AssignmentNo.1

Title: Write a program to compute the area of triangle and circle by overloading the area() function.

Problem Statement: Implement a C++ program to understand concept of Overloading.

Objective:

1. Understand the basic principles of object oriented programming
2. Implement the concepts of overloading.
3. Develop programs using object oriented concepts.

Theory:

Object-oriented programming (OOP) is a computer programming model that organizes software design around data, or objects, rather than functions and logic. An object can be defined as a data field that has unique attributes and behavior.

Basic concepts of Object-oriented programming are Inheritance, Encapsulation, Polymorphism, and Data abstraction.

Polymorphism is one of the core concepts of object-oriented programming (OOP) and describes situations in which something occurs in several different forms. In computer science, it describes the concept that you can access objects of different types through the same interface.

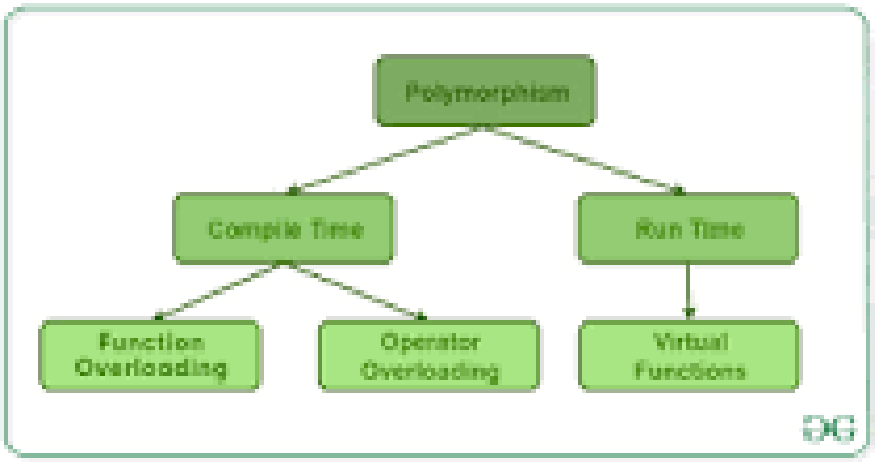


Fig. Polymorphism

Polymorphism in Function Overloading:

In computing, polymorphism is a property of object oriented programming in which a function can take different forms based on the number of arguments and their data types. All the functions will have the same name, but they will differ in their arguments.

Function overloading or method overloading is the ability to create multiple functions of the same name with different implementations. Calls to an overloaded function will run a specific implementation of that function appropriate to the context of the call, allowing one function call to perform different tasks depending on context.

Algorithm:

Step 1: Start the Program.

Step 2: Declare function area() to find area of triangle with two integer arguments.

Step 3: Declare function area() to find area of circle with one integer argument.

Step 4: Read value of base and height of a triangle from user.

Step 5: Call function area with above two values as arguments.

Step 6: Read value of radius of a circle from user.

Step 7: Call function area with above one value as argument.

Step 8: End of program.

Program:

Output:

Conclusion:

AssignmentNo.2

Title: Define a class to represent a bank account. Include the following members:

Data members: Name of depositor, Account number, Type of account, Balance amount in the account Member functions: To assign initial values, to deposit an amount, to withdraw an amount after checking the balance, to display name & balance. Write a main program to test program using class and object.

Problem Statement: Implement C++ programs which include a Class having data members and member function and Object to access these members of Class.

Objective:

1. Understand the basic principles of object oriented programming

2. Develop programs using object oriented concepts

Theory:

Object Oriented Programming

Object Oriented programming is a programming style that is associated with the concept of Class, Objects and various other concepts revolving around these two, like Inheritance, Polymorphism, Abstraction, Encapsulation etc.

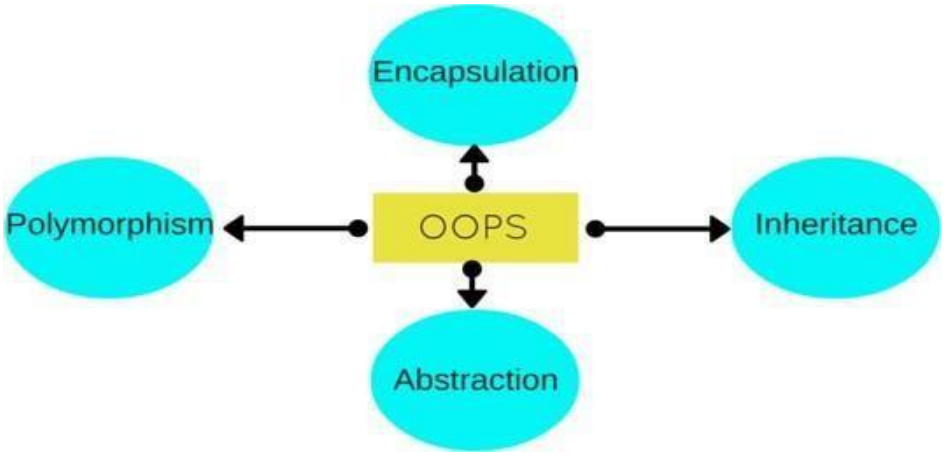


Fig. Features of OOP

Objects

Objects are the basic unit of OOP. They are instances of class, which have data members and use various member functions to perform tasks.

Class

It is similar to structures in C language. Class can also be defined as user defined data type but it also contains functions in it. So, class is basically a blueprint for object. It declare & defines what data variables the object will have and what operations can be performed on the class's object.

Abstraction

Abstraction refers to showing only the essential features of the application and hiding the details. In C++, classes can provide methods to the outside world to access & use the data variables, keeping the variables hidden from direct access, or classes can even declare everything accessible to everyone, or maybe just to the classes inheriting it. This can be done using access specifiers.

Encapsulation

It can also be said data binding. Encapsulation is all about binding the data variables and functions together in class.

Inheritance

Inheritance is a way to reuse once written code again and again. The class which is inherited is called the **Base** class & the class which inherits called the **Derived** class. They are also called parent and child class.

So when, a derived class inherits a base class, the derived class can use all the functions which are defined in base class, hence making code reusable.

Polymorphism

It is a feature, which lets us create functions with same name but different arguments, which will perform different actions. That means, functions with same name, but functioning in different ways. Or, it also allows us to redefine a function to provide it with a completely new definition. You will learn how to do this in details soon in coming lessons.

Algorithm:

Step 1: Declare a class BankAccount with data members Name of depositor, Account number, Type of account, Balance amount. and member functions - To assign initial values, to deposit an amount, to withdraw an amount after checking the balance, to display name & balance.

Step 2: Define parameterized constructor to assign values to data members.

Step 3: Define member function deposit() to deposit an amount in the account.

Step 4: Define member function withdraw() to withdraw an amount from the account.

Step 5: Define member function display() to display Account No, Account type and Balance.

Step 6: Write main() function and accept values of acc no, name, acc type and balance from user.

Step 7: Create object of class BankAccount.

Step 8: Call all member functions one by one.

Step 9: End of program.

Program:

Output:

Conclusion:

Assignment No.3

Title: Create two classes DM and DB which stores values of distances. DM stores distances in meters and centimeters and DB in feet and inches. Write a program that can read values for the class objects and add one object of DM with another object of DB. Use a friend function to carry out addition operation.

Problem Statement: Implement a C++ program to understand concept of Friend Function.

Objective:

1. Understand the basic principles of object oriented programming.
2. Apply the concepts of inheritance and Friend function.
3. Develop program using object oriented concepts.

Theory:

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. Such a function need not be a member of any of these classes. To make an outside function "friendly" to a class, we have to simply declare this function as a friend of the class as shown below:

```
class ABC
{
        ........
        ........
    public:
```

```
        .....

        ........

        friend void xyz(void);       // declaration

};
```

A friend function possesses certain special characteristics:

1. It is not in the scope of the class to which it has been declared as friend.

2. Since it is not in the scope of the class, it cannot be called using the object of that class.

3. Unlike member functions, it can not access the member names directly and has to use an object name and dot membership operator with each member name.

4. It can be declared either in the public or the private part of a class without affecting its meaning.

5. Usually, it has the objects as arguments.

Algorithm:

Step 1: Start the Program

Step 2: Declare class DM with data members meter, centimeter and member functions accept ( ), display( ).

Step 3: Declare class DB with data members feet, inches and member functions accept ( ), display ( ).

Step 4: Declare friend function add( ) in booth the classes.

Step 5: Define function accept ( ) to take distance values.

Step 6: Define function display to display distance values.

Step 7: Define friend function add( ) to calculate addition of distances by adding two objects.

Step 8: Create objects of class DB and DM

Step 9: Call member functions accept() and display( ).

Step 10: Call add function and display addition of distances.

Step 11: End of program.

Program:

Output:

Conclusion:

AssignmentNo.4

Title: Create a class MAT of size m * n. Define all possible matrix operations for MAT type objects

Problem Statement: Implement a C++ program to understand concept matrix operations.

Objective:

1. Understand the basic principles of object oriented programming
2. Apply the concepts of overloading, inheritance, polymorphism
3. Develop programs using object oriented concepts

Theory:

If we create two or more members having the same name but different in number or type of parameter, it is known as C++ overloading. In C++, we can overload:

- methods, *
- constructors, and
- indexed properties

It is because these members have parameters only.

Types of overloading in C++ are:
o Function overloading

o Operator overloading

Operator Overloading

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

Implementing Operator Overloading

Operator overloading can be done by implementing a function which can be :

1. Member Function
2. Non-Member Function
3. Friend Function

Operator overloading function can be a member function if the Left operand is an Object of that class, but if the Left operand is different, then Operator overloading function must be a non-member function.

Operator overloading function can be made friend function if it needs access to the private and protected members of class.

The **operator** keyword declares a function specifying what operator-symbol means when applied to instances of a class. This gives the operator more than one meaning, or "overloads" it. The compiler distinguishes between the different meanings of an operator by examining the types of its operands.

Algorithm:

Step 1: Declare class Matrix with data member as two dimensional array and member functions accept( ), display( ), Operator +( ), Operator–( ) and Operator*().

Step 2: Define all member functions.
Step 3: Create three objects of class Matrix.
Step 4: Ask user to enter elements of first matrix and accept it.
Step 5: Ask user to enter elements of second matrix and accept it.
Step 6: Assign M3=M1+M2 & display result.
Step 7: Assign M3=M-+M2 & display result.
Step 8: Assign M3=M1*M2 & display result.
Step 9: End of program.

Program:

Output:

Conclusion:

Assignment No. 5

Title: Create Stud class to display student information using constructor and destructor. (Default constructor, Multiple constructor, Copy constructor, Overloaded constructor)

Problem Statement: Implement a C++ program to understand concept of constructor and Destructor.

Objective:

1. Understand the basic principles of object oriented programming.
2. Apply the concepts of class, overloading, inheritance & polymorphism
3. Develop program using object oriented concepts.

Theory:

C++ Constructor:

In C++, constructor is a special method which is invoked automatically at the time of object creation. It is used to initialize the data members of new object generally. The constructor in C++ has the same name as class or structure.

There can be two types of constructors in C++.

o Default constructor

o Parameterized constructor

C++ Default Constructor:

A constructor which has no argument is known as default constructor. It is invoked at the time of creating object.

C++ Parameterized Constructor:

A constructor which has parameters is called parameterized constructor. It is used to provide different values to distinct objects.

Multiple Constructor:

In C++, We can have more than one constructor in a class with same name, as long as each has a different list of arguments. This concept is known as **Constructor Overloading** and is quite similar to function overloading . Overloaded constructors essentially have the same name (exact name of the class) and differ by number and type of arguments.

Copy constructor:

The copy constructor is a constructor which creates an object by initializing it with an object of the same class, which has been created previously. The copy constructor is used to −

-Initialize one object from another of the same type.

-Copy an object to pass it as an argument to a function.

-Copy an object to return it from a function.

If a copy constructor is not defined in a class, the compiler itself defines one.If the class has pointer variables and has some dynamic memory allocations, then it is a must to have a copy constructor.

C++ Destructor:

A destructor works opposite to constructor; it destructs the objects of classes. It can be defined only once in a class. Like constructors, it is invoked automatically. A destructor is defined like constructor. It must have same name as class. But it is prefixed with a tilde sign (~).

Algorithm:

Step 1: Start the program

Step 2: Declare class Stud with data members name, address & rollno and member functions read() , display() and construction stu() and destructor stu().

Step 3: Define constructor function stu()

Step 4: Define member functions read() and display().

Step 5: Define destructor function ~stu()

Step 6: Create object of class stu

Step 7: Call all member functions.

Step 8: End of program.

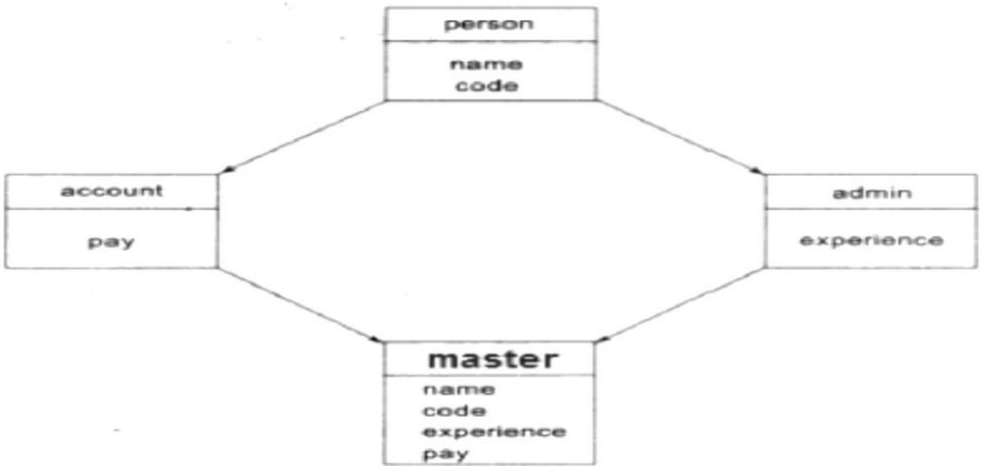
Program:

Output:

Conclusion:

Assignment No.6

Title: Consider class network of given figure. The class master derives information from both account and admin classes which in turn derive information from the class person. Define all the four classes and write a program to create, update and display the information contained in master objects.



Problem Statement: Implement a C++ program to understand concept of multiple and multilevel inheritance using Virtual function.

Objective:

1. Understand the basic principles of object oriented programming
2. Apply the concepts of inheritance.
3. Develop programs using object oriented concepts

Theory:

The Diamond Problem occurs when a child class inherits from two parent classes who both share a common grandparent class. This is illustrated in the diagram.

Here, we have a class **Child** inheriting from classes **Father** and **Mother**. These two classes, in turn, inherit the class **Person** because both Father and Mother are Person

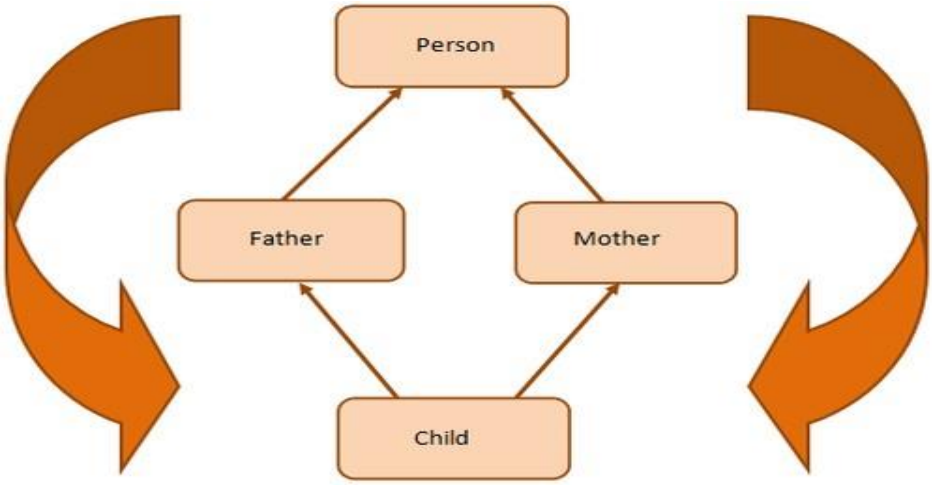


Fig. Inheritance

As shown in the figure, class Child inherits the traits of class Person twice—once from Father and again from Mother. This gives rise to ambiguity since the compiler fails to understand which way to go.

This scenario gives rise to a diamond-shaped inheritance graph and is famously called –The Diamond Problem.

The solution to the diamond problem is to use the **virtual** keyword. We make the two parent classes (who inherit from the same grandparent class) into virtual classes in order to avoid two copies of the grandparent class in the child class.

Here we have used the **virtual** keyword when classes Father and Mother inherit the Person class. This is usually called ―virtual inheritance," which guarantees that only a single instance of the inherited class (in this case, the Person class) is passed on.

In other words, the Child class will have a single instance of the Person class, shared by both the Father and Mother classes. By having a single instance of the Person class, the ambiguity is resolved.

Algorithm:

Step 1: Start the program

Step 2: Declare class Person with data members as Name and Code.

Step 3: Declare class account which is sub class of class person, with data member as pay

Step 4: Declare class admin which is also sub class of class person with data member experience.

Step 5: Declare class master which is sub class of class account and class admin.

Step 6: Create object of class master and access members of class person, admin and account.

Step 7: End of the program.

Program:

Output:

Conclusion:

Assignment No. 7

Title: A book shop sells both books and video tapes. Create a class media that stores the title and price of the publication. Create two derived classes, one for storing number of pages in the book and another for storing playing time of tape. A function display() must be defined in all classes to display class contents. Write a program using polymorphism and virtual function.

Problem Statement: Implement a C++ program to understand concept of polymorphism and virtual function.

Objective:

1. Understand the basic principles of object oriented programming
2. Apply the concepts of overloading, inheritance, polymorphism
3. Develop programs using object oriented concepts

Theory:

Static & Dynamic Binding

Polymorphism means „one name" -„multiple forms. The overloaded member functions are „selected" for invoking by matching arguments, both type and number. This information is known to the compiler at the compile time and compiler is able to select the appropriate function for a particular call at the compile time itself. This is called Early Binding or Static Binding or Static Linking. Also known as compile time polymorphism. Early binding means that an object is bound to its function call at the compile time. It would be nice if the appropriate member function could be selected while the program is running. This is known as runtime polymorphism. C++ supports a mechanism known as virtual function to achieve run time polymorphism. At the runtime, when it is known what class objects are under consideration, the appropriate version of the function is

invoked. Since the function is linked with a particular class much later after the compilation, this process is termed as late binding. It is also known as dynamic binding because the selection of the appropriate function is done dynamically at run time.

VIRTUAL FUNCTIONS

Polymorphism refers to the property by which objects belonging to different classes are able to respond to the same message, but different forms. An essential requirement of polymorphism is therefore the ability to refer to objects without any regard to their classes. When we use the same function name in both the base and derived classes, the function in the base class is declared as virtual using the keyword virtual preceding its normal declaration. When a function is made virtual, C++ determines which function to use at runtime based on the type of object pointed to by the base pointer, rather than the type of the pointer. Thus, by making the base pointer to point to different objects, we can execute different versions of the virtual function.

Rules For Virtual Functions:

When virtual functions are created for implementing late binding, observe some basic rules that satisfy the compiler requirements.

1. The virtual functions must be members of some class.

2. They cannot be static members.

3. They are accessed by using object pointers.

4. A virtual function can be a friend of another class.

5. A virtual function in a base class must be defined, even though it may not be used.

6. The prototypes of the base class version of a virtual function and all the derived class versions must be identical. C++ considers them as overloaded functions, and the virtual function mechanism is ignored.

7. We cannot have virtual constructors, but we can have virtual destructors.

8. While a base pointer points to any type of the derived object, the reverse is not true. i.e. we cannot use a pointer to a derived class to access an object of the base class type.

9. When a base pointer points to a derived class, incrementing or decrementing it will not make it to point to the next object of the derived class. It is incremented or decremented only relative to its base type. Therefore we should not use this method to move the pointer to the next object.

10. If a virtual function is defined in the base class, it need not be necessarily redefined in the derived class. In such cases, calls will invoke the base function.

Algorithm:

Step 1: Start the program

Step 2: Declare class media with data members title & price add member function display() in class.

Step 3: Declare class book, subclass of class media with data member pages and member function display()

Step 4: Declare class tape, subclass of class media with data member time and member function display()

Step 5: Define display() functions of class media, book and tape.

Step 6: Accept book details from user

Step 7: Create object of class book and initialize its data members

Step 8: Accept tape details from user

Step 9: Create object of class tape and initialize its data members

Step 10: Create object of class media

Step 11: Call display function of class book and class tape to display book details and tape details respectively with the help of object of class media.

Step 12: End of Program.

Program:

Output:

Conclusion:

Assignment No. 8

Title: Write a program to show use of this pointer, new and delete.

Problem Statement: Implement a C++ program to understand concept of memory allocation using new and delete keyword.

Objective:

1. Understand the basic principles of object oriented programming
2. Implement memory allocation techniques and usage of exception handling, generic programming
3. Develop programs using object oriented concepts

Theory:

C++ supports these functions, it defines two unary operators **new** and **delete** that perform the task of allocating and deallocating the memory in a better and easier way. An object can be created by using **new**, and destroyed by using **delete**. A data object created inside a block with **new**, will remain in existence until it is explicitly destroyed by using **delete**.

- **new operator:-**

new operator can be used to create objects of any type .Hence new operator allocates sufficient memory to hold data of objects and it returns address of the allocated memory.

Syntax:

pointer-variable = **new** data-type;

Ex: int *p = new int;

To create memory space for arrays:

pointer-variable = **new** data-type[size];

Ex: int *p = new int[10];

- d**elete operator:**

If the variable or object is no longer required or needed is destroyed by **–delete** operator, there by some amount of memory is released for future purpose.

Synatx:

delete pointer-variable;

Eg: delete p;

If we want to free a dynamically allocated array:

delete [size] pointer-variable;

Algorithm:

Step 1: Start of program

Step 2: Allocate memory with New operator.

Step 3: Insert some values in the memory

Step 4: Display values which are stored in memory

Step 5: Release memory with Delete operator.

Step 6: End of program.

Program:

Output:

Conclusion:

Assignment No. 9

Title: Write a function template for finding the minimum value contained in an array.

Problem Statement: Implement a C++ program to understand concept of template.

Objective:

1. Understand the basic principles of object oriented programming

2. Implement memory allocation techniques and usage of exception handling, generic programming

3. Develop programs using object oriented concepts

Theory:

Templates enable us to define generic classes and functions and thus provide support for generic programming. Generic programming is an approach where generic types are used as parameters in algorithms so that they work for a variety of suitable data types and data structures. A template can be used to create a family of classes or functions. For example, a class template for an array class would enable us to create arrays of various data types such as int array and float array. Similarly, we can define a template for a function say mul(), that would help us create various versions of mul() for multiplying int, float and double type values.

Features of templates:-

1. It eliminates redundant code

2. It enhances the reusability of the code.

3. It provides great flexibility to language

Templates are classified into two types. They are

1. Function templates

2. Class Templates.

Function Templates:

The templates declared for functions are called as function templates. A function template defines how an individual function can be constructed.

The general format of function template is:

template<class T>

returntype functionname (arguments of type T)

{

    //.....

    //Body of function with type T

    //.....

}

Algorithm:

Step 1: Start the program.

Step 2: Template<class T>

Step 3: Define function template minimum( ) with arguments array a[ ] and size of array.

Step 4: Assign min=0;

Step 5: Repeat steps 6 for I = 0 to size

Step 6: if a[i] <min then min=a[i];

Step 7: Write main function in which accept array size, integer array elements and floting array elements from user.

Step 8: Call template function for integer array and for floating array.

Step 9: Display result.

Step 10: End of program.

Program:

Output:

Conclusion:

Assignment No. 10

Title: Write a program containing a possible exception. Use a try block to throw it and catch block to handle it properly.

Problem Statement: Implement a C++ program to understand concept of error handling.

Objective:

1. Understand the basic principles of object oriented programming
2. Implement memory allocation techniques and usage of exception handling, generic programming
3. Develop programs using object oriented concepts

Theory:

Difference between errors and exceptions:

Exceptions are those which can be handled at the run time whereas errors cannot be handled. An exception is an Object of a type deriving from the System.Exception class. SystemException is thrown by the CLR (Common Language Runtime) when errors occur that are nonfatal and recoverable by user programs. It is meant to give you an opportunity to do something with throw statement to transfer control to a catch clause in a try block.

Exception handling

Exceptions: Exceptions are runtime anomalies or unusual conditions that a program may encounter while executing .Anomalies might include conditions such ass division by zero, accessing an array outside of its bounds or running out of memory or disk space. When a program encounters an exception condition, it must be identified and handled.

Exceptions provide a way to transfer control from one part of a program to another. C++ exception handling is built upon three keywords: try, catch, and throw.

Types of exceptions:

There are two kinds of exceptions
1.Synchronous exceptions
2.Asynchronous exceptions

1.Synchronous exceptions: Errors such as ―Out-of-range index‖ and ―over flow‖ are synchronous exceptions.

2.Asynchronous exceptions: The errors that are generated by any event beyond the control of the program are called asynchronous exceptions.

The purpose of exception handling is to provide a means to detect and report an exceptional circumstance.

Exception Handling Mechanism:

An exception is said to be thrown at the place where some error or abnormal condition is detected. The throwing will cause the normal program flow to be aborted, in a raised exception. An exception is thrown programmatic, the programmer specifies the conditions of a throw.

In handled exceptions, execution of the program will resume at a designated block of code, called a catch block, which encloses the point of throwing in terms of program execution. The catch block can be, and usually is, located in a different function than the point of throwing.

C++ exception handling is built upon three keywords: try, catch, and throw. Try is used to preface a block of statements which may generate exceptions. This block of statements is known as try block. When an exception is detected it is thrown by using throw statement in the try

block. Catch block catches the exception thrown by throw statement in the try block and handles it appropriately.

Algorithm:

Step 1: Create a class with two double type values.
Step 2: Input X & Y.
Step 3: Within try block check Y==0 or not.

```
Division= X/Y
    if Y==0
        throw(y) & catch corresponding exception.
    else
        Print Divison.
```

Step 4: End of program.

Program:

Output:

Conclusion: