int x, int y)
{
        cout<<x+y;
}
double sum(double x, double y)
{
        cout<<x+y;
}
int main()
{
        Sum (10, 20);
        sum(10.5,20.5);
}
```

Default Arguments

When we mention a default value for a parameter while declaring the function, it is said to be as default argument. In this case, even if we make a call to the function without passing any value for that parameter, the function will take the default value specified.

```
sum (int x, int y=0)
{
        cout<<x+y;
}
int main()
{
        sum (10);
        sum (10,0);
```

```
        sum (10,10);
    }
```

First two function calls will produce the exact same value, but for the third fu nction call, y will take 10 values and output will become 20. By setting default argument, we are also overloading the function.

Set A

1. Write a C++ program to print area of circle, square and rectangle usinginlinefunction.
2. Write a C++ program to subtract two integer numbers of two different classes using friend function.
3. Write a C++ program to overload function volume and find volume of cube, cylinder and sphere.
4. Write a C++ program to calculate function to determine simple interest byusing default arguments as follows
   int calculate(int p,int n=10,int r=7)- Returns SI by specifying no of years and rate of interest

Set B

1. Write a C++ program to create two classes Rectangle1 and Rectangle2.Compare area of both the rectangles using friend function.
2. Write a program to design a class complex to represent complex number. The complex class should use an external function (use it as a friend function) to add two complex number. The function should return an object of type complex representing the sum of two complex numbers.
3. Create a class telephone containing name, telephone number and city as a data member and write necessary member functions for the following (use function overloading).
   a. Search the telephone number with givenname
   b. Search the name with given telephone number
   c. Search all customer in a given city

Set C

1. Write a C++ program to implement a class „printdata" to overload „print" function as follows:
   void print(int) - outputs value - <int>, that is, value followed by the value of the integer. eg. print(10) outputs value -<10>
   void print(char *) – outputs value –"char*", that is, value followed by the string in double quotes. eg print("hi") outputs value-"hi"
   void print(int n, char *)- display first n characters from the given string. eg print(3,"Object")- outputs value –"Obj"
2. Write a C++ program to create two classes DM and DB which stores the value of distances. DM stores distance in m and cm and DB stores distance in feet and inches. Write a program that can read valuefor the class objects and add one object of DM with the other object of DB by using friend function.

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well Done [ ]

Signature of Instructor

Assignment No.	6	Session :	04
Assignment Name :	Functions in C++		

Prerequisites:

- Operator overloading
- Rules of Operator Overloading
- Unary operator overloading
- Overloading Arithmetic Operator
- Binary Operator Overloading
- Binary operator using friend function
- Overloading insertion and extraction operators
- Overloading Relational Operator

Guidelines for teacher/Instructor

- Demonstration of operator functions must with proper rules

Instruction to Students

- Student must read the theory, syntax and solved example of Operator overloading
- Solve SET A, SET B or SET C assign by instructor in allocated time slot only.

Operator Overloading

It is a type of polymorphism in which an operator is overloaded to give user defined meaning to it. Overloaded operator is used to perform operation on user-defined data type. For example '+' operator can be overloaded to perform addition on various data types, like for Integer, String etc.

Operator overloading is carried out by writing member functions using the operator keyword called operator functions.

Syntax of operator function

Return type class name :: operator Op (argument List)

{

```
//Function Body
```

}

Where, return type is the type of value returned by the specified operation Op is the operator being overloaded. Op is preceded by keyword operator.

The argument list will depend on whether the operator is unary or binary and whether the function is a member function or friend function.

For a unary operator, a non member function will have one argument while a member function will have no arguments as the class object itself is the object on which operator operates. For a binary operator, a non member function will have two arguments while a member function has one argument, the other implicitly being the class object itself.

Restrictions on Operator Overloading

Following are some restrictions to be kept in mind while implementing operator overloading.

Precedence and associatively of an operator cannot be changed.

Arity (numbers of Operands) cannot be changed. Unary operator remains unary, binary remains binary etc.

No new operators can be created, only existing operators can be overloaded.

Cannot redefine the meaning of a procedure. You cannot change how integers are added.

There are few operators in C++ that cannot be overloaded such as

ternary operator ?:,
sizeof,
scope resolution operator ::
membership operators . and .* .

Overloading Unary operators

Unary operators are Increment, Decrement and unary minus which can be overloaded. Example of overloading increment operator
The increment operator ++ is used in two ways: pre-increment (++d) and post-increment(d++).
To distinguish between pre and post increment operator overloading, dummy parameter of type int in the function heading of the post-increment operator function is used. Decrement operator can be overloaded similarly.

```
void operator++()
{
    ++day;
    ++month;
    ++year;
 }
    void operator++(int)
    {
            day++; month++; year++;
    }
```

Overloading arithmetic operators

Arithmetic operator is most commonly used operator in C++. Almost all arithmetic (+, - , *, /) operators are overloaded to perform arithmetic operation on user-defined data type.
Example of overloading binary + operator Overloading Binary + using member function Date operator + (Date D1)
{
Date D3;
D3.day = day + D1.day; D3.month = month + D1.month; D3.year = year + D1.year; Return D3;
}

Overloading Binary + using friend (non-member) function
friend Date operator + (Date D1,Date D2)
{
Date D3;
D3.day = D1.day + D2.day; D3.month = D1.month + D2.month; D3.year = D1.year + D2.year;
Return D3;
}

Overloading insertion and extraction operators

Overloading insertion(<<) operator and extraction (>>) operator is used to input and output objects using stream class library in the similar way as built in data types.

Example

<< operator is overloaded with ostream class object cout to print primitive type value output to the screen.

Similarly you can overload << operator in your class to print user-defined type to screen. For example we will overload << in Date class to display date object using cout.
For a class Date
Date D1, D2(2,3);
cin>> D1; cout<< D2;

```
friend ostream& operator<< (ostream&out, Date &D)
{
    out<<D.dayr<<D.month<<D.year ; return out;
}
    friend istream& operator >> (istream&in, Date &D)
    {
            in>>D.day ; in>>D.month; in>>D.year; return in;
    }
```

Overloading Relational operator

Relational operators like ==, > ,>=, <, <=, !, != are used to compare two user-defined objects.
Examples to compare two strings are equal or not

```
class comp
    {
            Public:
             Char *s;
            void getstring(char *str)
            {
                    strcpy(s,str);
            }

            void operator==(comp ob)
            {
                    if(strcmp(s,ob.s)==0)
                    cout<<"\nStrings are Equal";
                    else
                    cout<<"\nStrings are not Equal";
            }
    };

    void main()
    {
            comp ob, ob1;
            char *string1,*string2; cout<<"Enter First String:"; cin>>string1; ob.getstring(string1);
            cout<<"\nEnter Second String:"; cin>>string2; ob1.getstring(string2); ob==ob1;
     }
```

Set A

1. Write a C++ program to create a class Number. Write necessary member functions to overload the operator unary pre and post increment „++ “ for an integer number.

2. Write a C++ program to create a class employee containing salary as a data member. Write necessary member functions to overload the operator unary pre and post decrement “- -“ for incrementing and decrementing salary.

3. Write a C++ program to create a class Array that contains one float array as member. Overload the Unary ++ and -- operators to increase or decrease the value of each element of an array. Use friend function for operator function.

4. Write a program to design a class representing complex number and having the functionality of performing addition & multiplication of two complex number using operatoroverloading.

Set B

1. Write a program to overload operators like *, <<, >>using friend functions .the following overloaded operators should work for a class vector.

2. Write a program for developing matrix classes which can handle integer matrices of different dimensions. Also overload the operator for addition, multiplication and comparison of matrices.

3. Create a class String which contains a character pointer (Use new and delete operator). Write a C++ program to overload followingoperators:
   ! To reverse the case of each alphabet from given string
   [ ] To print a character present at specified index
   < To compare length of two strings
   == To check equality of two strings
   + To concatenate twostrings

Set C

1. Create a class Fraction that contains two data members as numerator and denominator. Write a C++ program to overload followingoperators
   ++ Unary(pre and post both)
   <<and>> Overload as friend functions
   < Returns 1 if first fraction is less than secondfraction

2. Create a class Rational to represent a Rational number. Perform the Basic Arithmetic operations: Addition, Subtraction, Multiplication and Division for two Rational Numbers.

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well Done [ ]

Signature of Instructor

Assignment No.	7	Session :	04
Assignment Name :	Inheritance		

Prerequisites

- Modes of Inheritance
- Types of inheritance:- Single,Multiple,Multilevel,Hirerachical,Hybrid
- Virtual Base Class, Uses of virtual Base Class
- Constructor and Destructor in derived class

Guidelines for Teachers/Instructors:

- Demonstration of object oriented concepts and their syntax using simple programs.

Instructions for Students:

- Students must read theory and syntax for solving programs before his/her practical slot.
- Solve SET A, B or C assigned by instructor in allocated slots only

Inheritance

Definition: Inheritance is one of the feature of Object Oriented Programming System (OOPs) it allows the child class to acquire the properties (the data members) and functionality (the member functions) of parent class.

What is child class?

A class that inherits another class is known as child class, it is also known as derived class or subclass.

What is parent class?

The class that is being inherited by other class is known as parent class, super class or base class.
Implementing inheritance in C++:

Syntax for creating a sub-class which is inherited from the base class:

Syntax

```
class subclass_name : access_mode base_class_name
{
        //body of subclass
};
```

Here, subclass_name is the name of the sub class, access_mode is the mode in which you want to inherit this sub class for example: public, private, protected. And base_class_name is the name of the base class from which you want to inherit the sub class.

Modes of Inheritance

1. **Public mode:** If a sub class is derived from base class in public mode then the public member of the base class will become public in the derived class and protected members of the base class will become protected in derived class. Private members of the base class will never get inherited in subclass.
2. **Protected mode:** If a sub class is derived from a base class in protected mode then both public member and protected members of the base class will become protected in derived class. Private members of the base class will never get inherited in sub class.
3. **Private mode:** If a sub class is derived from a base class in private mode then both public member and

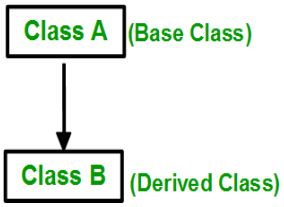
protected members of the base class will become Private in derived class. Private members of the base class will never get inherited in sub class.

The following table summarizes the above three modes and shows the access specifier of the members of base class in the sub class when derived in public, protected and private modes:

	Derived Class	Derived Class	Derived Class
Base class	**Public Mode**	**Private Mode**	**Protected Mode**
Private	Not Inherited	Not Inherited	Not Inherited
Protected	Protected	Private	Protected
Public	Public	Private	Protected

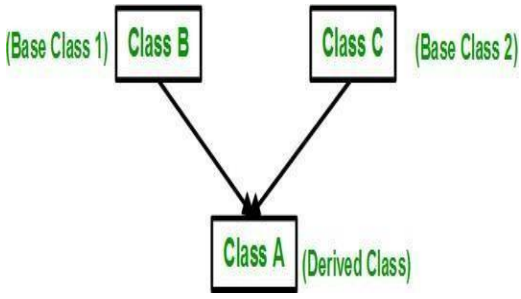
Types of Inheritance in C++

1. **Single Inheritance** : In single inheritance, a class is allowed to inherit from only one base class. i.e. one sub class is inherited by one base class only.



.

2. **Multiple Inheritance:** Multiple Inheritance is a feature of C++ where a class can inherit from more than one classes. i.e one **sub class** is inherited from more than one **base classes**.

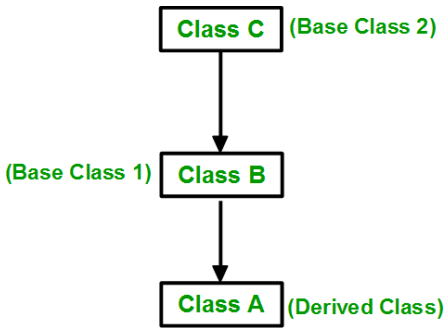


Syntax:

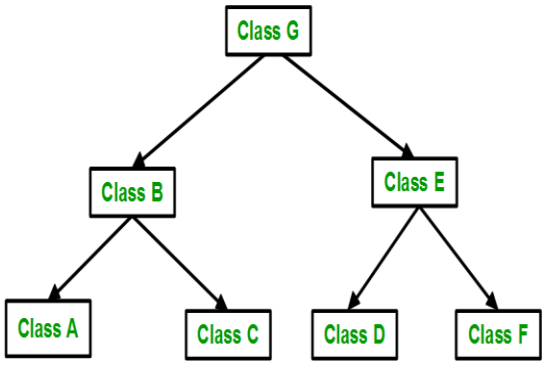
Class subclass_name : access_mode base_class1, access_mode base_class2, ...
{
//body of subclass
};

Here, the number of base classes will be separated by a comma („, „) and access mode forevery base class must be specified.

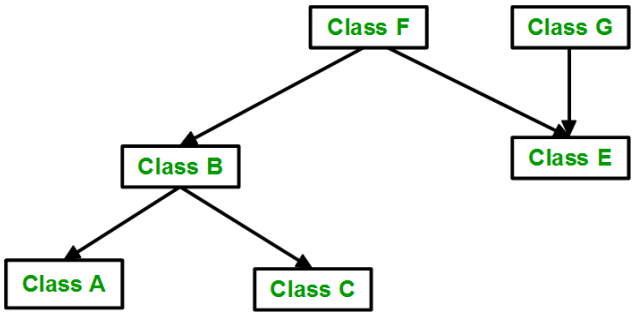
3. **Multilevel Inheritance** : In this type of inheritance, a derived class is created from another derived class.



4. **Hierarchical Inheritance**: In this type of inheritance, more than one sub class is inherited from a single base class. i.e. more than one derived class is created from a single base class.



5. **Hybrid (Virtual) Inheritance**: Hybrid Inheritance is implemented by combining more than one type of inheritance. For example: Combining Hierarchical inheritance and Multiple Inheritance. Below image shows the combination of hierarchical and multiple inheritance:



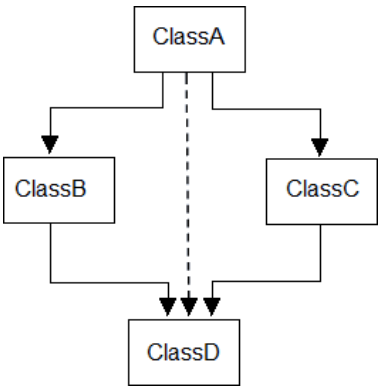
Virtual base class

An ambiguity can arise when several paths exist to a class from the same base class. This means that a child class could have duplicate sets of members inherited from a single base class.

C++ solves this issue by introducing a virtual base class. When a class is made virtual, necessary care is taken so that the duplication is avoided regardless of the number of paths that exist to the child class.

Uses of virtual base class:

When two or more objects are derived from a common base class, we can prevent multiple copies of the base class being present in an object derived from those objects by declaring the base class as virtual when it is being inherited. Such a base class is known as virtual base class. This can be achieved by preceding the base class‟ name with the word virtual.



Syntax:

```
class A
{
        //body of class  A

 }
        //body of class  B
}
} class C : public virtual A
{
          //body of class C
}
class D : public B, public C
{
          //body of class D
}
```

Constructor in derived class

1) While using constructors during inheritance, is that, as long as a base class constructor does not take any arguments, the derived class need not have a constructor function.
2) However, if a base class contains a constructor with one or more arguments, then it is mandatory for the derived class to have a constructor and pass the arguments to the base class constructor.
3) While applying inheritance, we usually create objects using derived class. Thus, it makes sense for the derived class to pass arguments to the base class constructor.
4) When both the derived and base class contains constructors, the base constructor is executed first and then the constructor in the derived class is executed.
5) In case of multiple inheritance, the base class is constructed in the same order in which they appear in the declaration of the derived class. Similarly, in a multilevel inheritance, the constructor will be executed in the order of inheritance.
6) The derived class takes the responsibility of supplying the initial values to its base class. The constructor of the derived class receives the entire list of required values as its argument and passes them on to the base constructor in the order in which they are declared in the derived class.
7) A base class constructor is called and executed before executing the statements in the body of the derived class.

Syntax

```
Derived-Constructor (ArgList2, ArgList2,……. ArgListN. ArgListD) :
Base1(ArgList1)
Base2(ArgList2)
.
.
```

BaseN(ArgListN)
{
// Body of Derived Constructor
}

Destructor in derived class

Destructors are invoked in the reverse order of the constructor invocation i.e. The destructor of the derived class is called first and the destructor of the base is callednext.

Syntax
~Derived-Destructor()
{
// Body of Derived Destructor
}

Set A

1. Design a base class Product(Product _Id, Product _Name, Price). Derive a class Discount (Discount_In_Percentage) from Product. A customer buys „n‟ Products. Calculate total price, total discount and display bill using appropriate manipulators.

2. Create three classes Car, Maruti and Maruti800.Class Maruti extends Car and Class Maruti80 0 extends Maruti.Maruti800 class is able to use the methods of both the classes (Car and Maruti).

3. Design a two base classes Employee (Name, Designation) and Project(Project_Id, title). Derive a class Emp_Proj(Duration) from Employee and Project. Write a menu driven programto
   Build a master table. Display a master table
   Display Project details in the ascending order of duration.

4. Create a Base class Flight containing protected data members as Flight_no, Flight_Name. Derive a class Route (Source, Destination) from class Flight. Also derive a class Reservation(Number_Of_Seats, Class, Fare, Travel_Date) from Route. Write a C++ program to perform following necessary functions:
   Enter details of „n‟ reservations Display details of all reservations
   Display reservation details of a Business class

Set B

1. Create a base class Account (Acc_Holder_Name, Acc_Holder_Contact_No). Derive two classes as Saving_Account(S_Acc_No., Balance) and Current_Account(C_Acc_No., Balance) from Account. The savings account provides interest and withdrawal facility. The current account provides no interest facility. Saving account maintains a minimum balance of 2000 and if the balance falls below this level, a service charge of Rs 500 is imposed. Current account maintains a minimum balance of 5000 and if the balance falls below this level, a service charge of Rs 1000 is imposed.
   Write a C++ menu driven program to perform following functions: Accepting Amount and deposit it into account
   Withdrawing amount from account.
   Calculating Interest and service charge where interest rate for Saving account is 10% of balance Displaying information of account.

2. Create a Base class Train containing protected data membersas Train_no,Train_Name.Derive a class Route(Route_id,Sorce,Destination) from Train class. Also derive a class Reservation(Number_of_Seats,Train_Class,Fare,Travel_Date) from Route.Write a to perform following necessary functions:
   a. Enter details of „n‟ reservations
   b. Display details of all reservations

c. Display reservation details of a specified Train class

Set C

1. Create two base classes Learn_Info(Roll_No, Stud_Name, Class, Percentage) and Earn_Info(No_of_hours_worked, Charges_per_hour). Derive a class Earn_Learn_info from above two classes. Write necessary member functions to accept and display Student information. Calculate total money earned by the student. (Use constructor in derived class)

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well Done [ ]

Signature of Instructor

Assignment No.	8	Session :	03
Assignment Name :	Virtual Function		

Prerequisites

- Virtual Function and rules of Virtual Function
- Virtual Function in derived class
- Pure Virtual Function
- Abstract Class

Guidelines for Teachers/Instructors:

- Demonstration of Virtual and Pure virtual function functions with their syntax using Simple programs.

Instructions for Students:

- Students must read theory and syntax for solving programs before his/her Practical slot.
- Solve SET A, B or C assigned by instructor in allocated slots only

Virtual Functions:

What is Virtual Functions?

A virtual function is a member function within the base class that we redefine in a derived class. It is declared using the virtual keyword.

When a class containing virtual function is inherited, the derived class redefines the virtual function to suit its own needs.

Rules for Virtual Function in C++:

1) They are always defined in a base class and overridden in derived class but it is not mandatory to override in the derived class.
2) The virtual functions must be declared in the public section of the class.
3) They cannot be static or friend function also cannot be the virtual function of another class.
4) The virtual functions should be accessed using a pointer to achieve run time polymorphism.

```
class Base
{
int a;
public:
        virtual void func();        //function in base class
};
class Dervied : public Base
{
int b;
public:
        voidfunc();        //function in derived class
}
```

The main thing to note about the program is, derived class function is called using a base class pointer. The idea is, virtual functions are called according to the type of object pointed or referred, not according to the type of pointer or reference. In other words, virtual functions are resolved late, at runtime.

Example:

```
#include<iostream.h>
 class Base
        {

      public:
        virtual void show() { cout<<" In Base \n"; }
        };

        class Derived: public Base
        {
        public:
        void show() { cout<<"In Derived \n"; }
        };

        int main(void)
        {
                Base *bp = new Derived;
                bp->show(); // RUN-TIME POLYMORPHISM
                return 0;
        }
```

Virtual functions in derived classes:

In C++, once a member function is declared as a virtual function in a base class, it becomes virtual in every class derived from that base class.

Example:

For example, the following program prints “C::fun() called” as B::fun() becomes virtual automatically.

```
#include<iostream>
 class A
{
  public:
    virtual void fun()
    { cout<<"\n A::fun() called ";}
};
 class B: public A {
  public:
    void fun()
    { cout<<"\n B::fun() called "; }
};

class C: public B {
  public:
    void fun()
    { cout<<"\n C::fun() called "; }
};

int main()
{
   C c; // An object of class C
   B *b = &c; // A pointer of type B pointing to c
   b->fun(); // this line prints "C::fun() called"
```

```
  return 0;
}
```

Pure Virtual Function and Abstract Class:

What is pure virtual function?

A pure virtual function is a type of function, which has only a function declaration. It does not have the function definition.

Abstract class:

An abstract class is one that is not used to create object. An abstract class is designed to act as a base class (to be inherited by other classes). The class, which contains pure virtual functions, is called as abstract class.

Syntax:

```
class Base
{
        inti;
        public:
                virtual void getData()=0;
};
class Derived : public Base
{};
```

Example:

```
#include<iostream.h>
class BaseClass
{
        Public:
                virtual void Display1() = 0; // Pure  virtual function
                virtual void Display2() = 0; // Pure  virtual function

                void Display3()
                {
                        cout<< "\n\t This is Display3() method of Base Class;
                }
};
class DerivedClass : public BaseClass
{
        public:
                void Display1()
                {
                        cout<< "\n\t This is Display1() method of Derived class";
                }
                void Display2()
                {
                        cout<<"\n\t This is Display2() method of Derived Class";
                }
};
void main()
{
        DerivedClass D;
        D.Display1();           //This  will invoke  Display1() method of Derived  Class
        D.Display2();           //This  will invoke  Display2() method of Derived  Class
```

D.Display3(); //This will invoke Display3() method of Base Class

}

Set A

1. Write a C++ program to create a class shape with functions to find area of the shapes and display the name of the shape and other essential component of the class. Create derived classes circle, rectangle and trapezoid each having overridden functions area and display. Write a suitable program to illustrate virtual functions.

2. A book (ISBN) and CD (data capacity) are both types of media (id, title) objects. A person buys 10 media items, each of which can be either book or CD. Display the list of all books and CD"s bought. Define the classes and appropriate member functions to accept and display data. Use pointers and concepts of polymorphism (virtual functions).

3. Create a base class Student (Roll_No, Name) which derives two classes Theory and Practical. Theory class contains marks of five Subjects and Practical class contains marks of two practical subjects. Class Result (Total_Marks, Class) inherits both Theory and Practical classes. (Use concept of Virtual Base Class)
   Write a C++ menu driven program to perform the following functions: Build a master table
   Display a master table
   Calculate Total_marks and class obtained

Set B

1. Write a program with Student as abstract class and create derive classes Engineering, Medicine and Science from base class Student. Create the objects of the derived classes and process them and access them using array of pointer of type base class Student.

2. Create a class called LIST with two pure virtual function store() and retrieve(). To store a value call store and to retrieve call retrieves function. Derive two classes stack and queue from it and orverride store and retrieve.

3. Create a base class Person (P_Code, P_Name). Derive two classes Account(Ac_No., Balance) and Official(Designation, Experience) from Person. Further derive another class Employee from both Account and Official classes. (Use Concept of Virtual Base Class)
   Write a C++ menu driven program to perform the following functions: Build a master table for „n" employees.
   Display a master table of „n" employees. Display employees whose designation is H.O.D.
4. Create a base class Student(Roll_No, Name, Class) which derives two classes Internal_Marks(IntM1, IntM2, IntM3, IntM4, IntM5) and External_Marks(ExtM1 ExtM2, ExtM3, ExtM4, ExtM5). Class Result(T1, T2, T3, T4, T5) inherits both Internal_Marks and External_Marks classes. (Use Virtual Base Class)
   Write a C++ menu driven program to perform the following functions: To Accept and display student details
   Calculate Subject wise total marks obtained. Check whether student has passed in Internal and External Exam of each subject. Also check whether he has passed in respective subject or not and display result accordingly.

Set C

1. Write a program to design a class representing the information regarding digital library (book, tape: book and tape should be separate classes having the base class as media). The class should have functionality for adding new items, issuing and deposit etc.

2. Write a program to calculate bonus of the employees. The class master derives the information from both admin and account classes which in turn derives information from class person. Create base and all derived classes having same member functions call getdata, display data and bonus. Create a pointer of base class that capable of accessing data of any class and calculates bonus of the specified employee. (Hint: Use virtual functions)

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well Done [ ]

Signature of Instructor

Assignment No.	9	Session :	03
Assignment Name :	Working with Files		

Prerequisites

- Types of files:- Text file and Binary file
- Operations performed on file.
- Different modes of file opening.
- Syntax for opening, closing, reading and writing to a file.
- Functions used for performing Random Access to a file.
- Syntax of seekg, seekp, tellg and tellp functions.

Guidelines for Teachers/Instructors:

- Demonstration of Opening a text file for reading/writing purpose.
- Demonstration of Opening a binary file for reading/writing purpose.
- Demonstration of use of file pointer for random access to file.
-

Instructions for Students:-

- Students should know the different file opening modes and the basic syntax of file opening, closing, reading and writing.
- Students should also know the concept of Random Access to file and the syntax of functions related to Random Access to file.
- Solve Set A, B and C assigned by instructor in allocated slots only.

We have used the iostream standard library till now which uses cin and cout methods for reading from standard input and writing to standard output respectively. In this assignment we will see how to read and write data from a file.

Files are used to store data in a storage device permanently**.**A file is a collection of related data stored in the particular area on the disk. The I/O system of C++ handles file operations in the same way as that of console input and output operations. It uses file stream as an interface between programs and files. C++ contains a set of classes that define the file handling method. These include ifstream, ofstream and fstream. These three classes are derived from fstreambase class.

- ofstream: This Stream class signifies the output file stream and is applied to create files for writing information to files.
- ifstream: This Stream class signifies the input file stream and is applied for reading information from files.
- fstream: This Stream class can beused for both reading and writing tofiles.

1. Opening file:- To open a particular file for reading or writing operation. We can open file by 2 ways

1. By passing file name in constructor at the time of object creation.
2. By using the open method

A file must be opened before you can read from it or write to it. Either ofstream or fstream object may be used to open a file for writing. ifstream object is used to open a file for reading purpose only. ofstream object is used to open file for writing purpose only.

Method I: -

1. *Using Constructor:* - Constructor is used to initialize an object. We need to use filename to initialize the file stream object.

For example: - ofstream outfile (“student”);

The above syntax declares outfile as an object of ofstream where student is passed as a parameter which indicates filename.

Similarly we may also write

ifstream ifile (“student”);

This syntax declares ifile as an object of ifstream where student is passed as a parameter which indicates file name.

Method II: -

2. *Using open ():-* We can also use open () to open files that uses the same stream object.

Syntax is: - filestreamclass stream object;
streamobject.open (“filename”);

For Example: - ofstream out;
out.open (“student”);

The above syntax declares out as a file stream object and then we use this out object to open a student file.

File Opening Modes: -

We have used ifstream, ofstream constructors and function open () to create a new file as well as to open an existing file. However these functions can also take two arguments where the second argument will be the mode of file. The general form is

Streamobject.open (“filename”,mode);

Here, the first argument specifies the name and location of the file to be opened and the second argument of the open () member function defines the mode in which the file should be opened.

Parameter	Meaning
ios::in	Opens a file for reading only.
ios::out	Opens a file for writing only.
ios::binary	Open a file in binary mode.
ios::ate	Goes to the end of file after opening. It allows contents to be added or modified anywhere within the file.
ios::app	Append contents to the end of the file.
ios::trunc	Deletes the contents of the file if it exists.

You can combine two or more of these modes. For example if you want to open a file in write mode and want to truncate it then the syntaxwill be

```
ofstream outfile;
outfile.open ("file.dat", ios::out | ios::trunc);
```

Similar way, you can open a file for reading and writing purpose as follows − fstream ifile;

```
ifile.open ("file.dat", ios::out | ios::in);
```

2. <u>Closing a File :-</u> When a C++ program terminates it automatically flushes all the streams, releaseall the allocated memory and close all the opened files. But it is always a good practice that a programmer should close all the opened files before program termination.

Following is the standard syntax for close () function, which is a member of fstream, ifstream, and ofstream objects.

```
void close ();
```

3. Reading from a File :- You read information from a file into your program using the stream extraction operator (>>) just as you use that operator to input information from the keyboard. The only difference is that you use an ifstream or fstream object instead of the cin object.

For example: - ifstream ifile;

```
ifile>>rollno>>name;
```

In the above example we have created ifile as an object for ifstream and using ifile object we are reading data from the file.

4. Writing to a File :- While doing C++ programming, you write information to a file from your program using the stream insertion operator (<<) just as you use that operator to output information to the screen. The only difference is that you use an ofstream or fstream object instead of the cout object.

For example: - ofstream ofile;

```
ofile<<rollno<<name;
```

Example: - Write a C++ program to read characters from a text file.

```
#include<iostream>
#include<fstream>
using namespace std;
int main ()
{
        char ch;
        ifstream ifile;
        ifile.open ("a.txt", ios::in);
        if (!ifile)
        {
                cout<<"error in opening file";
                exit(0);
        }
        while (!ifile.eof ())
        {
          ifile.get(ch;)
          cout<<ch;
ifile.close ();
}
```

Example: - Write a C++ program to write characters to a text file.

```
#include<iostream>
#include<fstream>
using namespace std;
void main ()
{
        char ch;
        ofstream fout("a.txt");
        cin>>ch;
        while(ch!='#')
        {
        fout<<ch;
        cin>>ch;
```

```
    }
fout.close();
}
```

File pointer

Each file stream class contains a file pointer that is used to keep track of the current read/write position within the file. When something is read from or written to a file, the reading/writing happens at the file pointer's current location. By default, when opening a file for reading or writing, the file pointer is set to the beginning of the file. However, if a file is opened in append mode, the file pointer is moved to the end of the file, so that writing does not overwrite any of the current contents ofthe file.

Random Access to File

Rather than reading all of the records until you get to the one you want, you can skip directly to the record you wish to retrieve. Each file has two pointers associated with it. One of them is input pointer that is the get pointer and other is output pointer that is the put pointer. The input pointer is used for reading the contents at the given location and output pointer is used for writing contents to the given file location.

To move a file pointer to the desired location following functions are used.

1. seekg () – Moves the get pointer to a specified location.
2. seekp () – Moves put pointer to a specified location.
3. tellg() – Gives the current position of get pointer
4. tellp() - Gives the current position of put pointer.

Binary Files

The functions specifically designed to read and write binary data sequentially are: write and read. The first one (write) is a member function of ostream (inherited by ofstream) and read is a member function of istream (inherited by ifstream). The write () and read () function reads and writes blocks of binary data.

Syntax of read and write () is as follows:-

```
file.read ((char *) &obj, sizeof (obj));
file.write ((char *) &obj, sizeof (obj));
```

Example: - Write a C++ program to read and write data from a binary file.

```
#include <iostream.h>
#include <fstream.h>
                                        //class employee declaration
class Employee {
            int     empID;
            char    empName[100] ;
public :
                                        //function to read employee details
void readEmployee ()
{
        cout<<"EMPLOYEE DETAILS"<<endl;
        cout<<"ENTER EMPLOYEE ID: " ;
        cin>>empID;
        cout<<"ENTER NAME OF THE EMPLOYEE : "
        cin>>empName;
}
```

```
                                        //function to write employee details
        void displayEmployee()
        {
                cout<<"EMPLOYEE ID: "<<empID<<endl;
                cout<<"EMPLOYEE NAME: "<<empName<<endl;
        }
};

int main ()
{
        Employee emp;                   //object of Employee class
        emp.readEmployee ();            //read employee details

                                        //write object into the file
        fstream file;
        file.open ("emp.dat",ios:: out|ios::binary);
        if (!file)
        {
                cout<<"Error in creating file...\n";
                exit(0);
        }

file.write ((char*)&emp,sizeof (emp));
file.close ();
cout<<"Date saved into the file.\n";
                                        //open file again
file.open ("emp.dat", ios::in|ios:: binary);
if(!file)
{
        cout<<"Error in opening file...\n";
        exit (0);
}

if(file.read((char*)&emp,sizeof(emp)))
{
                cout<<"Data from file..\n";
                                        //print the object
                emp.displayEmployee ();
}

file.close();
return 0;
}
```

Set A

1. Write a C++ program to read a text file and count number of Upper case Alphabets, Lower Case Alphabets Digits and Spaces using File Handling
2. Write a C++ program to read integers from a text file and display sum of all integers in that file.
3. Write a C++ program to count the occurrence of a word in a file using filehandling.
4. Write a C++ program using function to count and display the number of lines not starting with alphabet "A" in a text file.

Set B

1. Write a C++ program that copies the contents of one file to another.
2. Write a C++ program to merge two files into a single file using file handling.
   Assuming that a text file named FIRST.TXT contains some text written into it, write a function named vowelwords(), that reads the file FIRST.TXT and creates a new file named SECOND.TXT, to contain only those words from the file FIRST.TXT which start with a lower case vowel(i.e.,with'a','e','i','o','u').
   For example, if the file FIRST.TXT contains Carry umbrella and overcoat when it rains. Then the file SECOND.TXT shall contain umbrella, and, overcoat, it.
3. Write a C++ program to read student information such as rollno, name and percentage of n students. Write the student information using file handling.

Set C

1. Write a Menu driven Program to maintain book records using file handling. Program should contain the following menu
   a. Add new Record
   b. Displayrecord.
   c. Delete a particular record
   d. Search a record
   e. Update a record.
2. Write a C++ program to create a file which has information Name, Account number, Balance and perform following operations:
   a. Add new record
   b. Display content of file
   c. Display name of person having balance > 10,000

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well Done [ ]

Signature of Instructor