GROUP - B ASSIGNMENT -10 (CG)

Name: Adityas Onkar Patil
Batch: G4 (SE4)
Roll No: 21449
Performance Date:
Submission Date:

TITLE: Fractals and Curves

PROBLEM STATEMENT: write C++ program to generate snowflake using concept of fractals.

LEARNING OBJECTIVES: To learn the concept of fractals and curves as a part of computer graphics.

LEARNING OUTCOMES: After completion of this assignment, students will be able to:

1. Implement fractals and curves to form figures like snow flakes, Hilbert curves etc.

S/W AND H/W REQUIREMENTS: 64-bit open source Linux or its derivative, open source C++ programming tools like GCC/G++, Qt Creator, OpenGL.

REFERENCES: 1. Programming Principles and practice using C++, Bjarne Stroustrup.
2. www.qt.io.

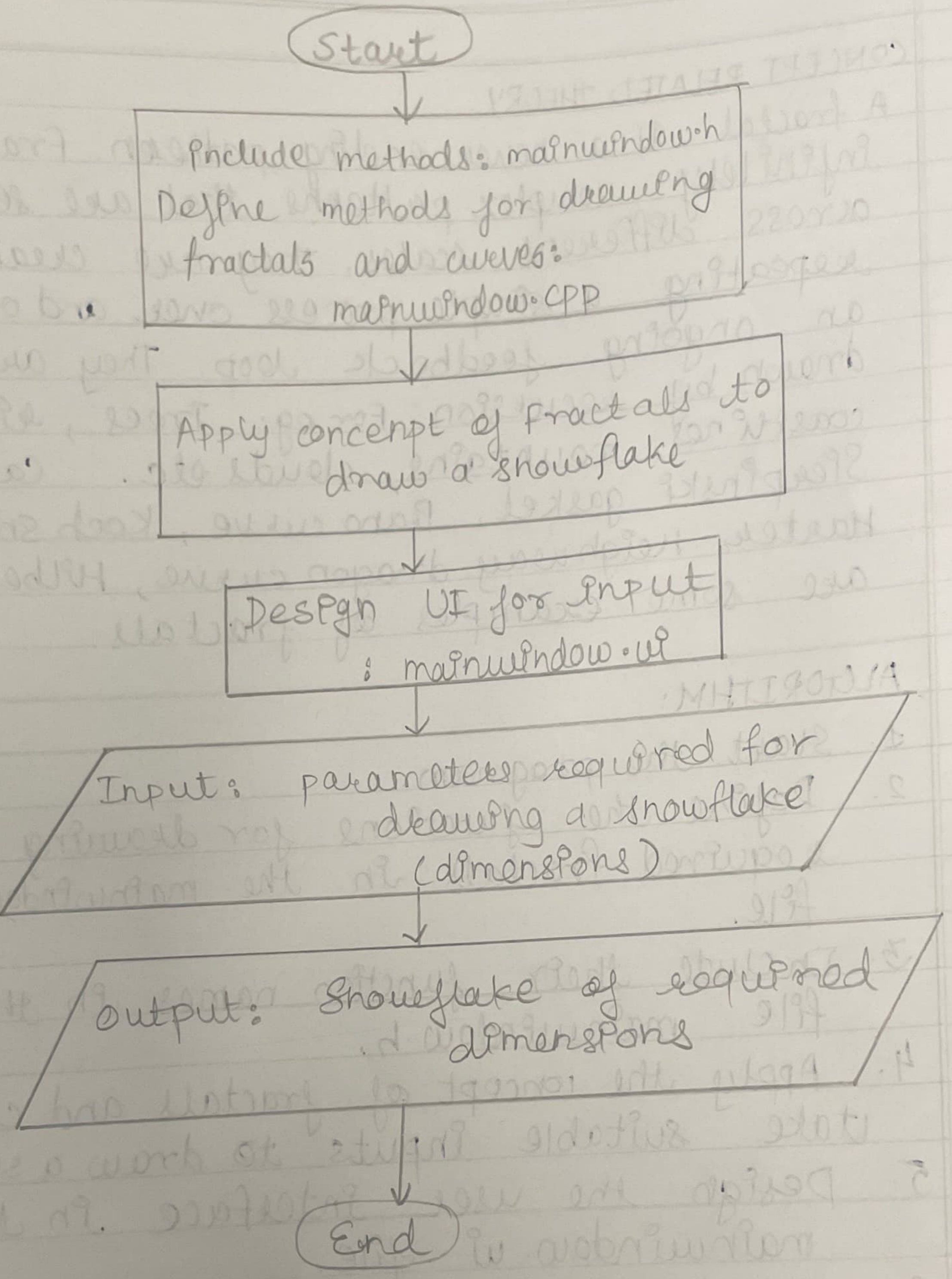
CONCEPT RELATED THEORY:

A fractal is a never ending pattern. Fractals are infinitely complex patterns that are self similar across different scales. They are created by repeating a simple process over and over in an ongoing feedback loop. They are usually drawn by recursion. For eg:- Trees, rivers, coastlines, mountains, clouds etc. Cantor set, Sierpinski gasket, Peano curve, Koch snowflake, Harter Heighway dragon curve, Hilbert curve are some examples of fractals.

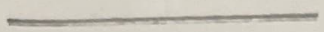
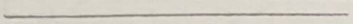
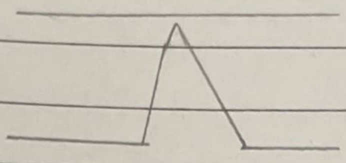
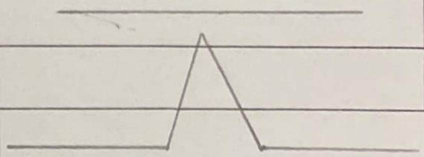
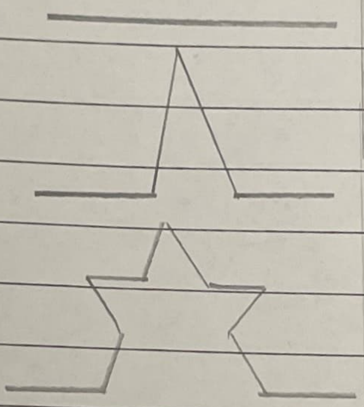
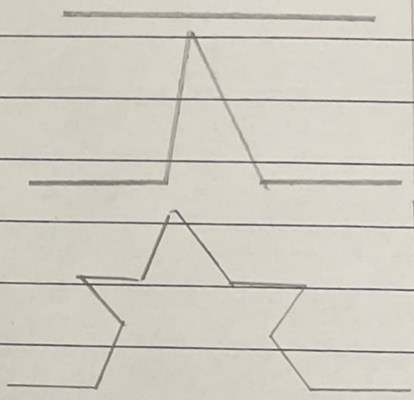
ALGORITHM:

1. Start the program.
2. Define the functions for drawing the required fractals in the mainwindow.cpp file.
3. Include their function names in the header file mainwindow.h.
4. Apply the concept of fractals and curves to take suitable inputs to draw a snowflake.
5. Design the user interface in the mainwindow.ui file.
6. End the program.

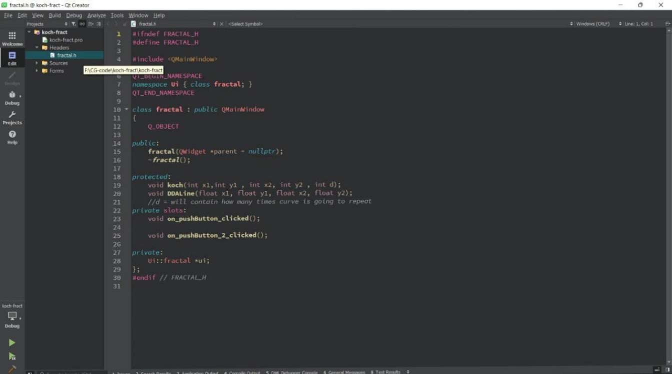
FLOWCHART:


Start
include methods: mainwindow.h
Define methods for drawing fractals and curves: mainwindow.cpp
Apply concenpt of fractals to draw a snowflake
Design UI for input : mainwindow.ui
Input: parameters required for drawing a snowflake (dimensions)
output: Snowflake of required dimensions
End

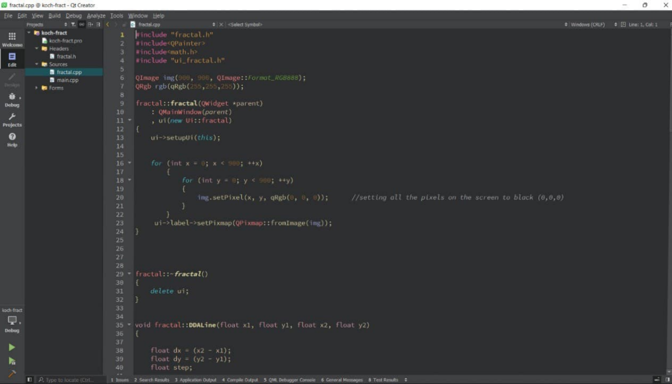
TEST CASES:

Sr.No.	Description	Expected Output	Actual Output	Status
1.	Order: 0			Pass
2.	Order: 1			Pass
3.	Order: 2.			Pass.

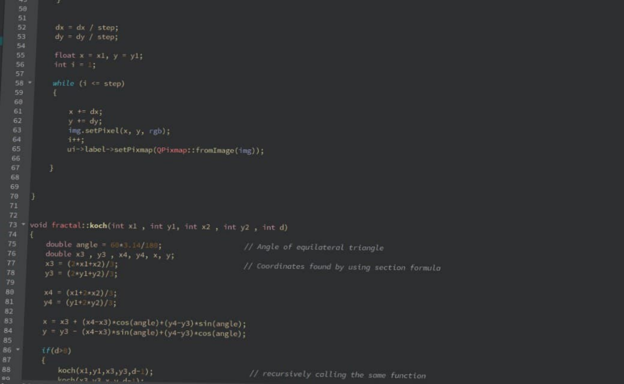
CONCLUSION:
The knowledge and implementation of koch curves using the concept of fractals was successfully implemented.



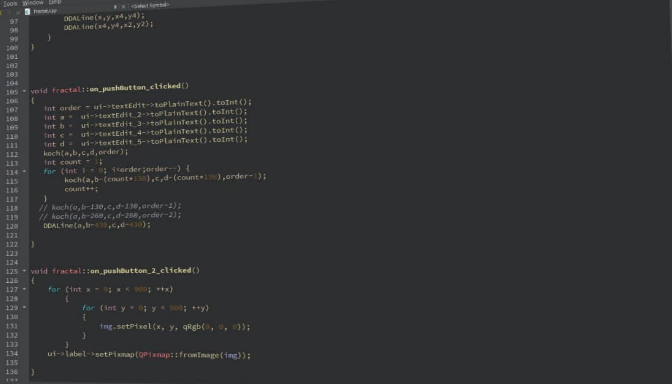
```
#ifndef FRACTAL_H
#define FRACTAL_H

#include <QMainWindow>

QT_BEGIN_NAMESPACE
namespace Ui { class fractal; }
QT_END_NAMESPACE

class fractal : public QMainWindow
{
    Q_OBJECT

public:
    fractal(QWidget *parent = nullptr);
    ~fractal();

protected:
    void koch(int x1,int y1 , int x2, int y2 , int d);
    void DDALine(float x1, float y1, float x2, float y2);
    //d = will contain how many times curve is going to repeat
private slots:
    void on_pushButton_clicked();

    void on_pushButton_2_clicked();

private:
    Ui::fractal *ui;
};
#endif // FRACTAL_H
```



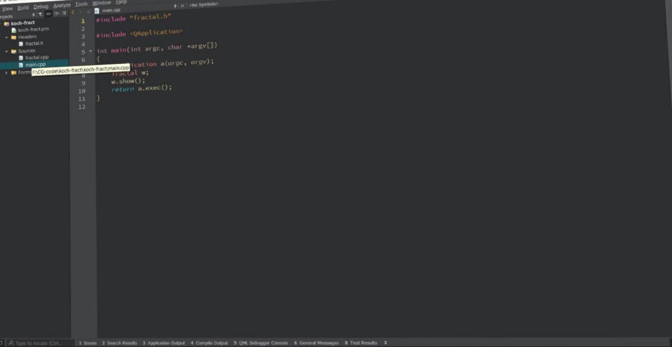
```cpp
#include "fractal.h"
#include<QPainter>
#include<math.h>
#include "ui_fractal.h"

QImage img(900, 900, QImage::Format_RGB888);
QRgb rgb(qRgb(255,255,255));

fractal::fractal(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::fractal)
{
    ui->setupUi(this);


    for (int x = 0; x < 900; ++x)
        {
            for (int y = 0; y < 900; ++y)
            {
                img.setPixel(x, y, qRgb(0, 0, 0));      //setting all the pixels on the screen to black (0,0,0)
            }
        }
     ui->label->setPixmap(QPixmap::fromImage(img));
}




fractal::~fractal()
{
    delete ui;
}


void fractal::DDALine(float x1, float y1, float x2, float y2)
{

    float dx = (x2 - x1);
    float dy = (y2 - y1);
    float step;
```



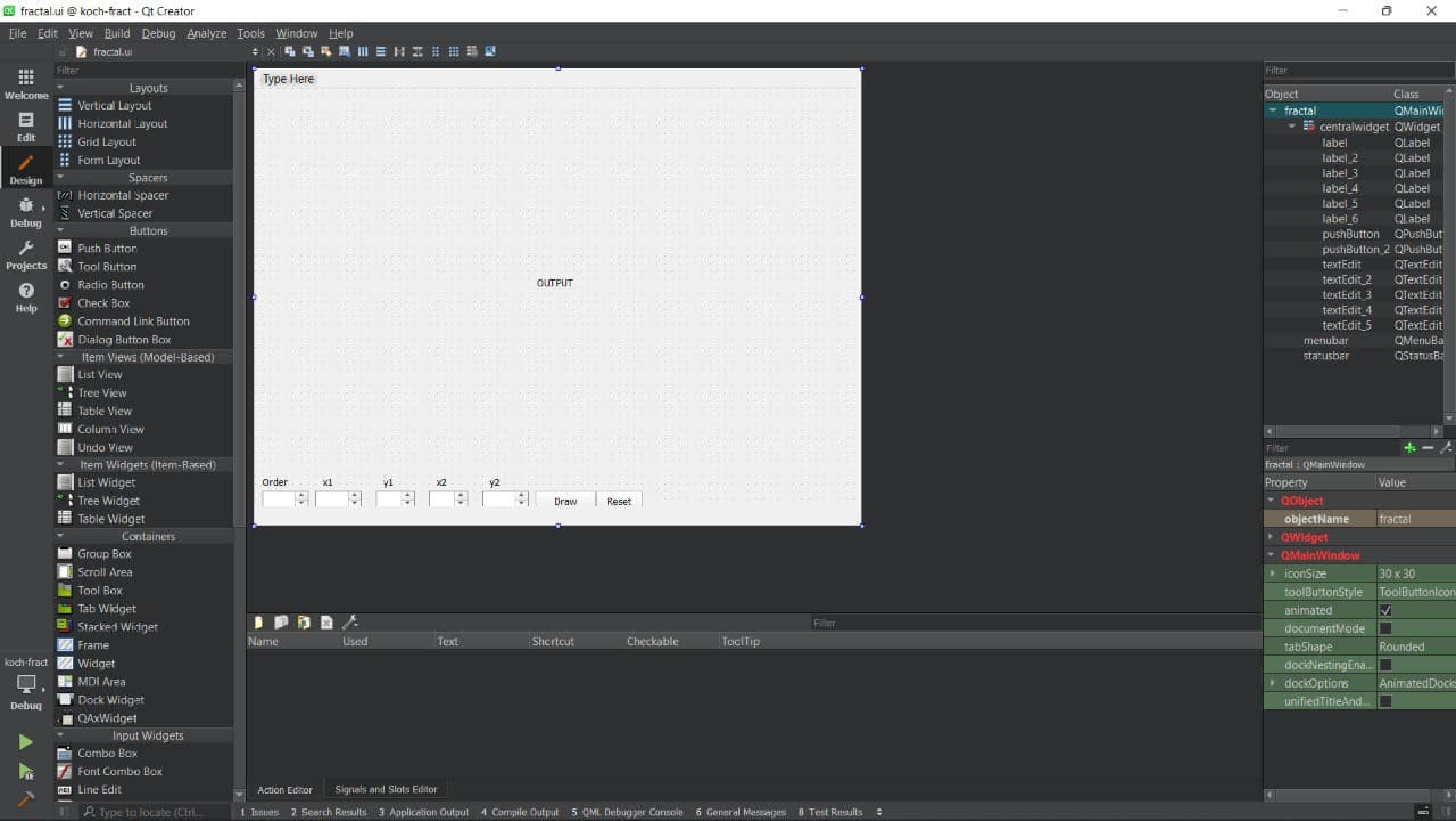
```
    }


    dx = dx / step;
    dy = dy / step;

    float x = x1, y = y1;
    int i = 1;

    while (i <= step)
    {

        x += dx;
        y += dy;
        img.setPixel(x, y, rgb);
        i++;
        ui->label->setPixmap(QPixmap::fromImage(img));

    }


}


void fractal::koch(int x1 , int y1, int x2 , int y2 , int d)
{
    double angle = 60*3.14/180;                     // Angle of equilateral triangle
    double x3 , y3 , x4, y4, x, y;
    x3 = (2*x1+x2)/3;                               // Coordinates found by using section formula
    y3 = (2*y1+y2)/3;

    x4 = (x1+2*x2)/3;
    y4 = (y1+2*y2)/3;

    x = x3 + (x4-x3)*cos(angle)+(y4-y3)*sin(angle);
    y = y3 - (x4-x3)*sin(angle)+(y4-y3)*cos(angle);

    if(d>0)
    {
        koch(x1,y1,x3,y3,d-1);                      // recursively calling the same function
        koch(x3,y3,x,y,d-1);
```



```
        DDALine(x,y,x4,y4);
        DDALine(x4,y4,x2,y2);
    }
}



void fractal::on_pushButton_clicked()
{
    int order = ui->textEdit->toPlainText().toInt();
    int a =  ui->textEdit_2->toPlainText().toInt();
    int b =  ui->textEdit_3->toPlainText().toInt();
    int c =  ui->textEdit_4->toPlainText().toInt();
    int d =  ui->textEdit_5->toPlainText().toInt();
    koch(a,b,c,d,order);
    int count = 1;
    for (int i = 0; i<order;order--) {
        koch(a,b-(count*130),c,d-(count*130),order-1);
        count++;
    }
  // koch(a,b-130,c,d-130,order-1);
  // koch(a,b-260,c,d-260,order-2);
    DDALine(a,b-430,c,d-430);

}


void fractal::on_pushButton_2_clicked()
{
    for (int x = 0; x < 900; ++x)
        {
            for (int y = 0; y < 900; ++y)
            {
                img.setPixel(x, y, qRgb(0, 0, 0));
            }
        }
    ui->label->setPixmap(QPixmap::fromImage(img));

}
```



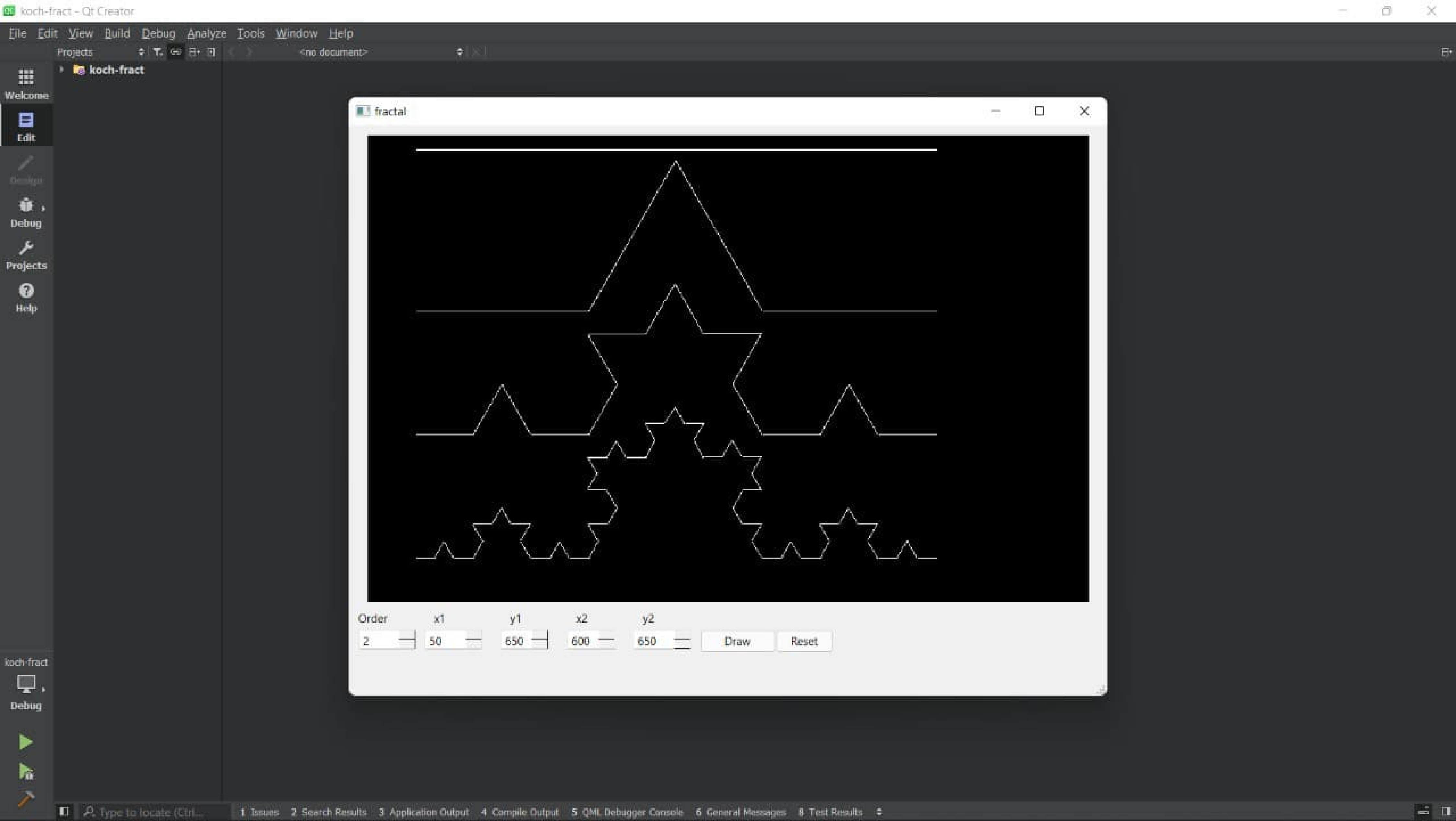
```cpp
#include "fractal.h"

#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    Fractal w;
    w.show();
    return a.exec();
}

```



Object	Class
fractal	QMainWi
centralwidget	QWidget
label	QLabel
label_2	QLabel
label_3	QLabel
label_4	QLabel
label_5	QLabel
label_6	QLabel
pushButton	QPushBut
pushButton_2	QPushBut
textEdit	QTextEdit
textEdit_2	QTextEdit
textEdit_3	QTextEdit
textEdit_4	QTextEdit
textEdit_5	QTextEdit
menubar	QMenuBa
statusbar	QStatusBa

fractal : QMainWindow

Property	Value
QObject	
objectName	fractal
QWidget	
QMainWindow	
iconSize	30 x 30
toolButtonStyle	ToolButtonIcon
animated	☑
documentMode	□
tabShape	Rounded
dockNestingEna...	□
dockOptions	AnimatedDock
unifiedTitleAnd...	□



koch-fract - Qt Creator
File Edit View Build Debug Analyze Tools Window Help
Projects
<no document>
koch-fract
Welcome
Edit
Design
Debug
Projects
Help
fractal
Draw
Reset
koch-fract
Debug
Type to locate (Ctrl...
1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 6 General Messages 8 Test Results

Order 2
x1 50
y1 650
x2 600
y2 650