

Dr. D. Y. Patil Pratishthan's

**DR. D. Y. PATIL INSTITUTE OF ENGINEERING, MANAGEMENT &
RESEARCH**

Approved by A.I.C.T.E, New Delhi , Maharashtra State Government, Affiliated to Savitribai Phule Pune University

Sector No. 29, PCNTDA , Nigidi Pradhikaran, Akurdi, Pune 411044. Phone: 020-27654470, Fax: 020-27656566

Website : www.dypiemr.ac.in Email : principal.dypiemr@gmail.com

Department of Computer Engineering

LAB MANUAL Object Oriented Programming and Computer Graphics Laboratory (SE) Semester I

Prepared By:

Ms. Shritika Waykar

Mr. Shivaji Vasekar

Mrs Manisha Ambekar



OOP & Computer Graphics Laboratory

Course Code	Course Name	Teaching Scheme (Hrs./ Week)	Credits
217523	Computer Graphics Laboratory	4	2

Course Objectives:

To understand basics of Computer Graphics, apply various methods and techniques for implementing line-circle drawing, projections, animation, shading, illumination and lighting using concepts of Object Oriented Programming.

Course Outcomes:

On completion of the course, learner will be able to—

CO1: Understand and **apply** the concepts like inheritance, polymorphism, exception handling and generic structures for implementing reusable programming codes.

CO2: Analyze the concept of file and **apply** it while storing and retrieving the data from secondary storages. **CO3: Analyze** and **apply** computer graphics algorithms for line-circle drawing, scan conversion and filling with the help of object oriented programming concepts.

CO4: Understand the concept of windowing and clipping and apply various algorithms to fill and clip polygons.


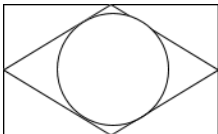
CO5: Apply logic to implement, curves, fractals, animation and gaming programs.

Operating System recommended: 64-bit Open source Linux or its derivative

Programming tools recommended: Use suitable programming language/Tool for implementation

Table of Contents

Sr. No	Title of Experiment	CO Mappin	Page No
Part I : Object Oriented Programming			
Group A			
1	Implement a class Complex which represents the Complex Number data type. Implement the following 1. Constructor (including a default constructor which creates the complex number 0+0i). 2. Overload operator+ to add two complex numbers. 3. Overload operator* to multiply two complex numbers. 4. Overload operators << and >> to print and read Complex Numbers.	CO1, CO2	0
2	Develop a program in C++ to create a database of student's information system containing the following information: Name, Roll number, Class, Division, Date of Birth, Blood group, Contact address, Telephone number, Driving license no. and other. Construct the database with suitable member functions. Make use of constructor, default constructor, copy constructor, destructor, static member functions, friend class, this pointer, inline code and dynamic memory allocation operators-new and delete as well as exception handling.	CO1, CO2	11
3	Imagine a publishing company which does marketing for book and audio cassette versions. Create a class publication that stores the title (a string) and price (type float) of publications. From this class derive two classes: book which adds a page count (type int) and tape which adds a playing time in minutes (type float). Write a program that instantiates the book and tape class, allows user to enter data and displays the data members. If an exception is caught, replace all the data member values with zero values.	CO1, CO2	15
Group B			
4	Write a C++ program that creates an output file, writes information to it, closes the file, open it again as an input file and read the information from the file.	CO1, CO2	18
5	Write a function template for selection sort that inputs, sorts and outputs an integer array and a float array.	CO1, CO2	21
Group C			

6	<p>Write C++ program using STL for sorting and searching user defined records such as personal records (Name, DOB, Telephone number etc) using vector container.</p> <p>OR</p> <p>Write C++ program using STL for sorting and searching user defined records such as Item records (Item code, name, cost, quantity etc) using vector container.</p>	CO1, CO2	24
7	Write a program in C++ to use map associative container. The keys will be the names of states and the values will be the populations of the states. When the program runs, the user is prompted to type the name of a state. The program then looks in the map, using the state name as an index and returns the population of the state.	CO1, CO2	27
Part II : Computer Graphics			
Group A			
8	Write C++ program to draw a concave polygon and fill it with desired color using scan fill algorithm.	CO4, CO5	31
9	Write C++ program to implement Cohen Southerland line clipping algorithm.	CO3, CO4, CO2	37
10	<p>Write C++ program to draw the following pattern. Use DDA line and Bresenham's circle drawing algorithm. Apply the concept of encapsulation.</p>  <p>OR</p> <p>Write C++ program to draw the following pattern. Use DDA line and Bresenham's circle drawing algorithm. Apply the concept of encapsulation</p> 	CO3	41
Group B			
11	<p>Write C++ program to draw 2-D object and perform following basic transformations: 1. Scaling 2. Translation 3. Rotation. Apply the concept of operator overloading.</p> <p>OR</p> <p>Write C++ program to implement translation, rotation and scaling transformations on equilateral triangle and rhombus. Apply the concept of operator overloading</p>	CO3	52

12	Write C++ program to generate snowflake using concept of fractals. OR Write C++ program to generate Hilbert curve using concept of fractals. OR Write C++ program to generate fractal patterns by using Koch curves.	CO3, CO5	63
	Group C		
13	Design and simulate any data structure like stack or queue visualization using graphics. Simulation should include all operations performed on designed data structure. Implement the same using OpenGL. OR Write C++ program to draw 3-D cube and perform following transformations on it using OpenGL i) Scaling ii) Translation iii) Rotation about an axis (X/Y/Z). OR Write OpenGL program to draw Sun Rise and Sunset.	CO3, CO5	73
14	Write a C++ program to control a ball using arrow keys. Apply the concept of polymorphism. OR Write a C++ program to implement bouncing ball using sine wave form. Apply the concept of polymorphism. OR Write C++ program to draw man walking in the rain with an umbrella. Apply the concept of polymorphism. OR Write a C++ program to implement the game of 8 puzzle. Apply the concept of polymorphism. OR Write a C++ program to implement the game Tic Tac Toe. Apply the concept of polymorphism.	CO3, CO5	79
	Mini- Project		
15	Design and implement game / animation clip / Graphics Editor using open source graphics library. Make use of maximum features of Object Oriented Programming.	CO1,2,3,4, 5	86

Lab Assignment No.	1
Title	Implement a class Complex which represents the Complex Number data type. Implement the following 1. Constructor (including a default constructor which creates the complex number 0+0i). 2. Overload operator+ to add two complex numbers. 3. Overload operator* to multiply two complex numbers. 4. Overload operators << and >> to print and read Complex Numbers.
Roll No.	
Class	SE
Date of Completion	
Subject	OOP & CG Lab
Assessment Marks	
Assessor's Sign	

LAB ASSIGNMENT 1

Problem Statement

Implement a class Complex which represents the Complex Number data type. Implement the following operations:

1. Constructor (including a default constructor which creates the complex number $0+0i$).
2. Overloaded operator+ to add two complex numbers.
3. Overloaded operator* to multiply two complex numbers.
4. Overloaded << and >> to print and read Complex Numbers.

Objectives:

1. Understand the concept of constructors and how to design it.
2. Understand the OOP concept of operator overloading.

Outcomes:

1. Will be able to design constructors for classes designed in programming.
2. Will be able to program using operator functions.

Software Requirements:

1. 64-bit Open source Linux or its derivative
2. Open Source C++ Programming tool like G++/GCC.

Theory:

C++ Class

A class is the collection of related data and function under a single name. A C++ program can have any number of classes. When related data and functions are kept under a class, it helps to visualize the complex problem efficiently and effectively.

Class is defined in C++ programming using keyword class followed by identifier(name of class). Body of class is defined inside curly brackets and terminated by semicolon at the end in similar way as structure.

```
class class_name
{
    // some data
    // some functions
};
```

Keyword private makes data and functions private and keyword public makes data and functions public. Private data and functions are accessible inside that class only whereas, public data and functions are accessible both inside and outside the class. This feature is known as data hiding.

C++ Objects

When class is defined, only specification for the object is defined. Object has same relationship to class as variable has with the data type. Objects can be defined in similar way as structure is defined.

```
class_name variable_name;
```

Data Member & Data Functions

The data within the class is known as data member. The function defined within the class is known as member function. Data members and member functions can be accessed in similar way the member of structure is accessed using member operator(.).

```
object_name.data_member;
```

Constructors:

Constructors are the special type of member function that initialises the object automatically when it is created. Compiler identifies that the given member function is a constructor by its name and return type. Constructor has same name as that of class and it does not have any return type.

Constructor can be overloaded in similar way as function overloading. Overloaded constructors have same name(name of the class) but different number of argument passed. Depending upon the number and type of argument passed, specific constructor is called. Since, constructor are called when object is created. Argument to the constructor also should be passed while creating object.

A object can be initialized with another object of same type.- Default Copy Constructor. If you want to initialise a object A3 so that it contains same value as A2.

```
Area A3(A2); /* Copies the content of A2 to A3 */
```

OR,

```
Area A3=A2; /* Copies the content of A2 to A3 */
```

Operator Overloading:

The meaning of operators are already defined and fixed for basic types like: int, float, double etc in C++ language. For example: If you want to add two integers then, + operator is used. But, for user-defined types(like: objects), you can define the meaning of operator, i.e, you can redefine the way that operator works. For example: If there are two objects of a class that contain string as its data member, you can use + operator to concatenate two strings. Suppose, instead of strings if that class contains integer data member, then you can use + operator to add integers. This feature in C++ programming that allows programmer to redefine the meaning of operator when they operate on class objects is known as operator overloading.

To overload a operator, a operator function is defined inside a class as:

```
class class_name
{
.....
public:
    return_type operator sign (argument/s)
    {
        .....
    }
.....
};
```

The return type comes first which is followed by keyword **operator**, followed by operator sign,i.e., the operator you want to overload like: +, <, ++ etc. and finally the arguments is passed. Then, inside the body of you want perform the task you want when this operator function is called.This operator function is called when, the operator(sign) operates on the object of that class class_name.

Algorithm

1. Design a class `Complex` which represents the Complex Number data type.

2. Create a constructor which creates the complex number `0+0i`

`Complex(): real(0), imag(0){ }`

3. Use operator overloading to overload operator `+` to add two complex numbers.

`Complex operator + (Complex c2)`

4. Use operator overloading to overload operator `*` to multiply two complex numbers.

`Complex operator * (Complex c3)`

5. Use operator overloading to overload operator `<<` and `>>` to print and read Complex Numbers.

`friend ostream& operator << (ostream &out, Complex &c); //overloading '<<' operator`

`friend istream& operator >> (istream &in, Complex &c); //overloading '>>' operator`

Conclusion:

Lab Assignment No.	2
Title	Develop a program in C++ to create a database of student's information system containing the following information: Name, Roll number, Class, Division, Date of Birth, Blood group, Contact address, Telephone number, Driving license no. and other. Construct the database with suitable member functions. Make use of constructor, default constructor, copy constructor, destructor, static member functions, friend class, this pointer, inline code and dynamic memory allocation operators-new and delete as well as exception handling.
Roll No.	
Class	SE
Date of Completion	
Subject	OOP & CG Lab
Assessment Marks	
Assessor's Sign	

LAB ASSIGNMENT 2

Problem Statement

Develop an object oriented program in C++ to create a database of student information system containing the following information: Name, Roll number, Class, division, Date of Birth, Blood group, Contact address, telephone number, driving license no. etc Construct the database with suitable member functions for initializing and destroying the data viz constructor, default constructor, Copy constructor, destructor, static member functions, friend class, this pointer, inline code and dynamic memory allocation operators-new and delete.

Objectives:

1. Understand the concept of constructor and its use
2. Understand the concept friend function
3. Understand the concept of copy constructor.
4. Understand the use of static and inline function
5. Understand the concept of exception handling

Outcomes:

1. Will be to make use of constructors
2. Will be able to design a program using friend functions.
3. Will be able to design a program using exception handling.

Software Requirements:

1. 64-bit Open source Linux or its derivative
2. Open Source C++ Programming tool like G++/GCC.

Theory:

Constructors:

A special method of the class that will be automatically invoked when an instance of the class is created is called as constructor. Following are the most useful features of constructor.

- 1) Constructor is used for Initializing the values to the data members of the Class.
- 2) Constructor is that whose name is same as name of class.
- 3) Constructor gets Automatically called when an object of class is created.

- 4) Constructors never have a Return Type even void.
- 5) Constructor is of Default, Parameterized and Copy Constructors.

The various types of Constructor are as follows:-

Constructors can be classified into 3 types

1. Default Constructor
2. Parameterized Constructor
3. Copy Constructor

1. Default Constructor:- Default Constructor is also called as Empty Constructor which has no arguments and It is Automatically called when we creates the object of class but Remember name of Constructor is same as name of class and Constructor never declared with the help of Return Type. Means we can't declare a Constructor with the help of void Return Type. , if we never Pass or declare any Arguments then this called as the Copy Constructors.

2. Parameterized Constructor: - This is another type constructor which has some Arguments and same name as class name but it uses some Arguments So For this We have to create object of Class by passing some Arguments at the time of creating object with the name of class. When we pass some Arguments to the Constructor then this will automatically pass the Arguments to the Constructor and the values will retrieve by the Respective Data Members of the Class.

3. Copy Constructor: - This is also another type of Constructor. In this Constructor we pass the object of class into the Another Object of Same Class. As name Suggests you Copy, means Copy the values of one Object into the another Object of Class .This is used for Copying the values of class object into an another object of class So we call them as Copy Constructor and For Copying the values We have to pass the name of object whose values we wants to Copying and When we are using or passing an Object to a Constructor then we must have to use the & Ampersand or Address Operator.

Destructor: As we know that Constructor is that which is used for Assigning Some Values to data Members and For Assigning Some Values this May also used Some Memory so that to free up the Memory which is Allocated by Constructor, destructor is used which gets Automatically Called at the End of Program and we doesn't have to Explicitly Call a Destructor and Destructor Cant be Parameterized or a Copy This can be only one Means Default

Destructor which Have no Arguments. For Declaring a Destructor we have to use ~tiled Symbol in front of Destructor

Friend Class A friend class can access private and protected members of other class in which it is declared as friend. It is sometimes useful to allow a particular class to access private members of other class. For example, a LinkedList class may be allowed to access private members of Node.

A static member function can access only the names of static members, enumerators, and nested types of the class in which it is declared. Suppose a static member function f() is a member of class X . The static member function f() cannot access the nonstatic members X or the nonstatic members of a base class of X

Inline function is one of the important feature of C++. So, let's first understand why inline functions are used and what is the purpose of inline function?

When the program executes the function call instruction the CPU stores the memory address of the instruction following the function call, copies the arguments of the function on the stack and finally transfers control to the specified function. The CPU then executes the function code, stores the function return value in a predefined memory location/register and returns control to the calling function. This can become overhead if the execution time of function is less than the switching time from the caller function to called function (callee). For functions that are large and/or perform complex tasks, the overhead of the function call is usually insignificant compared to the amount of time the function takes to run. However, for small, commonly-used functions, the time needed to make the function call is often a lot more than the time needed to actually execute the function's code. This overhead occurs for small functions because execution time of small function is less than the switching time.

C++ provides an inline functions to reduce the function call overhead. Inline function is a function that is expanded in line when it is called. When the inline function is called whole code of the inline function gets inserted or substituted at the point of inline function call. This substitution is performed by the C++ compiler at compile time.

Inline function may increase efficiency if it is small.
The syntax for defining the function inline is:

```
inline return-type function-name(parameters)
{
    // function code
}
```

Conclusion:

Lab Assignment No.	3
Title	Imagine a publishing company which does marketing for book and audio cassette versions. Create a class publication that stores the title (a string) and price (type float) of publications. From this class derive two classes: book which adds a page count (type int) and tape which adds a playing time in minutes (type float). Write a program that instantiates the book and tape class, allows user to enter data and displays the data members. If an exception is caught, replace all the data member values with zero values.
Roll No.	
Class	SE
Date of Completion	
Subject	OOP & CG Lab
Assessment Marks	
Assessor's Sign	

LAB ASSIGNMENT 3

Problem Statement

Imagine a publishing company that markets both book and audiocassette versions of its works. Create a class publication that stores the title (a string) and price (type float) of a publication. From this class derive two classes: book, which adds a page count (type int), and tape, which adds a playing time in minutes (type float). Each of these three classes should have a getdata() function to get its data from the user at the keyboard, and a putdata() function to display its data. Write a main() program to test the book and tape classes by creating instances of them, asking the user to fill in data with getdata(), and then displaying the data with putdata().

Objectives:

To learn the concept of Class, Object, Data Hiding, Inheritance

Outcomes:

The performer will be able to design a program in c++ using concept Class, Object, Data Hiding, Inheritance

Software Requirements:

1. 64-bit Open source Linux or its derivative
2. Open Source C++ Programming tool like G++/GCC.

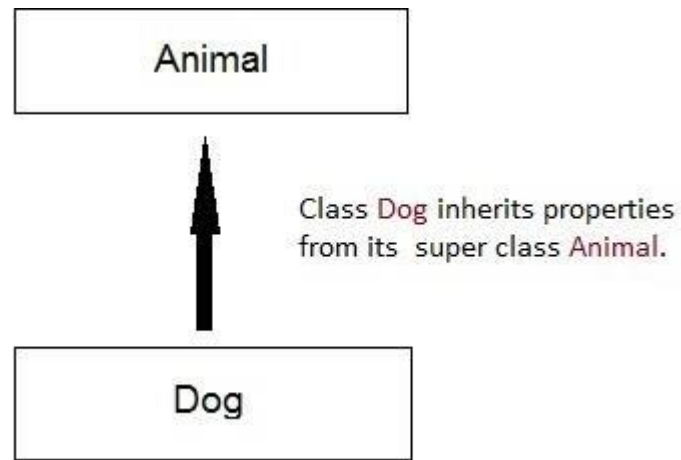
Theory:

Inheritance

Inheritance is the capability of one class to acquire properties and characteristics from another class. The class whose properties are inherited by other class is called the **Parent** or **Base** or **Super** class. And, the class which inherits properties of other class is called **Child** or **Derived** or **Sub** class. Inheritance makes the code reusable. When we inherit an existing class, all its methods and fields become available in the new class, hence code is reused.

class Subclass_name : access_mode Superclass_name

While defining a subclass like this, the super class must be already defined or atleast declared before the subclass declaration. Access Mode is used to specify, the mode in which the properties of superclass will be inherited into



subclass,
public,
private or
protected.

Depending on Access modifier used while inheritance, the availability of class members of Super class in the sub class changes. It can either be private, protected or public.

Base class		Derived Class		
	Public Mode	Private Mode	Protected Mode	
Protected	Protected	Private	Protected	

Algorithm:

1. Design a Base Class publication which contains fields to store title and price..
2. Design two derived classes to hold properties of Tape and Book respectively.
3. Create functions for getting data and putting data in the base class as well as in the derived class.
4. Create main() program to test the book and tape classes by creating instances of them, asking the user to fill in data with getdata(), and then displaying the data with putdata().

Conclusion:

Lab Assignment No.	4
Title	Write a C++ program that creates an output file, writes information to it, closes the file, open it again as an input file and read the information from the file.
Roll No.	
Class	SE
Date of Completion	
Subject	OOP & CG Lab
Assessment Marks	
Assessor's Sign	

LAB ASSIGNMENT 4

Problem Statement

Write a C++ program that creates an output file, writes information to it, closes the file and open it again as an input file and read the information from the file.

Objectives:

To learn the concept of File Handling in c++ programming

Outcomes:

The performer will be able to design a program in c++ using concept File Handling

Software Requirements:

1. 64-bit Open source Linux or its derivative
2. Open Source C++ Programming tool like G++/GCC.

Theory:

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

```
void open(const char *filename, ios::openmode mode);
```

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

When a C++ program terminates it automatically closes flushes all the streams, release all the allocated memory and close all the opened files. But it is always a good practice that a programmer should close all the opened files before program termination.

```
void close();
```

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

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

Algorithm:

1. Create a file “telephone.txt”
2. Read input from the user .
3. Write the data to the file using `ofstream`.
4. Read the input from the file using `ifstream`
5. Display the contents

Conclusion:

Lab Assignment No.	5
Title	Write a function template for selection sort that inputs, sorts and outputs an integer array and a float array.
Roll No.	
Class	SE
Date of Completion	
Subject	OOP & CG Lab
Assessment Marks	
Assessor's Sign	

LAB ASSIGNMENT 5

Problem Statement

Write a function template selection Sort. Write a program that inputs, sorts and outputs an int array and a float array.

Objectives:

To learn the use of Templates and sorting in c++ programming

Outcomes:

The performer will be able to design a program in c++ using Templates and Sorting .

Software Requirements:

1. 64-bit Open source Linux or its derivative
2. Open Source C++ Programming tool like G++/GCC.

Theory:

Templates

Templates are the foundation of generic programming, which involves writing code in a way that is independent of any particular type.

A template is a blueprint or formula for creating a generic class or a function. The library containers like iterators and algorithms are examples of generic programming and have been developed using template concept.

The general form of a template function definition is shown here:

```
template <class type> ret-type func-name(parameter list)
{
    // body of function
}
```

Here, type is a placeholder name for a data type used by the function. This name can be used within the function

definition.

Just as we can define function templates, we can also define class templates. The general form of a generic class declaration is shown here:

```
template <class type> class class-name {
```

Here, type is the placeholder type name, which will be specified when a class is instantiated. You can define more than one generic data type by using a comma-separated list.

Algorithm:

1. Create Template template <typename T>
 2. Create a function to perform sorting.
- ```
void Sort(T* const array,int size){
```

3. Create a function to display array
- ```
void printarray(T* const array,int size)
```

Conclusion:

Lab Assignment No.	6
Title	Write C++ program using STL for sorting and searching user defined records such as personal records (Name, DOB, Telephone number etc) using vector container. OR Write C++ program using STL for sorting and searching user defined records such as Item records (Item code, name, cost, quantity etc) using vector container.
Roll No.	
Class	SE
Date of Completion	
Subject	OOP & CG Lab
Assessment Marks	
Assessor's Sign	

LAB ASSIGNMENT 6

Problem Statement

Write C++ program using STL for Sorting and searching with user-defined records such as Person Record (Name, birth date, telephone no), item record (item code, item name, quantity and cost)

Objectives:

To learn the use of standard template library (STL) in c++ programming

Outcomes:

The performer will be able to design a program in c++ using standard template library (STL) .

Software Requirements:

1. 64-bit Open source Linux or its derivative
2. Open Source C++ Programming tool like G++/GCC.

Theory:

STL Container List:

Array and Vector are contiguous containers, i.e they store their data on continuous memory, thus the insert operation at the middle of vector/array is very costly (in terms of number of operation and process time) because we have to shift all the elements, linked list overcome this problem. Linked list can be implemented by using the list container.

```
list<T> lst;
```

Construct an empty list lst which can hold values of type T.

```
list<T> lst(inIterBegin, inIterEnd);
```

Construct lst containing values from the range[inIterBegin,inIterEnd) in another (not necessarily list) container, but whose component type is the same as the component type of lst.

```
lst.begin()
```

Return an iterator (or const_iterator) pointing to the first component of lst.

`lst.end()`

Return an iterator (or `const_iterator`) pointing to one-past-the-last component of `lst`.

`lst.rbegin()`

Return a `reverse_iterator` (or `const_reverse_iterator`) pointing to the last component of `lst`.

`lst.rend()`

Return a `reverse_iterator` (or `const_reverse_iterator`) pointing to one-before-the-first component of `lst`.

`lst.size()`

Return a value of type `size_type` giving the number of values currently in `lst`.

`lst.empty()`

Return true if `lst` is empty (contains zero values); otherwise return false.

`lst.insert(iter, val)`

Insert `val` into `lst` immediately before the value pointed to by `iter`, and return an iterator pointing at the new component with value `val`.

`lst.unique()`

Remove all but one of any group of consecutive components containing duplicate items.

`lst.merge(otherList)`

Merge the list contained in `otherList` with the invoking list and clear `otherList`. (Both lists are assumed to be ordered and the result is also ordered.)

`lst.reverse()`

Reverse the order of all components of a list.

`lst.sort()`

Sort the components of a list into ascending order.

Conclusion:

Lab Assignment No.	7
Title	Write a program in C++ to use map associative container. The keys will be the names of states and the values will be the populations of the states. When the program runs, the user is prompted to type the name of a state. The program then looks in the map, using the state name as an index and returns the population of the state.
Roll No.	
Class	SE
Date of Completion	
Subject	OOP & CG Lab
Assessment Marks	
Assessor's Sign	

LAB ASSIGNMENT 7

Problem Statement:

Write a program in C++ to use map associative container. The keys will be the names of states and the values will be the populations of the states. When the program runs, the user is prompted to type the name of a state. The program then looks in the map, using the state name as an index and returns the population of the state.

Objectives:

1. Understand the concept of map in associate container
2. Understand the concept friend function

Outcomes:

1. Will be to make use of constructors
2. Will be able to design a program using friend functions.
3. Will be able to design a program using exception handling.

Software Requirements:

3. 64-bit Open source Linux or its derivative
4. Open Source C++ Programming tool like G++/GCC.

Theory:

Maps are associative containers that store elements in a mapped fashion. Each element has a key value and a mapped value. No two mapped values can have same key values.

Some basic functions associated with Map:

begin() – Returns an iterator to the first element in the map
end() – Returns an iterator to the theoretical element that follows last element in the map
size() – Returns the number of elements in the map
max_size() – Returns the maximum number of elements that the map can hold
empty() – Returns whether the map is empty
pair insert(keyvalue, mapvalue) – Adds a new element to the map
erase(iterator position) – Removes the element at the position pointed by the iterator
erase(const g)– Removes the key value ‘g’ from the map
clear() – Removes all the elements from the map

Creating a Map in C++ STL

Maps can easily be created using the following statement :

```
map<key_type, value_tye> map_name;
```

Copy

This will create a map with key of type Key_type and value of type value_type. One thing which is to remembered is that key of a map and corresponding values are always inserted as a pair, you cannot insert only key or just a value in a map.

Conclusion:

Part II : Computer Graphics

Lab Assignment No.	8
Title	Write C++ program to draw a concave polygon and fill it with desired color using scan fill algorithm.
Roll No.	
Class	SE
Date of Completion	
Subject	OOP & CG Lab
Assessment Marks	
Assessor's Sign	

LAB ASSIGNMENT 8

TITLE: Write C++ program to draw a concave polygon and fill it with desired color using scan fill algorithm.

PROBLEM STATEMENT: To draw a concave polygon and fill it with desired color using seed fill algorithm

OBJECTIVES:

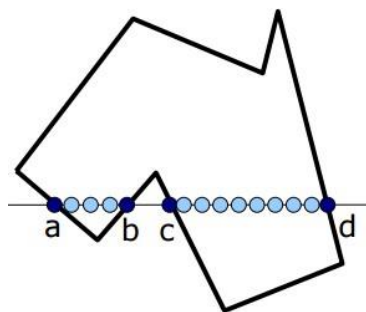
- To understand Scan Line drawing algorithms used for polygon filling computer graphics.
- To understand concept of different line styles using line algorithms for filling polygon with desired pattern.

OUTCOMES: Implement computer graphics programs in C++ using the polygon drawing algorithm and fill the polygon using scan fill algorithm.

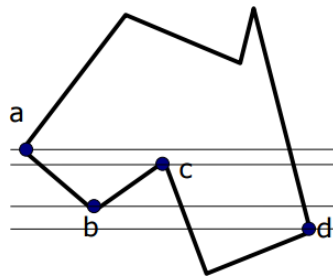
THEORY CONCEPTS/ LOGIC/ ALGORITHM:

Scan-line Polygon Fill

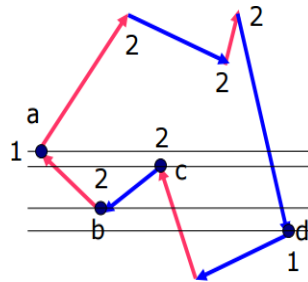
- For each scan-line:
 - Locate the intersection of the scan line with the edges ($y=y_s$)
 - Sort the intersection points from left to right.
 - Draw the interiors intersection points pairwise. (a-b), (c-d)
- Problem with corners. Same point counted twice or not?



- a , b , c and d are intersected by 2 line segments each.



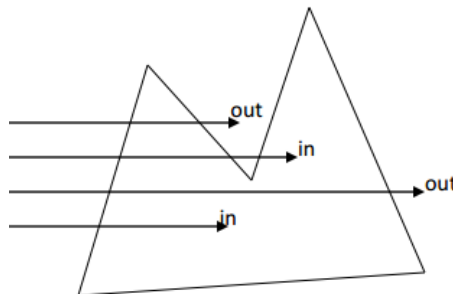
- Count b,c twice but a and d once. Why?



Solution: Make a clockwise or counter-clockwise traversal on edges. Check if y is monotonically increasing or decreasing. If direction changes, double intersection, otherwise single intersection.

Basic algorithm:

- Assume scan line start from the left and is outside the polygon.
- When intersect an edge of polygon, start to color each pixel (because now we're inside the polygon), when intersect another edge, stop coloring ...
- Odd number of edges: inside
- Even number of edges: outside
- **Advantage of scan-line fill:** It does fill in the same order as rendering, and so can be pipelined.



Scan-Line Polygon Fill Algorithm

• Odd-parity rule

Calculate span extrema (intersections) on each scan line

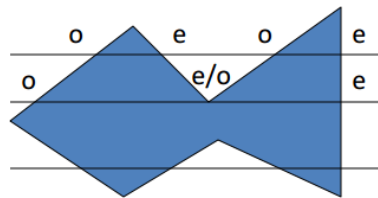
Using parity rule to determine whether or not a point is inside a region

Parity is initially even

◇ each intersection encountered thus inverts the parity bit

parity is odd -> interior point (draw)

parity is even-> exterior point (do not draw)

**setlinestyle****Syntax**

```
#include <graphics.h>
```

```
void setlinestyle(int linestyle, unsigned upattern, int thickness);
```

Description

setlinestyle sets the style for all lines drawn by line, lineto, rectangle, drawpoly, and so on.

The linesettingtype structure is defined in graphics.h as follows:

```
struct linesettingtype
{
    int linestyle;
    unsigned upattern;
    int thickness;
```

```
};
```

linestyle specifies in which of several styles subsequent lines will be drawn (such as solid, dotted, centered, dashed). The enumeration `line_styles`, which is defined in `graphics.h`, gives names to these operators:

Name	Value	Description
SOLID_LINE	0	Solid line
DOTTED_LINE	1	Dotted line
CENTER_LINE	2	Centered line
DASHED_LINE	3	Dashed line
USERBIT_LINE	4	User-defined line style

thickness specifies whether the width of subsequent lines drawn will be normal or thick.

Name	Value	Description
NORM_WIDTH	1	1 pixel wide
THICK_WIDTH	3	3 pixels wide

upattern is a 16-bit pattern that applies only if linestyle is `USERBIT_LINE` (4). In that case, whenever a bit in the pattern word is 1, the corresponding pixel in the line is drawn in the current drawing color. For example, a solid line corresponds to a upattern of `0xFFFF` (all pixels drawn), and a dashed line can correspond to a upattern of `0x3333` or `0x0F0F`. If the linestyle parameter to `setlinestyle` is not `USERBIT_LINE` (in other words, if it is not equal to 4), you must still provide the upattern parameter, but it will be ignored.

Note: The linestyle parameter does not affect arcs, circles, ellipses, or pie slices. Only the thickness parameter is used.

Return Value

If invalid input is passed to `setlinestyle`, `graphresult` returns -11, and the current line style remains unchanged.

CONCLUSION: Thus we have studied C++/Java program to draw a concave polygon and fill it with desired pattern using scan line algorithm.

Questions:

1. Explain concave polygons.
2. Explain different tests used for checking whether pixel is inside or outside the polygon.
3. Explain different ways of filling the polygon.
4. What is meant by Active Edge Table and Global Edge Table?
5. What are the different methods of polygon filling?
6. How scan line algorithm works?
7. What Is polygon clipping?
8. How you take input as a polygon?
9. What convex polygon?
10. What is meaning of seed point?

Lab Assignment No.	9
Title	Write C++ program to implement Cohen Southerland line clipping algorithm.
Roll No.	
Class	SE
Date of Completion	
Subject	OOP & CG Lab
Assessment Marks	
Assessor's Sign	

LAB ASSIGNMENT 9

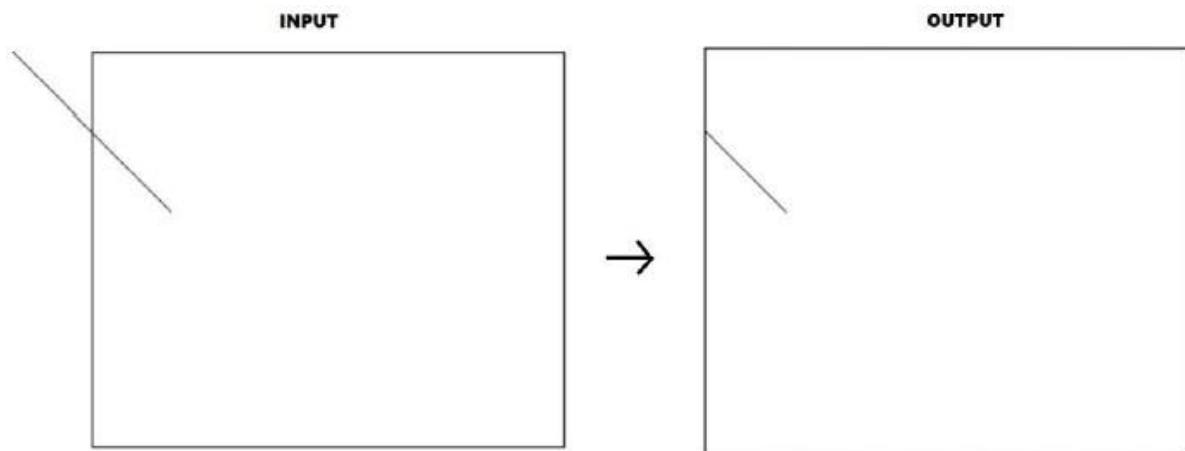
TITLE: Write C++ program to implement Cohen Southerland line clipping algorithm.

PROBLEM STATEMENT: Polygon clipping using Cohen Southerland line clipping algorithm

THEORY:

Cohen Sutherland Algorithm is a **line clipping algorithm** that cuts lines to portions which are within a rectangular area. It eliminates the lines from a given set of lines and rectangle area of interest (view port) which belongs outside the area of interest and clips those lines which are partially inside the area of interest.

Example:

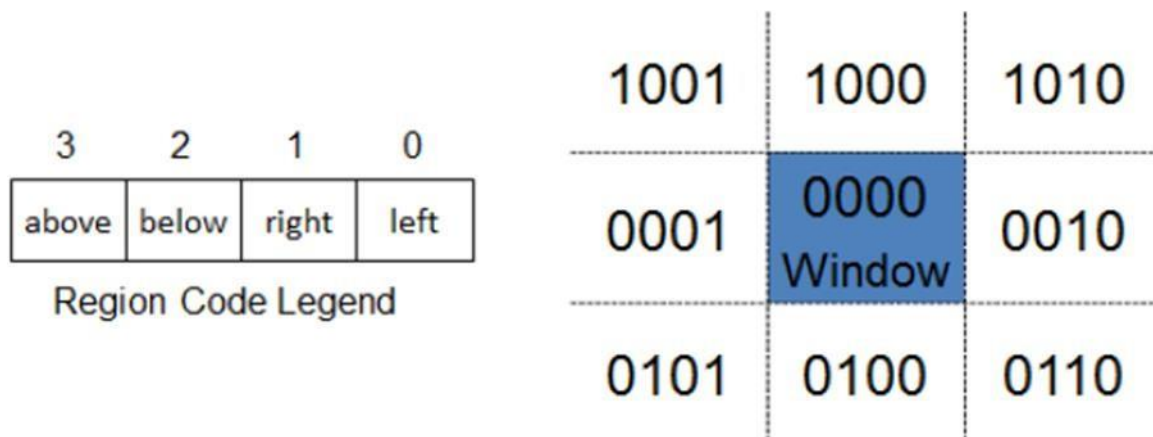


Cohen Sutherland Line Clipping Algorithm

Algorithm

The algorithm divides a **two-dimensional space** into **9 regions** (eight outside regions and one inside region) and then efficiently determines the lines and portions of lines that are visible in the central region of interest (the viewport).

Following image illustrates the 9 regions:

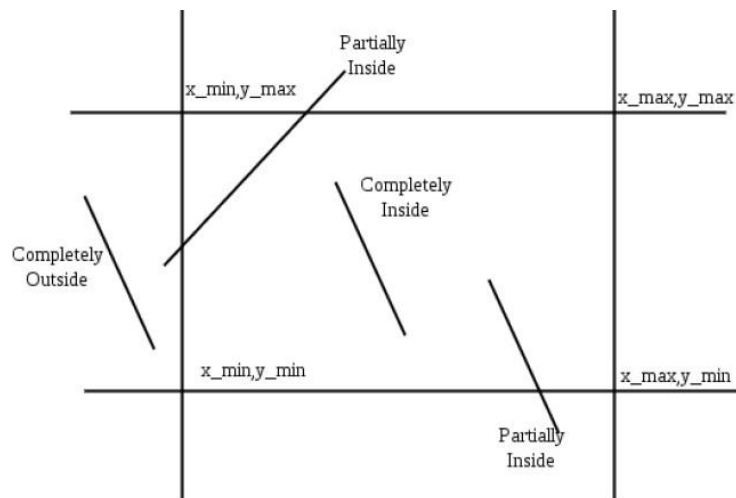


As you seen each region is denoted by a 4 bit code like 0101 for the bottom right region
Four Bit code is calculated by comparing extreme end point of given line (x,y) by four co-ordinates x_{min} , x_{max} , y_{max} , y_{min} which are the coordinates of the area of interest (0000)

Calculate the four bit code as follows:

- Set First Bit if 1 Points lies to left of window ($x < x_{min}$)
- Set Second Bit if 1 Points lies to right of window ($x > x_{max}$)
- Set Third Bit if 1 Points lies to left of window ($y < y_{min}$)

Set Forth Bit if 1 Points lies to right of window ($y > y_{max}$)



The more efficient Cohen-Sutherland Algorithm performs initial tests on a line to determine whether intersection calculations can be avoided.

Pseudocode

Step 1 : Assign a region code for two endpoints of given line

Step 2 : If both endpoints have a region code 0000 then given line is completely inside and we will keep this line.

Step 3: If step 2 fails, perform the logical AND operation for both region codes.

Step 3.1: If the result is not 0000, then given line is completely outside.

Step 3.2 : Else line is partially inside.

Step 3.2.a : Choose an endpoint of the line that is outside the given rectangle.

Step 3.2.b : Find the intersection point of the rectangular boundary (based on region code)

Step 3.2.c : Replace endpoint with the intersection point and upgrade the region code.


Step 3.2.d : Repeat step 2 until we find a clipped line either trivially accepted or rejected.

Step 4: Repeat step 1 for all lines.

CONCLUSION: Thus we have studied Cohen Sutherland Line Clipping Algorithm

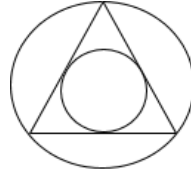
QUESTIONS:

1. What is the limitation of Cohen Sutherland Line Clipping algorithm?
2. What are the advantages of Cohen Sutherland Line Clipping?

Lab Assignment No.	10-A
Title	Write C++ program to draw the following pattern. Use DDA line and Bresenham's circle drawing algorithm. Apply the concept of encapsulation. 
Roll No.	
Class	SE
Date of Completion	
Subject	OOP & CG Lab
Assessment Marks	
Assessor's Sign	

LAB ASSIGNMENT 10-A

TITLE: Write C++ program to draw the following pattern. Use DDA line and Bresenham's circle drawing algorithm. Apply the concept of encapsulation.



PROBLEM STATEMENT: To draw the following pattern. Use DDA line and Bresenham's circle drawing algorithm

OBJECTIVE:

- To understand the DDA and Bresenham's algorithm Line drawing Algorithm.
- To understand the Midpoint and Bresenham's Circle drawing algorithm.
- To understand the basics of inscribed and Circumscribed Circles.

THEORY:**DDA Line Drawing Algorithm**

DDA algorithm is an incremental scan conversion method. Here we perform calculations at each step using the results from the preceding step. The characteristic of the DDA algorithm is to take unit steps along one coordinate and compute the corresponding values along the other coordinate. The unit steps are always along the coordinate of greatest change.

DDA Algorithm

```
Procedure DDA( x1, y1, x2, y2: integer);  
    dx, dy, steps: integer;  
    x_inc, y_inc, x, y: real;  
begin  
    dx := x2 - x1; dy := y2 - y1;  
    if abs(dx) > abs(dy) then  
        steps := abs(dx); {steps is larger of dx, dy}  
    else  
        steps := abs(dy);  
        x_inc := dx/steps; y_inc := dy/steps;  
        {either x_inc or y_inc = 1.0, the other is the slope}
```

```
x:=x1; y:=y1;  
set_pixel(round(x), round(y));  
for i := 1 to steps do  
begin  
    x := x + x_inc;  
    y := y + y_inc;  
    set_pixel(round(x), round(y));  
end;  
end;
```

Advantages of DDA Algorithm

- It is the simplest algorithm and it does not require special skills for implementation.
- It is a faster method
- for calculating pixel positions than the direct use of equation $y = mx + b$. It eliminates the multiplication in the equation by making use of raster characteristics, so that appropriate increments are applied in the x or y direction to find the pixel positions along the line path.

Disadvantages of DDA Algorithm

1. Floating point arithmetic in DDA algorithm is still time-consuming.
2. The algorithm is orientation dependent. Hence end point accuracy is poor.

Let us see few examples to illustrate this algorithm.

Circle Drawing Algorithms:

Bresenham's Circle drawing algorithm:-

This algorithm does only integer arithmetic which makes it faster than floating points. We are creating only one octant of the circle i.e., from 90 degree to 45 degree.

We can derive other seven octants of the circle by symmetry property.

The circle is generated by considering center points as origin with radius 'r'. This algorithm creates one pixel per step. From any point (x_n, y_n) on the circle, next point (x_{n+1}, y_{n+1}) must be either one on right or one to the right and down.

We cannot display a continuous arc on the raster display. Instead, we have to choose the nearest pixel position to complete the arc.

From the following illustration, you can see that we have put the pixel at (X, Y) location and now need to decide where to put the next pixel – at N (X+1, Y) or at S (X+1, Y-1).

Midpoint circle drawing algorithm

In computer graphics, the midpoint circle algorithm is an algorithm used to determine the points needed for drawing a circle. The algorithm is a variant of Bresenham's line algorithm, and is thus sometimes known as Bresenham's circle algorithm, although not actually invented by Jack E. Bresenham. The Midpoint circle drawing algorithm works on the same midpoint concept as the Bresenham's line algorithm. In the Midpoint circle drawing algorithm, you determine the next pixel to be plotted based on the position of the midpoint between the current and next consecutive pixel.

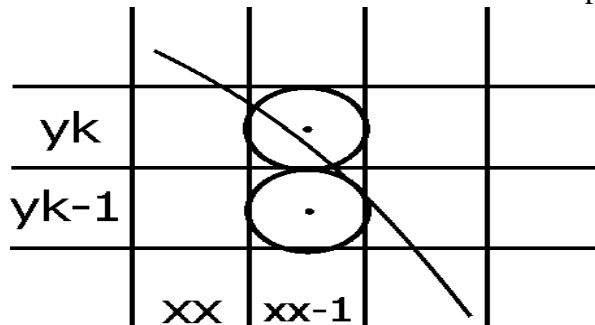


Fig 6.1 Midpoint circle Algorithm

Platform Required:

- Microsoft Windows 7/ Windows 8 Operating System onwards or 64-bit Open source Linux or its derivative.

Algorithm :

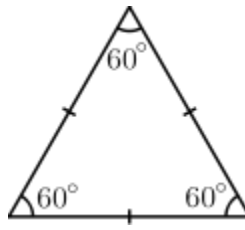
Bresenham's Algorithm to Draw a Circle.

1. Start.
2. Declare variables x,y,p and also declare gdriver=DETECT,gmode.
3. Initialise the graphic mode with the path location in TC folder.
4. Input the radius of the circle r.
5. Load x=0,y=r,initial decision parameter $p=1-r$.so the first point is (0,r).
6. Repeat Step 7 while $(x < y)$ and increment x-value simultaneously. 7. If $(p > 0)$,
do $p = p + 2*(x - y) + 1$.
8. Otherwise $p = p + 2*x + 1$
and
y is decremented simultaneously.

9. Then calculate the value of the function circlepoints() with parameters (x,y).
10. Place pixels using putpixel at points (x+300,y+300) in specified colour in circlepoints() function shifting the origin to 300 on both x-axis and y-axis.
11. Close Graph.
12. Stop.

Equilateral triangle

In geometry, an equilateral triangle is a triangle in which all three sides are equal. In the familiar Euclidean geometry, equilateral triangles are also equiangular; that is, all three internal angles are also congruent to each other and are each 60° . They are regular polygons, and can therefore also be referred to as regular triangles.



Equal cevians

Three kinds of cevians are equal for (and only for) equilateral triangles:

- The three altitudes have equal lengths.
- The three medians have equal lengths.
- The three angle bisectors have equal lengths.

Coincident triangle centers

Every triangle center of an equilateral triangle coincides with its centroid, which implies that the equilateral triangle is the only triangle with no Euler line connecting some of the centers. For some pairs of triangle centers, the fact that they coincide is enough to ensure that the triangle is equilateral. In particular:

- A triangle is equilateral if any two of the circumcenter, incenter, centroid, or orthocenter coincide.
- It is also equilateral if its circumcenter coincides with the Nagel point, or if its incenter coincides with its nine-point center.

Six triangles formed by partitioning by the medians

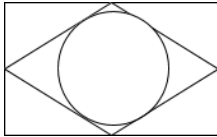
For any triangle, the three medians partition the triangle into six smaller triangles.

- A triangle is equilateral if and only if any three of the smaller triangles have either the same perimeter or the same inradius.
- A triangle is equilateral if and only if the circumcenters of any three of the smaller triangles have the same distance from the centroid.

CONCLUSION: Thus we have seen the two line drawing algorithms, the two Circle drawing algorithm and successfully implemented the given pattern.

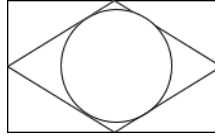
Questions:

- 1) What does DDA stand for?
- 2) What is faster DDA or bresenham's? why?
- 3) What occupies more memory – DDA or bresenham's? why?
- 4) Which gives better output - DDA or bresenham's? why?
- 5) What is abs()?
- 6) What is significance of sign() function in DDA?
- 7) What is the initial value of error variable for slope > 1 ?
- 8) What is the initial value of error variable for slope ≤ 1 ?
- 9) Which algorithm gives better output? Bresenham's or DDA?
- 10) Where is abs defined?

Lab Assignment No.	10-B
Title	Write C++ program to draw the following pattern. Use DDA line and Bresenham's circle drawing algorithm. Apply the concept of encapsulation 
Roll No.	
Class	SE
Date of Completion	
Subject	OOP & CG Lab
Assessment Marks	
Assessor's Sign	

LAB ASSIGNMENT 10-B

TITLE: Write C++ program to draw the following pattern. Use DDA line and Bresenham's circle drawing algorithm. Apply the concept of encapsulation



PROBLEM STATEMENT: To draw the following pattern. Use DDA line and Bresenham's circle drawing algorithm

OBJECTIVE:

- To understand the DDA and Bresenham's algorithm Line drawing Algorithm.
- To understand the Midpoint and Bresenham's Circle drawing algorithm.
- To understand the basics of inscribed and Circumscribed Circles.

THEORY:**DDA Line Drawing Algorithm**

DDA algorithm is an incremental scan conversion method. Here we perform calculations at each step using the results from the preceding step. The characteristic of the DDA algorithm is to take unit steps along one coordinate and compute the corresponding values along the other coordinate. The unit steps are always along the coordinate of greatest change.

DDA Algorithm

```
Procedure DDA( x1, y1, x2, y2: integer);
  dx, dy, steps: integer;
  x_inc, y_inc, x, y: real;
begin
  dx := x2 - x1; dy := y2 - y1;
  if abs(dx) > abs(dy) then
    steps := abs(dx); {steps is larger of dx, dy}
  else
    steps := abs(dy);
    x_inc := dx/steps; y_inc := dy/steps;
    {either x_inc or y_inc = 1.0, the other is the slope}
```



```
x:=x1; y:=y1;
set_pixel(round(x), round(y));
for i := 1 to steps do
begin
    x := x + x_inc;
    y := y + y_inc;
    set_pixel(round(x), round(y));
end;
end;
```

Advantages of DDA Algorithm

- It is the simplest algorithm and it does not require special skills for implementation.
- It is a faster method
- for calculating pixel positions than the direct use of equation $y = mx + b$. It eliminates the multiplication in the equation by making use of raster characteristics, so that appropriate increments are applied in the x or y direction to find the pixel positions along the line path.

Disadvantages of DDA Algorithm

1. Floating point arithmetic in DDA algorithm is still time-consuming.
2. The algorithm is orientation dependent. Hence end point accuracy is poor.

Let us see few examples to illustrate this algorithm.

Circle Drawing Algorithms:

Bresenham's Circle drawing algorithm:-

This algorithm does only integer arithmetic which makes it faster than floating points. We are creating only one octant of the circle i.e., from 90 degree to 45 degree.

We can derive other seven octants of the circle by symmetry property.

The circle is generated by considering center points as origin with radius 'r'. This algorithm creates one pixel per step. From any point (x_n, y_n) on the circle, next point (x_{n+1}, y_{n+1}) must be either one to the right or one to the left and down.

We cannot display a continuous arc on the raster display. Instead, we have to choose the nearest pixel position to complete the arc.

From the following illustration, you can see that we have put the pixel at (X, Y) location and now need to decide where to put the next pixel – at N (X+1, Y) or at S (X+1, Y-1).

Midpoint circle drawing algorithm

In computer graphics, the midpoint circle algorithm is an algorithm used to determine the points needed for drawing a circle. The algorithm is a variant of Bresenham's line algorithm, and is thus sometimes known as Bresenham's circle algorithm, although not actually invented by Jack E. Bresenham. The Midpoint circle drawing algorithm works on the same midpoint concept as the Bresenham's line algorithm. In the Midpoint circle drawing algorithm, you determine the next pixel to be plotted based on the position of the midpoint between the current and next consecutive pixel.

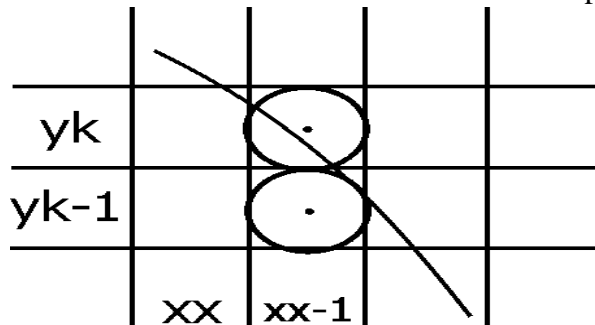


Fig 6.1 Midpoint circle Algorithm

Platform Required:

- Microsoft Windows 7/ Windows 8 Operating System onwards or 64-bit Open source Linux or its derivative.

Algorithm :

Bresenham's Algorithm to Draw a Circle.

7. Start.
8. Declare variables x,y,p and also declare gdriver=DETECT,gmode.
9. Initialise the graphic mode with the path location in TC folder.
10. Input the radius of the circle r.
11. Load x=0,y=r,initial decision parameter $p=1-r$.so the first point is (0,r).
12. Repeat Step 7 while $(x < y)$ and increment x-value simultaneously. 7. If $(p > 0)$,
do $p = p + 2*(x - y) + 1$.
13. Otherwise $p = p + 2*x + 1$
and
y is decremented simultaneously.

14. Then calculate the value of the function circlepoints() with parameters (x,y).
15. Place pixels using putpixel at points (x+300,y+300) in specified colour in circlepoints() function shifting the origin to 300 on both x-axis and y-axis.
16. Close Graph.
17. Stop.

CONCLUSION: Thus we have seen the two line drawing algorithms, the two Circle drawing algorithm and successfully implemented the given pattern.

Questions:

- 11) What does DDA stand for?
- 12) What is faster DDA or bresenham's? why?
- 13) What occupies more memory – DDA or bresenham's? why?
- 14) Which gives better output - DDA or bresenham's? why?
- 15) What is abs()?
- 16) What is significance of sign() function in DDA?
- 17) What is the initial value of error variable for slope > 1?
- 18) What is the initial value of error variable for slope <= 1?
- 19) Which algorithm gives better output? Bresenham's or DDA?
- 20) Where is abs defined?

Lab Assignment No.	11 A
Title	Write C++ program to draw 2-D object and perform following basic transformations: 1. Scaling 2. Translation 3. Rotation. Apply the concept of operator overloading. OR Write C++ program to implement translation, rotation and scaling transformations on equilateral triangle and rhombus. Apply the concept of operator overloading
Roll No.	
Class	SE
Date of Completion	
Subject	OOP & CG Lab
Assessment Marks	
Assessor's Sign	

LAB ASSIGNMENT 11-A

TITLE: Write C++ program to draw 2-D object and perform following basic transformations: 1. Scaling 2. Translation 3. Rotation. Apply the concept of operator overloading.

PROBLEM STATEMENT: To draw 2-D object and perform basic transformations

OBJECTIVES: Identify the 2-D object transformations of computer graphics

OUTCOMES: Implement computer graphics programs in C++ using the basic 2-D object transformations.

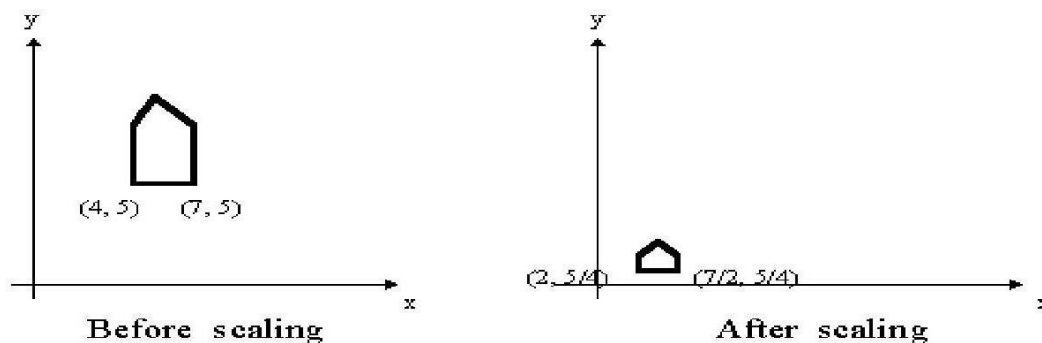
THEORY CONCEPTS/ LOGIC/ ALGORITHM:

We have to perform 2D transformations on 2D objects. Here we perform transformations on a line segment.

The 2D transformations are:

1. Scaling
2. Translation
3. Rotation

1. Scaling: scaling refers to changing the size of the object either by increasing or decreasing. We will increase or decrease the size of the object based on scaling factors along x and y-axis.



If (x, y) are old coordinates of object, then new coordinates of object after applying scaling transformation are obtained as:

$$x' = x * s_x$$

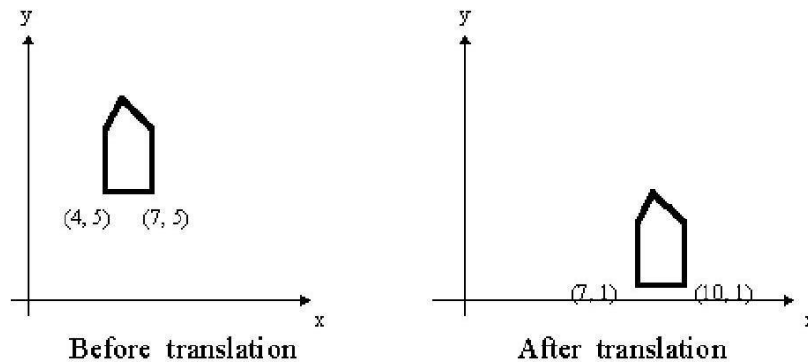
$$y' = y * s_y$$

s_x and s_y are scaling factors along x-axis and y-axis. we express the above equations in matrix form as:

[illegible]

Scaling Matrix

2. Translation: Translation is defined as moving the object from one position to another position along straight line path.



We can move the objects based on translation distances along x and y axis. t_x denotes translation distance along x-axis and t_y denotes translation distance along y axis.

Translation Distance: It is nothing but by how much units we should shift the object from one location to another along x, y-axis.

Consider (x,y) are old coordinates of a point. Then the new coordinates of that same point (x',y') can be obtained as follows:

$$X' = x + tx$$

$$Y' = y + ty$$

We denote translation transformation as P . we express above equations in matrix form as:

$$\begin{bmatrix} \mathbf{x}' \\ \mathbf{y}' \end{bmatrix} = \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

x, y ---old coordinates x', y' ---

new coordinates after

translation

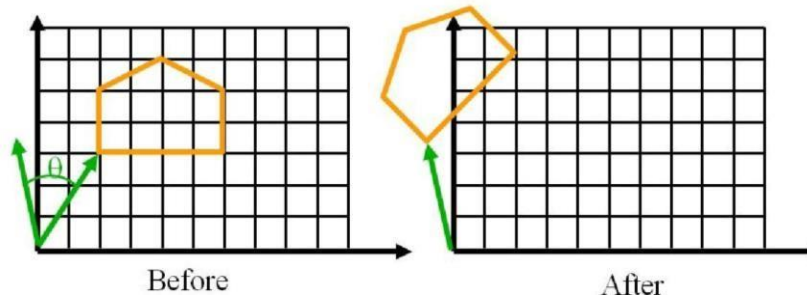
t_x, t_y —translation distances, T is

3. **Rotation:** A rotation repositions all points in an object along a circular path in the plane centered at the pivot point. We rotate an object by an angle θ .
New coordinates after rotation depend on *both* x and y

$$x' = x \cos \theta - y \sin \theta$$

$$y' = x \sin \theta + y \cos \theta$$

or in matrix form:



$$P' = R \cdot P,$$

$$R = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix}$$

R-rotation matrix.

$$\begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix}$$

Algorithm:

1. Start
2. Initialize the graphics mode.
3. Construct a 2D object (use Drawpoly()) e.g. (x,y)
4. A) Translation
 - a. Get the translation value t_x, t_y
 - b. Move the 2d object with t_x, t_y ($x' = x + t_x, y' = y + t_y$)
 - c. Plot (x', y')
5. B) Scaling
 - d. Get the scaling value S_x, S_y
 - e. Resize the object with S_x, S_y ($x' = x * S_x, y' = y * S_y$)

f. Plot (x',y')

6. C) Rotation

g. Get the Rotation angle

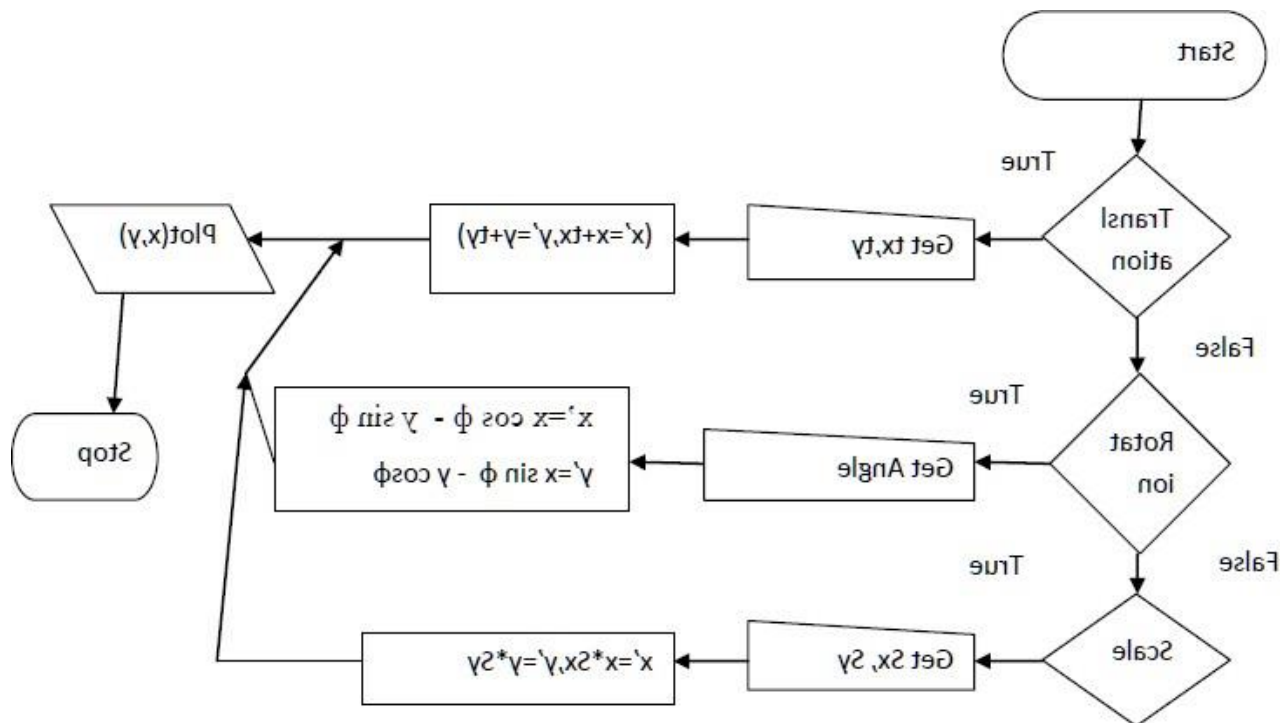
h. Rotate the object by the angle ϕ

$$x' = x \cos \phi - y \sin \phi$$

$$y' = x \sin \phi + y \cos \phi$$

i. Plot (x',y')

Flow chart for 2D Transformations



CONCLUSION: We have performed scaling, translation and rotation onto the polygon.

Oral Questions:

1. What is scaling?
2. What is Rotation?
3. What is translation?
4. What are the steps required for rotation about arbitrary point?
5. What is the meaning of x-shear and y-shear?
6. What is homogenous coordinate system?

7. What is the reflection?
8. How the reflection calculated at origin?
9. How we perform matrix multiplication?
10. What is difference between scaling and translation?

Lab Assignment No.	11 B
Title	Write C++ program to implement translation, rotation and scaling transformations on equilateral triangle and rhombus. Apply the concept of operator overloading
Roll No.	
Class	SE
Date of Completion	
Subject	OOP & CG Lab
Assessment Marks	
Assessor's Sign	

LAB ASSIGNMENT 11-B

TITLE: Write C++ program to implement translation, rotation and scaling transformations on equilateral triangle and rhombus. Apply the concept of operator overloading

PROBLEM STATEMENT: To implement following 2D transformation

- Simple Translation.
- Scaling and Rotation about origin
- Scaling and Rotation about arbitrary point

OBJECTIVES: Understand and Implement basic 2D transformations in Laboratory

OUTCOMES: Implement basic 2D transformations on equilateral triangle and rhombus.

THEORY CONCEPTS/ LOGIC/ ALGORITHM:**2D transformation:**

Transformation allows us to uniformly alter the entire picture instead of drawing a new picture.

Transformation means translation, scaling, rotation, shearing or combination of all.

Translation:

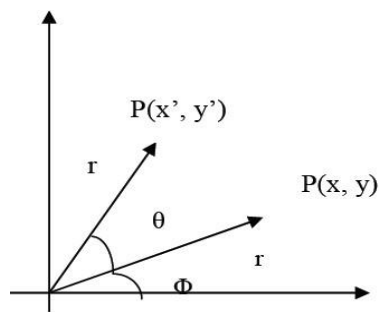
Translation is applied to an object by repositioning it along a straight line path from one coordinate location to another. This can be easily done by adding to each point the amount by which we want to shift the point

Translation Matrix:

$$T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ tx & ty & 1 \end{bmatrix}$$

Rotation:

The 2D rotation is applied to object by repositioning it along circular path in xy. To generate rotation we specify a rotation angle and position (xr , yr) of rotation point .



$$x' = x \cos \theta - y \sin \theta$$

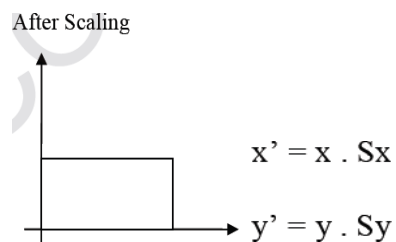
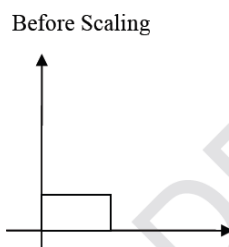
$$y' = x \sin \theta + y \cos \theta$$

Rotation Matrix :

$$R = \begin{bmatrix} \cos \theta & \sin \theta & 0 \\ -\sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Scaling:

Scaling transformation alters the size of an object. This operation can be carried out for polygon by multiplying coordinates valued (x, y) of each vertex by scaling factors S_x and S_y to produce the transformed coordinates

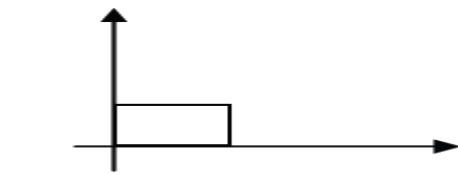
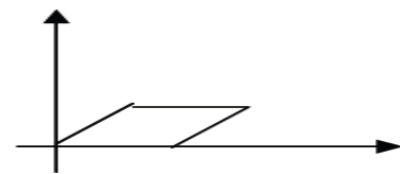


Scaling Matrix :

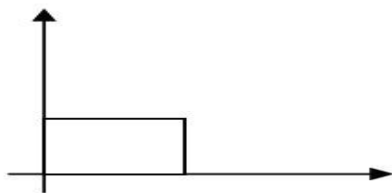
$$S = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Shearing:

A transformation that slants the shape of an object is called the shear transformation.

X Shear:**Original Object****Object after x shear**

$$X_sh = \begin{bmatrix} 1 & 0 & 0 \\ Shx & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Y Shear:**Original Object****Object after y shear**

$$Y_sh = \begin{bmatrix} 1 & Shy & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

CONCLUSION: Implement translation, sheer, rotation and scaling transformations on equilateral triangle and rhombus

Questions:

1. What is scaling? What is Rotation?
2. How to take the input as rhombus?
3. What is the logic to rotate the triangle by 90 Degree?
4. What is translation?
5. What are the steps required for rotation about arbitrary point by 90?
6. What is the meaning of x-shear and y-shear?

7. What is homogenous coordinate system?
8. What is the reflection? what are the steps for reflection of point along arbitrary axis?
9. How is the reflection calculated at origin?
10. How do we perform matrix multiplication?

Lab Assignment No.	12 A
Title	Write C++ program to generate snowflake using concept of fractals.
Roll No.	
Class	SE
Date of Completion	
Subject	OOP & CG Lab
Assessment Marks	
Assessor's Sign	

LAB ASSIGNMENT 12A

TITLE: Write C++ program to generate snowflake using concept of fractals.

PROBLEM STATEMENT: To generate snowflake using concept of fractals.

THEORY:

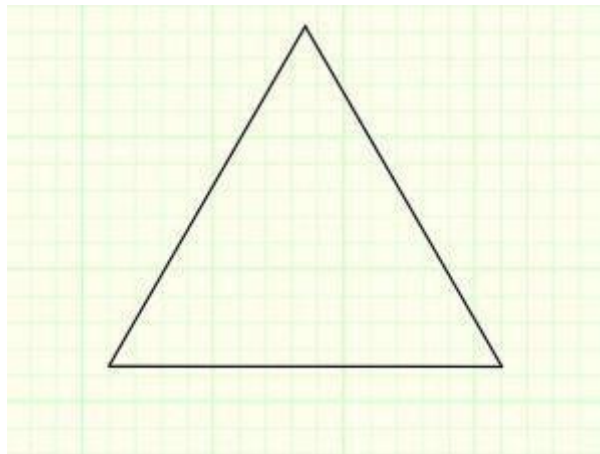
The Koch snowflake (also known as the Koch curve, Koch star, or Koch island) is a mathematical curve and one of the earliest fractal curves to have been described. It is based on the Koch curve, which appeared in a 1904 paper titled “On a continuous curve without tangents, constructible from elementary geometry” by the Swedish mathematician Helge von Koch. The progression for the area of the snowflake converges to $\frac{8}{5}$ times the area of the original triangle, while the progression for the snowflake’s perimeter diverges to infinity. Consequently, the snowflake has a finite area bounded by an infinitely long line.

Construction

Step1:

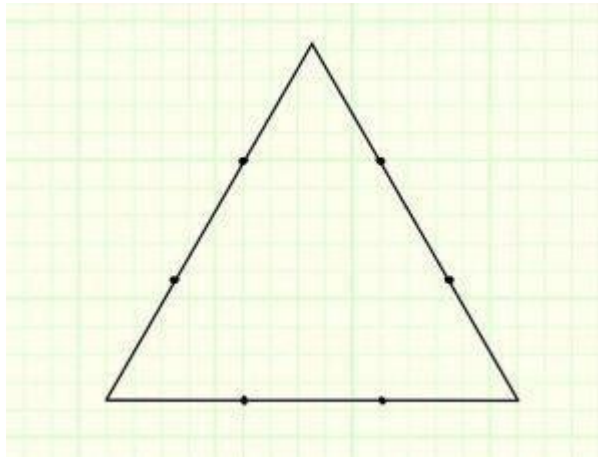
Draw an equilateral triangle. You can draw it with a compass or protractor, or just eyeball it if you don’t want to spend too much time drawing the snowflake.

Its best if the length of the sides is divisible by 3, because of the nature of this fractal. This will become clear in the next few steps.

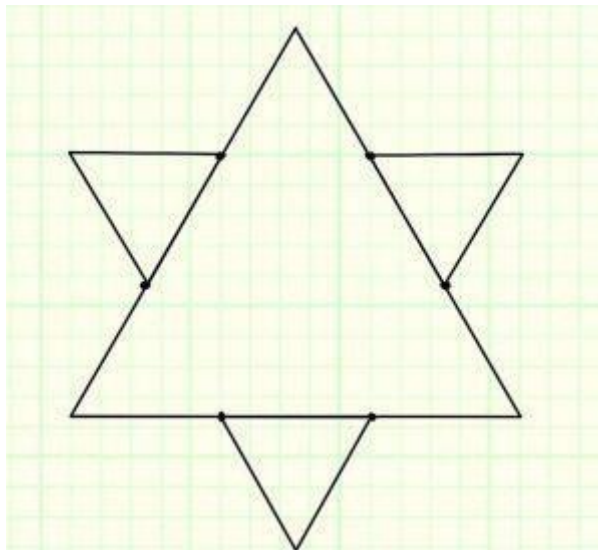


Step2:

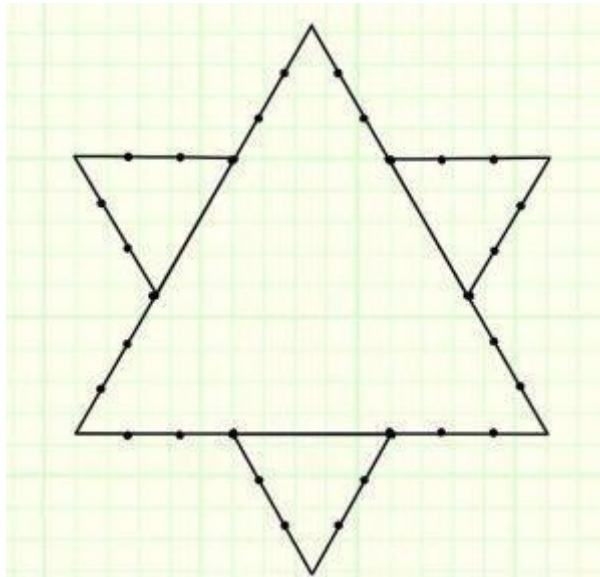
Divide each side in three equal parts. This is why it is handy to have the sides divisible by three.

**Step3:**

Draw an equilateral triangle on each middle part. Measure the length of the middle third to know the length

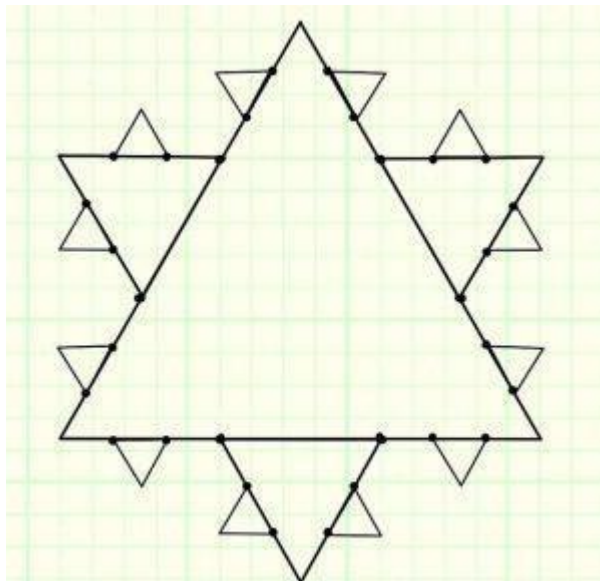
**Step4:**

Divide each outer side into thirds. You can see the 2nd generation of triangles covers a bit of the first. These three line segments shouldn't be parted in three.

**Step5:**

Draw an equilateral triangle on each middle part.

Note how you draw each next generation of parts that are one 3rd of the mast one.



CONCLUSION: We study concept of Fractals

Lab Assignment No.	12B
Title	Write C++ program to generate Hilbert curve using concept of fractals.
Roll No.	
Class	SE
Date of Completion	
Subject	OOP & CG Lab
Assessment Marks	
Assessor's Sign	

LAB ASSIGNMENT 12 B

TITLE: Write C++ program to generate Hilbert curve using concept of fractals.

PROBLEM STATEMENT: Program to generate Hilbert curve using concept of fractals.

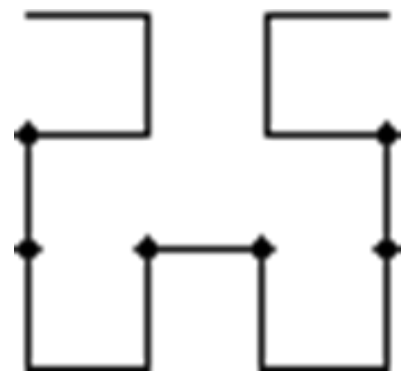
THEORY:

The Hilbert curve:

The Hilbert curve is a space filling curve that visits every point in a square grid with a size of 2×2 , 4×4 , 8×8 , 16×16 , or any other power of 2. It was first described by David Hilbert in 1892. Applications of the Hilbert curve are in image processing: especially image compression and dithering. It has advantages in those operations where the coherence between neighbouring pixels is important. The Hilbert curve is also a special version of a quadtree; any image processing function that benefits from the use of quadtrees may also use a Hilbert curve.

Cups and joins:

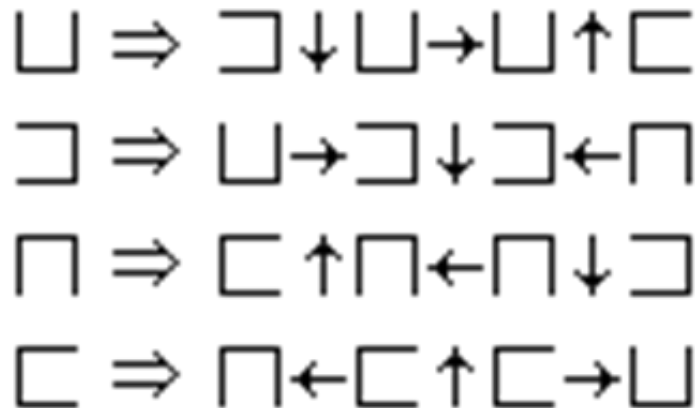
The basic elements of the Hilbert curves are what I call "cups" (a square with one open side) and "joins" (a vector that joins two cups). The "open" side of a cup can be top, bottom, left or right. In addition, every cup has two end-points, and each of these can be the "entry" point or the "exit" point. So, there are eight possible varieties of cups. In practice, a Hilbert curve uses only four types of cups. In a similar vein, a join has a direction: up, down, left or right.



A first order Hilbert curve is just a single cup (see the figure on the left). It fills a 2×2 space. The second order Hilbert curve replaces that cup by four (smaller) cups, which are linked together by three joins (see the figure on the right; the link between a cup and a join has been marked with a fat

dot in the figure). Every next order repeats the process of replacing each cup by four smaller cups and 3 joins

Cup subdivision rules



The function presented below (in the "C" language) computes the Hilbert curve. Note that the curve is symmetrical around the vertical axis. It would therefore be sufficient to draw half of the Hilbert curve.

CONCLUSION: Thus We have studied how to generate Hilbert curve using concept of fractals.

QUESTIONS:

1. What is the importance of curves and fractals in computer graphics?
2. What are applications of curves and fractals?

Lab Assignment No.	12 C
Title	Write C++ program to generate fractal patterns by using Koch curves.
Roll No.	
Class	SE
Date of Completion	
Subject	OOP & CG Lab
Assessment Marks	
Assessor's Sign	

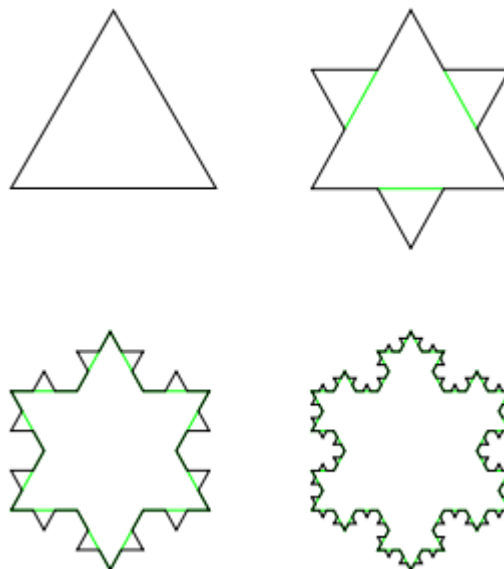
LAB ASSIGNMENT 12 C

TITLE: Write C++ program to generate fractal patterns by using Koch curves.

PROBLEM STATEMENT:9**THEORY:**

The Koch curve (also known as the Koch star, or Koch island) is a mathematical curve and one of the earliest fractal curves to have been described. It is based on the Koch curve, which appeared in a 1904 paper titled "On a continuous curve without tangents, constructible from elementary geometry".

The progression for the area of the snowflake converges to $8/5$ times the area of the original triangle, while the progression for the snowflake's perimeter diverges to infinity. Consequently, the snowflake has a finite area bounded by an infinitely long line.



The Koch snowflake can be constructed by starting with an equilateral triangle, then recursively altering each line segment as follows:

1. divide the line segment into three segments of equal length.
2. draw an equilateral triangle that has the middle segment from step 1 as its base and points outward.
3. remove the line segment that is the base of the triangle from step 2.

After one iteration of this process, the resulting shape is the outline of a hexagram. The Koch snowflake is the limit approached as the above steps are followed over and over again. The Koch curve originally described by Helge von Koch is constructed with only one of the three sides of the original triangle. In other words, three Koch curves make a Koch snowflake.

CONCLUSION: We implemented fractal patterns by using Koch curves.

Questions:

1. What is the meaning of fractal?
2. What are the different types of fractals?
3. What is the logic to generate Koch curve?
4. What is cups and joins?
5. List out the examples of fractals.
6. Why fractals are very important in computer graphics?
7. What is the difference between fractals and curves?
8. What are the different types of curves?
9. What is Bezier curve?
10. What are the different applications of fractals?

Lab Assignment No.	13 B
Title	Write C++ program to draw 3-D cube and perform following transformations on it using OpenGL i) Scaling ii) Translation iii) Rotation about an axis (X/Y/Z).
Roll No.	
Class	SE
Date of Completion	
Subject	OOP & CG Lab
Assessment Marks	
Assessor's Sign	

LAB ASSIGNMENT 13 B

TITLE: Write C++ program to draw 3-D cube and perform following transformations on it using OpenGL i) Scaling ii) Translation iii) Rotation about an axis (X/Y/Z).

PROBLEM STATEMENT: To perform 3 D transformations using OpenGL.

OBJECTIVES: Identify different 3D transformations of computer graphics

OUTCOMES: Implement computer graphics programs with 3 D transformations using OpenGL

THEORY:

OpenGL:

OpenGL is a software interface that allows the programmer to create 2D and 3D graphics images. OpenGL is both a standard API and the implementation of that API. You can call the functions that comprise OpenGL from a program you write and expect to see the same results no matter where your program is running. OpenGL is independent of the hardware, operating, and windowing systems in use. The fact that it is windowing-system independent, makes it portable. OpenGL program must interface with the windowing system of the platform where the graphics are to be displayed. Therefore, a number of windowing toolkits have been developed for use with OpenGL. OpenGL functions in a client/server environment. That is, the application program producing the graphics may run on a machine other than the one on which the graphics are displayed. The server part of OpenGL, which runs on the workstation where the graphics are displayed, can access whatever physical graphics device or frame buffer is available on that machine.

In OpenGL translation, rotation, and scaling are performed using commands such as:

`glTranslate{fd}(X,Y,Z) – glTranslatef(1.0, 2.5, 3.0)`

`glRotate{df}(Angle, X, Y, Z) - glRotatef(60.0, 0.0, 0.0, 1.0)`

`glScale{df}(X, Y, Z) - glScalef(1.0, 1.5, 2.0)`

3 D Transformations:

A points in 3-Dimensional space $[x \ y \ z]$ is represented by a four dimensional position vector.

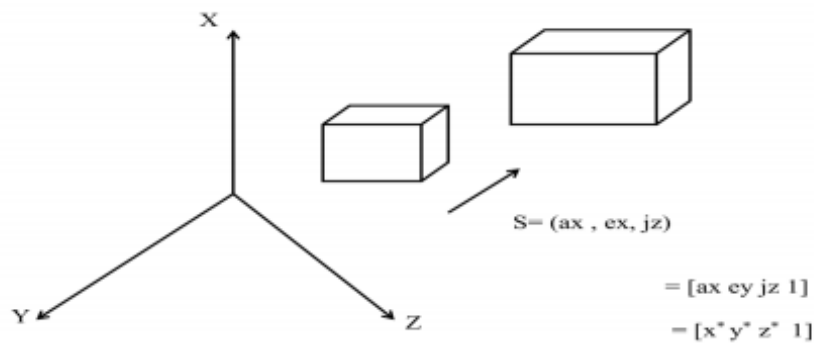
$$[x' \ y' \ z' \ h] = [x \ y \ z \ 1] [T]$$

The generalized 4*4 transformation matrix is,

$$[T] = \begin{bmatrix} a & b & c & p \\ d & e & f & q \\ g & i & j & r \\ l & m & n & s \end{bmatrix}$$

3D Scaling:- Diagonal terms produce local and overall scaling.

$$[x] \ [y] = [x \ y \ z \ 1] \begin{bmatrix} a & 0 & 0 & 0 \\ 0 & e & 0 & 0 \\ 0 & 0 & j & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1/2 \end{bmatrix}$$

Overall / Uniform Scaling

3 Dimensional shearing:- [off diagonal terms produce shearing in 3D]

$$[x] [y] = [x \ y \ z \ 1] \begin{bmatrix} 1 & b & c & 0 \\ d & 1 & f & 0 \\ g & i & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

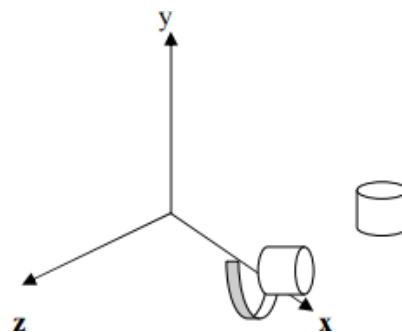
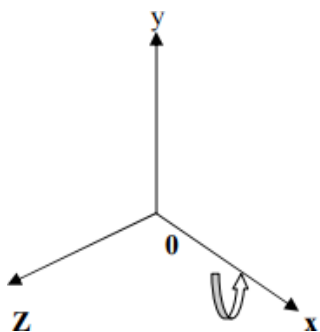
$$= [x+yd+gz \quad bx+y+iz \quad cx+fy+z \quad 1]$$

3 Dimensional rotation:-

For rotation about the x-axis, the x coordinates of the position vectors do not change the rotation occurs in planes perpendicular to the x-axis.

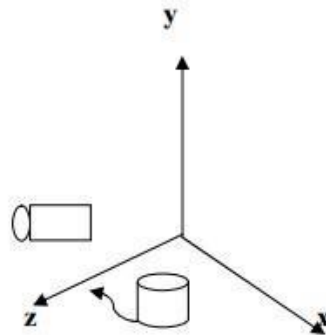
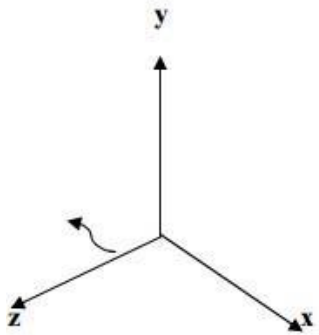
Transformation matrix for rotation about x axis by an angle is

$$[T] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \theta & \sin \theta & 0 \\ 0 & -\sin \theta & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

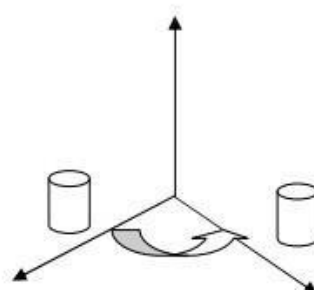
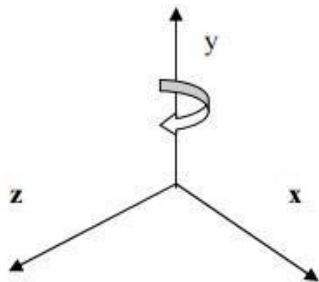


Rotation about z-axis:-

$$R_z = \begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Rotation about y-axis:-



$$R_y = \begin{bmatrix} \cos\theta & 0 & -\sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3 D Reflection

Reflection through the XY plane is

$$[T] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Reflection through YZ plane

$$[T] = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Reflection through the XZ plane

$$[T] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3D Translation:-

$$[T] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ h & m & n & 1 \end{bmatrix}$$

Translated Homogenous are

$$[x' \ y' \ z' \ h] = [x \ y \ z \ 1] \quad = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ h & m & n & 1 \end{bmatrix}$$

$$[x' \ y' \ z' \ h] = [(x+h) \ (y+m) \ (z+n) \ 1]$$

CONCLUSION: We have performed 3 D transformations using OpenGL.

