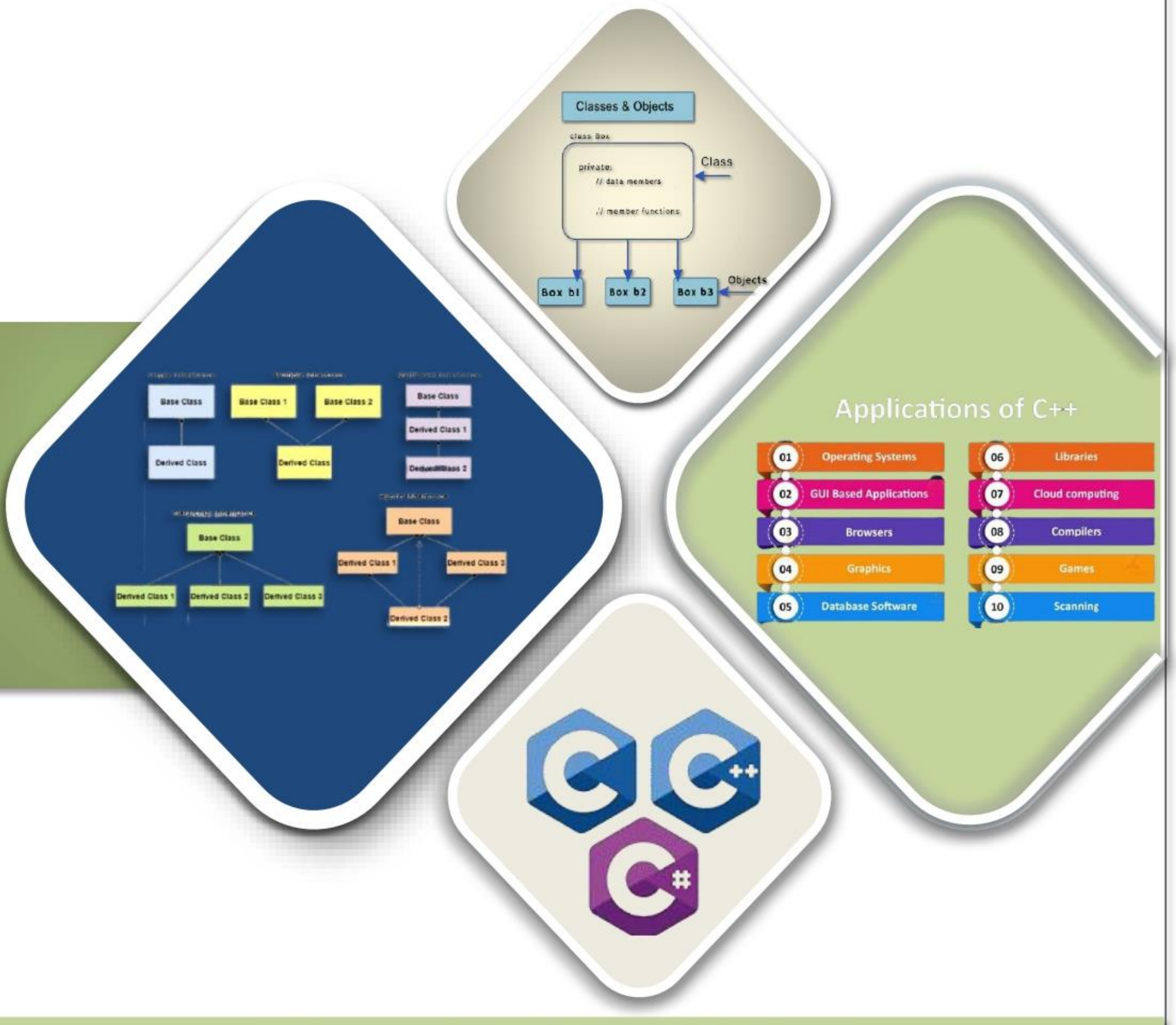
SCHEME : K

Name :	
Roll No. :	Year : 20___ 20___
Exam Seat No. :	

LABORATORY MANUAL FOR **OBJECT ORIENTED PROGRAMMING USING C++ (313304)**



COMPUTER ENGINEERING GROUP

MAHARASHTRA STATE BOARD OF TECHNICAL EDUCATION, MUMBAI
(Autonomous) (ISO 9001: 2015) (ISO/IEC 27001:2013)

VISION

To ensure that the Diploma level Technical Education constantly matches the latest requirements of technology and industry and includes the all-round personal development of students including social concerns and to become globally competitive, technology led organization.

MISSION

To provide high quality technical and managerial manpower, information and consultancy services to the industry and community to enable the industry and community to face the changing technological and environmental challenges.

QUALITY POLICY

We, at MSBTE are committed to offer the best-in-class academic services to the students and institutes to enhance the delight of industry and society. This will be achieved through continual improvement in management practices adopted in the process of curriculum design, development, implementation, evaluation and monitoring system along with adequate faculty development programs.

CORE VALUES

MSBTE believes in the followings:

- Education industry produces live products.
- Market requirements do not wait for curriculum changes.
- Question paper is the reflector of academic standards of educational organization.
- Well-designed curriculum needs effective implementation too.
- Competency based curriculum is the backbone of need based program.
- Technical skills do need support of life skills.
- Best teachers are the national assets.
- Effective teaching learning process is impossible without learning resources.

A Laboratory Manual

for

Object Oriented Programming Using C++

(313304)

Semester-III

(BD/IF/CO/CM/CW/HA/IH/TE)

Maharashtra State

Board of Technical Education, Mumbai

(Autonomous) (1SO:9001:2015) (1SO/IEC 27001:2013)

Maharashtra State Board of Technical Education,
(Autonomous) (ISO:9001: 2015) (ISO/IEC 27001 : 2013)
4th Floor, Government Polytechnic Building, 49, Kherwadi,
Bandra (East), Mumbai - 400051.
(Printed on July, 2024)

MAHARASHTRA STATE BOARD OF TECHNICAL EDUCATION

Certificate

This is to certify that Mr. / Ms...

Roll No......................................'. of Third Semester of Diploma 1n ……………………...……………………………………………… of Institute, (Inst. Code: ....) has completed the term work satisfactorily in course **Object Oriented Programming Using C++ (313304)** for the academic year 20........ to 20……... as prescribed in the curriculum.

Place: .... Enrollment No :................

Date: .... Exam. Seat No: ....

Subject Teacher **Head of the Department** **Principal**

Seal of Institution

Preface

The primary focus of any engineering laboratory/ field work in the technical education system is to develop the much-needed industry relevant competencies and skills. With this in view, MSBTE embarked on this innovative 'I' Scheme curricula for emerging diploma programs with outcome-based education as the focus and accordingly, relatively large amount of time is allotted for the practical work. This displays the great importance of laboratory work making each teacher; instructor and student to realize that every minute of the laboratory time need to be effectively utilized to develop these outcomes, rather than doing other mundane activities. Therefore, for the successful implementation of this outcome-based curriculum, every practical has been designed to serve as a ***'vehicle'*** to develop this industry identified competency in every student. The practical skills are difficult to develop through 'chalk and duster' activity in the classroom situation. Accordingly, the 'I' scheme laboratory manual development team designed the practical's to ***focus*** on the ***outcomes,*** rather than the traditional old practice of conducting practical's to 'verify the theory' (which may become a byproduct along the way).

This laboratory manual is designed to help all stakeholders, especially the students, teachers and instructors to develop in the student the pre-determined outcomes. It is expected from each student that at least a day in advance, they have to thoroughly read through the concerned practical procedure that they will do the next day and understand the minimum theoretical background associated with the practical. Every practical in this manual begins by identifying the competency, industry relevant skills, course outcomes and practical outcomes which serve as a key focal point for doing the practical. The students will then become aware about the skills they will achieve through procedure shown there and necessary precautions to be taken, which will help them to apply in solving real-world problems in their professional life.

This manual also provides guidelines to teachers and instructors to effectively facilitate student- centered lab activities through each practical exercise by arranging and managing necessary resources in order that the students follow the procedures and precautions systematically ensuring the achievement of outcomes in the students.

In the modern world of Information technology, the Object Oriented Programming has become the most preferred approach for software development. It offers a powerful way to cope up with complexity of real-world problems. Among the OOP languages available, C++ is the primitive language which develops fundamental understanding of Object Oriented Concepts. This course enables students to develop programs in 'C++' using Object Oriented Programming approach.

Although best possible care has been taken to check for errors (if any) in this laboratory manual, perfection may elude us as this is the first edition of this manual. Any errors and suggestions for improvement are solicited and highly welcome

Programme Outcomes (POs) to be achieved through Practical of this Course:

PO1: Basic and Discipline specific knowledge: Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems.

PO2: Problem analysis: Identify and analyze well-defined engineering problems using codified standard methods.

PO3: Design/ development of solutions: Design solutions for well-defined technical problems and assist with the design of systems components or processes to meet specified needs.

PO4: Engineering Tools, Experimentation and Testing: Apply modern tools and appropriate technique to conduct standard tests and measurements.

PO5: Engineering practices for society, sustainability and environment: Apply appropriate technology in context of society, sustainability, environment and ethical practices.

PO6: Project Management: Use engineering management principles individually, as a team member or a leader to manage projects and effectively communicate about well-defined engineering activities.

PO7: Life-long learning: Ability to analyse individual needs and engage in updating in the context of technological changes.

Practical- Course Outcome matrix

Course Outcomes (COs)

CO1 - Write C++ programs using classes and objects.
CO2 - Develop C++ programs using constructors.
CO3 - Implement Inheritance in C++.
CO4 - Implement Polymorphism in C++.
CO5 - Develop C++ programs to perform file operations.

Sr. No.	Practical Outcome	co1	co2	co3	co4	co5
1.	*Write programs to evaluate any expression using Input / Output functions.	√				
2.	*Write programs using- • Scope resolution operator • Memory management operator • Manipulators	√				
3.	Write programs to demonstrate use of- • Implicit type casting • Explicit type casting	√				
4.	Write programs to show use of classes and objects to define the function inside the class	√				
5.	*Write programs to define the function outside the class.	√				
6.	*Write programs to implement inline function.		√			
7.	*Write programs to implement friend function using- • Two different classes • External function		√			
8.	*Write programs to implement- • Static data member • Static member function		√			
9.	*Write programs to create array of objects.		√			
10.	*Write programs for- • Default constructor • Parameterized constructor • Copy constructor • Multiple constructor in one class		√			
11.	Write programs using- • Single level inheritance • Multilevel inheritance			√		

12.	*Write programs to implement multiple inheritance.			√		
13.	Write programs to implement hierarchical inheritance.			√		
14.	*Write programs to implement virtual base class.			√		
15.	Write programs which show the use of constructors in derived class.			√		
16.	*Write programs to implement- • Pointer to object • 'this' pointer				√	
17.	*Write programs for- • Pointer to derived class in single inheritance • Pointer to derived class in multilevel inheritance				√	
18.	Write programs which show the use of function overloading.				√	
19.	*Write programs to overload unary operator using- • Member function • Friend function				√	
20.	Write programs to overload binary operator using- • Member function • Friend function				√	
21.	*Write programs to implement virtual function.				√	
22.	Write programs to implement pure virtual function.				√	
23.	*Write programs to read and write from/to file using- • Constructor • open()					√
24.	*Write programs to copy the content of one file into another file using formatted input/output functions.					√
25.	Write file programs to implement sequential input and output operations on file.					√
26.	Write programs to perform input / output operations on binary files.					√

'' Marked Practical (LLOs) Are mandatory.*

Industry / Employer Expected Outcome

The aim of this course is to help the student to attain the student to attain the following industry identified outcomes through various teaching learning experiences:

- Develop 'C++' programs using classes and objects.
- Develop an application by implementing Inheritance.
- Develop an application by using Polymorphism.
- Use appropriate File handling operations for developing applications.

Guidelines to Teachers

1. There will be two sheets of blank pages after every practical for the student to report other matters (if any), which is not mentioned in the printed practicals.
2. For difficult practicals if required, teacher could provide the demonstration of the practical emphasizing of the skills which the student should achieve.
3. Teachers should give opportunity to students for hands-on after the demonstration.
4. Assess the skill achievement of the students and COs of each unit.
5. One or two questions ought to be added in each practical for different batches. For these teachers can maintain various practical related question bank for each course.
6. For effective implementation and attainment of practical outcomes, teacher ought to ensure that in the beginning itself of each practical, students must read through the complete write- up of that practical sheet.
7. During practical, ensure that each student gets chance and takes active part in taking observations/ readings and performing practical.
8. Teacher ought to assess the performance of students continuously according to the MSBTE guidelines.

Instructions for Students

Note: Kindly do add specific instructions for students for effective implementation of practical' s depending upon your course, if needed.

1. For incidental writing on the day of each practical session every student should maintain a ***dated log book*** for the whole semester, apart from this laboratory manual which has to ***submit for assessment to the teacher*** in the next practical session.
2. For effective implementation and attainment of practical outcomes, in the beginning itself of each practical, students need to read through the complete write-up including the practical related questions and assessment scheme of that practical sheet.
3. Student ought to refer the reference books, lab manuals, etc.
4. Student should not hesitate to ask any difficulties they face during the conduct of practical' s.

Content Page

List of Practical's and Progressive Assessment Sheet

Sr. No	Practical Outcome	Page No.	Date of performance	Date of submission	Assess-ment marks(25)	Dated sign. of teacher	Remarks (if any)
13	Write programs to implement hierarchical inheritance.	68					
14	*Write programs to implement virtual base class.	75					
15	Write programs which show the use of constructors in derived class.	81					
16	*Write programs to implement- • Pointer to object • 'this' pointer	88					
17	*Write programs for- • Pointer to derived class in single inheritance • Pointer to derived class in multilevel inheritance	95					
18	Write programs which show the use of function overloading.	101					
19	*Write programs to overload unary operator using- • Member function • Friend function	107					
20	Write programs to overload binary operator using- • Member function • Friend function	115					
21	*Write programs to implement virtual function.	122					
22	Write programs to implement pure virtual function.	129					
23	*Write programs to read and write from/to file using- • Constructor • Open ()	135					
24	*Write programs to copy the content of one file into another file using formatted input/output functions.	141					
25	Write file programs to implement sequential input and output operations on file.	145					
26	Write programs to perform input / output operations on binary files.	150					
	Total						

To be transferred to Proforma ofCIAAN-2017.

Practical No.1: Write programs to evaluate any expression using Input/Output functions

I Practical Significance:

This practical is useful for students to evaluate expressions using different types of operators and Input/output functions. Students will be able to evaluate any type of expressions in the C++ program.

II Industry/Employer Expected Outcome(s)

This practical is expected to develop the following skills as:
1. Develop C++ program using input-output functions.
2. Develop C++ program to evaluate any expressions using various operators.

III Course Level Learning Outcome(s)

Write C++ programs using classes and objects.

IV Laboratory Learning Outcome(s)

Develop program to evaluate expressions using various operators and Input/output functions.

V Relevant Affective Domain related outcome(s)

1. Select proper programming environment in C++.
2. Follow ethical practices.

VI Relevant Theoretical Background

Operators in C++:

An **operator** is a symbol that operates on a value to perform specific mathematical or logical computations. They form the foundation of any programming language. In C++, we have built-in operators to provide the required functionality.

An operator operates the **operands**. For example,

```
int c = a + b;
```

Here, '+' is the addition operator. 'a' and 'b' are the operands that are being 'added'.

Different types of operators are:

1. Arithmetic Operators
2. Relational Operators
3. Logical Operators
4. Bitwise Operators
5. Assignment Operators
6. Ternary or Conditional Operators

C++ Basic Input/Output:

C++ I/O operation is using the stream concept. Stream is the sequence of bytes or flow of data. It makes the performance fast. If bytes flow from main memory to device like printer, display screen, or a network connection, etc, this is called as **output operation**.

If bytes flow from device like printer, display screen, or a network connection, etc to main memory, this is called as **input operation**.

Header File	Function and Description
<iostream>	It is used to define the cout, cin and cerr objects, which correspond to standard output stream, standard input stream and standard error stream, respectively.
<iomanip>	It is used to declare services useful for performing formatted I/O, such as setprecision and setw.
<fstream>	It is used to declare services for user-controlled file processing.

Standard output stream (cout)

The cout is a predefined object of ostream class.

It is connected with the standard output device, which is usually a display screen. The cout is used in conjunction with stream insertion operator (<<) to display the output on a console.

Standard input stream (cin)

The cin is a predefined object of istream class.

It is connected with the standard input device, which is usually a keyboard.

The cin is used in conjunction with stream extraction operator (>>) to read the input from a console.

Standard end line (endl)

The endl is a predefined object of ostream class.
It is used to insert a new line character and flushes the stream

VII Resources Required

Sr. No	Name Resource	Specification	Quantity	Remarks
1	Computer System	Any desktop or laptop computer with basic configuration	One computer system for each student	
2	Operating system	Windows /LINUX	One for each computer system	
3	Software	Turbo C++ Version 3.0 or any other	One for each computer system	

VIII Precautions to be followed

1. Handle computer system and peripherals with care.
2. Follow safety practices.

IX Exercise:

1. Write a C++ code to evaluate following expression using input output function:
   y= 5*x-5 where value of x is taken from user. Find the value of y.
2. Write a program to print **"hi"** msg if entered value is more than 10 otherwise print **"bye"** msg on output screen. (Use of Relational operator)
3. Write a program to print largest number among two number using Conditional operator.

X C++ code:

Write "C++" Code for above exercise on the blank pages attached at the end of practical.

XI Resources Used

Sr. No.	Name of Resource	Suggested Broad Specification	Quantity

XII Result(s)

………

………..

XIII Conclusion

………

………

XIV Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

(Note: for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. Write a C++ program to evaluate the following expressions:
   X=(-b-(b^2-4ac))/2a

2. Complete the following table:

Program Code	Write & justify Output

a) #include<iostream.h>
#define PI 3.14159
int main ()
{
float r = 2;
float circle;
circle= 2 *PI* r;
cout << circle;
return 0;
}	

b) #include <iostream>
int main()
{
int x;
x=(3/2) + 2;
cout<<"Value of x is : "<<x;
return 0;
}	

c) #include <iostream>
int main()
{
int x;
int y=9;
x=y +int(10.0);
cout<<"Value of x : "<<x;
return 0;
}	

(Space for Answers)

XV References/Suggestions for further reading

1. https://www.geeksforgeeks.org/program-evaluate-simple-expressions
2. https://www.javatpoint.com/cpp-expression

XVI Assessment Scheme

	Performance Indicators	**Weightage**
	Process Related: 30 Marks	60 %
1	Logic formation	10%
2	Appropriate use of arithmetic expressions, operators and basic Input /output function.	20%
3	Debugging ability	20%
4	Follow ethical practices.	10%
	Product Related: 20 Marks	**40%**
5	Expected Output	20%
6	Submitting the Manual in time	10%
7	Answer to sample questions	10%
	Total (50 Marks)	**100 %**

Marks Obtained			**Dated signature of Teacher**
Process Related (30)	**Product Related (20)**	**Total (50)**	

Practical No.2: Write programs using-

- **Scope resolution operator**
- **Memory management operator**
- **Manipulators**

I Practical Significance:

This practical is useful for students to access the global variable, creation of dynamic memory allocation and displaying output in proper format. Students will be able to use different C++ operators in programming.

II Industry/Employer Expected Outcome(s)

This practical is expected to develop the following skills as:

1. Develop C++ program using Scope resolution operator.
2. Develop C++ program to using memory management operators.
3. Develop C++ program to display the output in proper format using different manipulators.

III Course Level Learning Outcome(s)

Write C++ programs using classes and objects.

IV Laboratory Learning Outcome(s)

Develop C++ program using special type of operators.

V Relevant Affective Domain related outcome(s)

1. Select proper programming environment in C++.
2. Follow ethical practices.

VI Relevant Theoretical Background

1. Scope resolution Operator:

The scope resolution operator is used to reference the global variable or member function that is out of scope. Therefore, we use the scope resolution operator to access the hidden variable or function of a program.

The operator is represented as the double colon **:: symbol.**

For example, when the **global and local variable or function** has the same name in a program, and when we call the variable, by default it only accesses the **inner or local variable** without calling the global variable. In this way, it hides the global variable or function. To overcome this situation, we use the scope resolution operator to fetch a program's hidden variable or function.

Uses of the scope resolution Operator

- It is used to access the hidden variables or member functions of a program.
- It defines the member function outside of the class using the scope resolution.
- It is used to access the static variable and static function of a class.
- The scope resolution operator is used to override function in the Inheritance.

2. Memory Management Operator:

Memory management is a process of managing computer memory, assigning the memory space to the programs to improve the overall system performance.

C++ a defines unary operators such as new and delete to perform the tasks, i.e., allocating and freeing the memory.

- **New operator**

A new operator is used to create the object while a delete operator is used to delete the object. When the object is created by using the new operator, then the object will exist until we explicitly use the delete operator to delete the object. Therefore, we can say that the lifetime of the object is not related to the block structure of the program

pointer_variable = new data-type

The above syntax is used to create the object using the new operator.

In the above syntax, **'pointer_variable'** is the name of the pointer variable, **'new' is the operator**, and **'data-type' defines the type of the data**.

Example 1:

```
int *p;
p = new int;
```

In the above example, **'p' is a pointer of type int.**

- **Delete operator**

When memory is no longer required, then it needs to be deallocated so that the memory can be used for another purpose. This can be achieved by using the delete operator, as shown below:

delete pointer_variable;

In the above statement, 'delete' is the operator used to delete the existing object, and 'pointer_variable' is the name of the pointer variable.

In the previous case, we have created pointer 'p' by using the new operator, and can be deleted by using the following statements:

delete p;

3. Manipulators:

Manipulators are helping functions that can modify the input/output stream. It does not mean that we change the value of a variable, it only modifies the I/O stream using insertion (<<) and extraction (>>) operators.

Manipulators are special functions that can be included in the I/O statement to alter the format parameters of a stream. Manipulators are operators that are used to format the data display. To access manipulators, the file **iomanip.h** should be included in the program.

- **endl C++ Manipulator:**

The word 'endl' in C++, a programming language, stands for end of line. Furthermore, the use of the endl C++ manipulators takes place to move the cursor to the beginning of the next line. Moreover, its working is similar to the '\n' escape sequence.

cout<<"object"<<endl<<"Oriented"<<endl<<"Programming";

- **setw Manipulator:**

The word 'setw' in C++ stands for set width. Furthermore, the use of the setw C++ manipulator takes place to set the output's field width on the output device. Moreover, by default, the displaying or printing of the output takes place right-justified within the specified field of setw C++ manipulators.

The use of the setw manipulator takes place to set the width of the output in a program. Furthermore, it takes up an argument 'n', the width of the field in which the displaying of the output is to take place. Moreover, the output in the field, by default, is right-aligned.

The general syntax of **setw** manipulator is as follows:

setw(n)

The **'n'** indicates the field width that is the number of columns and it happens to be an integer value. Also, if the specified field width is less in comparison to the width of the output data's width that is to be displayed, then the effect of setw manipulator will not take place. Moreover, there will be a normal displaying or printing of the output.

The **setw** C++ manipulator is also a component of **"iomanip.h"** header file. Furthermore, the inclusion of this header file must take place in the program to use **setw** manipulator.

Some important manipulators in <iomanip> are:

- setw (val): It is used to set the field width in output operations.
- setfill (c): It is used to fill the character 'c' on output stream.
- setprecision (val): It sets val as the new value for the precision of floating-point values.
- setbase(val): It is used to set the numeric base value for numeric values.
- setiosflags(flag): It is used to set the format flags specified by parameter mask.
- resetiosflags(m): It is used to reset the format flags specified by parameter mask.

X Resources Required

Sr. No	Name Resource	Specification	Quantity	Remarks
1	Computer System	Any desktop or laptop computer with basic configuration	One computer system for each student	
2	Operating system	Windows /LINUX	One for each computer system	
3	Software	Turbo C++ Version 3.0 or any other	One for each computer system	

XI Precautions to be followed

1. Handle computer system and peripherals with care.
2. Follow safety practices.

XII Exercise:

1. Write a C++ Program which show the use of function outside the class using scope resolution operator.
2. Write a program to display the massage **"Welcome to the world of C++"** using manipulators.
3. Write a program to create the memory using new operator and free the created memory using delete operator.

X C++ code:

Write "C++" Code for above exercise on the blank pages attached at the end of practical.

XI Resources Used

Sr. No.	Name of Resource	Suggested Broad Specification	Quantity

XII Result(s)

…………………………………………………………………………………………………..

....

....

XIII Conclusion

……………………………………………………………………………………………

……………………………………………………………………………………………

XIV Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

(Note: for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. Write a C++ program to access the global variable using scope resolution operator.

2. Format the following statement using manipulators.

   (Rate=412345 period=35 year=2024)

3. Complete the following table:

Program Code	**Write & justify Output**

a) #include<iostream>
 	
int x; // Global x
 	
int main()
{
int x = 10; // Local x
cout << "Value of global x is " << ::x;
cout << "\nValue of local x is " << x;
return 0;
}	

b. #include <iostream>
 	
class A {
public:
// Only declaration
void fun();
};
 	
//Definition outside class using ::
void A::fun() {
cout << "fun() called"; }
 	
int main()
{
A a;
a.fun();
return 0;
}	

c. #include <iostream>
#include <iomanip>
 	
int main (void)
{
int a,b;
a = 200;
b = 300;	

cout << setw (5) << a << setw (5) << b <<
endl;
 cout << setw (6) << a << setw (6) << b <<
endl;
  cout << setw (7) << a << setw (7) << b
<< endl;
  cout << setw (8) << a << setw (8) << b
<< endl;
return 0;
}	

(Space for Answers)

XV References/Suggestions for further reading

1. https://www.geeksforgeeks.org/scope-resolution-operator-in-

2.https://www.javatpoint.com/cpp-manipulator-setw-function

XVI Assessment Scheme

	Performance Indicators	**Weightage**
	Process Related: 30 Marks	60 %
1	Logic formation	10%
2	Appropriate use of arithmetic expressions, operators and basic Input /output function.	20%
3	Debugging ability	20%
4	Follow ethical practices.	10%
	Product Related: 20 Marks	**40%**
5	Expected Output	20%
6	Submitting the Manual in time	10%
7	Answer to sample questions	10%
	Total (50 Marks)	**100 %**

Marks Obtained			**Dated signature of Teacher**
Process Related (30)	**Product Related (20)**	**Total (50)**	

Practical No.3: Write programs to demonstrate use of

- **Implicit type casting**
- **Explicit type casting**

I Practical Significance:

This practical is useful for students to get accurate result while solving any expressions. Student will be able to use different types of type casting.

II Industry/Employer Expected Outcome(s)

This practical is expected to develop the following skills as:
1. Develop C++ program using implicit type casting
2. Develop C++ program using explicit type casting.

III Course Level Learning Outcome(s)

Write C++ programs using classes and objects.

IV Laboratory Learning Outcome(s)

Develop programs to implement type casting.

V Relevant Affective Domain related outcome(s)

1. Select proper programming environment in C++.
2. Follow ethical practices.

VI Relevant Theoretical Background

A type cast is basically a conversion from one type to another. There are two types of type conversion:

- Implicit Type Conversion.
- Explicit Type Conversion.

1. Implicit Type Conversion Also known as 'automatic type conversion'. Done by the compiler on its own, without any external trigger from the user. Generally takes place when in an expression more than one data type is present. In such condition type conversion (type promotion) takes place to avoid loss of data.

All the data types of the variables are upgraded to the data type of the variable with largest data type.

- bool -> char -> short int -> int ->
- unsigned int -> long -> unsigned ->
- long long -> float -> double -> long double

2. Explicit Type Conversion: This process is also called type casting and it is user-defined. Here the user can typecast the result to make it of a particular data type.

In C++, it can be done by two ways:

Converting by assignment: This is done by explicitly defining the required type in front of the expression in parenthesis. This can be also considered as forceful casting.

Syntax:

(type) expression;

VII Resources Required

Sr. No	**Name Resource**	**Specification**	**Quantity**	**Remarks**
1	Computer System	Any desktop or laptop computer with basic configuration	One computer system for each student	
2	Operating system	Windows /LINUX	One for each computer system	
3	Software	Turbo C++ Version 3.0 or any other	One for each computer system	

VIII Precautions to be followed

1. Handle computer system and peripherals with care.
2. Follow safety practices.

IX Exercise:

1. Find the area of the rectangle by casting double data into float and int type.
2. Write a program which show the use of implicit type casting to calculate the average of two number

X C++ code:

Write "C++" Code for above exercise on the blank pages attached at the end of practical.

XI Resources Used

Sr. No.	**Name of Resource**	**Suggested Broad Specification**	**Quantity**

XII Result(s)

………………………………………………………………………………………………..

………………………………………………………………………………………………..

…..………………………………………………………………………………………………

XIII Conclusion

……………………………………………………………………………………………………

……………………………………………………………………………………………………

XIV Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

(Note: for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. Calculate average of two numbers using explicit type casting

2. Write a program which display the percentage of students which accept marks of three subjects from user.(Show the use of Implicit type casting)

3.Complete the following table:

Program Code	**Write & justify Output**

a) #include <iostream>
 	
int main ()
{
    // declaration of the variables
    int a, b;
    float res;
    a = 21;
    b = 5;
    cout << " Implicit Type Casting: " << endl;
    cout << " Result: " << a / b << endl; // it loses some information
 	
    cout << " \n Explicit Type Casting: " << endl;
    // use cast () operator to convert int data to float
    res = (float) 21 / 5;
    cout << " The value of float variable (res): " << res << endl;
 	
    return 0;
}	

b) #include <iostream>
using namespace std;
int main ()
{
    short x = 200;
    int y;
    y = x;
    cout << " Implicit Type Casting " << endl;
    cout << " The value of x: " << x << endl;
    cout << " The value of y: " << y << endl;
 	
    int num = 20;
    char ch = 'a';	

int res = 20 + 'a';
   cout << " Type casting char to int data type ('a' to 20): " << res << endl;
 	
   float val = num + 'A';
   cout << " Type casting from int data to float type: " << val << endl;
   return 0;
}	

c)#include <iostream>
using namespace std;
 	
int main() {
 	
   int num_int;
   double num_double = 9.99;
 	
   // implicit conversion
   // assigning a double value to an int variable
   num_int = num_double;
 	
   cout << "num_int = " << num_int << endl;
   cout << "num_double = " << num_double << endl;
 	
   return 0;
}	

(Space for Answers)

XV References/Suggestions for further reading

1. https://www.javatpoint.com/type-casting-in-cpp
2. https://www.programiz.com/cpp-programming/type

XVI Assessment Scheme

	Performance Indicators	**Weightage**
	Process Related: 30 Marks	60 %
1	Logic formation	10%
2	Appropriate use of arithmetic expressions, operators and basic Input /output function.	20%
3	Debugging ability	20%
4	Follow ethical practices.	10%
	Product Related: 20 Marks	**40%**
5	Expected Output	20%
6	Submitting the Manual in time	10%
7	Answer to sample questions	10%
	Total (50 Marks)	**100 %**

Marks Obtained			**Dated signature of Teacher**
Process Related (30)	**Product Related (20)**	**Total (50)**	

Practical No.4: Write programs to show use of classes and objects to define the function inside the class

I Practical Significance:

The classes and objects help to represent real life entity with different attributes and related Member functions.

II Industry/Employer Expected Outcome(s)

This practical is expected to develop the following skills as:

i. Define and use classes and objects.
ii. Define the member function inside the class.

III Course Level Learning Outcome(s)

Write C++ programs using classes and objects.

IV Laboratory Learning Outcome(s)

Implement classes and objects to define the function inside class.

V Relevant Affective Domain related outcome(s)

1. Select proper programming environment in C++.
2. Follow ethical practices.

VI Relevant Theoretical Background

- **Class:.** Class is a user defined data type, which holds its own data members and associated member functions, which can be accessed and used by creating an instance of that class.

- **Object: 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.

```
 class  ClassName
{
Access specifier: Data members;              //can be private, public or protected
 Member     functions()                      // Variables to be used
                                             //Methods to access data members

  {

      I/body of the function
  }
 }; //Class ends with semicolon
```

Declaring Objects: When a class is defined, only the specification for the object is defined; no memory or storage is allocated. To use the data and access functions defined in the class, you need to create objects.

Syntax:
className ObjectName;
eg.
Student std;

Accessing data members and member functions: The data members and member functions of class can be accessed using the dot('.') operator with the object.

For example if the name of object is std and you want to access the member function with the name getData() then you will have to write std.getData() .

Member Functions in Classes: There are two ways to define functions that belongs to a class:

- Inside class definition
- Outside class definition

Define Member Function Inside the class:

Syntax: class className

```
{
 //other members declaration
 return type function_Name(list of parameters)
 {
 II body of function
 }
 //other members declaration
 };
```

Class Methods: A member function of a class can also be defined inside the class. However, when a member function is defined inside the class, the class name and the scope resolution operator are not specified in the function header. Moreover, the member functions defined inside a class definition are by default inline functions called.
In the following example, we define a function inside the class, and we name it **"myMethod".**

Note: You access methods just like you access attributes; by creating an object of the class and using the **dot syntax (.):**

Inside Example

```
class MyClass
{       // The class
        public:            // Access specifier
        void myMethod()
        {
        // Method/function defined inside the class
        cout << "Hello World!";
        }
};
```

```
int main()
{
MyClass myObj;     // Create an object of MyClass
myObj.myMethod();  // Call the method
return 0;
}
```

VII Resources Required

Sr. No	Name Resource	Specification	Quantity	Remarks
1	Computer System	Any desktop or laptop computer with basic configuration	One computer system for each student	
2	Operating system	Windows /LINUX	One for each computer system	
3	Software	Turbo C++ Version 3.0 or any other	One for each computer system	

VIII Precautions to be followed

1. Handle computer system and peripherals with care.
2. Follow safety practices.

IX Exercise:

1. Write syntax for define member function inside the class.
2. Write a program to define a class student having data members name and roll no. Accept and
   display data for one object. Define the member function inside the class.
3. Write a program to calculate square of number in which define class and define member function inside the class as per requirement.
4. Write a program to define a class student having data members name and roll no. Accept and display data for one object. Define the member function inside the c lass
5. What is the difference between struct and class in C++?

X C++ code:

Write "C++" Code for above exercise on the blank pages attached at the end of practical.

XI Resources Used

Sr. No.	Name of Resource	Suggested Broad Specification	Quantity

XII Result(s)

…………………………………………………………………………………………………..

..……………………………………………………………………………………………

..……………………………………………………………………………………………

....

....

XIII Conclusion

………………………………………………………………………………………………

………………………………………………………………………………………………

XIV Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

(Note: for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. Define a class Room with data members length, breadth and height. Member function calculate_area () and calculate_volume(). Calculate the area and volume of room. Define the member function inside the class.

2. Define a class mean in which assign two numbers in assign member function (i.e. assign(4,8)) passed value from main function and define assign member function inside the class and display the mean of two number on output screen.

3. Complete the following table:

Program Code	**Write & justify Output**

a) #include <iostream>
 	
class Student {
   public:
      int id;//data member (also instance variable)
      string name;//data member(also instance variable)
};
int main() {
    Student s1; //creating an object of Student
    s1.id = 201;
    s1.name = "Sonoo Jaiswal";
    cout<<s1.id<<endl;
    cout<<s1.name<<endl;
    return 0;
}	

```
b. #include <iostream>
class Student {
  public:
      int id;//data member (also instance variable)
      string name;//data member(also instance variable)
      void insert(int i, string n)
       {
          id = i;
          name = n;
       }
      void display()
       {
          cout<<id<<"  "<<name<<endl;
       }
};
int main(void) {
    Student s1; //creating an object of Student
    Student s2; //creating an object of Student
    s1.insert(201, "Sonoo");
    s2.insert(202, "Nakul");
    s1.display();
    s2.display();
    return 0;
}
```

(Space for Answers)

XV References/Suggestions for further reading

1. https://www.javatpoint.com/cpp-object-and-class2.
2. https://www.w3schools.com/cpp

XVI Assessment Scheme

Performance Indicators		Weightage
Process Related: 30 Marks		60 %
1	Logic formation	10%
2	Appropriate use of arithmetic expressions, operators and basic Input /output function.	20%
3	Debugging ability	20%
4	Follow ethical practices.	10%
Product Related: 20 Marks		**40%**
5	Expected Output	20%
6	Submitting the Manual in time	10%
7	Answer to sample questions	10%
Total (50 Marks)		**100 %**

Marks Obtained			**Dated signature of Teacher**
Process Related (30)	**Product Related (20)**	**Total (50)**	

Practical No.5: Write programs to define the function outside the class

I Practical Significance:

The classes and objects help to represent real life entity with different attributes and related Member functions.

II Industry/Employer Expected Outcome(s)

This practical is expected to develop the following skills as:

i. Define and use classes and objects.
ii. Define the member function Outside the class.
iii. Use of scope resolution operator.

III Course Level Learning Outcome(s)

Write C++ programs using classes and objects.

IV Laboratory Learning Outcome(s)

Implement classes and objects to define the function outside class..

V Relevant Affective Domain related outcome(s)

1. Select proper programming environment in C++.
2. Follow ethical practices.

VI Relevant Theoretical Background

Member Functions in Classes: There are two ways to define functions that belongs to a class:

- Inside class definition
- Outside class definition

Define Member Function Outside class definition:

Syntax: return type className :: functionName(list of parameters)

```
{
        //body of function

}
MyClass myObj;     // Create an object of MyClass
myObj.myMethod();  // Call the method
        return 0;
        }
```

Example:

```
1.#include<iostream.h>
class Class_name{

    public:

    return_type Method_name(); // method outside class definition

    };

    // Outside the Class using scope resolution operator

    return_type Class_name :: Method_name() {

      // body of member function

    }
```

```
2. #include <iostream>

    class Sq {
    public:
        int a;
        int square(); // Declaring function square with no argument and having
              return type 'int'.
    };

    int Sq::square()
    {
        return a * a;
    }
```

VII Resources Required

Sr. No	Name Resource	Specification	Quantity	Remarks
1	Computer System	Any desktop or laptop computer with basic configuration	One computer system for each student	
2	Operating system	Windows /LINUX	One for each computer system	
3	Software	Turbo C++ Version 3.0 or any other	One for each computer system	

VIII Precautions to be followed

1. Handle computer system and peripherals with care.
2. Follow safety practices.

IX Exercise:

1. Write syntax for define member function Outside the class.
2. Write a C++ program to declare a class "staff' having data members name, basic salary, DA, HRA and calculate gross salary. Accept and display data for one staff.

Where DA=74.5% of basic

1. HRA= 30% of basic.

ii. Gross_salary=basic+HRA+DA

X C++ code:

Write "C++" Code for above exercise on the blank pages attached at the end of practical.

XI Resources Used

Sr. No.	Name of Resource	Suggested Broad Specification	Quantity

XII Result(s)

……
……
……

XIII Conclusion

……
……

XIV Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

(Note: for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. Write a program to find area of circle such that the class circle must have three functions namely:

a)read() to accept the radius from user.

b)compute() for calculating the area

c)display() for displaying the result.(Use Scope resolution operator)

2. Define a class complex with data members real and imaginary, member function read() and write(). Write a program to perform the addition of two complex number and display the result.

3. Complete the following table:

Program Code	**Write & justify Output**

a) #include <iostream>
 	
struct X {
  int a, b ;// member function declaration
  int add();
};
 	
// global variable
int a = 10;
 	
// define member function outside class
int X::add() { return a + b; }
 	
int main() {
  int answer;
  X xobject;
  xobject.a = 1;
  xobject.b = 2;
  answer = xobject.add();
  cout << xobject.a << " + " << xobject.b << " = " << answer << endl;
}	

b. #include <iostream>
// Declaration of the class
class MyClass {
public:
   void myFunction(); // Function prototype declaration inside the class
};
 	
// Definition of the function outside the class
void MyClass::myFunction() {
   std::cout << "This is myFunction() defined outside the class." << std::endl;
}
 	
int main() {
   MyClass obj;
   obj.myFunction(); // Call the function
   return 0;
}	

(Space for Answers)

XV References/Suggestions for further reading

1. https://www.geeksforgeeks.org/cpp-class-methods/
2. https://www.ibm.com/docs/en/i/7.3?topic=only-memberscope-c

XVI Assessment Scheme

	Performance Indicators	**Weightage**
	Process Related: 30 Marks	60 %
1	Logic formation	10%
2	Appropriate use of arithmetic expressions, operators and basic Input /output function.	20%
3	Debugging ability	20%
4	Follow ethical practices.	10%
	Product Related: 20 Marks	**40%**
5	Expected Output	20%
6	Submitting the Manual in time	10%
7	Answer to sample questions	10%
	Total (50 Marks)	**100 %**

Marks Obtained			**Dated signature of Teacher**
Process Related (30)	**Product Related (20)**	**Total (50)**	

Practical No.6: Write programs to implement Inline functions.

I Practical Significance:

The classes and objects help to represent real life entity with different attributes and related member functions. The use of inline functions facilitates faster execution of the program

II Industry/Employer Expected Outcome(s)

An inline function that reduces the execution time of a program

III Course Level Learning Outcome(s)

Develop C++ programs using classes and objects.

IV Laboratory Learning Outcome(s)

Implement programs using Inline functions

V Relevant Affective Domain related outcome(s)

1. Select proper programming environment in C++.
2. Follow ethical practices.

VI Relevant Theoretical Background

Inline Function: C++ inline function is concept that is commonly used with classes. If a function is inline, the compiler places a copy of the code of that function at each point where the function is called at compile time.
A function definition in a class definition is an inline function definition, even without the use of the **inline** keyword.

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.
Declaring inline function:

Syntax:

```
class className
{
        //other members declaration public:
                inline return type FunctionName(list of parameters)
                {
                //body of inline function
                }
        //other members declaration
};
```

VII Resources Required

Sr. No	Name Resource	Specification	Quantity	Remarks
1	Computer System	Any desktop or laptop computer with basic configuration	One computer system for each student	
2	Operating system	Windows /LINUX	One for each computer system	
3	Software	Turbo C++ Version 3.0 or any other	One for each computer system	

VIII Precautions to be followed

1. Handle computer system and peripherals with care.
2. Follow safety practices.

IX Exercise:

1. Write a C++ program to create a class "Number" having data members n1 and n2 and perform mathematical operations like addition, subtraction, multiplication and division on two numbers using inline functions.

X C++ code:

1. Write “C++” Code for above exercise on the blank pages attached at the end of practical.

XI Resources Used

Sr. No.	Name of Resource	Suggested Broad Specification	Quantity

XII Result(s)

……………………………………………………………………………………………………..

..………………………………………………………………………………………………………
…..…………………………………………………………………………………………………

XIII Conclusion

……

……

XIV Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

(Note: for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. Write a C++ program to calculate area of Rectangle using Inline function.
2. Complete the following table:

Program Code	Write & justify Output

a) #include <iostream>
using namespace std;
 	
inline int exp(int x, int y, int z)
{
    return (x + y) * z;
}
int main()
{
    cout << exp(4,5,7) << endl;
    cout << exp(4,5,6) << endl;
    cout << exp(4,7,5) << endl;
    cout << exp(7,4,6) << endl;
    return 0;
}	

b)#include <iostream>
using namespace std;
 	
class Sample {
public:
  inline void myfun();
};
void inline Sample::myfun()
{
  cout << "Hello from inline function";
}
 	
int main()
{
  Sample S;
 	
  S.myfun();
  return 0;
}	

(Space for Answers)

XV References/Suggestions for further reading

1. https://www.tutorialspoint.com/cplusplus/cpp_inline_functions.htm
2. https://www.codesdope.com/cpp-inline-function/

XVI Assessment Scheme

Performance Indicators		Weightage
Process Related: 30 Marks		60 %
1	Logic formation	10%
2	Appropriate use of arithmetic expressions, operators and inline function.	20%
3	Debugging ability	20%
4	Follow ethical practices.	10%
Product Related: 20 Marks		**40%**
5	Expected Output	20%
6	Submitting the Manual in time	10%
7	Answer to sample questions	10%
Total (50 Marks)		**100 %**

Marks Obtained			**Dated signature of Teacher**
Process Related (30)	**Product Related (20)**	**Total (50)**	

Practical No.7: Write program to Implement Friend Function using

- **Two different classes**
- **External Function**

I Practical Significance:

The classes and objects help to represent real life entity with different attributes and related member functions.
The use of friend function allows non-member functions to access the private and protected data of the class.

II Industry/Employer Expected Outcome(s)

Friend function is widely used in cases when two or more classes contain the interrelated members relative to other parts of the program.

III Course Level Learning Outcome(s)

Develop C++ programs using constructors.

IV Laboratory Learning Outcome(s)

Implement programs using Friend functions

V Relevant Affective Domain related outcome(s)

1. Select proper programming environment in C++.
2. Follow ethical practices.

VI Relevant Theoretical Background

- **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. **friend** keyword is used to declare the function as friend of class.
- Declaring friend function
  Syntax:

```
class className
{
        //other members declaration
        public:
                **friend** return type friendFunctionName(list of parameters);
        //other members declaration
};
return type friendFunctionName(list of parameters)
{
        //body of friend function
}
```

VII Resources Required

Sr. No	Name Resource	Specification	Quantity	Remarks
1	Computer System	Any desktop or laptop computer with basic configuration	One computer system for each student	
2	Operating system	Windows /LINUX	One for each computer system	
3	Software	Turbo C++ Version 3.0 or any other	One for each computer system	

VIII Precautions to be followed

a) Handle computer system and peripherals with care.
b) Follow safety practices.

IX Exercise:

1. Write a C++ program to exchange the values of two variables using friend function.

2. WAP to create two classes test1 and test2 which stores marks of a student. Read value for class objects and calculate average of two tests using friend function.

X C++ code:

Write "C++" Code for above exercise on the blank pages attached at the end of practical.

XI Resources Used

Sr. No.	Name of Resource	Suggested Broad Specification	Quantity

XII Result(s)

…………………………………………………………………………………………………..

..…………………………………………………………………………………………………

..…………………………………………………………………………………………………

XIII Conclusion

…………………………………………………………………………………………………

…………………………………………………………………………………………………

XIV Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

(Note: for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. WAP to declare a class calculation. Display addition, subtraction, multiplication, division of two numbers. Use friend function.

2. Complete the following table:

Program Code	**Write & justify Output**

a) How many member functions are there in this C++ class excluding constructors and destructors?
`class Box`
`{`
`        int capacity;`
`   public:`
`        void print();`
`        friend void show();`
`        bool compare();`
`        friend bool lost();`
`};`	

b) What will be the output of the following C++ code?
`#include <iostream>`
`#include <string>`
`using namespace std;`
`class Box`
`{`
`int capacity;`
`   public:`	

```
                Box(int cap){
                        capacity = cap;
                }

                friend void show();
                };

                void show()
                {
                Box b(10);
                cout<<"Value   of   capacity
                is: "<<b.capacity<<endl;
                }

int main(int argc, char const *argv[])
{
          show();
          return 0;
```

(Space for Answers)

XV References/Suggestions for further reading

1. https://www.javatpoint.com/cpp-friend-function
2. https://www.tutorialspoint.com/cplusplus/cpp_friend_functions.htm
3. https://www.geeksforgeeks.org/friend-class-function-cpp/

XVI Assessment Scheme

Performance Indicators		Weightage
Process Related: 30 Marks		60 %
1	Logic formation	10%
2	Appropriate use of arithmetic expressions, operators and inline function.	20%
3	Debugging ability	20%
4	Follow ethical practices.	10%
Product Related: 20 Marks		**40%**
5	Expected Output	20%
6	Submitting the Manual in time	10%
7	Answer to sample questions	10%
Total (50 Marks)		**100 %**

Marks Obtained			**Dated signature of Teacher**
Process Related (30)	**Product Related (20)**	**Total (50)**	

Practical No.8: Write program to Implement-

- **Static Data Member**
- **Static Member Function**

I Practical Significance:

A typical use of static members is for recording data common to all objects of a class. Static data member as a counter to store the number of objects of a particular class type that are created.

II Industry/Employer Expected Outcome(s)

Static data member in C++ can be accessed anywhere in the program after the declaration of class either using the class instance or using scope resolution, class name, and variable name

III Course Level Learning Outcome(s)

Develop C++ programs using constructors.

IV Laboratory Learning Outcome(s)

Implement programs using Static Members and Static Member Functions

V Relevant Affective Domain related outcome(s)

1. Select proper programming environment in C++.
2. Follow ethical practices.

VI Relevant Theoretical Background

- Static data members are class members that are declared using static keywords. A static member has certain special characteristics which are as follows:

Only one copy of that member is created for the entire class and is shared by all the objects of that class, no matter how many objects are created.
It is initialized before any object of this class is created, even before the main starts.
It is visible only within the class, but its lifetime is the entire program.

- Declaring Static Data Member-

  Syntax:

  static data_type data_member_name;

- The static member functions are special functions used to access the static data members or other static member functions. A member function is defined using the static keyword. A static member function shares the single copy of the member function to any number of the class' objects. We can access the static member function using the class name or class' objects.

- Declaring Static Data Member-

  Syntax-

  class_name::function_name (parameter);

VII Resources Required

Sr. No	Name Resource	Specification	Quantity	Remarks
1	Computer System	Any desktop or laptop computer with basic configuration	One computer system for each student	
2	Operating system	Windows /LINUX	One for each computer system	
3	Software	Turbo C++ Version 3.0 or any other	One for each computer system	

VIII Precautions to be followed

1. Handle computer system and peripherals with care.
2. Follow safety practices.

IX Exercise:

1. Write a Program to define a class having data members principal, duration and rate of interest. Declare rate _of_ interest as static member variable .calculate the simple interest and display it.

X C++ code:

Write "C++" Code for above exercise on the blank pages attached at the end of practical.

XI Resources Used

Sr. No.	Name of Resource	Suggested Broad Specification	Quantity

XII Result(s)

……………………………………………………………………………………………………..

..……………………………………………………………………………………………………

..……………………………………………………………………………………………………

XIII Conclusion

………………………………………………………………………………………………………

………………………………………………………………………………………………………

XIV Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

(Note: for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. Write a Program to calculate weight of object at different planets using formula weight=m*g Where m=mass of object G=gravitational force Declare g as static member variable.

2. Complete the following table:

Program Code	**Write & justify Output**
a)Which is correct syntax to access the static member functions with class name?	

b)What will be the output of the following C++ code?
 	
class Test
{
private: static int x;
public: static void fun()
{
cout &lt;&lt; ++x &lt;&lt; " ";
}
};
int Test :: x =20;
void main()
{
Test x;
x.fun();
x.fun();
}	

c) What will be the output of the following C++ code?
class Test
{
public: Test()
{
cout &lt;&lt; "Test's Constructor is Called " &lt;&lt; endl;
}	

```
};

class Result
{
        static Test a;
        public:
        Result()
        {
                cout            &lt;&lt;
"Result's  Constructor  is  Called  "
&lt;&lt; endl;
        }
};

void main()
{
        Result b;
}
```

(Space for Answers)

XV References/Suggestions for further reading

1 https://www.javatpoint.com/static-member-function-in-cpp

2 https://www.geeksforgeeks.org/cpp-static-data-members/

XVI Assessment Scheme

	Performance Indicators	**Weightage**
	Process Related: 30 Marks	60 %
1	Logic formation	10%
2	Appropriate use of arithmetic expressions, operators and inline function.	20%
3	Debugging ability	20%
4	Follow ethical practices.	10%
	Product Related: 20 Marks	**40%**
5	Expected Output	20%
6	Submitting the Manual in time	10%
7	Answer to sample questions	10%
	Total (50 Marks)	**100 %**

Marks Obtained			**Dated signature of Teacher**
Process Related (30)	**Product Related (20)**	**Total (50)**	

Practical No.9: Write programs to create Array of Objects.

I Practical Significance:

The classes and objects help to represent real life entity with different attributes and related member functions.
Array of objects are used to represent the data of similar type.

II Industry/Employer Expected Outcome(s)

The array of objects represent storing multiple objects in a single name. Reduce the time and memory by storing the data in a single variable.

III Course Level Learning Outcome(s)

Develop C++ programs using constructors.

IV Laboratory Learning Outcome(s)

Write/ Compile/ debug/ Execute simple C++ program using classes and array of objects.

V Relevant Affective Domain related outcome(s)

1. Select proper programming environment in C++.
2. Follow ethical practices.

VI Relevant Theoretical Background

Array of objects: Arrays of variables of type "class" is known as "Array of objects".

Declaring Array of Objects: Syntax:

```
className  ObjectArrayName[size];
eg.
Student std[5];
```

Accessing data members and member functions: The data members and member functions of class can be accessed using the dot('.') operator with the object.
For example if the name of object is *std[2]* and you want to access the member function with the name *getDataO* then you will have to write *std[2].getData0*.

VII Resources Required

Sr. No	Name Resource	Specification	Quantity	Remarks
1	Computer System	Any desktop or laptop computer with basic configuration	One computer system for each student	
2	Operating system	Windows /LINUX	One for each computer system	
3	Software	Turbo C++ Version 3.0 or any other	One for each computer system	

VIII Precautions to be followed

1. Handle computer system and peripherals with care.
2. Follow safety practices.

IX Exercise:

Write a C++ program to declare class 'Account' having data members as Account_No and Balance. Accept this data for 10 accounts and display data of Accounts having balance greater than 10000.

X C++ code:

1 Write "C++" Code for above exercise on the blank pages attached at the end of practical.

XI Resources Used

Sr. No.	Name of Resource	Suggested Broad Specification	Quantity

XII Result(s)

……………………………………………………………………………………………………
……………………………………………………………………………………………………
……………………………………………………………………………………………………
………………………………………………………………………………………………

XIII Conclusion

……………………………………………………………………………………………………
……………………………………………………………………………………………………
……

XIV Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must

design more such questions so as to ensure the achievement of identified CO.

(Note: for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. Write a Program to declare a class birthday having data member day, month and year. Accept this info for object using pointer to array of object and display it.

2. Complete the following table:

Program Code	**Write** & **justify Output**
1) If array of objects is declared as given below, which is the limitation on objects? Class_name arrayName[size];	

2) What will be the output of the following C++ code?
#include<iostream>
using namespace std;
 	
class Employee
{
  int id;
  char name[30];
  public:
 	
    void getdata();
 	
    void putdata();
};
 	
void Employee::getdata()
{
  cout << "Enter Id : ";
  cin >> id;
  cout << "Enter Name : ";
  cin >> name;
}
 void Employee::putdata()
{
  cout << id << " ";
  cout << name << " ";
  cout << endl;
}
 int main()
{	

```
  Employee emp[30];
int n, i;
cout << "Enter Number of Employees - ";
cin >> n;
  for(i = 0; i < n; i++)
  emp[i].getdata();

cout << "Employee Data - " << endl;

  for(i = 0; i < n; i++)
  emp[i].putdata();
}
```

(Space for Answers)

XV References/Suggestions for further reading

1. https://www.geeksforgeeks.org/array-of-objects-in-c-with-examples/

XVI Assessment Scheme

	Performance Indicators	**Weightage**
	Process Related: 30 Marks	60 %
1	Logic formation	10%
2	Appropriate use of arithmetic expressions, operators and inline function.	20%
3	Debugging ability	20%
4	Follow ethical practices.	10%
	Product Related: 20 Marks	**40%**
5	Expected Output	20%
6	Submitting the Manual in time	10%
7	Answer to sample questions	10%
	Total (50 Marks)	**100 %**

Marks Obtained			**Dated signature of Teacher**
Process Related (30)	**Product Related (20)**	**Total (50)**	

Practical No.10: Write programs for –

- **Default Constructor**
- **Parameterized Constructor**
- **Copy Constructor**
- **Multiple Constructor in one class**

I Practical Significance:

The classes and objects help to represent real life entity with different attributes and related member functions.
The use of constructor functions facilitates initialization of instance variables.

II Industry/Employer Expected Outcome(s)

Constructor is to set the initial values of the object's attributes or perform any action that needs to be done for each object.

III Course Level Learning Outcome(s)

Develop C++ programs using constructors.

IV Laboratory Learning Outcome(s)

Write/ Compile/ debug *I* Execute simple C++ program using constructors and destructors.

V Relevant Affective Domain related outcome(s)

a. Select proper programming environment in C++.
b. Follow ethical practices.

VI Relevant Theoretical Background

- **Constructor:** Constructors are special member functions which performs initialization of every object. The Compiler calls the Constructor whenever an object is created. Constructors initialize values to instance variables after storage is allocated to the object.
  Constructors are of three types :
  1. Default Constructor
  2. Parameterized Constructor
  3. Copy Constructor

- **Declaring constructor function: Syntax:**
  class className
  {
  //other members

```
        declaration public:
                className(list of parameters)
                {
                        I/body of constructor function
                }
        //other members declaration
    };
```

- **Default Constructor In C++**
  Syntax of Default Constructor-

```
        class ClassName {
        public:
        ClassName(); // Default constructor declaration
        };
```

- **Parameterized Constructor In C++**
  Syntax of Parameterized Constructor

```
        class ClassName {
        public:
           ClassName(Type1 parameter1, Type2 parameter2, ...); // Parameterized
        //constructor declaration
        };
```

- **Copy Constructor In C++**
  Syntax of Copy Constructor

```
        class ClassName {
        public:
           ClassName(const ClassName& obj); // Copy constructor declaration
        };
```

VII Resources Required

Sr. No	**Name Resource**	**Specification**	**Quantity**	**Remarks**
1	Computer System	Any desktop or laptop computer with basic configuration	One computer system for each student	
2	Operating system	Windows /LINUX	One for each computer system	
3	Software	Turbo C++ Version 3.0 or any other	One for each computer system	

VIII Precautions to be followed

1. Handle computer system and peripherals with care.
2. Follow safety practices.

IX Exercise:

1. WAP to implement default constructor that initializes num1 and num2 as 10 and 20 and prints the values of num1 and num2.

2. Define a class student which contain member variables as rollno ,name and course. Write a program using constructor as "Computer Engineering" for course . Accept this data for objects of class and display the data.

X C++ code:

1. Write “C++” Code for above exercise on the blank pages attached at the end of practical.

XI Resources Used

Sr. No.	Name of Resource	Suggested Broad Specification	Quantity

XII Result(s)

…………………………………………………………………………………………………..

……………………………………………………………………………………………………

……………………………………………………………………………………………………

XIII Conclusion

……………………………………………………………………………………………………

……………………………………………………………………………………………………

XIV Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

(Note: for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. WAP to declare a class student having datamembers as name and percentage .Write constructor to initialise these data members. Accept and display data for one object.
2. WAP to declare class time having data members as hrs, min,sec. Write a constructor to accept data and display for two objects.

3. Complete the following table:

Program Code	**Write & justify Output**

1) What will be the output of the following C++ code?
 	
#include <iostream>
#include <string>
using namespace std;
class A{
  mutable int a;
public:
  A(){
    cout<<"Default constructor called\n";
  }
  A(const A& a){
    cout<<"Copy Constructor called\n";
  }
};
int main(int argc, char const *argv[])
{
  A obj;
  A a1 = obj;
  A a2(obj);
}	

2) What will be the output of the following C++ code?
 	
#include <iostream>
#include <string>
using namespace std;
class A{
int a;
public:
A(int i){
a = i;
}
void assign(int i){
a = i;
}
int return_value(){
return a;
}	

};
int main(int argc, char const *argv[])
{
A obj;
obj.assign(5);
cout<<obj.return_value();
}	

(Space for Answers)

XV References/Suggestions for further reading

1. https://www.geeksforgeeks.org/constructors-c/
2. https://www.javatpoint.com/cpp-tutorial

XVI Assessment Scheme

	Performance Indicators	**Weightage**
	Process Related: 30 Marks	60 %
1	Logic formation	10%
2	Appropriate use of arithmetic expressions, operators and inline function.	20%
3	Debugging ability	20%
4	Follow ethical practices.	10%
	Product Related: 20 Marks	**40%**
5	Expected Output	20%
6	Submitting the Manual in time	10%
7	Answer to sample questions	10%
	Total (50 Marks)	**100 %**

Marks Obtained			**Dated signature of Teacher**
Process Related (30)	**Product Related (20)**	**Total (50)**	

Practical No.11: Write programs for –

- **Single level Inheritance**
- **Multilevel Inheritance**

I Practical Significance:

The use of inheritance shows the reusability of the existing class properties and deriving a new class with additional properties.
Multiple Inheritances is a feature of C++ where a class can inherit from more than one classes.

II Industry/Employer Expected Outcome(s)

Constructor is to set the initial values of the object's attributes or perform any action that needs to be done for each object.

III Course Level Learning Outcome(s)

Implement Inheritance in C++.

IV Laboratory Learning Outcome(s)

Write/ Compile/ debug / Execute simple C++ program using multilevel inheritance.

V Relevant Affective Domain related outcome(s)

3. Select proper programming environment in C++.
4. Follow ethical practices.

VI Relevant Theoretical Background

Inheritance: It is the process of inheriting properties of objects of one class by objects of another class. The class which inherits the properties of another class is called Derived or Child or Sub class and the class whose properties are inherited is called Base or Parent or Super class. When a single class is derived from a single parent class, it is called Single inheritance. It is the simplest of all inheritance

Types of Inheritance:
1. Single Inheritance
2. Multiple Inheritance
3. Multilevel Inheritance
4. Hierarchical Inheritance
5. Hybrid Inheritance

Syntax of Single Inheritance:

```
class base  classname
{
   properties;
   member
   functions;
};
class derived_classname : visibility_mode base_classname
{
   properties;
member
functions;
};
```

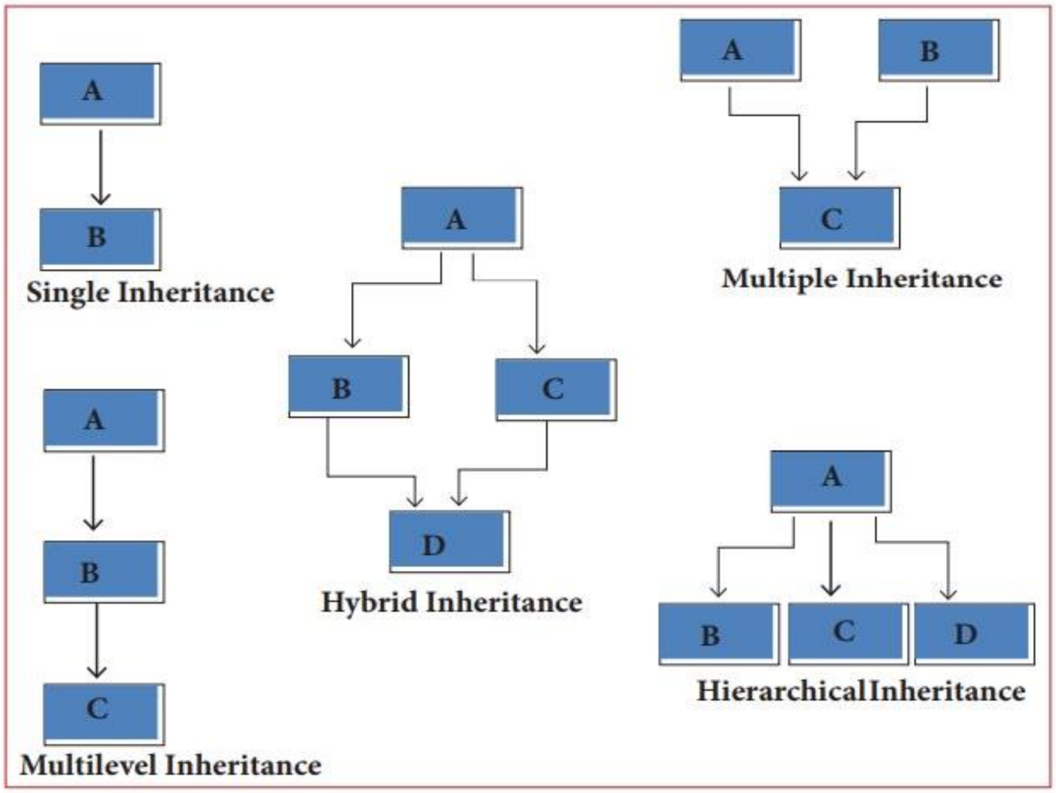
Single Inheritance

Multilevel Inheritance: In C++ programming, not only you can derive a class from the base class but you can also derive a class from the derived class. This form of inheritance is known as multilevel inheritance.

Syntax of multilevel Inheritance:

```
class base  classname
          {
             properties;
             member
             functions;
          };
          class derived_classname 1 : visibility_mode base_classname
          {
             properties;
             member
             functions;
          };
          class derived_classname2 : visibility_mode derived_classname1
          {
             properties;
             member
             functions;
          };
```

Pictorial Representation:



VII Resources Required

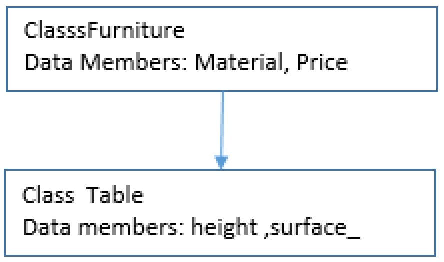
Sr. No	Name Resource	Specification	Quantity	Remarks
1	Computer System	Any desktop or laptop computer with basic configuration	One computer system for each student	
2	Operating system	Windows /LINUX	One for each computer system	
3	Software	Turbo C++ Version 3.0 or any other	One for each computer system	

VIII Precautions to be followed

1. Handle computer system and peripherals with care.
2. Follow safety practices.

IX Exercise:

1. WAP to implement inheritance shown below figure. Assume suitable member function.



2. Write a C++ program to define a class "Employee" having data members emp_no, emp_name and emp_designation. Derive a class "Salary" from "Employee" having data members basic, hra, da, gross_sal. Accept and display data for one employee.

X C++ code:

Write "C++" Code for above exercise on the blank pages attached at the end of practical.

XI Resources Used

Sr. No.	Name of Resource	Suggested Broad Specification	Quantity

XII Result(s)

……..

……

……

XIII Conclusion

………

………

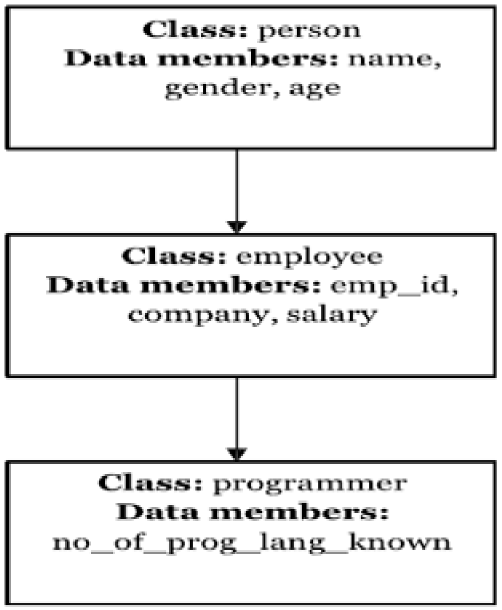
XIV Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

(Note: for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. Write a C++ program to define a class "Student" having data members roll_no, name. Derive a class "Marks" from "Student" having data members ml,m2,m3, total and percentage. Accept and display data for one student.

2. Write a C++ program to implement following Multilevel Inheritance.


Class: person
Data members: name, gender, age
Class: employee
Data members: emp_id, company, salary
Class: programmer
Data members: no_of_prog_lang_known

3. Complete the following table:

Program Code	Write & justify Output

a)
```
#include<iostream.h>

class Base

        public:
                Base(){}
                ~Base(){}
                protected:
                private:
  } ;
  class Derived:public Base
  {
        public:
                Derived(){}
                Derived(){}
                private:
                protected:
  } ;
  void main()

cout << "The program exceuted" <<
endl;

    }
```

	b) #include <iostream.h>
	class A
	{
	public:
	void print() {
	cout <<
	"print() in A";
	}
	};
	class B :
	private A
	{
	public:
	void print() {
	cout <<
	"print() in B";
	}
	};
	
	class C :
	public B
	{
	public:
	void print() {
	cout <<
	"print() in C";
	A::print();}
	};
	void main()
	{
	Cb; b.print();
	}	

(Space for Answers)

XV References/Suggestions for further reading

1.https://www.geeksforgeeks.org/constructors-c/

2.https://www.javatpoint.com/cpp-tutorial

XVI Assessment Scheme

Performance Indicators		Weightage
Process Related: 30 Marks		60 %
1	Logic formation	10%
2	Appropriate use of arithmetic expressions, operators and inline function.	20%
3	Debugging ability	20%
4	Follow ethical practices.	10%
Product Related: 20 Marks		**40%**
5	Expected Output	20%
6	Submitting the Manual in time	10%
7	Answer to sample questions	10%
Total (50 Marks)		**100 %**

Marks Obtained			**Dated signature of Teacher**
Process Related (30)	**Product Related (20)**	**Total (50)**	

Practical No.12: Write programs to implement Multiple Inheritance

I Practical Significance:

Multiple Inheritances is a feature of C++ where a class can inherit from more than one classes.

II Industry/Employer Expected Outcome(s)

Multiple inheritance allows you to mix properties and behaviors from different classes.

III Course Level Learning Outcome(s)

Implement Multiple Inheritance in C++ program.

IV Laboratory Learning Outcome(s)

Write/Compile/debug/Execute simple C++ program using Multiple inheritance.

V Relevant Affective Domain related outcome(s)

1. Select proper programming environment in C++.
2. Follow ethical practices.

VI Relevant Theoretical Background

1. **Inheritance** :-It is the process of inheriting properties of objects of one class by objects of another class. The class which inherits the properties of another class is called Derived or Child or Sub class and The class whose properties are inherited is called Base or Parent or Super class.

2. **Multiple Inheritance** :-When a class is derived from two or more base classes, such inheritance is called Multiple Inheritance. It allow us to combine the features of several existing classes into a single class.

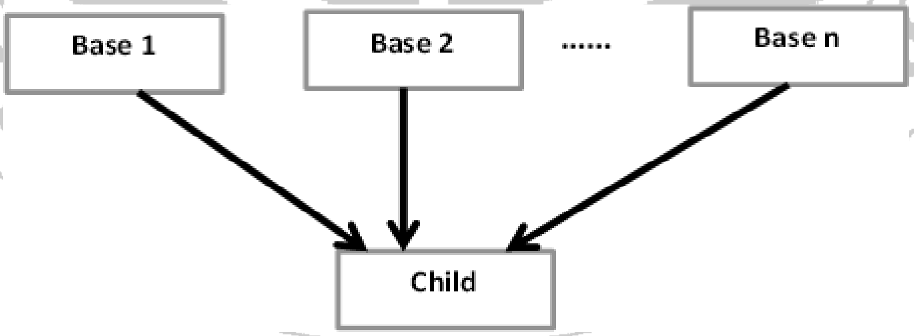
Syntax of Multiple Inheritance

```
class base  class1
{
   properties;
   member
   functions;
};

class base  class2
{
   properties;
   member
   functions;
};


class base  classN
{
   properties;
   member
   functions;
};

class derived_classname : visibility_mode base_class1,
visibility_mode base_class2,... ,visibility_mode base_classN
{
   properties;
   member
   functions;
};
```



The different access specifiers for inheritance concept-

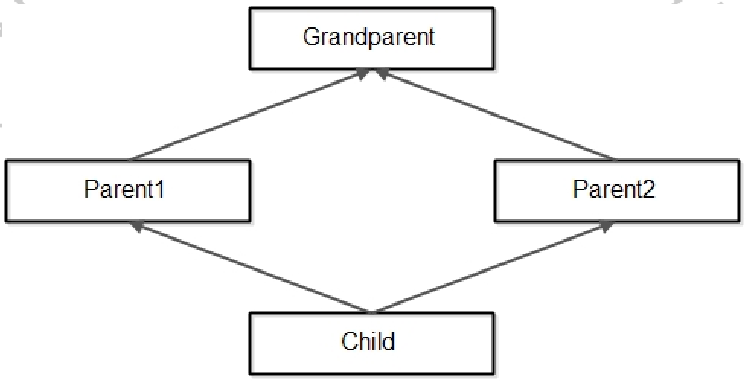
Access	**public**	**protected**	**Private**
Same class	Yes	Yes	Yes
Derived classes	Yes	Yes	No
Outside classes	Yes	No	No

Virtual Base Class:

1. 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.
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's name with the word **virtual.**

Syntax:
```
  class derived_class_name: virtual visibility_mode base_class
  {


     ---------------//members of derived class
  };
```



VII Resources Required

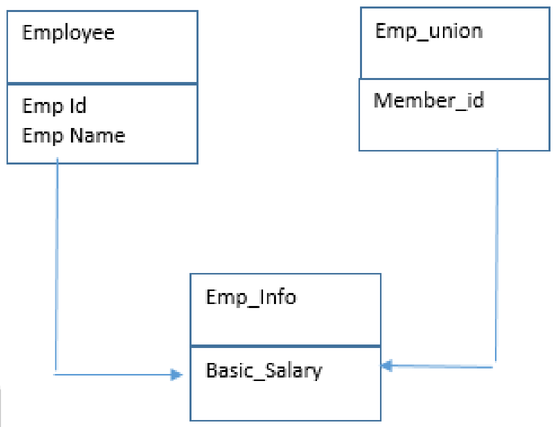
Sr. No	Name Resource	Specification	Quantity	Remarks
1	Computer System	Any desktop or laptop computer with basic configuration	One computer system for each student	
2	Operating system	Windows /LINUX	One for each computer system	
3	Software	Turbo C++ Version 3.0 or any other	One for each computer system	

VIII Precautions to be followed

1. Handle computer system and peripherals with care.
2. Follow safety practices.

IX Exercise:

1. Identify the following type of Inheritance shown . Write a definition of each class .Write suitable member functions to accept and display data for each class.



2. Write a Program to get the average marks of six subjects using the Multiple Inheritance

X C++ code:

Write “C++” Code for above exercise on the blank pages attached at the end of practical.

XI Resources Used

Sr. No.	Name of Resource	Suggested Broad Specification	Quantity

XII Result(s)

……………………………………………………………………………………………..

..…………………………………………………………………………………………

..…………………………………………………………………………………………

XIII Conclusion

……………………………………………………………………………………………

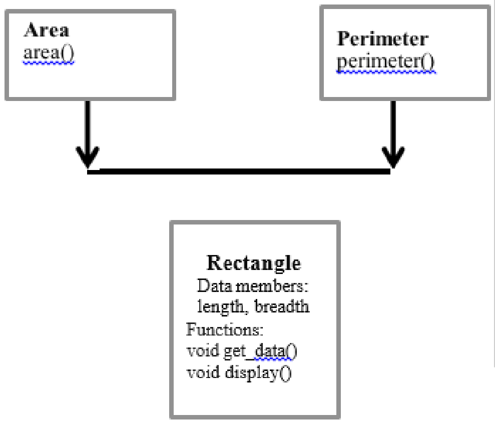
……………………………………………………………………………………………

XIV Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

(Note: for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. Write a C++ program to calculate the area and perimeter of rectangles using concept of inheritance.



2. Complete the following table:

Program Code	Write & justify Output

a) #include <iostream>
#include <string>
using namespace std;
class A
{
int a, b;
float d;
public:
void change(int i){
a = i;	

```
}
void value_of_a(){
cout<<a;
}
};
class B: private A
{
};
int main(int argc, char const *argv[])
{
B b;
cout<<sizeof(B);
return 0;
}
```

```
b)#include <iostream>
  #include <string>
     using namespace std;
     class A
     {
          float d;
        public:
          int a;
          void change(int i){
                a = i;
          }
          void value_of_a(){
                cout<<a;
          }
     };
      class B: public A
     {
          int a = 15;
        public:
          void print(){
                cout<<a;
          }
     };
     int main(int argc, char const *argv[])
     {
          B b;
          b.change(10);
          b.print();
          b.value_of_a();

          return 0;
     }
```

(Space for Answers)

XV References/Suggestions for further reading

1. https://www.geeksforgeeks.org/constructors-c/
2. https://www.javatpoint.com/cpp-tutorial

XVI Assessment Scheme

	Performance Indicators	**Weightage**
	Process Related: 30 Marks	60 %
1	Logic formation	10%
2	Appropriate use of arithmetic expressions, operators and inline function.	20%
3	Debugging ability	20%
4	Follow ethical practices.	10%
	Product Related: 20 Marks	**40%**
5	Expected Output	20%
6	Submitting the Manual in time	10%
7	Answer to sample questions	10%
	Total (50 Marks)	**100 %**

Marks Obtained			**Dated signature of Teacher**
Process Related (30)	**Product Related (20)**	**Total (50)**	

Practical No.13: Write programs to implement Hierarchical Inheritance

I Practical Significance:

Hierarchical Inheritance is a type of inheritance in C++ where one base class is inherited by more than one derived class.

II Industry/Employer Expected Outcome(s)

Hierarchical Inheritance allows for code reuse and refactoring, as the same code can be reused in multiple classes.

III Course Level Learning Outcome(s)

Implement Hierarchical Inheritance in C++ program.

IV Laboratory Learning Outcome(s)

Write/ Compile/ debug *I* Execute simple C++ program using Hierarchical Inheritance.

V Relevant Affective Domain related outcome(s)

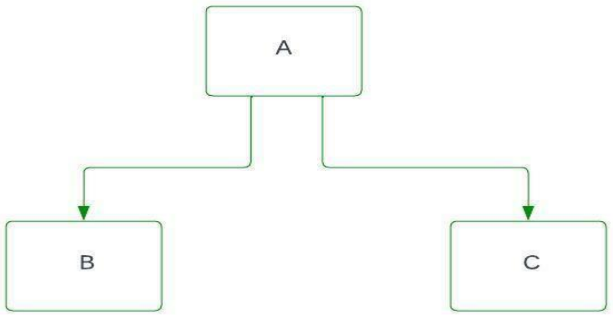
a. Select proper programming environment in C++.
b. Follow ethical practices.

VI Relevant Theoretical Background

1. **Inheritance** :-It is the process of inheriting properties of objects of one class by objects of another class. The class which inherits the properties of another class is called Derived or Child or Sub class and The class whose properties are inherited is called Base or Parent or Super class.
2. **Hierarchical Inheritance** –It is feature of Object-Oriented-programming in which a derived class (child class) inherits the property (data member and member functions) of the Base class (parent class). For example, a child inherits the traits of their parents.

Syntax of Multiple Inheritance

```
Class A
{
   ...........
};
Class B: access_specifier A
{
  .........
};
Class C: access_specifier A
{
  .............
};
```



Visibility Modes in Hierarchical Inheritance in C++ :
The visibility modes in hierarchical inheritance in C++ are Public, Protected, and Private. This means that the base class members (which includes functions and data variables) can be inherited as public, protected, or private according to the user's needs.
Public: Base class members declared public will be inherited as public members in derived classes and can be accessed by both the derived class and any external user.
Protected: Base class members declared protected will only be accessible within the derived classes(of that base class) unless it is explicitly declared public within the derived classes.
Private: If we derive a subclass from a Private base class. Then both public member and protected members of the base class will become Private in the derived class.

VII Resources Required

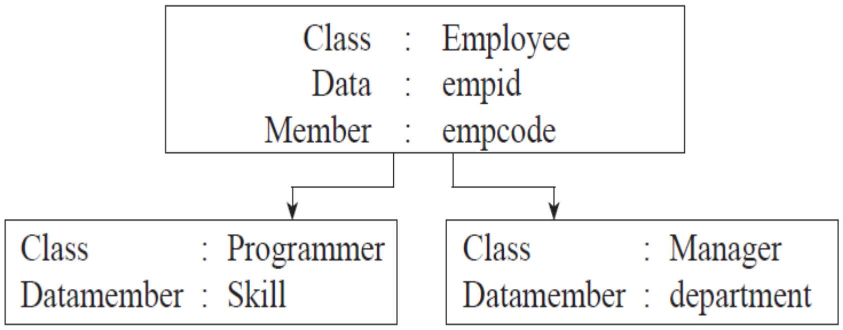
Sr. No	Name Resource	Specification	Quantity	Remarks
1	Computer System	Any desktop or laptop computer with basic configuration	One computer system for each student	
2	Operating system	Windows /LINUX	One for each computer system	
3	Software	Turbo C++ Version 3.0 or any other	One for each computer system	

VIII Precautions to be followed

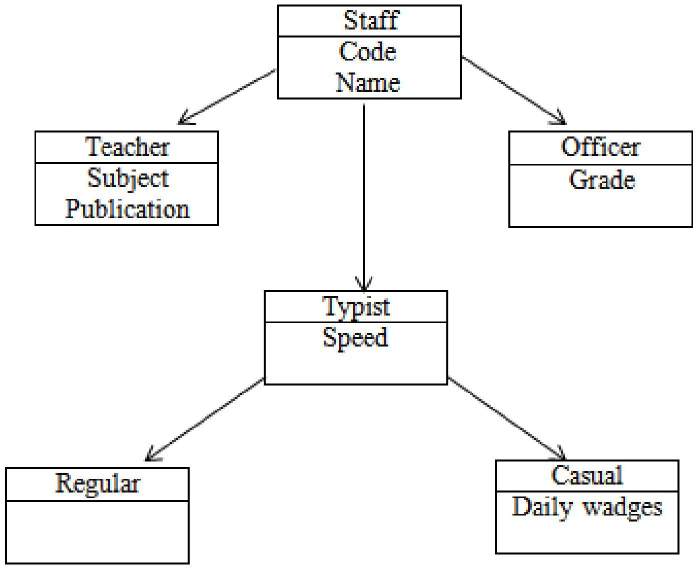
1. Handle computer system and peripherals with care.
2. Follow safety practices.

IX Exercise:

1. Write a C++ program to implement the following inheritance. Accept and display data for one programmer and one manager. Make display function virtual.



2. An Educational institute wishes to store the data of its employees. The Hierarchical relationships of its related classes are shown in the figure below. Define all the classes to represent hierarchy and define functions to retrieve individual information as and when required.



X C++ code:

Write "C++" Code for above exercise on the blank pages attached at the end of practical.

XI Resources Used

Sr. No.	Name of Resource	Suggested Broad Specification	Quantity

XII Result(s)

……………………………………………………………………………………………………..

..…………………………………………………………………………………………………

..…………………………………………………………………………………………………

XIII Conclusion

……………………………………………………………………………………………………

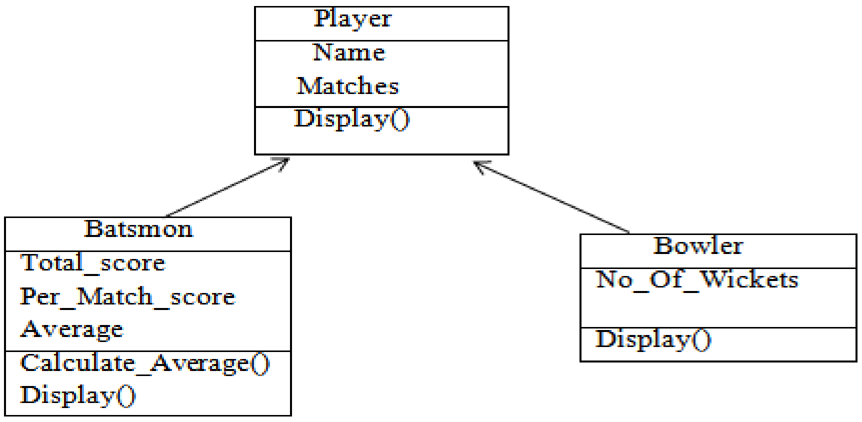
……………………………………………………………………………………………………

XIV Practical Related Questions

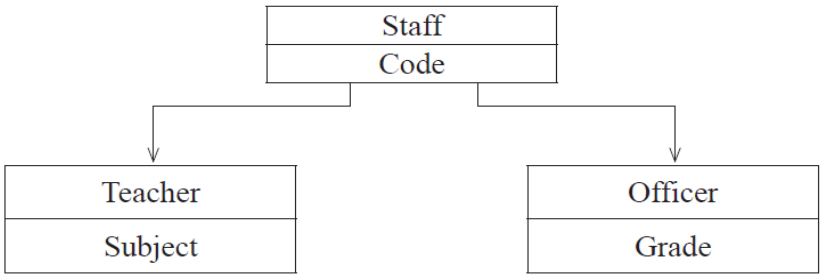
Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

(Note: for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. Write a C++ program to implement given class hierarchy.



2. Write a C++ program to implement given figure class hierarchy.



3. Complete the following table:

Program Code	**Write & justify Output**

a) #include <iostream>
using namespace std;
class Animal {
public:
    void info() {
        cout << "I am an animal." <<
        endl;
    }
};
class Dog : public Animal {
public:
    void bark() {
        cout << "I am a Dog. Woof
        woof." << endl;
    }
};
class Cat : public Animal {
public:
    void meow() {
        cout << "I am a Cat. Meow." <<
        endl;
    }
};
int main() {
    class
    Dog dog1;
    cout << "Dog Class:" << endl;
    dog1.info(); // parent Class function
    dog1.bark();
class
    Cat cat1;
    cout << "\nCat Class:" << endl;
    cat1.info(); // parent Class function
    cat1.meow();
    return 0;
}	

```
b)#include<iostream.h>
#include<conio.h>
class emp
{
int id;
char name[20];
int sal;
public:
void accept()
{
cout<<"enter id, name and salary"<<endl;
cin>>id>>name>>sal;
}
void dis()
{
cout<<"id"<<id;
cout<<"name--"<<name<<endl;
cout<<"salary"<<sal;
}
};
class manager : public emp
{
public:
acc1()
{
accept();
dis();
}
} ;
class clerk :public emp
{
public: acc2()
{
accept();
dis() ;
}
};
void main()
{
manager m;
clerk c;
m.acc1();
c.acc2();
getch();
}
```

(Space for Answers)

XV References/Suggestions for further reading

1. https://www.javatpoint.com/hierarchical-inheritance-in-cpp
2. https://www.geeksforgeeks.org/cpp-hierarchical-inheritance/
3. https://www.w3schools.in/cplusplus/inheritance

XVI Assessment Scheme

	Performance Indicators	**Weightage**
	Process Related: 30 Marks	60 %
1	Logic formation	10%
2	Appropriate use of arithmetic expressions, operators and inline function.	20%
3	Debugging ability	20%
4	Follow ethical practices.	10%
	Product Related: 20 Marks	**40%**
5	Expected Output	20%
6	Submitting the Manual in time	10%
7	Answer to sample questions	10%
	Total (50 Marks)	**100%**

Marks Obtained			**Dated signature of Teacher**
Process Related (30)	**Product Related (20)**	**Total (50)**	

Practical No.14: Write programs to implement Virtual Base Class.

I Practical Significance:

Virtual Base Class is that a child class could have duplicate sets of members inherited from a single base class.

II Industry/Employer Expected Outcome(s)

Virtual base classes used to prevent multiple instances of a given class from appearing in an inheritance hierarchy when using multiple inheritances.

III Course Level Learning Outcome(s)

Implement Virtual Base Class in C++ program.

IV Laboratory Learning Outcome(s)

Write/ Compile/ debug *I* Execute simple C++ program using Virtual Base Class.

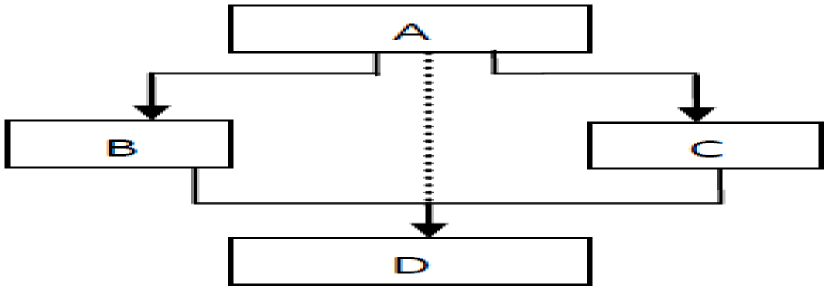
V Relevant Affective Domain related outcome(s)

a. Select proper programming environment in C++.
b. Follow ethical practices.

VI Relevant Theoretical Background

Virtual Base Class:

1. **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.
   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's name with the word **virtual.**



Syntax:

```
class derived_class_name: virtual visibility_mode base_class
{


  ---------------//members of derived class
};
```

Example:

```
       class A
 {
        Public:
        Int I;
 };
 class B: virtual public A
 {
        Public:
        Int j;
 };
```

VII Resources Required

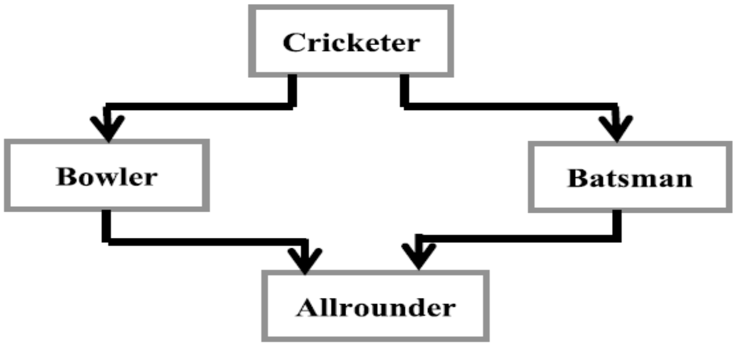
Sr. No	Name Resource	Specification	Quantity	Remarks
1	Computer System	Any desktop or laptop computer with basic configuration	One computer system for each student	
2	Operating system	Windows /LINUX	One for each computer system	
3	Software	Turbo C++ Version 3.0 or any other	One for each computer system	

VIII Precautions to be followed

1. Handle computer system and peripherals with care.
2. Follow safety practices.

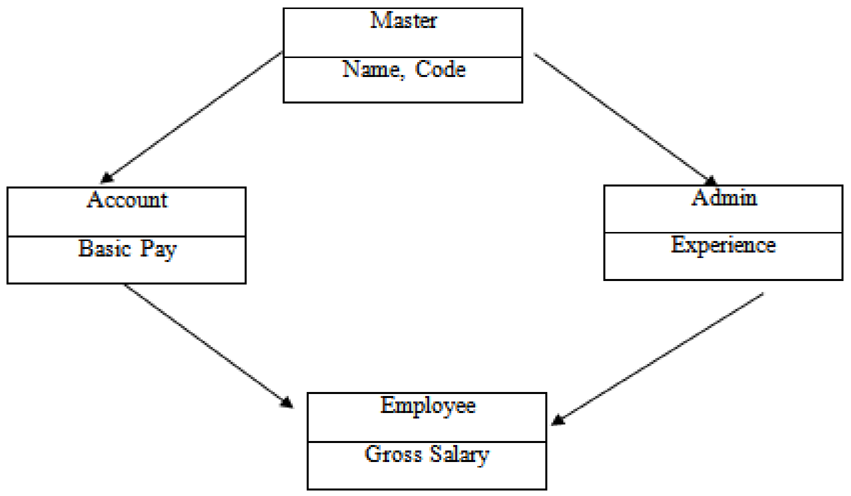
IX Exercise:

1. Write a C++ program to implement the concept of Virtual Base Class for following figure.
   Assume suitable data and function members.



2. Write a C++ program to implement the concept of Virtual Base Class for following figure.

Accept and Display information of one employee with his name,code,basic Pay,Experience and Gross Salary with the object of Employee class.



X C++ code:

Write "C++" Code for above exercise on the blank pages attached at the end of practical.

XI Resources Used

Sr. No.	Name of Resource	Suggested Broad Specification	Quantity

XII Result(s)

……………………………………………………………………………………………………

……………………………………………………………………………………………………

……………………………………………………………………………………………………

XIII Conclusion

……………………………………………………………………………………………………

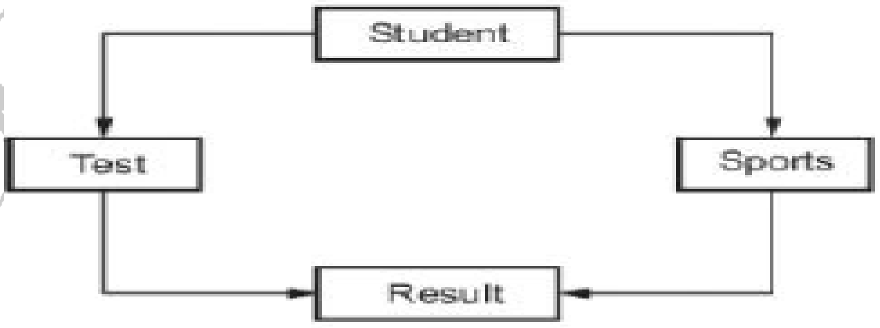
……………………………………………………………………………………………………

XIV Practical Related Questions

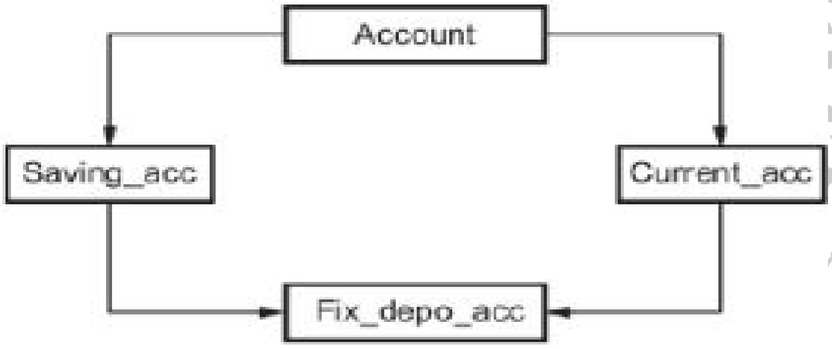
Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

(Note: for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1.Write a C++ program to implement the concept of virtual base class for following figure.



2.Write a C++ program to implement given figure class hierarchy. Assume suitable member variables and member functions.



XIII Complete the following table:

Program Code	**Write & justify Output**
a) #include <iostream.h> struct a { int count; } ; Struct b	

{
 int* value;
 } ;
struct c
public a,
public b
 } ;
int main()
 {
c* p new =c;
p->value=0;
cout<< "Inherited";
return 0;
}	

b)#include<iostream.h>
 class A
 {
 public:
 void display()
 {
 cout<< “A\n”;
 };
class B : public A
{
 public:
 void display()
 {
 cout<<”B\n”;
 }
};
 	
int main()
{
 B b;
 b.display();
 b.A::display();
 b.B::display();
 return 0;
}	

(Space for Answers)

XV References/Suggestions for further reading

1. https://www.geeksforgeeks.org/virtual-base-class-in-c/
2. https://www.w3schools.in/cplusplus/virtual-function

XVI Assessment Scheme

Performance Indicators		Weightage
Process Related: 30 Marks		60 %
1	Logic formation	10%
2	Appropriate use of Virtual class.	20%
3	Debugging ability	20%
4	Follow ethical practices.	10%
Product Related: 20 Marks		**40%**
5	Expected Output	20%
6	Submitting the Manual in time	10%
7	Answer to sample questions	10%
Total (50 Marks)		**100 %**

Marks Obtained			**Dated signature of Teacher**
Process Related (30)	**Product Related (20)**	**Total (50)**	

Practical No.15: Write programs which show use of constructors in derived class.

I Practical Significance:

Constructors in derived classes are special member functions used to initialize objects of derived classes. When you create an object of a derived class, its constructor is called to initialize both the base class part and the derived class part of the object

II Industry/Employer Expected Outcome(s)

When a derived class is extended from the base class, the constructor of the base class is executed first followed by the constructor of the derived class.

III Course Level Learning Outcome(s)

Implement constructors in derived class in C++ program.

IV Laboratory Learning Outcome(s)

Write/ Compile/ debug / Execute simple C++ program using Constructors in derived class.

V Relevant Affective Domain related outcome(s)

c. Select proper programming environment in C++.
d. Follow ethical practices.

VI Relevant Theoretical Background

Constructors in Derived Class in C++

- We can use constructors in derived classes in C++
- If the base class constructor does not have any arguments, there is no need for any constructor in the derived class
- But if there are one or more arguments in the base class constructor, derived class need to pass argument to the base class constructor
- If both base and derived classes have constructors, base class constructor is executed first

Syntax:

```
Derived-Constructor (arg1, arg2, arg3….): Base 1-Constructor (arg1,arg2),
Base 2-Constructor(arg3,arg4)
{
….
} Base 1-Constructor (arg1,arg2)
```

VII Resources Required

Sr. No	Name Resource	Specification	Quantity	Remarks
1	Computer System	Any desktop or laptop computer with basic configuration	One computer system for each student	
2	Operating system	Windows /LINUX	One for each computer system	
3	Software	Turbo C++ Version 3.0 or any other	One for each computer system	

VIII Precautions to be followed

1. Handle computer system and peripherals with care.
2. Follow safety practices.

IX Exercise:

1. Write output of following code

```
#include <iostream>
using namespace std;

// base class
class Parent
{
 public:

  // base class constructor
   Parent()
   {
   cout << "Inside base class" << endl;
   }
};

// sub class
class Child : public Parent
{
    public:

   //sub class constructor
    Child()
    {
        cout << "Inside sub class" << endl;
    }
};
// main function
int main() {
    // creating object of sub class
    Child obj;

    return 0;
}
2. #include <iostream>
using namespace std;

// first base class
class Parent1
{

    public:

    // first base class's Constructor
    Parent1()
    {
        cout << "Inside first base class" << endl;
    }
};

// second base class
class Parent2
{
    public:

    // second base class's Constructor
    Parent2()
    {
        cout << "Inside second base class" << endl;
    }
};

// child class inherits Parent1 and Parent2
class Child : public Parent1, public Parent2
{
    public:

    // child class's Constructor
    Child()
    {
        cout << "Inside child class" << endl;
    }
};
```

```
        // main function
        int main() {
                // creating object of class Child
           Child obj1;
           return 0;
        }
```

X C++ code:

Write “C++” Code for above exercise on the blank pages attached at the end of practical.

XI Resources Used

Sr. No.	Name of Resource	Suggested Broad Specification	Quantity

XII Result(s)

……………………………………………………………………………………………………

……………………………………………………………………………………………………

XIII Conclusion

……………………………………………………………………………………………………

……………………………………………………………………………………………………

XIV Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

(**Note:** for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. Write a program which show the use of constructor in derived class.

2.Complete the following table:

Program Code	Write & justify Output

a)class Base1{
        int data1;
        public:
           Base1(int i){
              data1 = i;
              cout<<"Base1 class constructor called"<<endl;
           }
           void printDataBase1(void){
              cout<<"The value of data1 is	

```
"<<data1<<endl;
            }
    };

    class Base2{
        int data2;

        public:
            Base2(int i){
                data2 = i;
                cout << "Base2 class constructor
called" << endl;
            }
            void printDataBase2(void){
                cout << "The value of data2 is " <<
data2 << endl;
            }
    };

    class Derived: public Base2, public Base1{
        int derived1, derived2;
        public:
            Derived(int a, int b, int c, int d) :
Base2(b), Base1       {
                derived1 = c;
                derived2 = d;
                cout<< "Derived class constructor
called"<<endl;
            }
            void printDataDerived(void)
            {
                cout << "The value of derived1 is " <<
derived1 << endl;
                cout << "The value of derived2 is " <<
derived2 << endl;
            }
    };
```

```
b)  #include <iostream>
c)    using namespace std;

      class A
      {
            public:
            A(int n )
            {
                cout << n;
            }
        };
```

```
class B: public A
{
   public:
   B(int n, double d)
   : A(n)
   {
      cout << d;
   }
};
class C: public B
{
   public:
   C(int n, double d, char ch)
   : B(n, d)
   {
      cout <<ch;
   }
};
int main()
{
   C c(5, 4.3, 'R');
   return 0;
}
```

(Space for Answers)

XV References/Suggestions for further reading

1. https://www.tutorialspoint.com/cplusplus/cpp_constructor_destructor.htm
2. https://www.codewithharry.com/videos/cpp-tutorials-in-hindi-48/

VI Assessment Scheme

Performance Indicators		Weightage
Process Related: 30 Marks		60 %
1	Logic formation	10%
2	Appropriate use of Constructors in derived class	20%
3	Debugging ability	20%
4	Follow ethical practices.	10%
Product Related: 20 Marks		**40%**
5	Expected Output	20%
6	Submitting the Manual in time	10%
7	Answer to sample questions	10%
Total (50 Marks)		**100 %**

Marks Obtained			**Dated signature of Teacher**
Process Related (30)	**Product Related (20)**	**Total (50)**	

Practical No.16: Write programs to implement-

- **Pointer to object**
- **'this' pointer**

I Practical Significance:

The pointer to objects helps to improve the efficiency of the program and access the objects dynamically.

II Industry/Employer Expected Outcome(s)

This practical is expected to develop the following skills as:

1. Define real life problems using pointer to objects.

ii. Use of this pointer

III Course Level Learning Outcome(s)

Implement Polymorphism in C++.

IV Laboratory Learning Outcome(s)

- Implement pointer arithmetic in a program.
- Implement pointer to object in a program.
- Implement 'this' pointer in a program.

V Relevant Affective Domain related outcome(s)

1.Select proper programming environment in C++.

2.Follow ethical practices.

VI Relevant Theoretical Background

Pointers to objects: -

A pointer is a variable that stores the memory address of another variable (or object) as its value. A pointer aims to point to a data type which may be int, character, double, etc.

Pointers to objects aim to make a pointer that can access the object, not the variables. Pointer to object in C++ refers to accessing an object.

There are two approaches by which you can access an object. One is directly and the other is by using a pointer to an object in C++.

A pointer to an object in C++ is used to store the address of an object. For creating a pointer to an object in C++, we use the following syntax:

Class name *pointer to object;

For storing the address of an object into a pointer in c++, we use the following syntax:

Pointer to object=&object- name;

The above syntax can be used to store the address in the pointer to the object. After storing the address in the pointer to the object, the member function can be called using the pointer to the object with the help of an arrow operator.

Examples

Example 1. In the below example, a simple class named My_Class is created. An object of the class is defined as named object. Here a pointer is also defined named p. In the program given below program, it is shown how can we access the object directly and how can we use the pointer to access the object directly.

```
// Example using an object pointer.

#include <iostream>
using namespace std;

class My_Class {
   int num;
    public:
   void set_number(int value)
  {
   num = value;
   }
   void show_number();
   };

  void My_Class::show_number()
{
  cout << num << "\n";
}

int main()
  {
 My_Class object, *p; // an object is declared and a pointer to it

 object.set_number(1); // object is accessed directly
 object.show_number();

 p = &object; //  the address of the object is assigned to p
   p->show_number(); // object is accessed using the pointer

   return 0;
  }
```

b. this pointer:

In C++ programming, **this** is a keyword that refers to the current instance of the class. There can be 3 main usages of this keyword in C++.

- It can be used **to pass current object as a parameter to another method.**
- It can be used **to refer current class instance variable.**
- It can be used **to declare indexers.**

VII Resources Required

Sr. No	**Name Resource**	**Specification**	**Quantity**	**Remarks**
1	Computer System	Any desktop or laptop computer with basic configuration	One computer system for each student	
2	Operating system	Windows /LINUX	One for each computer system	
3	Software	Turbo C++ Version 3.0 or any other	One for each computer system	

VIII Precautions to be followed

1.Handle computer system and peripherals with care.
2.Follow safety practices.

IX Exercise:

1. State output of the following code:

```
#include <iostream.h>
        #include<conio.h>
  class myclass
              {
              int i;
              public:
              void read(int j)
              {
              i= j;
              }
              int getint()
              {
              return i;
              }
              };
              void main()
              {
              clrscr();
              rnyclass ob, *objectPointer;
```

```
objectPointer = &ob;
objectPointer->read(lO);
cout<<objectPointer->getint();
getch();
```

2. Which is the pointer which denotes the object calling the member function?
   a. Variable pointer b) This pointer
   c) Null pointer d) Zero pointer

3. A pointer can be initialized with ________

   a) Null b) zero

   c)Address of an object of same type d) All of them

4. Write a program which show the use of this pointer.

X C++ code:

Write "C++" Code for above exercise on the blank pages attached at the end of practical.

XI Resources Used

Sr. No.	Name of Resource	Suggested Broad Specification	Quantity

XII Result(s)

……

……

XIII Conclusion

……

……

XIV Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

(Note: for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1 Write a C++ program to declare a class "Book" contammg data members

book_name, auther_name, and price . Accept this information for one object of the class using pointer to that object.

2 Write a C++ program to declare a class "Box" having data members height, width and breadth. Accept this information for one object using pointer to that object . Display the area and volume of that object.

3. Write a C++ program to declare a class birthday having data members day, month, year. Accept this information for five objects using pointer to the array of objects

4. Complete the following table:

Program Code	**Write & justify Output**

a) #include <iostream>
 #include <iostream>
using namespace std;
class Employee {
public:
int id; //data member
string name; //data member
float salary;
Employee(int id, string name, float salary)
{
this->id = id;
this->name = name;
this->salary = salary;
}
void display()
{
cout<<id<<" "<<name<<" "<<salary<<endl;
}
};
int main(void) {
Employee e1 =Employee(101, "Sonoo", 890000); //creating an object of Employee
Employee e2=Employee(102, "Nakul", 59000); //creating an object of Employee
e1.display();
e2.display();
return 0;
}	

b)#include <iostream>
#include <string>
using namespace std;
class student
{
private:
int rollno;
string name;
public:
student():rollno(0),name(1111)
{}
student(int r, string n): rollno(r),name (n)
{}
void get()
cout<<"enter roll no"; cin>>rollno; cout<<"enter name"; cin>>name;
void print()
cout<<"roll no is	

11<<rollno;
cout<<"name is "<<name;
} ;
void main ()
student *ps=new student; (*ps).get();
(*ps).print(); delete ps;	

(Space for Answers)

XV References/Suggestions for further reading

1. https://www.scaler.com/topics/pointer-to-object-in-cpp/
2. https://www.ibm.com/docs/en/i/7.3?topic=only-member--c

XVI Assessment Scheme

Performance Indicators		**Weightage**
Process Related: 30 Marks		60 %
1	Logic formation	10%
2	Appropriate use of arithmetic expressions, operators and basic Input /output function.	20%
3	Debugging ability	20%
4	Follow ethical practices.	10%
Product Related: 20 Marks		**40%**
5	Expected Output	20%
6	Submitting the Manual in time	10%
7	Answer to sample questions	10%
Total (50 Marks)		**100 %**

Marks Obtained			**Dated signature of Teacher**
Process Related (30)	**Product Related (20)**	**Total (50)**	

Practical No.17: Write programs for-

- **Pointer to derived class in single inheritance**
- **Pointer to derived class in multilevel inheritance**

I Practical Significance:

The key feature of class inheritance is that a pointer to a derived class is type-compatible with a pointer to its base class.
.

II Industry/Employer Expected Outcome(s)

This practical is expected to develop the following skills as:

1. Define pointer to derived class in single inheritance.
2. Define pointer to derived class in multilevel inheritance

III Course Level Learning Outcome(s)

Implement Polymorphism in C++.

IV Laboratory Learning Outcome(s)

Implement program to use pointer to derived class.

V Relevant Affective Domain related outcome(s)

1.Select proper programming environment in C++.
2.Follow ethical practices.

VI Relevant Theoretical Background

C++ allows base class pointers to point to derived class objects.

Let we have-

```
class base { .. . };

class derived : public base { ... }; Then we can write -

base *pl;

derived d_obj;

pl=&d_obj;

base *p2 = new derived;

ll
```

Using a base class pointer (pointing to a derived class object) we can access only those members of the derived object that were inherited from the base.

- It is different from the behaviour that Java shows.
- We can get Java-like behaviour using virtual functions.
- This is because the base pointer has knowledge only of the base class.
- It knows nothing about the members added by the derived class.

VII Resources Required

Sr. No	Name Resource	Specification	Quantity	Remarks
1	Computer System	Any desktop or laptop computer with basic configuration	One computer system for each student	
2	Operating system	Windows /LINUX	One for each computer system	
3	Software	Turbo C++ Version 3.0 or any other	One for each computer system	

VIII Precautions to be followed

1. Handle computer system and peripherals with care.
2. Follow safety practices.

IX Exercise:

1.State output of the following code:

```
#include<iostream.h>
class base {
public:
void show()
cout << "base\n";
} ;
class derived: public base
public:
void show() {
cout << "derived\n";
} ;
void main()
{
base bl;
bl.show();
derived dl;
dl.show();
base *pb = &bl;
pb->show();
pb = &dl;
pb->show();
}
```

2. A pointer to the base class can hold address of
   A) only base class object B) only derived class object
   C) base class object as well as derived class object D) None of the above

3. A pointer can be initialized with ________

   a) Null b) zero
   c)Address of an object of same type d) All of them

3.Which variable stores the memory address of another variable?
   A) Reference B) Pointer
   C) Array D) None of the above

X C++ code:

Write “C++” Code for above exercise on the blank pages attached at the end of practical.

XI Resources Used

Sr. No.	**Name of Resource**	**Suggested Broad Specification**	**Quantity**

XII Result(s)

……………………………………………………………………………………………………
..……………………………………………………………………………………………
..……………………………………………………………………………………………

XIII Conclusion

……………………………………………………………………………………………………
……………………………………………………………………………………………………

XIV Practical Related Questions

***Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.*(Note:** for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. Write a C++ program declare a class "polygon" having data members width and height. Derive classes "rectangle" and "triangle" from "polygon" having area() as a member function. Calculate area of triangle and rectangle using pointer to derived class object.

2. Write a program which show the use of Pointer to derived class in multilevel inheritance.

3.Complete the following table:

Program Code	**Write & justify Output**

a) #include<iostream.h>
class base
{
public:
int nl;
void show()
{
cout<<"\nnl"<<nl;
}
} ;
class derive:public base
{
public:
int n2;
void show()
{
cout<<"\nnl   "<<nl;
 cout<<"\nn2 "<<n2;
}
} ;
 	
int main()
{
base b;
base *bptr;
cout<<"Pointer of base class points to it";
bptr=&b;
bptr->n1=44;
bptr->show();
derived d;
cout<<"\n";
bptr=&d;
bptr->n1=66;
bptr->show();
return 0;
}	

b) #include <iostream.h>
class BaseClass {
int x;
public:
void setx(int i)
X = i;
int getx()
return x;
} ;
 	
class DerivedClass : public BaseClass
{
int y;
public:
void sety(int i)
{
y = i;
 	
}
 	
int gety()
{
return y;
}
} ;
 	
int main()
{
BaseClass *p;
BaseClass baseObject;
DerivedClass derivedObject;
 	
p = &baseObject;
p->setx(l0);
cout << "Base object x: "<< p->getx() <<"\n" ;
p = &derivedObject;
p->setx(99);
derivedObject.sety(88);
cout << "Derived object x: " << p->getx () <<"\n" ;
cout << "Derived object y: "<< derivedObject.gety() << '\n';
return 0;
}	

(Space for Answers)

XV References/Suggestions for further reading

1. https://www.scaler.com/topics/pointer-to-object-in-cpp/

2.https://www.geeksforgeeks.org/base-class-pointer-pointing-to-derived class object-in-cpp/

XVI Assessment Scheme

	Performance Indicators	**Weightage**
	Process Related: 30 Marks	60 %
1	Logic formation	10%
2	Appropriate use of arithmetic expressions, operators and basic Input /output function.	20%
3	Debugging ability	20%
4	Follow ethical practices.	10%
	Product Related: 20 Marks	**40%**
5	Expected Output	20%
6	Submitting the Manual in time	10%
7	Answer to sample questions	10%
	Total (50 Marks)	**100%**

Marks Obtained			**Dated signature of Teacher**
Process Related (30)	**Product Related (20)**	**Total (50)**	

Practical No.18: Write programs which show the use of function overloading

I Practical Significance:

The concept of function overloading helps to use more than one definition for a function name in the same scope and helps to extend the concept of polymorphism.
.

II Industry/Employer Expected Outcome(s)

This practical is expected to develop the following skills as:

1. Define more than one definition for a function name in the same scope to implement function overloading.

III Course Level Learning Outcome(s)

Implement Polymorphism in C++.

IV Laboratory Learning Outcome(s)

Implement function overloading in a program.

V Relevant Affective Domain related outcome(s)

1. Select proper programming environment in C++.
2.Follow ethical practices.

VI Relevant Theoretical Background

Function overloading.

- Function overloading is a feature in C++ where two or more functions can have the same name but different parameters.
- The C++ compiler selects the proper function by examining the number, types and order of the arguments in the call.
- Function overloading is commonly used to create several functions of the same name that perform similar tasks but on different data types.
- Function overloading can be considered as an example of polymorphism feature in C++.
- There are two ways to overload the method in C++
    a. By changing number of arguments or parameters
    b. By changing the data type

Syntax

```
class class_Name
{
retumtype method()
{

}
returntype method(datatype1 variable1)
{

}
returntype method(datatype1 variable1, datatype2 variable2)
{


}
};
```

Examples

void display();   //function with no arguments
void display(int); //function with one integer type arguments void display(float);   //function with one floating point arguments
void display(int, float); //function with one floating and one integer type argument

VII Resources Required

Sr. No	Name Resource	Specification	Quantity	Remarks
1	Computer System	Any desktop or laptop computer with basic configuration	One computer system for each student	
2	Operating system	Windows /LINUX	One for each computer system	
3	Software	Turbo C++ Version 3.0 or any other	One for each computer system	

VIII Precautions to be followed

1. Handle computer system and peripherals with care.
2. Follow safety practices.

IX Exercise:

1. Does constructor overloading is implementation of function overloading?
2. State output of the following code:

```
#include<iostream.h> #include<conio.h>
        int operate (int a, int b)
        {
        return (a *  b);

        float operate (float a, float b)
        {
        return (a / b);

        int main()
```

```
{
    int  X   = 5,  y  =  2; float n = 5.0,
    m = 2.0;
    cout << operate(x, y)        <<"\t";   cout
    << operate (n, m);
    return 0;
```

3. Which of the following in Object Oriented Programming is supported by Function overloading and default arguments features of C++?
   a) Inheritance   b) Polymorphism
   c) Encapsulation   d) None of these

4. Overloaded functions are
   a) Very long functions that can hardly run
   b) One function containing another one or more functions inside it.
   c) Two or more functions with the same name but different number parameters or type.
   d) None of these

X C++ code:

Write "C++" Code for above exercise on the blank pages attached at the end of practical.

XI Resources Used

Sr. No.	Name of Resource	Suggested Broad Specification	Quantity

XII Result(s)

……………………………………………………………………………………………………

..………………………………………………………………………………………………

XIII Conclusion

……………………………………………………………………………………………………

……………………………………………………………………………………………………

XIV Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

(Note: for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. Write a C++ Program to interchange the values of two int , float and char using function overloading.

2. Write a C++ Program that find the sum of two int and double number using function overloading.

3. Write C++ program to find the area of various geometrical shapes by Function overloading. (eg.Area of circle, circumference of circle etc…)

4. Complete the following table:

Program Code	Write & justify Output

a) #include<iostream.h>
 int absolute(int);
float absolute(float);
{
int main()
int a= -5;
float b = 5.5;
cout << "Absolute value of"<< a<<"   "<<   absolute(a)
<< endl;
cout << "Absolute value of"<< b <<"   "<< absolute(b);
return 0;
}
int absolute(int var)
{
if (var < 0)
{
var= -var;
return var;
}
float absolute(float var)
{
if (var< 0.0)
{
var= -var;
return var;
}
}	

b) #include <iostream.h>
class Test
{
public:
int main(int s)
{
cout << s << "\n";
 return 0;	

```
}
int main(char *s)
{
cout << s << endl;
return 0;
}
int main(int s ,int m)
{
cout << s <<" "<< m;
return 0;
}
} ;
int main()
{
Test obj;
obj.main(3);
obj.main("! like C++");
obj.main(9, 6);
return 0;
}
```

(Space for Answers)

XV References/Suggestions for further reading

1. https://www.w3schools.com/cpp/cpp_function_overloading.asp

2. https://www.javatpoint.com/cpp-overloading

XVI Assessment Scheme

Performance Indicators		**Weightage**
	Process Related: 30 Marks	60 %
1	Logic formation	10%
2	Appropriate use of arithmetic expressions, operators and basic Input /output function.	20%
3	Debugging ability	20%
4	Follow ethical practices.	10%
	Product Related: 20 Marks	**40%**
5	Expected Output	20%
6	Submitting the Manual in time	10%
7	Answer to sample questions	10%
	Total (50 Marks)	**100 %**

Marks Obtained			**Dated signature of Teacher**
Process Related (30)	**Product Related (20)**	**Total (50)**	

Practical No.19: Write programs to overload unary operator using-

- **Member function**
- **Friend function**

I Practical Significance:

The concept of operator overloading helps to assign the new meaning to the existing operator and helps to extend the concept of polymorphism.

II Industry/Employer Expected Outcome(s)

This practical is expected to develop the following skills as:

1. provide additional meaning to the unary operator using member function and friend function.
2. Define and use the overloaded operator function in the class.

III Course Level Learning Outcome(s)

Implement Polymorphism in C++.

IV Laboratory Learning Outcome(s)

1. Implement unary operator overloading using member function.
2. Implement unary operator overloading using friend function.

V Relevant Affective Domain related outcome(s)

1. Select proper programming environment in C++.
2. Follow ethical practices.

VI Relevant Theoretical Background

Operator overloading

Operator overloading is a type of polymorphism m which an operator is overloaded to give user defined meaning to it. The mam purpose of operator overloading is to perform operation on user defined data type. For eg. The '+' operator can be overloaded to perform addition on various data types. Operator overloading is used by the programmer to make a program clearer. It is an important concept in C++.

Syntax:

```
Retum_type classname :: operator OperatorSymbol (Argument_List)
{

//Statements;
```

}

The operator keyword is used for overloading the operators.

There are a few operators which cannot be overloaded are follows,

1. Scope resolution operator (::)
2. sizeof
3. member selector(.)
4. member pointer selector (*)
5. ternary operator (? :)

There are some restrictions considered while implementing the operator overloading,

1. The number of operands cannot be changed. Unary operator remains unary, binary remains binary etc.
2. Only existing operators can be overloaded.
3. The precedence and associativity of an operator cannot be changed.
4. Cannot redefine the meaning of a procedure.

The Unary Operators can be overloaded by using:

1. Member Function
2. Friend Function

- In the case of a non-static member function, the unary operator should not have an argument.
- In the case of a friend function, the unary operator should have only one argument.
- They perform various operations such as incrementing/decrementing a value by one, negating an expression, or inverting the value of a boolean

- **Unary Operator Overloading Algorithm/Steps:**

Step **1:** Start the program.
Step 2: Declare the class.
Step 3: Declare the variables and its member function.
Step4: Using the function getvalue() to get the two numbers.
Step 5: Define the function operator ++ to increment the values
Step 6: Define the function operator - -to decrement the values.
Step 7: Define the display function.
Step 8: Declare the class object.
Step 9: Call the function getvalue()
Step 10: Call the function operator++ () by incrementing the class object and call the function display.
Step **11:** Call the function operator -- () by decrementing the class object and call the function display.
Step 12: Stop the program.

VII Resources Required

Sr. No	Name Resource	Specification	Quantity	Remarks
1	Computer System	Any desktop or laptop computer with basic configuration	One computer system for each student	
2	Operating system	Windows /LINUX	One for each computer system	
3	Software	Turbo C++ Version 3.0 or any other	One for each computer system	

VIII Precautions to be followed

1. Handle computer system and peripherals with care.
2. Follow safety practices.

IX Exercise:

1. What is function overloading and operator overloading?
2.State output of the following code:

```
#include <iostream.h> class
Distance {
    private:
        int  feet;  int
        inches;

    public:
        // required constructors Distance()
        {
            feet= 0;
            inches= 0;

        Distance(int     f,     inti)
            feet= f;
            inches= i;


        void displayDistance() {
            cout << "F: "<<feet<<" I:"<< inches <<endl;


        // overloaded minus (-) operator Distance
        operator- () {
            feet= -feet;  inches=
            -inches;
            return Distance(feet, inches);

 } ;
int main()
    Distance Dl(ll, 10), D2(-5, 11);
     -Dl;
    Dl.displayDistance();
```

```
        -D2;
        D2.displayDistance();
        return 0;
      }
```

2. Overload the unary operator –(minus) using member function and friend function

X C++ code:

Write "C++" Code for above exercise on the blank pages attached at the end of practical.

XI Resources Used

Sr. No.	Name of Resource	Suggested Broad Specification	Quantity

XII Result(s)

....

....

....

XIII Conclusion

....

....

....

XIV Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

(Note: for all relevant programming exercise use blank pages provided or attach more pages if needed.)

Write a C++ program to overload unary operators (++) increment and (--) decrement by using member function and friend function.

1. Write C++ program to find the area of various geometrical shapes by function overloading. (eg. Area of circle, circumference of circle etc…)

3.Complete the following table:

Program Code	**Write & justify Output**

a)#include<iostream.h>
class 3D
{
int x, y, z;
 public:
3D (int a=O, int b=O, int c=O)
{
x=a;
y=b;
z = c;
}
 	
3D operator++()
{
X=X + 1;
y=y + 1;
z=z + 1;
return *this;
}
3D operator ++(int)
{
3D t = *this;
X=X + 1;
y = y + 1;
z = z + 1;
return t;
}
3D show()
{
cout<<"The elements are:\n" cout<<"x:"<<this-x<<", y:<<this-y <<",z:"<<this-z;
 	
} ;
int main()
{
3D pt1(2,4,5), pt2(7,1,3);
cout<<"Point one's dimensions before increment are:"<< ptl.show();
++ptl;
cout<<"Point one's dimensions after increment are:"<< ptl.show();
cout<<"Point two's dimensions before increment are:"<< pt2.show();
pt2++;	

cout<<"Point two's dimensions after increment are:"<< pt2.show();
return 0;
}	

b) #include <iostream.h>
// C++ program to show unary
// operator overloading
#include <iostream>
using namespace std;
 	
class Distance {
public:
    int feet, inch;
 	
    // Constructor to initialize
    // the object's value
    Distance(int f, int i)
    {
        this->feet = f;
        this->inch = i;
    }
 	
    // Overloading(-) operator to
    // perform decrement operation
    // of Distance object
    void operator-()
    {
        feet--;
        inch--;
        cout << "\nFeet & Inches(Decrement): " <<
                feet << "'" << inch;
    }
};
 	
// Driver Code
int main()
{
    Distance d1(8, 9);
 	
    // Use (-) unary operator by
    // single operand
    -d1;
    return 0;
}	

(Space for Answers)

XV References/Suggestions for further reading

1. https://www.geeksforgeeks.org/types-of-operator-overloading-in-cpp/2.
2. http://www.careerride.com/C++-what-is-overloading-unary-operator.aspx
3. http://www.leamcpp.com/cpp-tutorial/95-overloading-unary-operators/
4. http://web.itu.edu.tr/bkurt/Courses/blg252e/blg252e_mod05.pdf
5. https://www.tutorialride.com/cpp-operator-overloading-programs/unary-operator- overloading-c-program.htm
6. https://www.tutorialride.com/cpp-operator-overloading-programs/accept- display-compare-time-with-operator-overloading.htm
7. https://www.tutorialride.com/cpp-operator-overloading-programs/overload-unary- minus-operator-c-program.htm

XVI Assessment Scheme

Performance Indicators		Weightage
Process Related: 30 Marks		60 %
1	Logic formation	10%
2	Appropriate use of arithmetic expressions, operators and basic Input /output function.	20%
3	Debugging ability	20%
4	Follow ethical practices.	10%
Product Related: 20 Marks		**40%**
5	Expected Output	20%
6	Submitting the Manual in time	10%
7	Answer to sample questions	10%
Total (50 Marks)		**100 %**

Marks Obtained			**Dated signature of Teacher**
Process Related (30)	**Product Related (20)**	**Total (50)**	

Practical No.20: Write programs to overload binary operator using-

- **Member function**
- **Friend function**

I Practical Significance:

The concept of operator overloading helps to assign the new meaning to the existing operator and helps to extend the concept of polymorphism.

II Industry/Employer Expected Outcome(s)

This practical is expected to develop the following skills as:

1. provide additional meaning to the unary operator using member function and friend function.
2. Define and use the overloaded operator function in the class.

III Course Level Learning Outcome(s)

Implement Polymorphism in C++.

IV Laboratory Learning Outcome(s)

1. Implement binary operator overloading using member function.
2. Implement binary operator overloading using friend function.

V Relevant Affective Domain related outcome(s)

1. Select proper programming environment in C++.
2. Follow ethical practices.

VI Relevant Theoretical Background

Binary Operator Overloading:

An operator which contains two operands to perform a mathematical operation is called the Binary Operator Overloading. It is a polymorphic compile technique where a single operator can perform various functionalities by taking two operands from the programmer or user. There are multiple binary operators like +, -, *, /, etc., that can directly manipulate or overload the object of a class.

For example, suppose we have two numbers, 5 and 6; and overload the binary (+) operator. So, the binary (+) operator adds the numbers 5 and 6 and returns 11. Furthermore, we can also perform subtraction, multiplication, and division operation to use the binary operator for various calculations.

Syntax of the Binary Operator Overloading using member function

Following is the Binary Operator Overloading syntax in the C++ Programming language.

```
return_type :: operator binary_operator_symbol (arg)
{
// function definition
}
```

Here,

return_type: It defines the return type of the function.

operator: It is a keyword of the function overloading.

binary_operator_symbol: It represents the binary operator symbol that overloads a function to perform the calculation.

arg: It defines the argument passed to the function.

Operator Overloading with Friend Functions:

Another method for overloading operators with non-member functions designated as friends inside the class is friend functions. Although friend functions are not class members, they can access the class's private members.

The general syntax for utilizing a friend function to overload an operator is as follows:

friend return_type operator op(parameters);

The function is declared a 'friend' of the class with this syntax, enabling it to access its private members. The remaining portions of the grammar resemble member function overloading.

Binary Operator Overloading Algorithm/Steps:

```
Step 1: Start the program.
Step 2: Declare the class.
Step 3: Declare the variables and its member function.
Step 4: Using the function get value() to get the two numbers.
Step 5: Define the function operator +() to add two complex numbers.
Step 6: Define the function operator -()to subtract two complex numbers.
Step 7: Define the display function.
Step 8: Declare the class objects objl,obj2 and result.
Step 9: Call the function get value using obj1 and obj2
Step 10: Calculate the value for the object result by calling the
         function operator+ and operator-.
Step 11: Call the display function using obj1 and obj2 and result.
Step 12: Return the values.
Step 13: Stop the program
```

VII Resources Required

Sr. No	Name Resource	Specification	Quantity	Remarks
1	Computer System	Any desktop or laptop computer with basic configuration	One computer system for each student	
2	Operating system	Windows /LINUX	One for each computer system	
3	Software	Turbo C++ Version 3.0 or any other	One for each computer system	

VIII Precautions to be followed

1. Handle computer system and peripherals with care.
2.Follow safety practices.

IX Exercise:

1.What is the difference between binary operator overloading with member function and binary operator overloading with friend function in C++?State output of the following code:

```
#include <iostream.h>
class Distance {
 private:
int feet;
inches;
public:
// required constructors Distance() {
feet= 0;
inches= 0;
Distance(int f, inti)
{
 feet= f;
inches= i;
}


void displayDistance() {
cout << "F: "<<feet<<" I:"<< inches <<endl;

// overloaded minus (-) operator Distance operator- ()
{
feet= -feet;
inches= -inches;
return Distance(feet, inches);
} ;
int main()
Distance Dl(ll, 10), D2(-5, 11);
-Dl;
Dl.displayDistance();
```

```
-D2;
D2.displayDistance();
return 0;
}
class overloading {
int value;
public:

void setValue(int temp) {
value= temp;
overloading operator+(overloading ob)
{
overloading t;
t.value =value+ ob.value;
return (t);
}

void display()
{
cout <<value<< endl;
}
} ;

void main() {
overloading objl, obj2, result;
int a, b;
cout << "Enter the value of Complex Numbers a,b:";
cin >> a>>b;
objl.setValue(a);
obj2.setValue(b);
result= objl + obj2;
cout << "Input Values:\n";
objl.display();
obj2.display();
cout << "Result:";
result.display();
getch();
}
```

X C++ code:

Write "C++" Code for above exercise on the blank pages attached at the end of practical.

XI Resources Used

Sr. No.	Name of Resource	Suggested Broad Specification	Quantity

XII Result(s)

…………………………………………………………………………………………

..………………………………………………………………………………………

..………………………………………………………………………………………

XIII Conclusion

…………………………………………………………………………………………

…………………………………………………………………………………………

XIV Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

(Note: for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. Write a C++ program to add two complex numbers using operator overloading by a friend function.
2. Write a C++ program to subtract two complex numbers using operator overloading by using member function.
3. Write a C++ program to compare two strings using'==' operator overloading.
4. Complete the following table:

Program Code	**Write & justify Output**

a)// Write a program to demonstrate binary operator overloading
#include <iostream>
using namespace std;
class complex
{
   int a, b;
   public:
 	
   void get_data(){
      cout << "Enter the value of Complex Numbers a, b: ";
      cin >> a >> b;
   }
 	
   complex operator+(complex ob)// overaloded operator	

```
function +
   {
      complex t;
      t.a = a + ob.a;
      t.b = b + ob.b;
      return (t);
   }

   complex operator-(complex ob)// overaloded operator
function -
   {
      complex t;
      t.a = a - ob.a;
      t.b = b - ob.b;
      return (t);
   }

   void display(){
      cout << a << "+" << b << "i" << "\n";
   }
};

int main()
{
   complex obj1, obj2, result, result1;
   obj1.get_data();
   obj2.get_data();

   result = obj1 + obj2;
   result1 = obj1 - obj2;

   cout << "\n\nInput Values:\n";

   obj1.display();
   obj2.display();

   cout << "\nResult:";

   result.display();
   result1.display();

   return 0;
}
```

(Space for Answers)

XV References/Suggestions for further reading

1. https://www.javatpoint.com/binary-operator-overloading-in-cpp
2. https://www.tutorialride.com/cpp-operator-overloading-programs/demonstrating-operator-overloading-by-using-friend-function.htm
3. https://www.tutorialride.com/cpp-operator-overloading-programs/overload- arithmetic-insertion-and-extraction-operators.htm
4. https://www.tutorialride.com/cpp-operator-overloading-programs/compare-perform-arithmetic-operations-on-two-fractions.htm
5. http://www.includehelp.com/cpp-programs/cpp-program-to-add-two- objects-using-binary-plus-operator-overloading.aspx

XVI Assessment Scheme

	Performance Indicators	**Weightage**
	Process Related: 30 Marks	60 %
1	Logic formation	10%
2	Appropriate use of arithmetic expressions, operators and basic Input /output function.	20%
3	Debugging ability	20%
4	Follow ethical practices.	10%
	Product Related: 20 Marks	**40%**
5	Expected Output	20%
6	Submitting the Manual in time	10%
7	Answer to sample questions	10%
	Total (50 Marks)	**100 %**

Marks Obtained			**Dated signature of Teacher**
Process Related (30)	**Product Related (20)**	**Total (50)**	

Practical No.21: Write programs to implement virtual function

I Practical Significance:

The concept of virtual function helps to learn the concept of run time polymorphism in C++.

II Industry/Employer Expected Outcome(s)

This practical is expected to develop the following skills as:

1. Implementation of function overriding
2. Same function can be used for different purpose at run time.

III Course Level Learning Outcome(s)

Implement Polymorphism in C++.

IV Laboratory Learning Outcome(s)

Develop program using virtual function.

V Relevant Affective Domain related outcome(s)

1.Select proper programming environment in C++.
2.Follow ethical practices.

VI Relevant Theoretical Background:

Virtual Function:

A virtual function (also known as virtual methods) is a member function that is declared within a base class and is re-defined (overridden) by a derived class. When you refer to a derived class object using a pointer or a reference to the base class, you can call a virtual function for that object and execute the derived class's version of the method.

Virtual functions ensure that the correct function is called for an object, regardless of the type of reference (or pointer) used for the function call. They are mainly used to achieve Runtime polymorphism.

Functions are declared with a virtual keyword in a base class. The resolving of a function call is done at runtime.

Rules for Virtual Functions:

The rules for the virtual functions in C++ are as follows:

- Virtual functions cannot be static.
- A virtual function can be a friend function of another class.
- Virtual functions should be accessed using a pointer or reference of base class type to achieve runtime polymorphism.
- The prototype of virtual functions should be the same in the base as well as the derived class.

- They are always defined in the base class and overridden in a derived class. It is not mandatory for the derived class to override (or re-define the virtual function), in that case, the base class version of the function is used.
- A class may have a virtual destructor but it cannot have a virtual constructor.

Compile time (early binding) VS runtime (late binding) behaviour of Virtual Functions

Consider the following simple program showing the runtime behaviour of virtual functions.

```
// C++ program to illustrate
// concept of Virtual Functions
 #include <iostream>
using namespace std;
 class base
{
public:
   virtual void print()
{
cout << "print base class\n";
}
void show()
{
cout << "show base class\n";
}
};

class derived : public base
{
public:
   void print()
{
 cout << "print derived class\n";
}
 void show()
{
cout << "show derived class\n";
}
};

int main()
{
   base* bptr;
   derived d;
   bptr = &d;
    // Virtual function, binded at runtime
   bptr->print();
    // Non-virtual function, binded at compile time
   bptr->show();
```

```
        return 0;
    }
```

VII Resources Required

Sr. No	Name Resource	Specification	Quantity	Remarks
1	Computer System	Any desktop or laptop computer with basic configuration	One computer system for each student	
2	Operating system	Windows /LINUX	One for each computer system	
3	Software	Turbo C++ Version 3.0 or any other	One for each computer system	

VIII Precautions to be followed

1.Handle computer system and peripherals with care.
2.Follow safety practices.

IX Exercise:

1.What is function overriding?
2. Write output for following code:

```
#include<iostream.h>
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()
{
   // An object of class C
   C c;
    // A pointer of class B pointing
   // to memory location of c
   B* b = &c;

   // this line prints "C::fun() called"
   b->fun();

   getchar(); // to get the next character
   return 0;
}
```

3. There are the following statements that are given below, which of them are correct about virtual function in C++?

A. The virtual function is the type of runtime polymorphism in C++.
B. The virtual function is used in inheritance where a function is declared in the base and it is overridden in child class.
C. For virtual function implementation we need did not require any special keyword.
D. All of the above
Options:

1. A
2. A and B
3. A, B, and C
4. D

X C++ code:

Write "C++" Code for above exercise on the blank pages attached at the end of practical.

XI Resources Used

Sr. No.	Name of Resource	Suggested Broad Specification	Quantity

XII Result(s)

……………………………………………………………………………………………………

……………………………………………………………………………………………………

XIII Conclusion

……………………………………………………………………………………………………

……………………………………………………………………………………………………

……………………………………………………………………………………………………

XIV Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

(Note: for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. Define a class parent in which use read function, define another class child use same read function. Display the data of both the read function on output screen using virtual function.

2. Write a program which shows the use of function overriding.

3. Complete the following table:

Program Code	**Write & justify Output**

a) #include <iostream>
using namespace std;
 	
class Base {
  public:
   virtual void print() {
      cout << "Base Function" << endl;
   }
};
 	
class Derived : public Base {
  public:
   void print() override {
      cout << "Derived Function" << endl;
   }
};
 	
int main() {
   Derived derived1;
 	
   // pointer of Base type that points to derived1
   Base* base1 = &derived1;
 	
   // calls member function of Derived class
   base1->print();
 	
   return 0;
}	

b. #include <iostream>
using namespace std;
 	
class Base {	

```
public:
   virtual void display()
   {
      cout << "Base class function";
   }
};

class Child : public Base {
public:
   virtual int display()
   {
      cout << "Child class function";
   }
};

int main()
{
   Child C;

   C.display();

   return 0;
}
```

(Space for Answers)

XV References/Suggestions for further reading

1. https://www.geeksforgeeks.org/virtual-function-cpp/
2. https://www.geeksforgeeks.org/virtual-functions-in-derived-classes-in-cpp/?ref=lbp
3. https://www.programiz.com/cpp-programming/virtual-functions
4. https://www.includehelp.com/cpp-programming/virtual-function-aptitude-questions-and-answers.aspx

XVI Assessment Scheme

	Performance Indicators	**Weightage**
	Process Related: 30 Marks	60 %
1	Logic formation	10%
2	Appropriate use of arithmetic expressions, operators and basic Input /output function.	20%
3	Debugging ability	20%
4	Follow ethical practices.	10%
	Product Related: 20 Marks	**40%**
5	Expected Output	20%
6	Submitting the Manual in time	10%
7	Answer to sample questions	10%
	Total (50 Marks)	**100 %**

Marks Obtained			**Dated signature of Teacher**
Process Related (30)	**Product Related (20)**	**Total (50)**	

Practical No.22: Write programs to implement pure virtual function

I Practical Significance:

The concept of pure virtual function helps to learn the concept of run time polymorphism in C++.

II Industry/Employer Expected Outcome(s)

This practical is expected to develop the following skills as:
A pure virtual function is implemented by classes that are derived from an Abstract class.

III Course Level Learning Outcome(s)

Implement Polymorphism in C++.

IV Laboratory Learning Outcome(s)

Develop program using pure virtual function

V Relevant Affective Domain related outcome(s)

1. Select proper programming environment in C++.
2. Follow ethical practices.

VI Relevant Theoretical Background:

Pure Virtual Function:

Sometimes implementation of all functions cannot be provided in a base class because we don't know the implementation. Such a class is called an abstract class. For example, let Shape be a base class. We cannot provide the implementation of function draw() in Shape, but we know every derived class must have an implementation of draw(). Similarly, an Animal class doesn't have the implementation of move() (assuming that all animals move), but all animals must know how to move. We cannot create objects of abstract classes.

A pure virtual function (or abstract function) in C++ is a virtual function for which we can have an implementation, But we must override that function in the derived class, otherwise, the derived class will also become an abstract class. A pure virtual function is declared by assigning 0 in the declaration.

Example of Pure Virtual Functions
C++

```
// An abstract class
class Test {
   // Data members of class
public:
        // Pure Virtual Function
        virtual void show() = 0;

        /* Other members */
    };
```

A pure virtual function is implemented by classes that are derived from an Abstract class.

VII Resources Required

Sr. No	Name Resource	Specification	Quantity	Remarks
1	Computer System	Any desktop or laptop computer with basic configuration	One computer system for each student	
2	Operating system	Windows /LINUX	One for each computer system	
3	Software	Turbo C++ Version 3.0 or any other	One for each computer system	

VIII Precautions to be followed

1. Handle computer system and peripherals with care.
2. Follow safety practices.

IX Exercise:

1.What is pure virtual function?

2. Write output for following code:

```
#include <iostream>
using namespace std;

class Base {
    // private member variable
    int x;

public:
    // pure virtual function
    virtual void fun() = 0;

    // getter function to access x
    int getX() { return x; }
};

// This class inherits from Base and implements fun()
class Derived : public Base {
    // private member variable
    int y;

public:
    // implementation of the pure virtual function
    void fun() { cout << "fun() called"; }
};
```

```
int main(void)
{
   // creating an object of Derived class
   Derived d;

   // calling the fun() function of Derived class
   d.fun();

   return 0;
}
3. #include <iostream>
using namespace std;

class Base {
public:
   virtual void show() = 0;
};

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

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

X C++ code:

Write “C++” Code for above exercise on the blank pages attached at the end of practical.

XI Resources Used

Sr. No.	**Name of Resource**	**Suggested Broad Specification**	**Quantity**

XII Result(s)

....

....

....

XIII Conclusion

....

....

....

XIV Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

(Note: for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. Write a program which show the use of Abstract class.

2.Complete the following table:

Program Code	Write & justify Output

a)#include<iostream>
using namespace std;
class B {
  public:
    virtual void s() = 0; // Pure Virtual Function
};
 	
class D:public B {
  public:
    void s() {
      cout << "Virtual Function in Derived class\n";
    }
};
 	
int main() {
  B *b;
  D dobj;
  b = &dobj;
  b->s();
}	

b)#include <iostream.h>
using namespace std;
 	
class Test {	

```
   int x;

public:
   virtual void show() = 0;
   int getX() { return x; }
};

int main(void)
{
   Test t;
   return 0;
}
```

(Space for Answers)

XV References/Suggestions for further reading

1. https://www.tutorialspoint.com/pure-virtual-functions-and-abstract-classes-in-cplusplus
2. https://www.geeksforgeeks.org/pure-virtual-functions-and-abstract-classes/
3. https://ww.geeksforgeeks.org/c-interview-questions-on-virtual-function-and-abstract-class/

XVI Assessment Scheme

	Performance Indicators	Weightage
	Process Related: 30 Marks	60 %
1	Logic formation	10%
2	Appropriate use of arithmetic expressions, operators and basic Input /output function.	20%
3	Debugging ability	20%
4	Follow ethical practices.	10%
	Product Related: 20 Marks	**40%**
5	Expected Output	20%
6	Submitting the Manual in time	10%
7	Answer to sample questions	10%
	Total (50 Marks)	**100 %**

Marks Obtained			**Dated signature of Teacher**
Process Related (30)	**Product Related (20)**	**Total (50)**	

Practical No.23: Write programs to read and write from/to file using –

- **Constructor**
- **Open()**

I Practical Significance:

Files store the output of a program and help to perform various operations on it. Files help store these data permanently on a storage device.

II Industry/Employer Expected Outcome(s)

Virtual base classes used to prevent multiple instances of a given class from appearing in an inheritance hierarchy when using multiple inheritances.

III Course Level Learning Outcome(s)

Implement Read and Write operation From/to file in C++ program.

IV Laboratory Learning Outcome(s)

1. Write/ Compile/ debug/ Execute simple C++ program using read and write data from a file.

V Relevant Affective Domain related outcome(s)

1. Select proper programming environment in C++.
2. Follow ethical practices.

VI Relevant Theoretical Background

File Handling:

1. This concept in C++ language is used for store a data permanently in computer.
2. Using file handling we can store our data in Secondary memory (Hard disk).

Standard File handling Classes

1. Ofstream: This file handling class in C++ signifies the output file stream and is applied to create files for writing information to files.
2. lfstream: This file handling class in C++ signifies the input file stream and is applied for reading information from files.
3. Fstream: This file handling class in C++ signifies the file stream generally, and has the capabilities for representing both ofstream and ifstream.

Opening a File:

1. The first operation generally performed on an object of one of these classes to use a file is the procedure known as to opening a file.
2. An open file is represented within a program by a stream and any input or output task performed on this stream will be applied to the physical file associated with it.
3. The syntax of opening a file in C++ is:
   open (filename, mode);
4. There are some mode flags used for file opening. ios::app: append mode
   ios::ate: open a file in this mode for output and read/write controlling to the end of the file 10s::m: open file in this mode for reading ios::out: open file in this mode for writing
   ios::trunk: when any file already exists, its contents will be truncated before file opening

Closing a File:

1. When any C++ program terminates, it automatically flushes out all the streams releases all the allocated memory and closes all the opened files.
2. But it is good to use the close() function to close the file related streams and it is a member of ifsream, ofstream and fstream objects.
3. The structure of using this function is:

void

close(); **General functions used for**

file handling:

1. open(): To create a file
2. close(): To close an existing file
3. get(): to read a single character from the file
4. put(): to write a single character in the file
5. read(): to read data from a file
6. write(): to write data into a file

VII Resources Required

Sr. No	Name Resource	Specification	Quantity	Remarks
1	Computer System	Any desktop or laptop computer with basic configuration	One computer system for each student	
2	Operating system	Windows /LINUX	One for each computer system	

3	Software	Turbo C++ Version 3.0 or any other	One for each computer system	

VIII Precautions to be followed

1. Handle computer system and peripherals with care.
2. Follow safety practices.

IX Exercise:

1. Write output for following code

```
#include <iostream>
#include <fstream>
using namespace std;
int main () {
ofstream filestream("testout.txt");
if (filestream.is_open())
{
filestream << "Welcome to javaTpoint.\n";
filestream << "C++ Tutorial.\n";
filestream.close();
}
else cout <<"File opening is fail.";
return 0;
```

2.Write a program in C++ in which open a file read and write mode ad display the content on
3. output screen using function open()
4.Write a program in C++ in which open a file read and write mode ad display the content
on output screen using constructor.

X C++ code:

Write “C++” Code for above exercise on the blank pages attached at the end of practical.

XI Resources Used

Sr. No.	Name of Resource	Suggested Broad Specification	Quantity

XII Result(s)

……………………………………………………………………………………………………

…..…………………………………………………………..…………………………………………

…..…………………………………………………………..…………………………………………

XIII Conclusion

………………………………………………………………………………………………………

………………………………………………………………………………………………………

………………………………………………………………………………………………………

XIV Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

(Note: for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. Which operator is used to insert the data into file?
a) >> b) <<
c) < d) none of the mentioned

2. Which function is used to position back from the end of file object?
a) seekg () b) seekp()
c) both seekg() & seekp() d) none of the mentioned

3. Write a C++ program to Read and Display File's Content.

4. Write a C++ program to List Files in Current Directory.

5.What is the return type open() method?
a)int
b)char
c)bool
d) float

6. Complete the following table:

Program Code	**Write & justify Output**

a)#include<iostream.h>
#include<fstream.h>
#include<cctype.h>
int main ()
{
ifstream ifile; ifile.open ("text.txt");
char last; ifile.ignore (256, ' '); last= ifile.get();
cout << "Your initial is"<< last<< '\n';
ifile.close();
return 0	

```
    }
```

```
b) #include <iostream>
   #include <fstream>
   using namespace std;
   int main ()
   {
      int length;
      char * buffer;
      ifstream is;
      is.open ("sample.txt", ios :: binary );
      is.seekg (0, ios :: end);
      length = is.tellg();
      is.seekg (0, ios :: beg);
      buffer = new char [length];
      is.read (buffer, length);
      is.close();
      cout.write (buffer, length);
      delete[] buffer;
      return 0;
   }
```

(Space for Answers)

XV References/Suggestions for further reading

1. https://www.geeksforgeeks.org/file-handling-c-
2. https://www.javatpoint.com/cpp-files-and-streams
3. https://www.tutorialspoint.com/cplusplus/cpp_files_streams.h

XVI Assessment Scheme

	Performance Indicators	**Weightage**
	Process Related: 30 Marks	60 %
1	Logic formation	10%
2	Appropriate use of Virtual class.	20%
3	Debugging ability	20%
4	Follow ethical practices.	10%
	Product Related: 20 Marks	**40%**
5	Expected Output	20%
6	Submitting the Manual in time	10%
7	Answer to sample questions	10%
	Total (50 Marks)	**100 %**

Marks Obtained			**Dated signature of Teacher**
Process Related (30)	**Product Related (20)**	**Total (50)**	

Practical No.24: Write programs to copy the contents of one file into another file using formatted input/output function.

I Practical Significance:

To copy the file using C++, we read the contents of the source file and write it into the destination file.

II Industry/Employer Expected Outcome(s)

Copying is a file operation that creates a new file which has the same content as an existing file

III Course Level Learning Outcome(s)

Implement copy the contents of one file into another file using formatted input/output Function in C++ program.

IV Laboratory Learning Outcome(s)

Write/ Compile/ debug/ Execute simple C++ program to copy the contents of one file into another file using formatted input/output function.

V Relevant Affective Domain related outcome(s)

1. Select proper programming environment in C++.
2. Follow ethical practices.

VI Relevant Theoretical Background

Formatted I/O in C++

C++ helps you to format the I/O operations like determining the number of digits to be displayed after the decimal point, specifying number base etc.

Example:

- If we want to add + sign as the prefix of out output, we can use the formatting to do so:
- stream.setf(ios::showpos)

  If input=100, output will be +100
- If we want to add trailing zeros in out output to be shown when needed using the formatting:
- stream.setf(ios::showpoint)

  If input=100.0, output will be 100.000

C++ Program to Copy One File into Another File

To copy the text/contents of one file to another file, we should know the basics of reading and writing a text file in C++. To copy the file using C++, we read the contents of the source file and write it into the destination file.

Steps to copy one file to another in C++:
1. Create objects of ifstream and ofstream classes.
2. Check if they are connected to their respective files. If so, go ahead otherwise check the filenames twice.
3. Read the contents of the source file using the getline() method and write the same to the destination using the << operator (i.e. copy each line from ifstream object to ofstream object).
4. Close files after the copy using the close() method.
5. End the program.

VII Resources Required

Sr. No	Name Resource	Specification	Quantity	Remarks
1	Computer System	Any desktop or laptop computer with basic configuration	One computer system for each student	
2	Operating system	Windows /LINUX	One for each computer system	
3	Software	Turbo C++ Version 3.0 or any other	One for each computer system	

VIII Precautions to be followed

2. Handle computer system and peripherals with care.
3. Follow safety practices.

IX Exercise:

1. Write programs to copy the contents of one file into another file using formatted input/output function

X C++ code:

Write "C++" Code for above exercise on the blank pages attached at the end of practical.

XI Resources Used

Sr. No.	Name of Resource	Suggested Broad Specification	Quantity

XII Result(s)

……………………………………………………………………………………………

..…………………………………………………………………………………………

XIII Conclusion

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

XIV Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

(**Note:** for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. Write a C++ program to Read Content From One File and Write it Into Another File.
2. Write a C++ Program to Copy the Contents of One File Into Another File.
3. Complete the following table:

Program Code	Write & justify Output
a) What is the use of ios::trunc mode? a) To open a file in input mode b) To open a file in output mode c) To truncate an existing file to half d) To truncate an existing file to zero	
b) Which of the following is not a file opening mode a. ios::ate b. ios::nocreate c. ios::noreplace d. ios::truncate	

State output

(Space for Answers)

XV References/Suggestions for further reading

1. https://www.geeksforgeeks.org/cpp-program-to-copy-one-file-into-another-file/
2. https://www.tutorialspoint.com/c-program-to-copy-the-contents-of-one-file-to-another file

XVI Assessment Scheme

Performance Indicators		Weightage
Process Related: 30 Marks		60 %
1	Logic formation	10%
2	Appropriate use of File copy functions.	20%
3	Debugging ability	20%
4	Follow ethical practices.	10%
Product Related: 20 Marks		**40%**
5	Expected Output	20%
6	Submitting the Manual in time	10%
7	Answer to sample questions	10%
Total (50 Marks)		**100 %**

Marks Obtained			**Dated signature of Teacher**
Process Related (30)	**Product Related (20)**	**Total (50)**	

Practical No.25: Write file programs to implement sequential input and output operations on file.

I Practical Significance:

Input and output operations are performed using streams, which are sequences of bytes that can be read from or written to.

II Industry/Employer Expected Outcome(s)

File can only be accessed sequentially from the beginning to the end.

III Course Level Learning Outcome(s)

Implement sequential input and output operations on file in C++ program.

IV Laboratory Learning Outcome(s)

Write/ Compile/ debug/ Execute simple C++ program to implement sequential input and output operations on file.

V Relevant Affective Domain related outcome(s)

1. Select proper programming environment in C++.
2. Follow ethical practices.

VI Relevant Theoretical Background

C++ Sequential I/O Operations with Files:
The file stream classes support a number of member functions for performing input and output operations on files. The functions get() and put() are capable of handling a single character at a time. The function getline() lets you handle multiple characters at a time. Another pair of functions, i.e., read() and write(), are capable of reading and writing blocks of binary data.

The get(), getline(), and put() functions:
The functions get() and put() are byte-oriented. That is, get() will read a byte of data, and put() will write a byte of data. The get() method has many forms, but the most commonly used version is shown here, along with put():

```
istream &get(char &ch);
ostream &put(char ch);
```

The get() function takes a single character as its input and reads it from the associated stream. It then stores the character's value in the variable ch. It provides a reference to the stream as its return value. The put() method returns a reference to the stream after it has been updated with the value of ch that was passed to it.

VII Resources Required

Sr. No	Name Resource	Specification	Quantity	Remarks
1	Computer System	Any desktop or laptop computer with basic configuration	One computer system for each student	
2	Operating system	Windows /LINUX	One for each computer system	
3	Software	Turbo C++ Version 3.0 or any other	One for each computer system	

VIII Precautions to be followed

1. Handle computer system and peripherals with care.
2. Follow safety practices.

IX Exercise:

1. Write file programs to implement sequential input and output operations on file.

X C++ code:

Write "C++" Code for above exercise on the blank pages attached at the end of practical.

XI Resources Used

Sr. No.	Name of Resource	Suggested Broad Specification	Quantity

XII Result(s)

…………………………………………………………………………………………………

…………………………………………………………………………………………………

XIII Conclusion

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

XIV Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

(Note: for all relevant programming exercise use blank pages provided or attach more pages if needed.)

a. Write a C++ program to Read Content from One File and Write it Into another File.

b. Write a C++ Program to Copy the Contents of One File Into Another File.

c. Complete the following table:

Program Code	Write & justify Output

a) #include<iostream>
#include <fstream>using namespace std;
int main()
{
int length;
char * buffer ;
ifstream is;is.open ("sample.txt",
ios:: binary);
is.seekg (0, ios:: end);
length= is.tellg();
is.seekg (0, ios:: beg);
 buffer= new char [length];
is.read(buffer,length);is.close();
cout.write(buffer,length);
delete[] buffer ;
return 0;
}
a)This is sample
b) sample
c) Error
d) Runtime error	

b) #include <iostream>
#include <fstream>
int main() {
  // Writing data sequentially
  std::ofstream outfile("data.txt");
  if (outfile.is_open()) {
    outfile << "Record 1\n";
    outfile << "Record 2\n";
    outfile << "Record 3\n";
    outfile.close();
  }
 	
  // Reading data sequentially
  std::ifstream infile("data.txt");
  if (infile.is_open()) {
    std::string line;
    while (std::getline(infile, line)) {	

`            std::cout << line << std::endl;`
`        }`
`        infile.close();`
`    }`
`    return 0;`
`}`	

(Space for answers)

XVI References/Suggestions for further reading

1. https://codescracker.com/cpp/cpp-sequential-io-with-files.htm
2. https://programmingknow.com/cpp-sequential-input-output/

XVII Assessment Scheme

Performance Indicators		Weightage
Process Related: 30 Marks		60 %
1	Logic formation	10%
2	Appropriate use of File copy functions.	20%
3	Debugging ability	20%
4	Follow ethical practices.	10%
Product Related: 20 Marks		**40%**
5	Expected Output	20%
6	Submitting the Manual in time	10%
7	Answer to sample questions	10%
Total (50 Marks)		**100 %**

Marks Obtained			**Dated signature of Teacher**
Process Related (30)	**Product Related (20)**	**Total (50)**	

Practical No.26: Write programs to perform input/output operations on Binary Files.

I Practical Significance:

Binary file is a file with data stored in raw format, the way it is stored in memory.

II Industry/Employer Expected Outcome(s)

Binary files can be used to store any data. The data inside a binary file is stored as raw bytes, which is not human readable.

III Course Level Learning Outcome(s)

Implement Input/output operation on Binary file in C++ program.

IV Laboratory Learning Outcome(s)

1. Write/ Compile/ debug/ Execute simple C++ program to perform input/output operations on Binary Files.

V Relevant Affective Domain related outcome(s)

a. Select proper programming environment in C++.
b. Follow ethical practices.

VI Relevant Theoretical Background

Binary File

Binary file is a file with data stored in raw format, the way it is stored in memory.
For example, numbers are stored in binary in memory. They are not converted to text (ASCII characters) when writing to binary file.
The data in binary format is not human readable and cannot be read or modified using text editors.
See following example to understand this.
Consider integer value 65536. When read from keyboard, it is entered using keys ‘6’, ‘5’, ‘5’, ‘3’, ‘6’ as ASCII characters. But in memory, data is stored in binary format as 0x’0001FFFF’.
Integer 65536 stored as text (using 5 ASCII characters)

Character	‘6’	‘5’	‘5’	‘3’	‘6’
ASCII in Binary	00 00 01 10	00000 101	00000 101	00000 011	00000 110
ASCII in Hexa	06	05	05	03	06

Integer 65536 stored as binary in memory (assuming long int, using 4 bytes):

In Binary	00000000	00000001	11111111	11111111
In Hexa	00	01	FF	FF

When writing the output on screen, it requires converting data stored in memory to text.
Thus, 0x’0001FFFF’ is converted to 5 characters ‘6’, ‘5’, ‘5’, ‘3’, ‘6’.

Thus, working with text files requires conversion from binary to text or vice versa.

When data is stored in binary file, it does not require such conversion. Integer 65536 is stored as 0x’0001FFFF’ using 4 bytes without any conversion.

Working with binary files

Working with files in C++ requires the use of file-oriented streams based on classes:
ifstream, ofstream and fstream.

- ifstream (input file stream) for reading only
- ofstream (output file stream) for writing only
- fstream (file stream) for reading as well as writing

Operations on Binary File:

Opening Binary file-

Opening binary file is similar to opening text files, but requires specifying ios::binary as additional open mode. A file stream object can be opened in one of two ways.

First, supply a file name along with an i/o mode parameter to the constructor when declaring an object:

Syntax: FileStream FileObject (const char *filename[, int mode][, int prot]);

Ex. ifstream myFile (“data.bin”, ios::in | ios::binary);

Second, call open method after declaring a file stream object.

Syntax of open(): void open(const char *filename[, int mode][, int prot]);

Ex. ofstream myFile; … myFile.open ("data2.bin", ios::out | ios::binary);

Ex. fstream myFile ("data.bin", ios::in | ios::out | ios::binary);

Either approach will work equally well with ifstream, ofstream or fstream object. In the syntax given here, [] indicates that the mode and prot arguments are optional. Default mode is ios::in for ifstream, ios::out for ofstream and ios::in | ios::out for fstream.

For binary I/O, opening with mode flag ios::binary indicate to suppress formatting and conversion.

Note: When working with text files, one may omit the second parameter (the i/o mode parameter) to use the default mode. However, in order to manipulate binary files, it is necessary to specify the i/o mode also along with ios::binary mode.

Writing data to binary file-

Method write() can be used to write data to binary file associated with ofstrem or fstrem object.

The write() method causes specified size of bytes to be written from the given memory location buf to binary
file on given stream and moves the file pointer size bytes ahead.

Following example writes two numbers in file. First is of type int, and second is of type double. Note that it requires typecasting to (char *) as per syntax. To know the size of an item to be written, one may use sizeof() operator.

Example code segment:

```
int i = 1234;

double d = 12.34;
ofstream fout("data.bin", ios::out | ios::binary);
fout.write((char *)&i, sizeof(int));
fout.write((char *)&d, sizeof(double));
```

Reading data from binary file-

Use read() method to read from a binary file associated with fstream or ifstream object.

Function read() extracts a given number of bytes from the specified stream and store it into the memory pointed to by the first parameter which is a C-type array of characters.

Following example reads two numbers (int and double) from binary file. Note typecasting (char *) used to convert address of int or double to pointer to char.

Example code:
```
int i;
double d;
ifstream fin("data.bin", ios::in | ios::binary);
fin.read((char *)&i, sizeof(int));
fin.read((char *)&d, sizeof(double));
```

VII Resources Required

Sr. No	Name Resource	Specification	Quantity	Remarks
1	Computer System	Any desktop or laptop computer with basic configuration	One computer system for each student	
2	Operating system	Windows /LINUX	One for each computer system	
3	Software	Turbo C++ Version 3.0 or any other	One for each computer system	

VIII Precautions to be followed

1. Handle computer system and peripherals with care.
2. Follow safety practices.

IX Exercise:

1. Write program to copy contents of one file to another file using get () and put () methods.

2. Write programs to perform input/output operations on Binary Files.

X C++ code:

Write "C++" Code for above exercise on the blank pages attached at the end of practical.

XI Resources Used

Sr. No.	Name of Resource	Suggested Broad Specification	Quantity

XII Result(s)

…………………………………………………………………………………………………

…………………………………………………………………………………………………

XIII Conclusion

……………………………………………………………………………………………………

……………………………………………………………………………………………………

XIV Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

(Note: for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1 Write C++ Program to write and read objects in binary file using write () and read () methods.

3. Write a C++ program to copy entire content of one binary file at a time to another file using read () and write ()

3. Complete the following table:

Program Code	**Write & justify Output**

a)#include <iostream>
#include<fstream>
using namespace std;
 	
int main()
{
ifstream fin;
fin.open(“file1.pdf”, ios::in | ios::binary);
 	
ifstream fout;
fout.open(“file2.pdf”, ios::out | ios::binary);
if (! (fin.is_open() && fout.is_open()))
{ cout << “Error opening file…”,
exit(1);
}
char ch;
while(!fin.eof())
{ ch = fin.get(); fout.put(ch); }
 	
fin.close();
fout.close();
return 0;
}	

```
b)#include<iostream.h>
#include<stdlib.h>
#include <fstream.h>

class Student
{
  private:
    int number;
    char name[50];
    float gpa;
  public:
    Student(int n, const char *s, float g);
    void save(ofstream& of);
    void load(ifstream& inf);
};

main()
{
  Student me(11321, "Myself", 4.3);
  ofstream myfile;
  myfile.open("silly.dat", ios::binary | ios::out);
  me.save(myfile);
  myfile.close();
  return(0);
}

void Student::save(ofstream& of)
{
  of.write(&number, sizeof(number));
  of.write(name, sizeof(name));
  of.write(&gpa, sizeof(gpa));
}

void Student::load(ifstream& inf)
{
  inf.read(&number, sizeof(number));
  inf.read(name, sizeof(name));
  inf.read(&gpa, sizeof(gpa));
}
```

(Space for Answers)

……
……
……
……
……

……
……

XV References/Suggestions for further reading

1. https://www.geeksforgeeks.org/file-handling-c-classes/
2. https://ebooks.inflibnet.ac.in/itp1/chapter/binary-files-in-c/

XVI Assessment Scheme

	Performance Indicators	**Weightage**
	Process Related: 30 Marks	60 %
1	Logic formation	10%
2	Appropriate use of Virtual class.	20%
3	Debugging ability	20%
4	Follow ethical practices.	10%
	Product Related: 20 Marks	**40%**
5	Expected Output	20%
6	Submitting the Manual in time	10%
7	Answer to sample questions	10%
	Total (50 Marks)	**100 %**

Marks Obtained			**Dated signature of Teacher**
Process Related (30)	**Product Related (20)**	**Total (50)**	