
LNCTE
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
Object oriented programming & Methodology (CS-305)

Lakshmi Narain College of Technology Excellence

Department of Computer Science and Engineering

Lab Manual

Subject Name: Object Oriented Programming Methodology

Course Code: CS 305

Course: B.Tech

Session: 2023-24

Prepared By

TABLE OF CONTENT

Vision and Mission of the Institute

Vision of the institute

To become a pioneer institute in technical education and innovations to build competent technocrats and leaders for the nation.

Mission of the institute

M1. To enhance the academic environment with innovative teaching learning processes and modern tools.

M2. To Practice and nurture high standards of human values, transparency and accountability.

M3. To collaborate with other academic and research institutes as well as industries in order to strengthen education and research.

M4. To uphold skill development for employability and entrepreneurship for interdisciplinary research and innovations.

Vision and Mission of the Department

Vision of the Department

To be a centre of excellence for providing quality technical education to develop future leaders with the aspects of research & computing, Software product development and entrepreneurship.

Mission of the Department

Mission No.	Mission Statements
M1	To offer academic program with state of art curriculum having flexibility for accommodating the latest developments in the areas of computer science engineering.
M2	To conduct research and development activities in contemporary and emerging areas of computer science & engineering.
M3	To inculcate moral values & entrepreneurial skills to produce professionals capable of providing socially relevant and sustainable solutions.

COURSE OUTCOMES: CS-305 Object Oriented Programming Methodology

CO305.1	Apply Procedural programming methodologies for simple arithmetic problem solving.
CO305.2	Illustrate fundamental class operations using C++ program.
CO305.3	Demonstrate inheritance, its types and virtual function with programming examples.
CO305.4	Examine types of polymorphism with C++ programming constructs.
CO305.5	Construct program to understand Exception handling and string manipulation functions in C++.

Course Articulation Matrix

CO's	Program Outcomes(PO's)											
	PO 1	**PO 2**	**PO 3**	**PO 4**	**PO 5**	**PO 6**	**PO 7**	**PO 8**	**PO 9**	**PO 10**	**PO 11**	**PO 12**
CO305.1	3	3	3	2	1	2	2	-	2	1	2	3
CO305.2	3	3	3	1	2	2	2	-	2	1	2	3
CO305.3	3	2	3	2	1	2	1	-	2	1	2	3
CO305.4	3	3	2	2	2	2	2	-	2	1	1	3
CO305.5	3	3	3	1	1	2	2	2	2	1	1	3
	3	**2.8**	**2.8**	**1.6**	**1.4**	**2**	**1.8**	**2**	**2**	**1**	**1.6**	**3**

Program Outcomes as defined by NBA (PO)

Engineering Graduates will be able to:

1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

2. Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

3. Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

4. Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

5. Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

6. The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

7. Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

10. Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12. Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Program Specific Outcomes (PSO)

A graduate of Computer Science and Engineering Program will develop

PSO1: An ability to apply technical knowledge of computer science and engineering fundamentals to become employable in industry.

PSO2: An ability to develop programming skills using modern software tools and techniques.

PSO3: An ability to develop real time projects for problem solving of domains such as Machine Learning, Cyber security, block chain and big data.

PSO4: An ability to grab research, higher studies and entrepreneurship opportunities towards society with moral values and ethics.

Program Educational Objectives (PEO):

PEO 1: Evolve as globally competent computer professionals, researchers and entrepreneurs possessing collaborative and leadership skills, for developing innovative solutions in multidisciplinary domains.

PEO 2: Excel as socially committed computer engineers having mutual respect, effective communication skills, high ethical values and empathy for the needs of society.

PEO 3: Involve in lifelong learning to foster the sustainable development in the emerging areas of technology

List of Program to be performed: -

1	Write a program in C++ to add two numbers accept through keyboard.	CO1
2	Write a program in C++ to swap 2 numbers without using third variable.	CO1
3	Write a program to create a Rectangle class, objects, and implement method to calculate area of circle.	CO2
4	Write a program to Computes time difference of two time period. (Time periods are entered by the user)	CO2
5	Write a program to Single level inheritance and Multilevel inheritance.	CO3
6	Write a program in which a class has three data members: name, roll no, marks of 5 subjects and a member function Assign() to assign the streams on the basis of table given below Avg marks Stream • 90% and more Computers • 80-89% Electronics • 75-79% Mechanical • 70-74% Electrical	CO3
7	Write a program for function overloading and operator overloading.	CO4
8	Write a program for run time polymorphism.	CO4
9	Construct the program to Exception Handling.	CO5
10	Write a C++ program to reverse a given string.	CO5
11	Write a program in C++ for File Handling using ifstream & ofstream class object to write the content in file then to read the content of file.	CO5
12	Write a program in C++ to implement the ATM Management System.	CO5

Object-Oriented Programming

Object-Oriented Programming or OOPs refers to languages that use objects in programming. Object-oriented programming aims to implement real-world entities like inheritance, hiding, polymorphism, etc in programming. The main aim of OOP is to bind together the data and the functions that operate on them so that no other part of the code can access this data except that function.

Overview of C++ language:

1. C++ can be considered as an incremental version of c language which consist all programming language constructs with newly added features of object oriented programming.
2. C++ is structure (procedure) oriented and objects oriented programming language.
3. The file extension of C++ program is ".CPP"
4. Function overloading and operator overloading are possible.
5. Variables can be declared in inline i.e. when required
6. In C++ more emphasis is give on data rather than procedures
7. Polymorphism, encapsulation and inheritance are possible.
8. Data abstraction property is supported by c++.
9. Data access is limited. It can be accessed by providing various visibility modes both for dataand member functions. there by providing data security by data hiding
10. Dynamic binding is supported by c++.
11. It supports all features of c language
12. It can be called as an incremental version of c language

Difference between Procedure Oriented Programming (POP) & Object Oriented Programming (OOP)

Parameter	OOP	POP
Definition	This type of programming language uses objects and classes for creating models.	This programming language uses a step-by-step approach for breaking down a task into a collection of routines and variables by following a

		sequence of instructions.
Abbreviation	Object-Oriented Programming	Procedure Oriented Programming
Approach	Bottom-up approach	Top-down approach
Polymorphism	Method overloading and overriding are used in OOP to achieve polymorphism.	It doesn't support polymorphism.
Inheritance	Supports	Do not support
Code Reusability	Supports	Don't support
Security	Data handling is possible in OOP due to programming.	It is less secure than OOP.
Problem-Solving	Used for solving big problems.	Not suitable for big problems.
Example	C++, JAVA, C#, .NET	C, FORTRAN

OOPs Concepts:

- Class
- Objects
- Data Abstraction
- Encapsulation
- Inheritance
- Polymorphism

- Dynamic Binding
- Message Passing

Syntax :

```
#include <iostream>
using namespace std;
 int main()
{
        //body
        return 0;
}
```

EXPERIMENT-1

Aim: Write a program in C++ to add two numbers accept through keyboard.

```
#include <iostream>
using namespace std;
 int main()
{
        int num1, num2, sum;
        cout << "\n Sum of two numbers :\n";
        cout << "------------------------\n";
        cout << " Input 1st number : ";
        cin >> num1 ;
        cout << " Input 2nd number : ";
        cin >> num2;
        sum = num1 + num2;
        cout <<" The sum of the numbers is : " << sum << endl;
        cout << endl;
        return 0;
}
```

Output :

```
        Sum of two numbers :
        ------------------------
```

Input 1st number : 20
Input 2nd number : 30
The sum of the numbers is : 50

- Write a program to calculate average age of 5 people.
- Write a C++ Program to Check Whether a Number is a Positive or Negative Number.

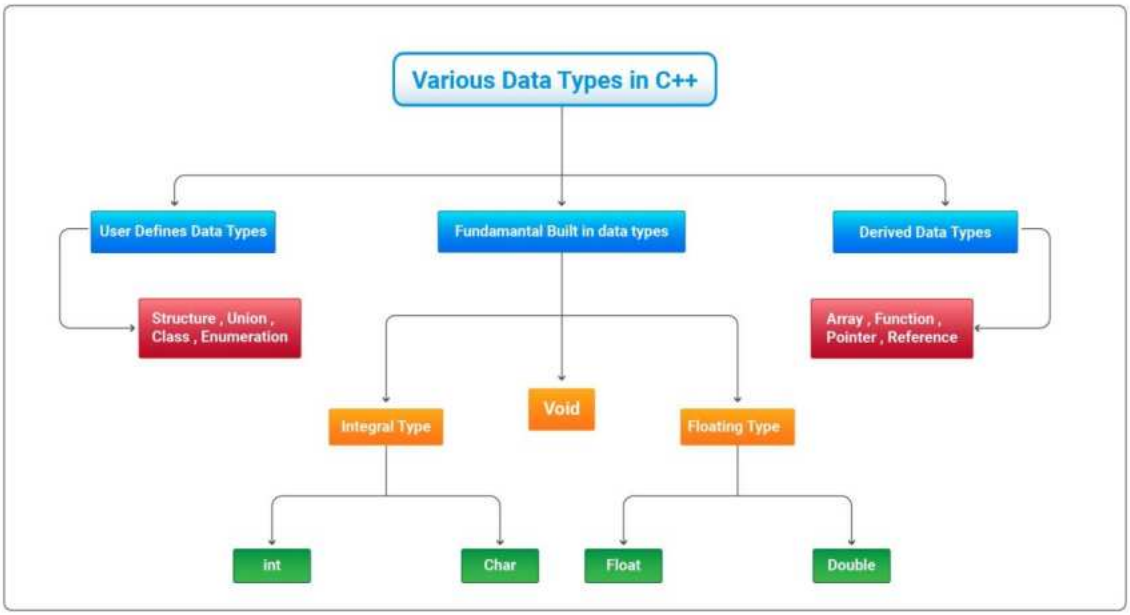
Viva Voce questions-

- What is the difference between OOP and SOP?
- What is Object Oriented Programming?
- Why use OOPs?
- What are the main features of OOPs?

DATA TYPES:

A data type is used to indicate the type of data value stored in a variable. All Cpp compilers support a variety of data types. This variety of data types allows the programmer to select the type appropriate to the needs of the application as well as the machine. ANSI C supports the following classes of datatypes:

- Primary (fundamental) data types.
- Derived data types.
- User-defined data types



Variables:

A named memory location is called variable.

OR

It is an identifier used to store the value of particular data type in the memory. Since variable name is identifier we use following rules which are same as of identifier

Rules for declaring Variables names:

- The first character must be an alphabet or underscore.
- It must consist of only letters, digits and underscore.
- Identifiers may have any length but only first 31 characters are significant.
- It must not contain white space or blank space.
- We should not use keywords as identifiers.
- Upper and lower case letters are different.
- Variable names must be unique in the given scope

Syntax:

Ex: int a,b,a;//is in valid

int a,b;//is valid

Variable declaration: The declaration of variable gives the name for memory location and its size andspecifies the range of value that can be stored in that location.

Syntax:

Data –type variable-name;

Ex-

int a;

int x=20;

float pi=3.14;

OPERATORS AND EXPRESSIONS

An *operator* is a symbol which represents a particular operation that can be performed on data. An *operand* is the object on which an operation is performed.

By combining the operators and operands we form an *expression*. An *expression* is a sequenceof operands and operators that reduces to a single value.

Cpp operators can be classified as

1. Arithmetic operators
2. Relational operators
3. Logical operators
4. Assignment operators
5. Increment or Decrement operators
6. Conditional operator
7. Bit wise operators
8. unary operator
9. Special operators
10. Additional operators in C++

1. ARITHMETIC OPERATORS : All basic arithmetic operators are present in C.

operator	meaning
+	add
-	subtract
*	multiplication
/	division
%	modulo division(remainder)

An arithmetic operation involving only real operands(or integer operands) is called real arithmetic(orinteger arithmetic). If a combination of arithmetic and real is called mixed mode arithmetic.

EXPERIMENT-2

Aim: Write a program in C++ to swap 2 numbers without using third variable.

```
#include <iostream>
using namespace std;
int main()
{
int a,b;
cout<<"Input 2 Number"<<endl;
cin>>a>>b;
cout<<"Before swap a= "<<a<<" b= "<<b<<endl;
a=a*b;
b=a/b;
a=a/b;
cout<<"After swap a= "<<a<<" b= "<<b<<endl;
return 0;
}
```

Output :

```
Input 2 Number
25
36
Before swap a= 25 b= 36
After swap a= 36 b= 25
```

- Write a program to swap two number without using 3rd variable.
- Write a program To Check Whether Number is Even Or Odd.

Viva Voce questions-

- What is data type?
- What is size of int data type in cpp?
- What is variable?
- What is operator?

Introduction of Class:

An object oriented programming approach is a collection of objects and each object consists of corresponding data structures and procedures. The program is reusable and more maintainable.
The important aspect in oop is a class which has similar syntax that of structure.

Class: It is a collection of data and member functions that manipulate data. The data components of class are called data members and functions that manipulate the data are called member functions.

It can also call as blue print or prototype that defines the variables and functions common to all objects of certain kind. It is also known as user defined data type or ADT(abstract data type) A classis declared by the keyword class.

Syntax:-

```
class class_name
{
Access specifier :
                    Variable declarations;


};
```

Objects - A class provides the blueprints for objects, so basically an object is created from a class. We declare objects of a class with exactly the same sort of declaration that we declare variables of basic types.

Syntax:-

Class-name Object-name;

Access Control:

Access specifies or access modifiers are the labels that specify type of access given to members of aclass. These are used for data hiding. These are also called as visibility modes. There are three types of

Access modifiers

1.private 2.public 3.protected

1.Private:

If the data members are declared as private access then they cannot be accessed from other functions outside the class. It can only be accessed by the functions declared within the class. It is declaredby the key word "private" .

2.Public:

If the data members are declared public access then they can be accessed from other functions outside the class. It is declared by the key word public.

3.Protected:

The access level of protected declaration lies between public and private. This accessspecifies is used at the time of inheritance.

Note:- If no access specifier is specified then it is treated by default as private
The default access specifier of structure is public where as that of a class is "private".

Syntax :

```
class class-name
{
public :
        datamember;
public :
        memberfunction();
}
```

EXPERIMENT-3

Aim: Write a program to create a Rectangle class, objects, and implement method to calculate area of circle.

```
#include <iostream>
using namespace std;
class Rectangle{
   private:
      int l, b;
   public:
      void input(int len, int wd){
         l = len;
         b = wd;
      }
      int area(){
         return l * b;
      }
};
int main(){
   Rectangle r1, r2;
   int l,b;
   cout<<"Input value of length and width "<<endl;
   cin>>l>>b;
   r1.input(l,b);
```

```
    r2.input(8, 6);
    cout << "Area of r1: " << r1.area() << endl;
    cout << "Area of r2: " << r2.area() << endl;
}
```

Output :

Input value of length and width

21

9

Area of r1: 189

Area of r2: 48

- Write a Program to Create an object of a class and access class attributes
- Write a program to create a class for student to get and print details of a student.

Viva Voce questions-

- What is an object?
- What is a class?
- What is the difference between a class and a structure?
- What is access modifiers?

EXPERIMENT-4

Aim: Write a program to Computes time difference of two time period. (Time periods are entered by the user)

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

class Time {
public :
  int hour;
  int mins;
  int secs;
};

Time findTimeDifference(Time t1, Time t2);

int main() {
   Time t1, t2, diff;

   cout << "Enter earlier time in hours, minutes and seconds\n";
   cin >> t1.hour >> t1.mins >> t1.secs;

   cout << "Enter current time in hours, minutes and seconds\n";
   cin >> t2.hour >> t2.mins >> t2.secs;

   diff = findTimeDifference(t1, t2);

   cout << "Difference = "<< diff.hour << ":"
      << diff.mins << ":" << diff.secs;
   return 0;
}

Time findTimeDifference(Time t1, Time t2){
```

```
 Time diff;
    if(t2.secs > t1.secs){
        --t1.mins;
        t1.secs += 60;
    }

    diff.secs = t1.secs - t2.secs;
    if(t2.mins > t1.mins) {
        --t1.hour;
        t1.mins += 60;
    }

    diff.mins = t1.mins-t2.mins;
    diff.hour = t1.hour-t2.hour;

    return diff;
}
```

Output :

Enter earlier time in hours, minutes and seconds

5 15 40

Enter current time in hours, minutes and seconds

2 40 14

Difference = 2:35:26

Inheritance

The capability of a class to derive properties and characteristics from another class is called Inheritance. Inheritance is one of the most important features of Object-Oriented Programming.

Inheritance is a feature or a process in which, new classes are created from the existing classes. The new class created is called “derived class” or “child class” and the existing class is known as the “base class” or “parent class”. The derived class now is said to be inherited from the base class.

When we say derived class inherits the base class, it means, the derived class inherits all the properties of the base class, without changing the properties of base class and may add new features to its own. These new features in the derived class will not affect the base class. The derived class is the specialized class for the base class.

Sub Class: The class that inherits properties from another class is called Subclass or Derived Class.

Super Class: The class whose properties are inherited by a subclass is called Base Class or Super class.

Importance of Inheritance in C++

Instead of trying to replicate what already exists, it is always ideal to reuse it since it saves time and enhances reliability. In C++, inheritance is used to reuse code from existing classes.

C++ highly supports the principle of reusability. Inheritance is used when two classes in a program share the same domain, and the properties of the class and its super class should remain the same.

Inheritance is a technique used in C++ to reuse code from pre-existing classes. C++ actively supports the concept of reusability.

Implementing Inheritance in C++

To create a parent class that is derived from the base class below is a syntax that you should follow:

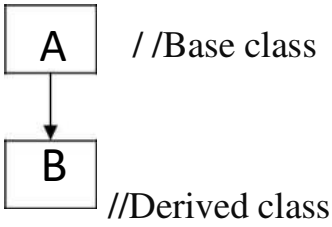
Syntax

```
class <derived_class_name> : <access-specifier> <base_class_name>
{
  / /body
}
```

Types of Inheritance:

1. Single Inheritance
2. Multi level Inheritance
3. Mutiple Inheritance
4. Hybrid inheritance
5. Hierarchical Inheritance.

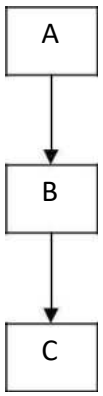
1. SINGLE INHERITANCE: one derived class inherits from only one base class. It is the mostsimplest form of Inheritance.



Syntax:

```
class D
{
……
}
class E : visibility D
{
……..
}
```

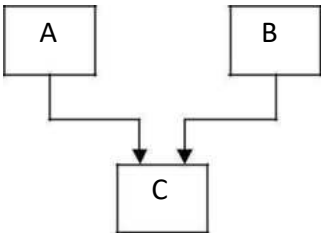
2. MULTILEVEL INHERITANCE: In this type of inheritance the derived class inherits from a class, which in turn inherits from some other class. The Super class for one, is sub class for the other.



Syntax –

```
class A
{
}
class B : public A
{
}
class c: public B
{
}
```

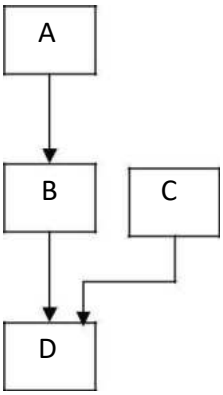
3. Multiple Inheritance:In this type of inheritance a single derived class may inherit from two or morethan two base classes.



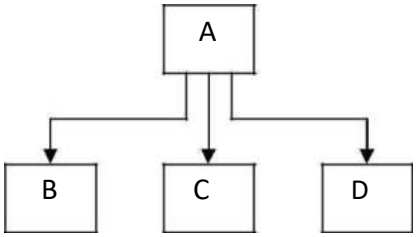
Syntax:

```
class A
{
}
class B
{
}
class C : visibility A, visibility B,….
{
……
}
```

4. Hybrid Inheritance: Hybrid inheritance is combination of two or more inheritances such as single, multiple, multilevel or Hierarchical inheritances.



5. Hierarchical Inheritance:- Inheriting is a method of inheritance where one or more derivedclasses is derived from common base class.



Syntax:

```
class A
{
}
class B : public A
{
}
class C : public A
{
}
class D : public A
{
```

```
}
```

EXPERIMENT-5

Aim: Write a program to Single level inheritance and Multi level inheritance.

Single level inheritance

```
#include <iostream>
using namespace std;
class A
{
public:
   int k = 1000;
   float salary = 80000;
};
class B : public A
{
public:
   float bonus = 8000;
   void ts()
   {
      cout << "Total salary.." << (salary + bonus)
          << endl;
   }
};
int main()
{
   B b1;
   cout << "Salary:" << b1.salary << endl;
   cout << "Bonus:" << b1.bonus << endl;
   b1.ts();
   return 0;
}
```

Output :

```
Salary:80000

Bonus:8000
```

Total salary..88000

Multi level inheritance.

```
#include <iostream>
using namespace std;
class base //single base class
{
        public:
        int x;
        void getdata()
        {
        cout << "Enter value of x= ";
        cin >> x;
        }
};
class derive1 : public base // derived class from base class
{
        public:
        int y;
        void readdata()
        {
           cout << "\nEnter value of y= "; cin >> y;
        }
};
class derive2 : public derive1   // derived from class derive1
{
        private:
```

```
        int z;
        public:
        void indata()
        {
        cout << "\nEnter value of z= "; cin >> z;
        }
        void product()
        {
            cout << "\nProduct= " << x * y * z;
        }
};
int main()
{
    derive2 a;      //object of derived class
    a.getdata();
    a.readdata();
    a.indata();
    a.product();
    return 0;
}
```

Output :

```
Enter value of x= 10
Enter value of y= 20
Enter value of z= 30
```

Product= 6000

EXPERIMENT-6

Aim: Write a program in which a class has three data members: name, roll no, marks of 5 subjects and a member function Assign() to assign the streams on the basis of table given below Avg marks Stream

- **90% and more Computers**
- **80-89% Electronics**
- **75-79% Mechanical**
- **70-74% Electrical**

```
#include <iostream>
#include<string>
using namespace std;
class student
{
public :
   string  name;
   int rollno;
   int i;
   int a;
   int marks[5];
   int total=0;
   int perc=0;
 void assign()
   {
    cout<<"Enter Name of Student"<<endl;
    getline(cin, name);

    cout<<"Enter Roll number of Student"<<endl;
    cin>>rollno;

    cout<<"Enter 5 Subject Marks out of 100"<<endl;
    for(i=0;i<5;i++)
    {
       cin>>marks[i];
```

```
    }

     for(i=0;i<5;i++)
    {
      total +=marks[i];
    }
    perc= total*100/500;
    cout<<"----------Output----------"<<endl;
    cout<<"Name of Student : "<<name<<endl;
    cout<<"Roll number of Student :"<<rollno<<endl;
    cout<<"total :"<<total<<endl;
    cout<<"Percentage :"<<perc<<endl;
   if(perc>=90 && perc<=100)
      {
          cout<<"Stream :Computers"<<endl;
      }
    else if(perc>=80 && perc<=89)
      {
          cout<<"Stream :Electronics"<<endl;
      }
      else if(perc>=75 && perc<=79)
      {
          cout<<"Stream :Mechanical"<<endl;
      }
   else if(perc>=70 && perc<=74)
      {
          cout<<"Stream :Electrical"<<endl;
      }
    }
};

int main()
{
   student st;
   st.assign();
   cout<<"Thanks";
```

```
    return 0;
}
```

Output :

Enter Name of Student

Atul Verma

Enter Roll number of Student

12365

Enter 5 Subject Marks out of 100

80

84

73

71

69

Output

Name of Student : Atul Verma

Roll number of Student :12365

total :377

Percentage :75

Stream :Mechanical

Thanks

- Write program to read and print student's information using two classes and simple inheritance.

- Write a program to read and print employee information using multiple inheritance.

Viva Voce questions-

- What is inheritance? How many types of inheritance?
- What is the difference between multiple and multilevel inheritance?
- What are the advantages of inheritance?
- What is a super class and sub class?

Polymorphism

The word "polymorphism" means having many forms. In simple words, we can define polymorphism as the ability of a message to be displayed in more than one form. A real-life example of polymorphism is a person who at the same time can have different characteristics. A man at the same time is a father, a husband, and an employee. So the same person exhibits different behavior in different situations. This is called polymorphism. Polymorphism is considered one of the important features of Object-Oriented Programming.

Types of Polymorphism

- Compile-time Polymorphism
- Runtime Polymorphism

1. Compile-Time Polymorphism

This type of polymorphism is achieved by function overloading or operator overloading.

A. Function Overloading

When there are multiple functions with the same name but different parameters, then the functions are said to be overloaded, hence this is known as Function Overloading. Functions can be overloaded by changing the number of arguments or/and changing the type of arguments. In simple terms, it is a feature of object-oriented programming providing many functions that have the same name but distinct parameters when numerous tasks are listed less than one function name. There are certain Rules of Function Overloading that should be followed while overloading a function.

Below is the C++ program to show function overloading or compile-time polymorphism:

B. Operator Overloading

C++ has the ability to provide the operators with a special meaning for a data type, this ability is known as operator overloading. For example, we can make use of the addition operator (+) for string class to concatenate two strings. We know that the task of this operator is to add two operands. So a single operator '+', when placed

between integer operands, adds them and when placed between string operands, concatenates them.

Below is the C++ program to demonstrate operator overloading:

EXPERIMENT-7

Aim:_Write a program for function overloading and operator overloading

```
#include <iostream>

using namespace std;

void display(int var1, double var2)

{

   cout << "Integer number: " << var1;

   cout << " and double number: " << var2 << endl;

}

void display(double var)

{

   cout << "Double number: " << var << endl;

}

void display(int var)

{

   cout << "Integer number: " << var << endl;

}

int main() {

   int a;

   double b;

   cout<<"Enter Ist value"<<endl;

   cin>>a;
```

```
    cout<<"Enter IInd value"<<endl;
    cin>>b;
          display(a);
          display(b);
         display(a, b);
    return 0;
}
```

Output :

Enter Ist value

3

Enter IInd value

6.3

Integer number: 3

Double number: 6.3

Integer number: 3 and double number: 6.3

C++ program to overload the binary operator + this program adds two complex numbers

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

class Complex {
  private:
   float real;
   float imag;

  public:
   Complex() : real(0), imag(0) {}          // Constructor to initialize real and imag to 0
   void input() {
      cout << "Enter real and imaginary parts respectively: ";
      cin >> real;
      cin >> imag;
   }

   Complex operator + (const Complex& obj)    // Overload the + operator
{
      Complex temp;
      temp.real = real + obj.real;
      temp.imag = imag + obj.imag;
      return temp;
   }

   void output() {
      if (imag < 0)
         cout << "Output Complex number: " << real << imag << "i";
      else
         cout << "Output Complex number: " << real << "+" << imag << "i";
   }
```

```
};
int main() {
    Complex complex1, complex2, result;
    cout << "Enter first complex number:\n";
    complex1.input();

    cout << "Enter second complex number:\n";
    complex2.input();

   // complex1 calls the operator function
   // complex2 is passed as an argument to the function
    result = complex1 + complex2;
    result.output();

    return 0;
}
```

Output :

```
Enter first complex number:
Enter real and imaginary parts respectively: 2
3.3
Enter second complex number:
Enter real and imaginary parts respectively: 4
3.3
Output Complex number: 6+6.6i
```

- Write a program to calculate area of square, cube and rectangle using function overloading.
- Program to add two numbers using the binary operator overloading.

Viva Voce questions-

- What is polymorphism?
- What is static or compile time polymorphism?

- What is method overloading?
- What is operator overloading?

Runtime Polymorphism

This type of polymorphism is achieved by Function Overriding. Late binding and dynamic polymorphism are other names for runtime polymorphism. The function call is resolved at runtime in runtime polymorphism. In contrast, with compile time polymorphism, the compiler determines which function call to bind to the object after deducting it at runtime.

A. Function Overriding

Function Overriding occurs when a derived class has a definition for one of the member functions of the base class. That base function is said to be overridden.

B. Virtual Function

A virtual function is a member function that is declared in the base class using the keyword virtual and is re-defined (Overridden) in the derived class.

Some Key Points About Virtual Functions:

- Virtual functions are Dynamic in nature.
- They are defined by inserting the keyword “virtual” inside a base class and are always declared with a base class and overridden in a child class
- A virtual function is called during Runtime

EXPERIMENT-8

Aim: Write a program for run time polymorphism

```
// VIRTUAL FUNCTION


#include <iostream>
using namespace std;
class Base {
  public:
   virtual void print() {
       cout << "Base Function" << endl;
   }
};
class Derived : public Base {
  public:
   void print() {
       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;
}
```

Output :

Derived Function

- Write a program for function overriding with data members.
- Write a program for virtual function.

Viva Voce questions-

- What is dynamic or run time polymorphism?
- What is method overriding?
- What is virtual function?
- What is pure virtual function?
- Differentiate between overloading and overriding.

Exception Handling

Errors can be broadly categorized into two types. We will discuss them one by one.

1.Compile Time Errors

2.Run Time Errors

Compile Time Errors – Errors caught during compiled time is called Compile time errors. Compile time errors include library reference, syntax error or incorrect class import.
Run Time Errors - They are also known as exceptions. An exception caught during run time creates serious issues.

Errors hinder normal execution of program. Exception handling is the process of handling errors and exceptions in such a way that they do not hinder normal execution of the system. For example, User divides a number by zero, this will compile successfully but an exception or run time error will occur due to which our applications will be crashed. In order to avoid this we'll introduce exception handling techniques in our code.

Exception Handling Keywords

Exception handling in C++ revolves around these three keywords:

- throw– when a program encounters a problem, it throws an exception. The throw keyword helps the program perform the throw.
- catch– a program uses an exception handler to catch an exception. It is added to the section of a program where you need to handle the problem. It’s done using the catch keyword.
- try– the try block identifies the code block for which certain exceptions will be activated. It should be followed by one/more catch blocks.

EXPERIMENT-9

Aim: Construct the program to Exception Handling.

```
// program to divide two numbers throws an exception when the divisor is 0
#include <iostream>
using namespace std;
int main() {
    double numerator, denominator, divide;
    cout << "Enter numerator: ";
    cin >> numerator;
    cout << "Enter denominator: ";
    cin >> denominator;
    try {
        // throw an exception if denominator is 0
        if (denominator == 0)
            throw 0;
        // not executed if denominator is 0
        divide = numerator / denominator;
        cout << numerator << " / " << denominator << " = " << divide << endl;
    }
    catch (int num_exception) {
        cout << "Error: Cannot divide by " << num_exception << endl;
    }
```

```
    return 0;
}
```

Output 1 :

```
Enter numerator: 72
Enter denominator: 0
ERROR!
Error: Cannot divide by 0
```

Output 2 :

```
Enter numerator: 72
Enter denominator: 3
72 / 3 = 24
```

- Write a program to illustrates multiple catch statements.
- Write a program for the concept of re-throwing an exception.

Viva Voce questions-

- What is the difference between an error and an exception?
- What is Exception Handling?
- What is a try, catch, throw block?
- What is a final and finally block?

What is a String in C++?

A string variable is a group of characters surrounded by double quotations. It is necessary to include the C++ string header < string> at the beginning of the program to use the string data type. Moreover, you must include using namespace std; in order to display the brief name string without utilizing the annoying std::string.

Syntax:

string str_name = " This is the page of Unstop" ;

Here is a list of some in-built string functions in C++

length()	The length of the string is returned by this function.
Size()	This function is used to find the size of the string in terms of bytes.

EXPERIMENT-10

Aim: Write a C++ program to reverse a given string.

```
#include <iostream>
#include <string>

using namespace std;

int main()
{
   string str;
   char temp;
   cout<<"Enter any string:";
   getline (cin, str);

   int len = str.length();

   for (int i = 0; i < len / 2; i++)
{
      temp = str[i];
      str[i] = str[len - i - 1];
      str[len - i - 1] = temp;
   }
   cout << str << endl;
   return 0;
}
```

Output :

Enter any string: gaurav saxena

anexas varuag

- Write a program to print a string entered by user.
- Write a program to enter a string s1 and copy it to another string s2.

Viva Voce questions-

- What is a String?
- What is length() function in string?
- What is size() function in string?
- What is reverse() function in string?

What is File Handling in C++?

File handling in C++ is a mechanism to store the output of a program in a file and help perform various operations on it. Files help store these data permanently on a storage device.

fstream library

Before diving into each sub-topics, let us first learn about the header file we will be using to gain access to the file handling method. In C++, fstream library is used to handle files, and it is dealt with the help of three classes known as ofstream, ifstream and fstream.

Ofstream: This class helps create and write the data to the file obtained from the program's output. It is also known as the input stream.

Ifstream: We use this class to read data from files and also known as the input stream.

Fstream: This class is the combination of both ofstream and ifstream. It provides the capability of creating, writing and reading a file.

File Operations in C++

C++ provides us with four different operations for file handling. They are:

1.open() – This is used to create a file.
2.read() – This is used to read the data from the file.
3.write() – This is used to write new data to file.
4.close() – This is used to close the file.

We will look into each of these and try to understand them better.

Opening files in C++

To read or enter data to a file, we need to open it first. This can be performed with the help of 'ifstream' for reading and 'fstream' or 'ofstream' for writing or appending to the file. All these three objects have open() function pre-built in them.

Syntax: open(Filename, mode);

Here:
FileName – It denotes the name of file which has to be opened.
Mode – There different mode to open a file and it explained in this article.

Writing to File

Till now, we learned how to create the file using C++. Now, we will learn how to write data to file which we created before. We will use fstream or ofstream object to write data into the file and to do so; we will use stream insertion operator (<<) along with the text enclosed within the double-quotes.
With the help of open() function, we will create a new file named 'FileName' and then we will set the mode to 'ios::out' as we have to write the data to file.
Syntax:

```
FileName<<"Insert the text here";
```

Reading from file in C++

Getting the data from the file is an essential thing to perform because without getting the data, we cannot perform any task. But don't worry, C++ provides that option too. We can perform the reading of data from a file with the CIN to get data from the user, but then we use CIN to take inputs from the user's standard console. Here we will use fstream or ifstream.
Syntax: FileName>>Variable;

Closing a file in C++

Closing a file is a good practice, and it is must to close the file. Whenever the C++ program comes to an end, it clears the allocated memory, and it closes the file. We can perform the task with the help of close() function.
Syntax: FileName.close();

EXPERIMENT-11

Aim: Write a program in C++ for File Handling using ifstream & ofstream class objec to write the content in file then to read the content of file.

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

int main()
{
   ofstream fout;

   string line;
   fout.open("sample.txt");

      while (fout) {

   getline(cin, line);

      if (line == "-1")
         break;

      fout << line << endl;
   }

  fout.close();

   ifstream fin;

    fin.open("sample.txt");

   while (getline(fin, line)) {

      cout << line << endl;
   }
```

```
    fin.close();

    return 0;
}
```

Output :

hello this is LNCT College Bhopal

and I am student of B.Tech

-1

hello this is LNCT College Bhopal

and I am student of B.Tech

- Write a program to create a new file.
- Write a program to read and write data on a file.

Viva Voce questions-

- What is file handling in C++?
- What is I/O stream in C++?
- What is read() and write() function?
- What is the use of open() and close() function?

Case Study - ATM System

ATMs are Automated Teller Machines that are used to carry day-to-day financial transactions. ATMs can be used to withdraw money or to deposit money or even to know the information of an account like the balance amount, etc. They are convenient and easy to use, and it allows consumers to perform quick self-service transactions.

In this article, we will discuss the ATM Management System in C++ which is an application that provides users with every aspect that an actual Automated Teller Machine i.e., ATM should have. It is a menu-driven program having ATM functions which include:

1. *Enter Name, Account number, Account type to be shown during transactions.*
2. *Shows the information about the person who is doing the transaction.*
3. *Enter amount to deposited in the account.*
4. *Shows the Balance in the account.*
5. *Enter amount to be withdrawn from the account, and then it shows available balance.*
6. *Cancel the transaction.*

Approach: This program uses basic concepts of class, Access Modifiers in C++, data types, variables, Switch Case, etc. Below are the functionalities that are to be implemented:

- setvalue(): This function is used here to set the data using basic input and output method in C++ i.e., cout and cin statements which display and take input from the keyboard i.e., from the user respectively.
- showvalue(): This function is used to print the data.
- deposit(): This function helps to deposit money in a particular account.
- showbal(): This function shows the total balance available after deposition.
- withdrawl(): This function helps to withdraw money from the account.
- main(): In this function, there is a simple switch case (to make choices) inside an infinite while loop so that every time user gets to select choices.

EXPERIMENT-12

Aim: Write a program in C++ to implement the ATM Management System.

```
// C++ program to implement the ATM
// Management System

#include <iostream>
#include <stdlib.h>
#include <string.h>
using namespace std;
class Bank {

    // Private variables used inside class
private:
    string name;
    long long accnumber;
    char type[10];
    long long amount = 0;
    long long tot = 0;

    // Public variables
public:
    // Function to set the person's data
    void setvalue()
    {
        cout << "Enter name\n";
        cin.ignore();

        // To use space in string
        getline(cin, name);

        cout << "Enter Account number\n";
        cin >> accnumber;
        cout << "Enter Account type\n";
        cin >> type;
        cout << "Enter Balance\n";
```

```
        cin >> tot;
    }

    // Function to display the required data
    void showdata()
    {
        cout << "Name:" << name << endl;
        cout << "Account No:" << accnumber << endl;
        cout << "Account type:" << type << endl;
        cout << "Balance:" << tot << endl;
    }

    // Function to deposit the amount in ATM
    void deposit()
    {
        cout << "\nEnter amount to be Deposited\n";
        cin >> amount;
    }

    // Function to show the balance amount
    void showbal()
    {
        tot = tot + amount;
        cout << "\nTotal balance is: " << tot;
    }

    // Function to withdraw the amount in ATM
    void withdrawl()
    {
        int a, avai_balance;
        cout << "Enter amount to withdraw\n";
        cin >> a;
        avai_balance = tot - a;
        cout << "Available Balance is" << avai_balance;
    }
};
```

```
// Driver Code
int main()
{
    // Object of class
    Bank b;
     int choice;
     // Infinite while loop to choose
    // options everytime
    while (1) {
        cout << "\n~~~~~~~~~~~~WELCOME~~~~~~~~~~~~\n\n";
        cout << "Enter Your Choice\n";
        cout << "\t1. Enter name, Account "
             << "number, Account type\n";
        cout << "\t2. Balance Enquiry\n";
        cout << "\t3. Deposit Money\n";
        cout << "\t4. Show Total balance\n";
        cout << "\t5. Withdraw Money\n";
        cout << "\t6. Cancel\n";
        cin >> choice;

        // Choices to select from
        switch (choice) {
        case 1:
            b.setvalue();
            break;
        case 2:
            b.showdata();
            break;
        case 3:
            b.deposit();
            break;
        case 4:
            b.showbal();
            break;
        case 5:
```

```
                b.withdrawl();
                break;
            case 6:
                exit(1);
                break;
            default:
                cout << "\nInvalid choice\n";
            }
        }
}
```

Output :

```
~~~~~~~~~~~~WELCOME~~~~~~~~~~~~

Enter Your Choice
        1. Enter name, Account number, Account type
        2. Balance Enquiry
        3. Deposit Money
        4. Show Total balance
        5. Withdraw Money
        6. Cancel
1
Enter name
Gaurav Saxena
Enter Account number
253614
Enter Account type
Saving
Enter Balance
25000
```

~~~~~~~~~~~~WELCOME~~~~~~~~~~~~~

Enter Your Choice

1. Enter name, Account number, Account type
2. Balance Enquiry
3. Deposit Money
4. Show Total balance
5. Withdraw Money
6. Cancel

2

Name:Gaurav Saxena

Account No:253614

Account type:Saving

Balance:25000

~~~~~~~~~~~~WELCOME~~~~~~~~~~~~~

Enter Your Choice

1. Enter name, Account number, Account type
2. Balance Enquiry
3. Deposit Money
4. Show Total balance
5. Withdraw Money
6. Cancel

Viva Voce questions-

- What is ATM management system?
- What is switch case?
- What is getline() function?

VISION OF THE DEPARTMENT

To be a centre of excellence for providing quality technical education to develop future leaders with the aspects of research &computing, Software product development and entrepreneurship.

MISSION OF THE DEPARTMENT

M1: To offer academic programme with state of art curriculum having flexibility for accommodating the latest developments in the areas of Computer science engineering

M2: To conduct research and development activities in contemporary and emerging areas of computer science & engineering.

M3: To inculcate moral values & entrepreneurial skills to produce professionals capable of providing socially relevant and sustainable solutions.