DRONACHARYA
College of Engineering

LABORATORY MANUAL

B.Tech. Semester- IV

OBJECT ORIENTED PROGRAMMING LAB USING C++
Subject code: LC-CSE-214G

Prepared by:	**Checked by:**	**Approved by:**
Prof. Ashwani Kumar	Prof. Yashwardhan Soni	Name : Prof. (Dr.) Isha Malhotra
Sign.: ……	**Sign.: ……**	**Sign.: ……**

DEPARTMENT OF CSE/CSIT/IT/IOT
DRONACHARYA COLLEGE OF ENGINEERING
KHENTAWAS, FARRUKH NAGAR, GURUGRAM (HARYANA)

Table of Contents

Vision and Mission of the Institute

VISION OF INSTITUTE

"Empowering human values and advanced technical education to navigate and address global challenges with excellence."

MISSION OF INSTITUTE

- M1: Seamlessly integrate human values with advanced technical education.
- M2: Supporting the cultivation of a new generation of innovators who are not only skilled but also ethically responsible.
- M3: Inspire global citizens who are equipped to create positive and sustainable impact, driving progress towards a more inclusive and harmonious world.

VISION OF DEPARTMENT

"Steering the future of computer science through innovative advancements, fostering ethical values and principles through technical education."

MISSION OF THE DEPARTMENT

M1: Directing future innovations in computer science through revolutionary progress.

M2: Instilling a foundation of ethical values and principles in every technologist.

M3: Offering a comprehensive technical education to equip individuals for a meaningful and influential future.

Programme Educational Objectives (PEOs)

PEO1: Apply the technical competence in Computer Science and Engineering for solving problems in the real world.

PEO2: Carry out research and develop solutions on problems of social applications.

PEO3: Work in a corporate environment, demonstrating team skills, work morals, flexibility and lifelong learning.

Programme Outcomes (POs)

PO1: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2: 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.

PO3: 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.

PO4: 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.

PO5: 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.

PO6: 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.

PO7: 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.

PO8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9: Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10: 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.

PO11: 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.

PO12: 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 (PSOs)

PSO1: Exhibit design and programming skills to develop and mechanize business solutions using revolutionary technologies.

PSO2: Learn strong theoretical foundation leading to brilliance and enthusiasm towards research, to provide well-designed solutions to complicated problems.

PSO3: Work effectively with diverse Engineering fields as a team to design, build and develop system applications.

University Syllabus

1. [Classes and Objects] Write a program that uses a class where the member functions are defined inside a class.

2. [Classes and Objects] Write a program that uses a class where the member functions are defined outside a class.

3. [Classes and Objects] Write a program to demonstrate the use of static data members.

4. [Classes and Objects] Write a program to demonstrate the use of const data members.

5. [Constructors and Destructors] Write a program to demonstrate the use of zero argument and parameterized constructors.

6. [Constructors and Destructors] Write a program to demonstrate the use of dynamic constructor.

7. [Constructors and Destructors] Write a program to demonstrate the use of explicit constructor.

8. [Initializer Lists] Write a program to demonstrate the use of initializer list.

9. [Operator Overloading] Write a program to demonstrate the overloading of increment and decrement operators.

10. [Inheritance] Write a program to demonstrate the multilevel inheritance.

11. [Exception Handling] Write a program to demonstrate the exception handling.

12. [Templates and Generic Programming] Write a program to demonstrate the use of function template.

Course Outcomes (COs)

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

C214.1: Understand the features of C++ supporting object oriented programming

C214.2: Understand the relative merits of C++ as an object oriented programming language

C214.3: Understand how to produce object-oriented software using C++.

C214.4 Understand how to apply the major object-oriented concepts to implement object

. C214.5: Understand advanced features of C++ specifically stream I/O, templates and operator overloading

CO-PO Mapping

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
C214.1	1		1		3			2	3	1	3	1
C214.2	1		3		3			2	3	1	3	1
C214.3	1		3		3			2	3	1	3	1
C214.4	1		3		3			2	3	1	3	1
C214.5	1		1		3			2	3	1	3	1

CO-PSO Mapping

	PSO1	PSO2	PSO3
C214.1	2		2
C214.2	2		2
C214.3	2		2
C214.4	2		2
C214.5	2		2

Course Overview

The Object-Oriented Programming using C++ course is designed to provide students with a solid foundation in object-oriented programming principles and the C++ programming language. Throughout the course, students will learn how to design, implement, and manipulate objects to solve complex problems. The course will cover key concepts such as classes, objects, inheritance, polymorphism, encapsulation, and abstraction.

1. Lab Setup: The lab should provide a computer environment equipped with the necessary software tools and compilers for C++ programming. This includes an integrated development environment (IDE) or a text editor, a C++ compiler, and debugging tools.
2. Programming Exercises: The lab sessions typically involve a series of programming exercises that cover various aspects of C++ programming. These exercises may focus on topics like basic syntax, control structures, functions, arrays, classes, inheritance, polymorphism, data structures, algorithms, file handling, and more.
3. Hands-on Coding: In the lab, students are encouraged to actively write code to solve programming problems and complete the given exercises. They can experiment with different programming techniques, implement algorithms, and practice coding best practices.
4. Debugging and Error Handling: Debugging is an essential part of programming, and the lab provides an environment for students to practice debugging techniques. Students learn to identify and fix errors in their programs, use debugging tools to trace program execution, and handle exceptions that may occur during program execution.
5. Lab Assignments/Projects: Lab courses often include assignments or projects that require students to work on more extensive programming tasks. These assignments may involve developing software applications, implementing algorithms, solving coding problems, or working on group projects to encourage teamwork and collaboration.
6. Code Review and Feedback: Lab instructors or teaching assistants provide guidance, review students' code, and offer feedback on their programming solutions. This helps students improve their coding style, understand best practices, and learn from their mistakes.

List of Experiments mapped with COs

Si No.	Name of the Experiment	Course Outcome
1	Write a program that uses a class where the member functions are defined inside a class.	C214.1, C214.3
2	Write a program that uses a class where the member functions are defined outside a class.	C214.3
3	Write a program to demonstrate the use of static data members.	C214.4
4	Write a program to demonstrate the use of const data members.	C214.3
5	Write a program to demonstrate the use of zero argument and parameterized constructors.	C214.2
6	Write a program to demonstrate the use of dynamic constructor.	C214.3
7	Write a program to demonstrate the use of explicit constructor.	C214.5
8	Write a program to demonstrate the use of initializer list	C214.4
9	Write a program to demonstrate the overloading of increment and decrement operators	C2144
10	Write a program to demonstrate the multilevel inheritance.	C214.5
11	Write a program to demonstrate the exception handling.	C214.4
12	Write a program to demonstrate the use of function template.	C214.5

DOs and DON'Ts

DOs

1. Login-on with your username and password.
2. Log off the computer every time when you leave the Lab.
3. Arrange your chair properly when you are leaving the lab.
4. Put your bags in the designated area.
5. Ask permission to print.

DON'Ts

1. Do not share your username and password.
2. Do not remove or disconnect cables or hardware parts.
3. Do not personalize the computer setting.
4. Do not run programs that continue to execute after you log off.
5. Do not download or install any programs, games or music on computer in Lab.
6. Personal Internet use chat room for Instant Messaging (IM) and Sites is strictly prohibited.
7. No Internet gaming activities allowed.
8. Tea, Coffee, Water & Eatables are not allowed in the Computer Lab.

General Safety Precautions

Precautions (In case of Injury or Electric Shock)

1. To break the victim with live electric source, use an insulator such as fire wood or plastic to break the contact. Do not touch the victim with bare hands to avoid the risk of electrifying yourself.
2. Unplug the risk of faulty equipment. If main circuit breaker is accessible, turn the circuit off.
3. If the victim is unconscious, start resuscitation immediately, use your hands to press the chest in and out to continue breathing function. Use mouth-to-mouth resuscitation if necessary.
4. Immediately call medical emergency and security. Remember! Time is critical; be best.

Precautions (In case of Fire)

1. Turn the equipment off. If power switch is not immediately accessible, take plug off.
2. If fire continues, try to curb the fire, if possible, by using the fire extinguisher or by covering it with a heavy cloth if possible isolate the burning equipment from the other surrounding equipment.
3. Sound the fire alarm by activating the nearest alarm switch located in the hallway.
4. Call security and emergency department immediately:

Emergency : Reception

Security : Front Gate

Guidelines to students for report preparation

All students are required to maintain a record of the experiments conducted by them. Guidelines for its preparation are as follows: -

1) All files must contain a title page followed by an index page. ***The files will not be signed by the faculty without an entry in the index page.***

2) Student's Name, Roll number and date of conduction of experiment must be written on all pages.

3) For each experiment, the record must contain the following

(i) Aim/Objective of the experiment
(ii) Pre-experiment work (as given by the faculty)
(iii) Lab assignment questions and their solutions
(iv) Test Cases (if applicable to the course)
(v) Results/ output

Note:

1. Students must bring their lab record along with them whenever they come for the lab.

2. Students must ensure that their lab recod is regularly evaluated.

Lab Assessment Criteria

An estimated 10 lab classes are conducted in a semester for each lab course. These lab classes are assessed continuously. Each lab experiment is evaluated based on 5 assessment criteria as shown in following table. Assessed performance in each experiment is used to compute CO attainment aswell as internal marks in the lab course.

Grading Criteria	Exemplary (4)	Competent (3)	Needs Improvement (2)	Poor (1)
AC1: Pre-Lab written work (this may be assessed through viva)	Complete procedure with underlined concept is properly written	Underlined concept is written but procedureis incomplete	Not able to write concept and procedure	Underlined concept is not clearly understood
AC2: Program Writing/ Modeling	Assigned problem is properly analyzed, correct solution designed, appropriate language constructs/ tools are applied, Program/solution written is readable	Assigned problem is properly analyzed, correct solution designed, appropriate language constructs/ tools are applied	Assigned problem is properly analyzed & correct solution designed	Assigned problem is properly analyzed
AC3: Identification & Removal of errors/ bugs	Able to identify errors/ bugs and remove them	Able to identify errors/ bugs and remove them with little bit of guidance	Is dependent totally on someone for identification of errors/ bugs and their removal	Unable to understand the reason for errors/ bugs even after they are explicitly pointed out
AC4:Execution & Demonstration	All variants of input /output are tested, Solution is well demonstrated and implemented concept is clearly explaine	All variants of input /output are not tested, However, solution is well demonstrated and implemented concept is clearly explained	Only few variants of input /output are tested, Solution is well demonstrated but implemented concept is not clearly explained	Solution is not well demonstrated and implemented concept is not clearly explained

AC5:Lab Record Assessment	All assigned problems are well recorded with objective, design constructs and solution along with Performance analysis using all variants of input and output	More than 70 % of the assigned problems are well recorded with objective, design contracts and solution along with Performance analysis is done with all variants of input and output	Less than 70 % of the assigned problems are well recorded with objective, design contracts and solution along with Performance analysis is done with all variants of input and output	Less than 40 % of the assigned problems are well recorded with objective, design contracts and solution along with Performance analysis is done with all variants of input and output

LAB EXPERIMENTS

LAB EXPERIMENT 1

OBJECTIVE: Write a program that uses a class where the member functions are defined inside a class.

BRIEF DESCRIPTION:

Member functions are defined inside a class and provide the behavior or actions that objects of that class can perform. They are also referred to as methods. Member functions can access and manipulate the data members (variables) of the class and can interact with other objects of the same or different classes.

PRE-EXPERIMENT QUESTIONS:

1. What is Class and Object?
2. What is global variables.?
3. Difference between normal function and member function of the class?

Explanation:

```
#include<iostream.h>
class car
{
Private:
Int car_number;
Char car_model[10];
Public:
void getdata()
{
cout<<"Enter car number: "; cin>>car_number;

cout<<"\n Enter car model: "; cin>>car_model;
}
void showdata()
{
cout<<"Car number is "<<car_number;
cout<<"\n Car model is "<<car_model;
}
};
// main function start
Int main()
```

```
{
Car c1;
c1.gatdata()
c1.showdata();
return 0;
}
```

Output:

Enter car number: 6325
Enter car model: 2021
Car number is 6325
Car model is 2021

POST EXPERIMENT QUESTIONS:

1. Explain the oops, how it is used?
2. Explain the concepts of data members in class .?

Explain object

LAB EXPERIMENT 2

OBJECTIVE:

Write a program that uses a class where the member functions are defined outside a class.

BRIEF DESCRIPTION:

Defining member functions outside the class in C++ is a way to separate the declaration and implementation of the functions. This approach provides modularity and improves code organization, especially when dealing with large classes.
To define a member function outside the class ,you need to follow these steps:-

1. **Declare the member function inside the class declaration, including the function's return type, name, and parameters.**
2. **Outside the class declaration, use the scope resolution operator :: followed by the class name and the member function's name to indicate that you are defining the function belonging to that class.**
3. **Provide the function definition, including the return type, name, and parameters, just as you would for any other function.**
4. **Implement the function's logic within the definition. You can access the class's private members and perform any necessary operations.**

.

PRE EXPERIMENT QUESTIONS:

1. Explain the member function?
2. Difference between global and local variables of the class?
3. What are different member function in c++?

Explanation:

```
#include<iostream.h>
class car
{
Private:
Int car_number;
Char car_model[10];
Public:void getdata();
        void showdata();
```

```
};

void  car ::getdata()
{
cout<<"Enter car number: "; cin>>car_number;

cout<<"\n Enter car model: "; cin>>car_model;
}
void car ::showdata()
{
cout<<"Car number is "<<car_number;
cout<<"\n Car model is "<<car_model;
}

// main function start
Int main()
{
car c1;
c1.gatdata()
c1.showdata();
return 0;
}
```

Output:

```
Enter car number: 6325
Enter car model: 2021
Car number is 6325
Car model is 2021
```

POST EXPERIMENT QUESTIONS:

1.Enumerate the differences between java and c++?
2.What are various data types in c++?
3.What is data hiding ?
4. What is Encapsulation.?

LAB EXPERIMENT 3

OBJECTIVE:

Write a program to demonstrate the use of static data members.

BRIEF DESCRIPTION: C++, a static data member is a class member that is shared among all instances (objects) of the class. It is associated with the class itself rather than with individual objects. Here's an overview of static data members in C++.

PRE-EXPERIMENT QUESTIONS:

1. What is Static Data Member?
2. What are the different ways of define data members?

Explanation:

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

class A {
public:
   A()
   {
     cout << "A's Constructor Called " <<
             endl;
   }
};

class B {
   static A a;

public:
   B()
   {
     cout << "B's Constructor Called " <<
             endl;
   }
};

// Driver code
```

```
int main()
{
   B b;
   return 0;
}
```

POST EXPERIMENT QUESTIONS:

1.How to define static data member .

2.What is static ?

LAB EXPERIMENT 4

OBJECTIVE: Write a program to demonstrate the use of const data members..

BRIEF DESCRIPTION:

In C++, a const data member is a class member whose value cannot be modified once it is initialized. Here's an overview of const data members in C++:
Declaration and Initialization: A const data member is declared in the class declaration and must be initialized at the point of declaration or in the constructor's initializer list. It is typically declared as private to ensure that its value remains constant.

PRE EXPERIMENT QUESTIONS:

1. What are const?
2. what are the different types data members?
3. How to declare const data member in c++?

```
#include<iostream>
using namespace std;
class Demo {
  int val;
  public:
  Demo(int x = 0) {
    val = x;
  }
  int getValue() const {
    return val;
  }
};
int main() {
  const Demo d(28);
  Demo d1(8);
  cout << "The value using object d : " << d.getValue();
  cout << "\nThe value using object d1 : " << d1.getValue();
  return 0;
}
```

POST EXPERIMENT QUESTIONS:

1. How to declare const data type.
2. What are the difference between static and const data members?

LAB EXPERIMENT 5

OBJECTIVE:

Write a program to demonstrate the use of zero argument and parameterized constructors.

BRIEF DESCRIPTION:

In C++, a parameterized constructor is a special member function of a class that is used to initialize objects of that class with specific values. It accepts parameters as arguments, allowing the caller to provide values during object creation. Here's an overview of parameterized constructors:
Declaration and Definition: A parameterized constructor is declared within the class declaration like any other member function. It has parameters corresponding to the values needed to initialize the object. The constructor definition provides the implementation for the constructor.

PRE EXPERIMENT QUESTIONS:

1. What is an constructor?
2. What are the various types of constructor in c++ 3.How can we determine the width and height of triangle using constructor?

EXPLANATION:

```
class math
{
private:
   int a,b,c;
public:
   math(int x,int y)
   {
      a=x;
      b=y;
   }
   void add()
   {
      c=a+b;
      cout<<"Total : "<<c;
   }
};
int main()
{
   math o(10,25);
```

```
   o.add();
  return 0;
}
```

Output: Total : 35

POST EXPERIMENT QUESTIONS:

1. Explain how to set the constructor.
2. What is the difference between an constructor and destructor?

LAB EXPERIMENT 6

OBJECTIVE:

Write a program to demonstrate the use of dynamic constructor..

BRIEF DESCRIPTION:

C++ provides different types of constructors, such as default constructors, parameterized constructors, copy constructors, and move constructors. These constructors are called implicitly based on the object creation syntax or specific scenarios.

Dynamic memory allocation in C++ involves the use of operators like new and delete to allocate and deallocate memory at runtime. However, this is not directly related to constructor.

PRE EXPERIMENT QUESTIONS:

1. What is the correct syntax of new and delete operator
2. What id dynamic memory allocation?
3. What is memory management concepts?

Explanation:

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


class example
{
  const char* ptr;


public:


  // default constructor
  example()
  {
    // allocating memory at run time by using the new keyword
    ptr = new char[15];
    ptr = "Hi from example constructor ";
  }
```

```
  void display()
  {
    cout << ptr;
  }
};


int main()
{
  example obj1;
  obj1.display();
}
```

POST EXPERIMENT QUESTIONS:

1. What is pointer?
2. What are steps of define pointer to pointer?

LAB EXPERIMENT_7

OBJECTIVE:

Write a program to demonstrate the use of explicit constructor..

BRIEF DESCRIPTION:

In C++, the explicit keyword is used to qualify a constructor declaration. It affects the way the constructor is used for implicit type conversions. Here's an overview of explicit constructors:

Implicit Type Conversions: By default, constructors that can be called with a single argument can also be used for implicit type conversions. For example, if a class has a constructor that takes an int parameter, it can be used to implicitly convert an int value to an object of the class

PRE EXPERIMENT QUESTIONS:

1. How call a member function in c++?
2. What is explicit call?
3. How do we define constructor in c++?

Explanation:

```
#include<iostream.h>

class A
{
  int data;

public:
  A(int a):data(a)
  {
     cout<<"A::Construcor...\n";
  };

  friend void display(A obj);
};

void display(A obj)
{
   cout<<"Valud of data in obj :="<< obj.data<<endl;
```

```
}

int main()
{
  //Call display with A object i.e. a1
  display(5000);

  return (0);
}
```

Output A::Construcor...
B::Construcor...
function display..

POST EXPERIMENT QUESTIONS:

1. What is implicit call?
2. Difference between implicit and explicit call in c++

LAB EXPERIMENT 8

OBJECTIVE:

Write a program to demonstrate the use of initializer list

BRIEF DESCRIPTION:

In C++, the initializer list is a special syntax used in constructors to initialize the data members of a class. It provides a concise and efficient way to initialize member variables when an object is created. Here's an overview of the initializer list:
Syntax: The initializer list is placed after the constructor's parameter list and is enclosed within parentheses. Each member variable is initialized using a comma-separated list of member initializations.

```
Constructor(datatype value1, datatype value2):datamember(value1),datamember(value2);
{
Body of function
}
```

PRE EXPERIMENT QUESTIONS:

1. What is **datamembers**?
2. What is initializer list?

EXPLANATION:

```
#include<iostream>
using namespace std;
class Point {
private:
int x;
int y;
public:
Point(int i = 0, int j = 0):x(i), y(j) {}
/* The above use of Initializer list is optional as the
constructor can also be written as:
Point(int i = 0, int j = 0) {
x = i;
y = j;
}
*/
int getX() const {return x;}
```

```
    int getY() const {return y;}
};
      int main()
      {
      Point t1(21, 15);
      cout<<"x = "<<t1.getX()<<", ";
      cout<<"y = "<<t1.getY();
      return 0;
      }
```

OUTPUT:

x = 21, y = 15

POST EXPERIMENT QUESTIONS

1. How to define data member in constructor ?
2. What is initializer list?
3. What are the features initializer list in c++?

LAB EXPERIMENT 9

OBJECTIVE:

Write a program to demonstrate the overloading of increment and decrement operators.

BRIEF DESCRIPTION:

Operator overloading in C++ allows you to define the behavior of operators when applied to user-defined types. It enables you to extend the functionality of operators beyond their built-in capabilities. Here's an overview of operator overloading:

Syntax: Operator overloading is performed by defining a function or a method that is associated with a specific operator. The function or method is invoked when the corresponding operator is used with operands of the user-defined type.

PRE-EXPERIMENT QUESTIONS:

Q1. What is operator overloading?
Q2. What is the binary and unary operator ?

Explanation:

```
#include<iostream.h>
Class Check
{
Private: int i;
Public: Check():i(0){ }
Void operator ++()
{++i;}
void Display()
{ cout<<"i="<<i<<endl;}
```

```
};
int main()
{
Check obj;
Obj.Display()
++obj;
Obj.Display()
Return 0;
}
```

Output

```
i=0
i=1
```

POST EXPERIMENT QUESTIONS:

Q1. Write a program to overload -unary operator.

Q2. What is overloading?

LAB EXPERIMENT 10

OBJECTIVE:

Write a program to demonstrate the multilevel inheritance..

BRIEF DESCRIPTION:

In C++, multilevel inheritance is a type of inheritance where a derived class is derived from another derived class. It involves creating a class hierarchy with multiple levels of derived classes. Here's an overview of multilevel inheritance:

Syntax: In multilevel inheritance, each derived class serves as the base class for the next level of derived class.

```
class A
{
.....
}
class B: public A
{
.....
}
Class C: public B
{
.....
}
```

PRE-EXPERIMENT QUESTIONS:

Q1. What is Inheritance ?

Q2. Which inheritance is most important explain.

Explanation:

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

class Vehicle{
   public:
      void vehicle(){
         cout<<"I am a vehicle\n";
      }
};

class FourWheeler : public Vehicle{
   public:
```

```
    void fourWheeler(){
       cout<<"I have four wheels\n";
    }
};

class Car : public FourWheeler{
  public:
    void car(){
       cout<<"I am a car\n";
    }
};
int main(){
  Car obj;
  obj.car();
  obj.fourWheeler();
  obj.vehicle();
  return 0;
}
```

Input

```
Car obj;
obj.car();
obj.fourWheeler();
obj.vehical()
```

Output

I am a car
I have four wheels
I am a vehicle

POST-EXPERIMENT QUESTIONS:

Q1. Different types of inheritance.

Q2. What is hybrid inheritance?.

LAB EXPERIMENT 11

OBJECTIVE:

Write a program to demonstrate the exception handling.

BRIEF DESCRIPTION: Exception handling in C++ provides a mechanism to handle runtime errors and exceptional conditions that may occur during program execution. It allows you to catch and handle exceptions, preventing program termination and providing a way to recover from errors. Here's an overview of exception handling

PRE-EXPERIMENT QUESTIONS:

1. What is exception ?
2. How to handle exception in c++?
3. What is try block?

Explanation:

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

double division(int a, int b) {
  if( b == 0 ) {
     throw "Division by zero condition!";
  }
  return (a/b);
}

int main () {
  int x = 50;
  int y = 0;
  double z = 0;

  try {
     z = division(x, y);
     cout << z << endl;
  } catch (const char* msg) {
    cerr << msg << endl;
  }
```

```
  return 0;
}
```

Output: Division by zero condition

POST EXPERIMENT QUESTIONS:

1. What are the try and catch block?
2. How to find error?
3. What are the advantages of throw block?

LAB EXPERIMENT 12

OBJECTIVE: Write a program to demonstrate the use of function template.

BRIEF DESCRIPTION:

Function templates in C++ provide a mechanism for writing generic functions that can operate on different data types without explicitly specifying each type. They allow you to define a blueprint for a function that can be instantiated with different types at compile time.

PRE-EXPERIMENT QUESTIONS:

1. What is generic programming?
2. Why do we need to templates?
3. What is class template?

Explanation:

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

template <class T>
T GetMax (T a, T b) {
  T result;
  result = (a>b)? a : b;
  return (result);
}

int  main  () {
  int i=5, j=6, k;
  long l=10, m=5, n;
  k=GetMax<int>(i,j);
  n=GetMax<long>(l,m);
  cout << k << endl;
  cout << n << endl;
  return 0;
}
```

POST EXPERIMENT QUESTIONS:

1. Write a program to add int float values using template function?
2. Diffreence between class and function template in c++.
3. What is STL in c++
4. What is algorithm.h in STL.

This lab manual has been updated by

Mr.Ashwani Kumar
(Ashwani.kumar@ggnindia.dronacharya.info)

Crosschecked By
HOD CSE

Please spare some time to provide your valuable feedback.