Министерство образования и науки Российской Федерации

КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ

ИНСТИТУТ ФИЗИКИ

КАФЕДРА РАДИОФИЗИКИ

ОТЧЕТ

По лабораторной работе №1 по предмету

Цифровая обработка изображений

Студент 4 курса,

группы 06-110

А.Ф.Хаматзянова

Казань – 2014

Постановка задачи:

Напишите программу, реализующую алгоритм заполнения многоугольника со списком, упорядоченным по координате.

Код программы:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace helloworld

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

Random rnd = new Random();

Random rn = new Random();

int k;

int q;

Point[] myPointArray;

Point[] myPoint;

Stack <Point> myStack =new Stack <Point> ();

Point[] myPoints;

Stack <Point>myPointsCDA = new Stack <Point>();

Point[] Points;

public void swap(int x1, int x2)

{

int x = myPointArray[x1].X;

int y = myPointArray[x1].Y;

myPointArray[x1].X = myPointArray[x2].X;

myPointArray[x1].Y = myPointArray[x2].Y;

myPointArray[x2].X = x;

myPointArray[x2].Y = y;

}

public void cda(double x1, double y1, double x2, double y2)

{

double mX, mY, l;

double dx, dy,k, x, y;

myPoints = new Point[1];

mX = x2 - x1;

mY = y2 - y1;

if (Math.Abs(mX) > Math.Abs(mY))

{

l = Math.Abs(mX);

}

else

{

l = Math.Abs(mY);

}

dx =mX / l;

dy = mY / l;

x = x1;

y = y1;

for (k = 1; k <= l; k++)

{

// PutPixel(g, Color.Red, x, y, 255);

myPoints[0]= new Point ((int)x,(int)y);

myPointsCDA.Push(myPoints[0]);

x += dx;

y += dy;

Array.Clear(myPoints, 0, 1);

}

}

public void filling()

{

Graphics g = pictureBox1.CreateGraphics();

Pen myPen = new Pen(Color.Red);

int c = myPointsCDA.Count();

Points = new Point[c];

myPointsCDA.CopyTo(Points,0);

myPointsCDA.Clear();

Stack <Point> UsedY=new Stack <Point> () ;

List<Point> myList = new List<Point>();

int i = 0;

UsedY.Push(Points[i]);

while (i < c)

{

for (int j = i ; j < c; j++)

{

if ((Points[i].Y == Points[j].Y) && (!UsedY.Contains(Points[j])))

{

myPoints[0]= Points[j];

myPoints[0].Y=(int)(myPoints[0].Y+0.5);

myList.Add(myPoints[0]);

UsedY.Push(Points[j]);

}

}

myList.Sort(delegate(Point one, Point two)

{

return one.Y.CompareTo(two.Y);

});

if (myList.Count() % 2 == 0)

{

int z = 0;

int check = 0; ;

while (check < myList.Count())

{

g.DrawLine(myPen, myList.ElementAt(z), myList.ElementAt(z + 1));

z = z + 2;

check = check + 2;

}

}

else

{

int z = 0;

int check = 0;

while (check < myList.Count() -1)

{

g.DrawLine(myPen, myList.ElementAt(z), myList.ElementAt(z + 1));

z = z + 2;

check = check + 2;

}

}

myList.Clear();

i++;

}

UsedY.Clear();

Array.Clear(Points,0,c);

Array.Clear(myPoints,0,1);

}

public double rotate( int x1,int x2, int x3)

{

double a = (myPointArray[x2].X - myPointArray[x1].X) \* (myPointArray[x3].Y - myPointArray[x1].Y) - (myPointArray[x3].X - myPointArray[x1].X) \* (myPointArray[x2].Y - myPointArray[x1].Y);

return a;

}

public double rotateForStack(PointF x1, PointF x2, PointF x3)

{

double a = (x2.X - x1.X) \* (x3.Y - x1.Y) - (x3.X - x1.X) \* (x2.Y - x1.Y);

return a;

}

private void button1\_Click(object sender, EventArgs e)

{

k= rnd.Next(3,11);

myPointArray = new Point[k];

myPoint = new Point[k];

for (int i = 0; i <k; i++)

{

int x= rn.Next(400);

int y= rn.Next(200);

myPointArray[i] = new Point(x, y);

}

Graphics g = pictureBox1.CreateGraphics();

Pen myPen = new Pen(Color.Blue);

for (int i = 1; i < k ; i++)

{

if (myPointArray[i].X < myPointArray[0].X)

{

swap(0,i);

}

}

int j;

for (int i = 2; i < k ; i++)

{

j = i;

while ((j > 1) && (rotate(0,j-1,j) > 0))

{

swap(j-1,j);

j--;

}

}

myStack.Push(myPointArray[0]);

myStack.Push(myPointArray[1]);

j = 2;

for (int i = 2; i < k-1 ; i++)

{

while (rotateForStack(myStack.ElementAt(j - 2), myStack.ElementAt(j - 1), myPointArray[i]) < 0)

{

myStack.Pop();

myStack.Push(myPointArray[i]);

}

myStack.Push(myPointArray[i]);

j++;

}

myStack.Push(myPointArray[0]);

myStack.CopyTo(myPoint,0);

q = myStack.Count();

if (q >= 4)

{

g.DrawPolygon(myPen, myPoint);

}

myStack.Clear();

Array.Clear(myPointArray,0,k);

}

private void pictureBox1\_Click(object sender, EventArgs e)

{

}

private void Form1\_Load(object sender, EventArgs e)

{

}

private void button2\_Click(object sender, EventArgs e)

{

Graphics g = pictureBox1.CreateGraphics();

Pen myPen = new Pen(Color.Red);

for (int i = 0; i < k-1; i++)

{

if (myPoint[i].Y != myPoint[i+1].Y)

cda(myPoint[i].X, myPoint[i].Y, myPoint[i+1].X, myPoint[i+1].Y);

}

if (myPoint[0].Y != myPoint[k-1].Y)

{

cda(myPoint[0].X, myPoint[0].Y, myPoint[k-1].X, myPoint[k-1].Y);

}

filling();

Array.Clear(myPoint,0,k);

}

private void button3\_Click(object sender, EventArgs e)

{

Graphics g = pictureBox1.CreateGraphics();

g.Clear(Color.White);

}

}

}

Результат выполнения программы:

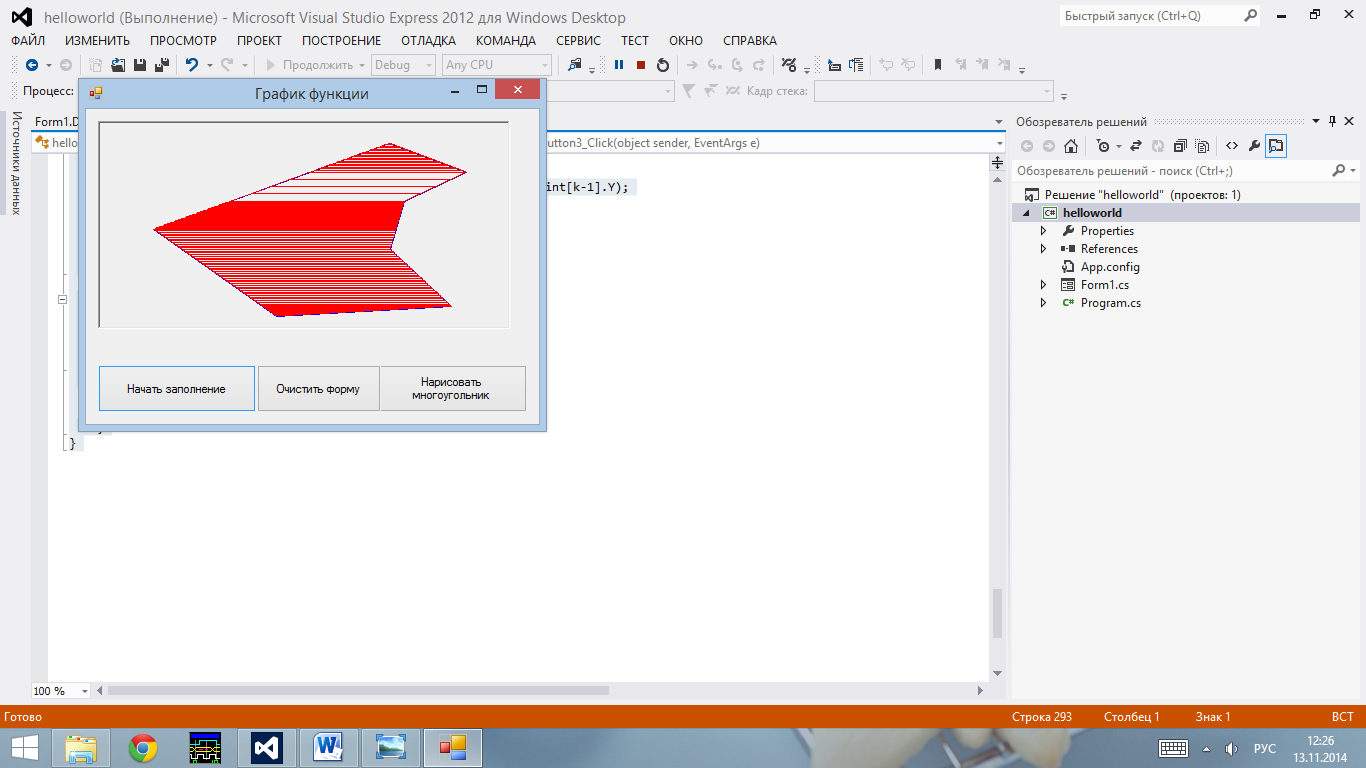


рис.1. Заполнение семиугольника

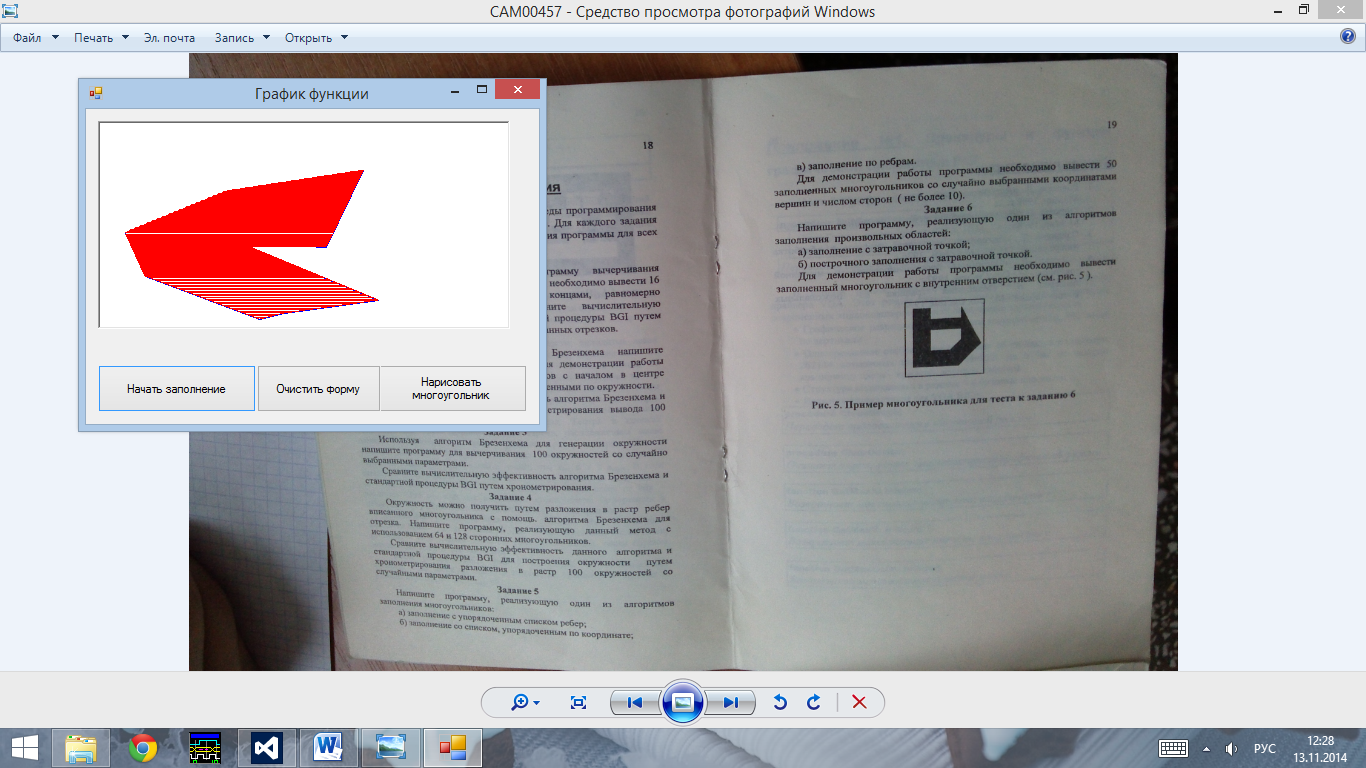


рис.2. Заполнение восьмиугольника

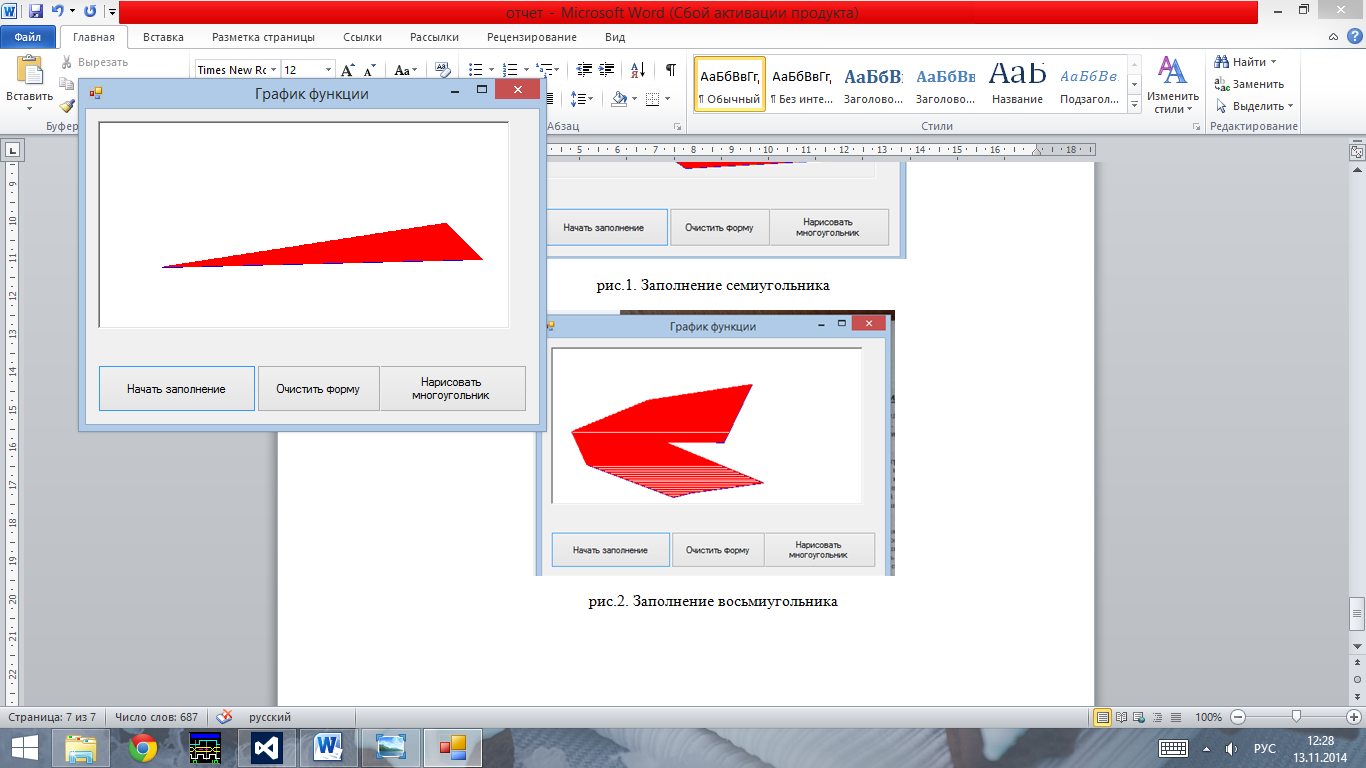


рис.3. Заполнение треугольника

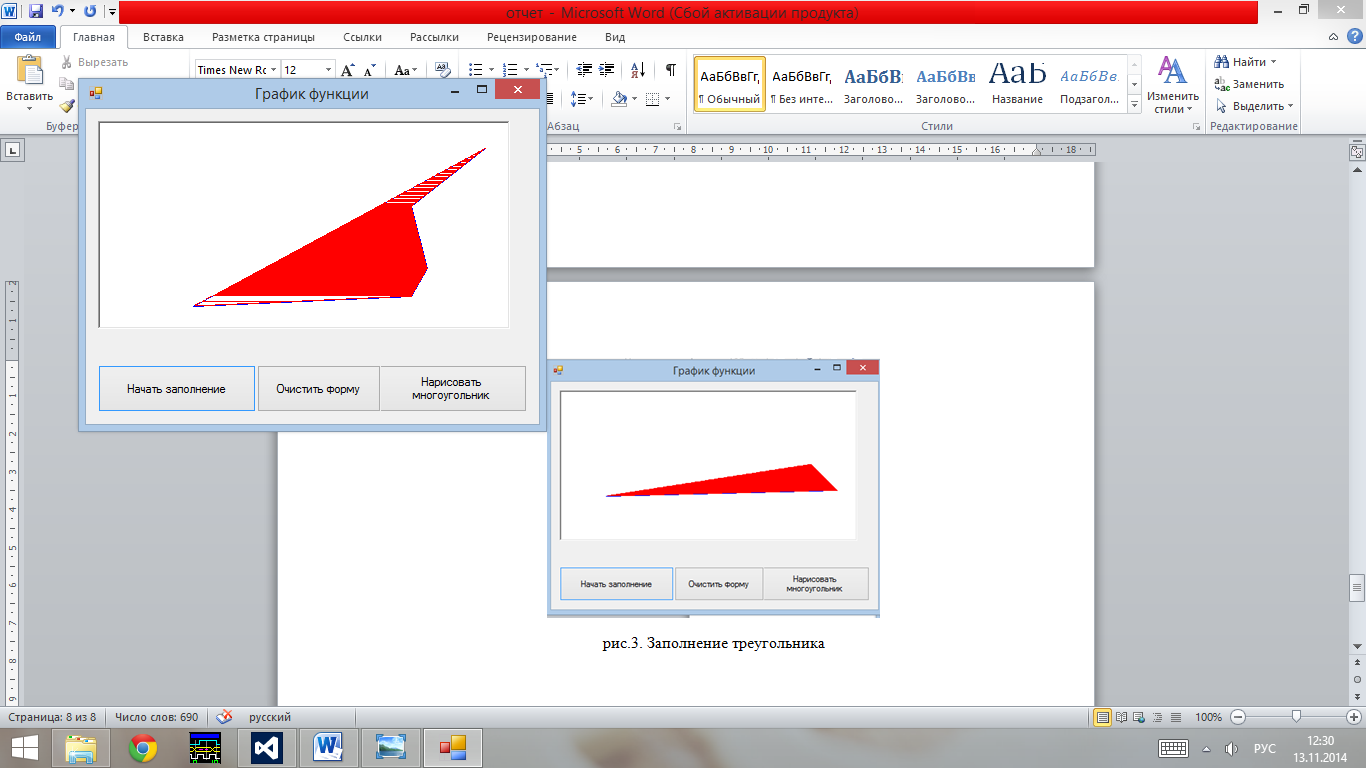


рис.4. Заполнение пятиугольника

Вывод: в результате выполнения данной лабораторной работы я изучила алгоритм заполнения многоугольника со списком, упорядоченным по координате и алгоритм Грэхема, позволяющий построить многоугольник без самопересечений. На рисунке 1, рисунке 2 и рисунке 4 показаны ошибки заполнения, связанные с переходом из векторной графики в растровую. На рисунке 2 и рисунке 4 показаны ошибки заполнения, связанные с заполнением тех строк многоугольника, в которых наблюдается пересечение строки растра с ребром многоугольника и вершиной, в этом случае пересечение с вершиной учитывается два раза, а пересечение с ребром – один раз. Алгоритм учитывает парные пересечения, в результате, некоторые строки не заполняются.