- **Source Code**

- ***MainWindow.h***

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>

QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE

class MainWindow : public QMainWindow
{
    Q_OBJECT

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

private slots:
    void on_pushButton_clicked();
    void DDALine(float x1,float y1,float x2,float y2, int flag);
    void mousePressEvent(QMouseEvent *event);
    void boundary(int flag);
    void clear_boundary();
    int compute_code(double x,double y);
    void cohenSutherlandClip(double x1,double y1,double x2,double y2);

private:
    Ui::MainWindow *ui;
    float VerticesX[10];
    float VerticesY[10];
    int numVertices = 0;

    const int INSIDE = 0;
    const int LEFT = 1;
    const int RIGHT = 2;
    const int BOTTOM = 4;
    const int TOP = 8;
```

```
    const int x_min = 100;
    const int y_min = 100;
    const int x_max = 200;
    const int y_max = 200;
};
#endif // MAINWINDOW_H
```

- ***MainWindow.cpp***

```
#include "mainwindow.h"
#include "./ui_mainwindow.h"
#include "QMouseEvent"

QImage img(300,300,QImage::Format_RGB888);

MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    boundary(0);
    ui->label->setPixmap(QPixmap::fromImage(img));
    ui->label_2->setPixmap(QPixmap::fromImage(img));

}

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


void MainWindow::on_pushButton_clicked()
{
    ui->label_2->setPixmap(QPixmap::fromImage(img));
    clear_boundary();
    boundary(1);
    for(int i = 1; i <= 10; i++)
    {
        if(i % 2 != 0)
        {
```

```
            cohenSutherlandClip(VerticesX[i],VerticesY[i],VerticesX[i
- 1],VerticesY[i - 1]);
        }
    }
}

void MainWindow::DDALine(float x1, float y1, float x2, float y2,int
flag)
{
    float xinc,yinc,dx,dy,length,x,y;
    dx = x2 - x1;
    dy = y2 - y1;
    if(abs(dx) > abs(dy))
    {
        length = abs(dx);
    }
    else
    {
        length = abs(dy);
    }
    xinc = dx/length;
    yinc = dy/length;
    int i = 1;
    x = x1;
    y = y1;
    while(i <= length)
    {
        img.setPixel(x,y,qRgb(0,255,0));
        x += xinc;
        y += yinc;
        i++;
    }
    if(flag==0)
        ui->label->setPixmap(QPixmap::fromImage(img));
    else{
        ui->label_2->setPixmap(QPixmap::fromImage(img));
    }
}

void MainWindow::mousePressEvent(QMouseEvent *event)
{
    bool start = true;
```

```
    if(start)
    {
        int x = event->pos().x();
        int y = event->pos().y();
        VerticesX[numVertices] = x;
        VerticesY[numVertices] = y;
        if(numVertices > 0 && numVertices %2 != 0)
        {
            DDALine(VerticesX[numVertices - 1],VerticesY[numVertices - 1],
            VerticesX[numVertices],VerticesY[numVertices],0);
        }
        if(event->button() == Qt::RightButton)
        {
            start = false;
        }
        numVertices++;
    }

}

void MainWindow::boundary(int flag)
{
    DDALine(x_min,y_min,x_min,y_max,flag);
    DDALine(x_min,y_min,x_max,y_min,flag);
    DDALine(x_min,y_max,x_max,y_max,flag);
    DDALine(x_max,y_min,x_max,y_max,flag);
    //    ui->label_2->setPixmap(QPixmap::fromImae(img));

}

void MainWindow::clear_boundary()
{
    for(int i = 0; i < 300; i++)
    {
        for(int j = 0; j < 300; j++)
        {
            img.setPixel(i,j,qRgb(0,0,0));
        }
    }
    //    ui->label_2->setPixmap(QPixmap::fromImage(img));
}
```

```
int MainWindow::compute_code(double x, double y)
{
    int code = INSIDE;
    if(x < x_min)
    {
        code |= LEFT;
    }
    else if(x > x_max)
    {
        code |= RIGHT;
    }
    if(y < y_min)
    {
        code |= BOTTOM;
    }
    else if(y > y_max)
    {
        code |= TOP;
    }
    return code;
}

void MainWindow::cohenSutherlandClip(double x1, double y1, double x2,
double y2)
{
    bool accept = false;
    int code1,code2;
    code1 = compute_code(x1,y1);
    code2 = compute_code(x2,y2);
    while (true)
    {
        if((code1 == 0) && (code2 == 0))
        {
            accept = true;
            break;
        }
        else if(code1 & code2)
        {
            break;
        }
        else
        {
```

```
int code_out;
double x,y;
if(code1 != 0)
{
    code_out = code1;
}
else
{
    code_out = code2;
}
if(code_out & TOP)
{
    x = x1 + (x2 - x1) * (y_max - (y1)) / (y2 - y1);
    y = y_max;
}
else if(code_out & BOTTOM)
{
    x = x1 + (x2 - x1) * (y_min - y1) / (y2 - y1);
    y = y_min;

}
else if(code_out & LEFT)
{
    x = x_min;
    y = y1 + (y2 - y1) * (x_min - x1) / (x2 - x1);
}
else if(code_out & RIGHT)
{
    x = x_max;
    y = y1 + (y2 - y1) * (x_max - x1) / (x2 - x1);

}
if(code_out == code1)
{
    x1 = x;
    y1 = y;
    code1 = compute_code(x1,y1);
}
else
{
    x2 = x;
    y2 = y;
```

```
                code2 = compute_code(x2,y2);

            }
        }

    }

    if(accept)
    {
        DDALine(x1,y1,x2,y2,1);
    }
}
```

- **Output**