Университет ИТМО

**Лабораторная работа №1**

**по дисциплине «Технологии программирования»**

*вариант 22*

Выполнил:

Припадчев Артём

Преподаватель:

Лабода Ю.А.

Санкт-Петербург

2015

**Задание**

Написать приложение Windows, которое по заданным в файле исходным данным строит горизонтальную или вертикальную столбиковую диаграмму или график.

Этапы выполнения работы

1) Создать главное окно. Заголовок содержит ФИО, гр., вар.

Цвет фона окна - белый.

2) Создать меню вида: Prepare Draw About Exit

File

Choose

По Exit приложение завершается. Пункт Draw запрещен.

По About выдается информация о теме разработки.

3) По File считываются данные из текстового файла, содержащего значения X и Y ( могут быть как положительные, так и отрицательные).

Файл должен быть доступен для корректировки преподавателем.

3) Пo Choose открывается диалоговое окно вида:

┌──────────────────────────────────┐

│ ┌─1──────────┬┐ ┌────────╖ │

│ ╞════════════╡│ │ O K ║ │

│ ├────────────┤│ ╘════════╝ │

│ ├────────────┤│ ┌────────╖ │

│ ├────────────┤│ │ Cancel ║ │

│ └────────────┴┘ ╘════════╝ │

└──────────────────────────────────┘

где 1 - окно-многострочный редактор, содержащее в 1-й строке число, определяющее вид диаграммы, а в следующих 3-х строках - значения составляющих цвета графика в формате RGB (числа от 0 до 255).

Виды графика: 1 - вертикальная столбиковая диаграмма, 2 – горизонтальная столбиковая диаграмма, 3 - ломаная линия.

OK, Cancel - кнопки типа BS\_PUSHBUTTON.

При инициализации диалога фокус ввода должно иметь окно 1.

По OK Пункт Draw разрешается, по Cancel остается в предыдущем состоянии и выбор цвета и вида графика игнорируется.

4) По Draw в главном окне приложения строится соответствующий график на основе значений из файла выбранным цветом цветом и выбранного типа.

При этом окно должно содержать заголовок, наименование и градацию осей.

Цвет осей, рисок градации и всей текстовой информации - синий.

Толщина осей - 2 пиксела, а рисок - 1 пиксел.

Точка пересечения осей должна соответствовать координатам 0,0.

При этом максимальные и минимальные значения по осям должны выбираться в соответствии с максимальными и минимальными значениями, заданными в файле (если все значения по оси отрицательные, то махимальное значение равно 0, а если все значения по оси положительные, то минимальное значение равно 0).

Расположение точки пересечения осей в рабочей области окна также должно выбираться с учетом значений исходных данных(так, например,если максимальное положительное значение по оси больше максимального(по модулю) отрицательного, то и область для вывода этих значений должна быть пропорционально больше;при этом если отрицательные или положительные значения по оси отсутствуют,то и соответсвующий участок этой оси не рисуется.

Таким образом достигается оптимальное заполнение рабочей области окна полезной информацией.

5) Первоначально окно должно располагаться в центре экрана и иметь ширину, равную половине ширины экрана, и высоту - половине высоты экрана.

Обратите внимание на то,что отрисовка осей и построение графика по новым выбранным значениям осуществляется только после входа в пункт Draw.

Изображение должно масштабироваться при изменении размеров окна.

Исходный код

public class Point

{

public int X { get; set; }

public int Y { get; set; }

public Point(int x, int y)

{

this.X = x;

this.Y = y;

}

}

public class RGBColor

{

public int R { get; set; }

public int G { get; set; }

public int B { get; set; }

public RGBColor(int r, int g, int b)

{

this.R = r;

this.G = g;

this.B = b;

}

}

public partial class Form1 : Form

{

private List<Point> points;

private bool isDraw = false;

private bool isData = false;

private static string windowName = "Припадчев Артём Александрович, группа P3415, вариант 22";

private static string verticalDiagram = "Вертикальная столбиковая диаграмма";

private static string horizontalDiagram = "Горизонтальная столбиковая диаграмма";

private static string xoy = "Ломаная линия";

public RGBColor rgbColor = new RGBColor(0, 0, 0);

public int Type = 1;

public Form1()

{

InitializeComponent();

}

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

{

MessageBox.Show("Технологии программирования\r\nЛабораторная 1", "Info", MessageBoxButtons.OK, MessageBoxIcon.Information, MessageBoxDefaultButton.Button1);

}

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

{

Application.Exit();

}

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

{

Form2 f = new Form2(Type, rgbColor);

f.ShowDialog();

if (f.DialogResult == System.Windows.Forms.DialogResult.OK)

{

drawToolStripMenuItem.Enabled = true;

this.Type = f.Type;

this.rgbColor = new RGBColor(f.R, f.G, f.B);

}

}

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

{

points = new List<Point>();

List<string> lines = new List<string>();

try

{

using (StreamReader sr = new StreamReader("data.txt"))

{

while (!sr.EndOfStream) lines.Add(sr.ReadLine());

}

}

catch (FileNotFoundException)

{

MessageBox.Show("File data.txt not found.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

try

{

foreach (var line in lines)

{

string[] temp = line.Split(new[] { " " }, StringSplitOptions.RemoveEmptyEntries);

Point p = new Point(int.Parse(temp[0]), int.Parse(temp[1]));

points.Add(p);

}

isData = true;

MessageBox.Show("File loaded!", "OK", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

catch

{

MessageBox.Show("File is corrupted", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

public Image DrawGraph(int w, int h)

{

Pen pen2 = new Pen(Color.Blue, 2);

Pen pen1 = new Pen(Color.Blue, 1);

Bitmap image = new Bitmap(w, h);

Graphics g = Graphics.FromImage(image);

g.SmoothingMode = SmoothingMode.AntiAlias;

if (Type == 1)

{

int maxPositive = points.Max(p => p.Y);

maxPositive = maxPositive < 0 ? 0 : maxPositive;

int maxNegative = points.Min(p => p.Y);

bool isNegative = maxNegative < 0;

int maxH = isNegative ? maxPositive + Math.Abs(maxNegative) : maxPositive;

int hInterval = (h - 4) / maxH;

int xInterval = (w - 4) / points.Count;

int bottomPadding = isNegative ? hInterval \* Math.Abs(maxNegative) : 0;

SolidBrush rectangleBrush = new SolidBrush(Color.FromArgb(rgbColor.R, rgbColor.G, rgbColor.B));

for (int i = 0; i < points.Count; i++)

{

if (points[i].Y > 0)

{

g.FillRectangle(rectangleBrush, i \* xInterval + 2, h - 2 - bottomPadding - hInterval \* points[i].Y, xInterval, hInterval \* points[i].Y);

}

else

{

g.FillRectangle(rectangleBrush, i \* xInterval + 2, h - 2 - bottomPadding, xInterval, hInterval \* Math.Abs(points[i].Y));

}

}

//X

g.DrawLine(pen2, 2, h - 2 - bottomPadding, w - 2, h - 2 - bottomPadding);

//Y

g.DrawLine(pen2, 2, h - 2, 2, 2);

for (int i = h - 2 - bottomPadding; i > 0; i -= hInterval)

{

g.DrawLine(pen1, 0, i, 4, i);

}

if (isNegative)

for (int i = h - 2 - bottomPadding; i < h; i += hInterval)

{

g.DrawLine(pen1, 0, i, 4, i);

}

g.DrawString("0", SystemFonts.DefaultFont, Brushes.Blue, new PointF(2, h - 2 - bottomPadding));

g.DrawString("X", SystemFonts.DefaultFont, Brushes.Blue, new PointF(w - 15, h - 2 - bottomPadding));

g.DrawString("Y", SystemFonts.DefaultFont, Brushes.Blue, new PointF(4, 10));

this.Text = windowName + " - " + verticalDiagram;

}

if (Type == 2)

{

int maxPositive = points.Max(p => p.X);

maxPositive = maxPositive < 0 ? 0 : maxPositive;

int maxNegative = points.Min(p => p.X);

bool isNegative = maxNegative < 0;

int maxW = isNegative ? maxPositive + Math.Abs(maxNegative) : maxPositive;

int hInterval = (h - 4) / points.Count;

int wInterval = (w - 4) / maxW;

int leftPadding = isNegative ? wInterval \* Math.Abs(maxNegative) : 0;

SolidBrush rectangleBrush = new SolidBrush(Color.FromArgb(rgbColor.R, rgbColor.G, rgbColor.B));

for (int i = 0; i < points.Count; i++)

{

if (points[i].X > 0)

{

g.FillRectangle(rectangleBrush, 2 + leftPadding, i \* hInterval + 2, points[i].X \* wInterval, hInterval);

}

else

{

g.FillRectangle(rectangleBrush, 2 + leftPadding - Math.Abs(points[i].X) \* wInterval, i \* hInterval + 2, Math.Abs(points[i].X) \* wInterval, hInterval);

}

}

//X

g.DrawLine(pen2, 2, 2, w - 2, 2);

//Y

g.DrawLine(pen2, 2 + leftPadding, 2, 2 + leftPadding, h - 2);

for (int i = 2 + leftPadding; i < w; i += wInterval)

{

g.DrawLine(pen1, i, 0, i, 4);

}

if (isNegative)

{

for (int i = 2 + leftPadding; i > 0; i -= wInterval)

{

g.DrawLine(pen1, i, 0, i, 4);

}

}

g.DrawString("0", SystemFonts.DefaultFont, Brushes.Blue, new PointF(6 + leftPadding, 6));

g.DrawString("Y", SystemFonts.DefaultFont, Brushes.Blue, new PointF(6 + leftPadding, h - 16));

g.DrawString("X", SystemFonts.DefaultFont, Brushes.Blue, new PointF(w - 28, 4));

this.Text = windowName + " - " + horizontalDiagram;

}

if (Type == 3)

{

int maxPositiveY = points.Max(p => p.Y);

maxPositiveY = maxPositiveY < 0 ? 0 : maxPositiveY;

int maxNegativeY = points.Min(p => p.Y);

bool isNegativeY = maxNegativeY < 0;

int maxPositiveX = points.Max(p => p.X);

maxPositiveX = maxPositiveX < 0 ? 0 : maxPositiveX;

int maxNegativeX = points.Min(p => p.X);

bool isNegativeX = maxNegativeX < 0;

int maxH = isNegativeY ? maxPositiveY + Math.Abs(maxNegativeY) : maxPositiveY;

int maxW = isNegativeX ? maxPositiveX + Math.Abs(maxNegativeX) : maxPositiveX;

int hInterval = (h - 8) / maxH;

int wInterval = (w - 8) / maxW;

int bottomPadding = isNegativeY ? hInterval \* Math.Abs(maxNegativeY) : 0;

int leftPadding = isNegativeX ? wInterval \* Math.Abs(maxNegativeX) : 0;

SolidBrush brush = new SolidBrush(Color.FromArgb(rgbColor.R, rgbColor.G, rgbColor.B));

int centerX = 4 + leftPadding;

int centerY = h - 4 - bottomPadding;

Point firstP;

Point secondP;

Pen linePen = new Pen(Color.FromArgb(rgbColor.R, rgbColor.G, rgbColor.B), 2);

for (int i = 1; i < points.Count; i++)

{

firstP = points[i - 1];

secondP = points[i];

g.DrawLine(linePen, centerX+firstP.X\*wInterval, centerY - firstP.Y\*hInterval,

centerX + secondP.X\*wInterval, centerY - secondP.Y\*hInterval);

}

for (int i = h - 4 - bottomPadding; i > hInterval; i -= hInterval)

{

g.DrawLine(pen1, centerX - 2, i, centerX + 2, i);

}

if (isNegativeY)

for (int i = h - 4 - bottomPadding; i < h - hInterval; i += hInterval)

{

g.DrawLine(pen1, centerX - 2, i, centerX + 2, i);

}

for (int i = 4 + leftPadding; i < w - wInterval; i += wInterval)

{

g.DrawLine(pen1, i, centerY - 2, i, centerY + 2);

}

if (isNegativeX)

{

for (int i = 4 + leftPadding; i > wInterval; i -= wInterval)

{

g.DrawLine(pen1, i, centerY - 2, i, centerY + 2);

}

}

//X

g.DrawLine(pen2, 4, h - 4 - bottomPadding, w - 4, h - 4 - bottomPadding);

g.DrawLine(pen2, w - 4, h - 4 - bottomPadding, w - 8, h - 1 - bottomPadding);

g.DrawLine(pen2, w - 4, h - 4 - bottomPadding, w - 8, h - 7 - bottomPadding);

//Y

g.DrawLine(pen2, 4 + leftPadding, 4, 4 + leftPadding, h - 4);

g.DrawLine(pen2, 4 + leftPadding, 4, 1 + leftPadding, 7);

g.DrawLine(pen2, 4 + leftPadding, 4, 7 + leftPadding, 7);

g.DrawString("0", SystemFonts.DefaultFont, Brushes.Blue, new PointF(centerX + 2, centerY + 4));

g.DrawString("Y", SystemFonts.DefaultFont, Brushes.Blue, new PointF(6 + leftPadding, 8));

g.DrawString("X", SystemFonts.DefaultFont, Brushes.Blue, new PointF(w - 15, h - 2 - bottomPadding));

this.Text = windowName + " - " + xoy;

}

return image;

}

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

{

if (isData)

{

isDraw = true;

GraphPanel.BackgroundImage = DrawGraph(GraphPanel.Width, GraphPanel.Height);

}

else

{

MessageBox.Show("No input data!", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void GraphPanel\_SizeChanged(object sender, EventArgs e)

{

if (isDraw)

GraphPanel.BackgroundImage = DrawGraph(GraphPanel.Width, GraphPanel.Height);

}

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

{

this.Text = windowName;

this.Width = Screen.PrimaryScreen.Bounds.Width / 2;

this.Height = Screen.PrimaryScreen.Bounds.Height / 2;

this.CenterToScreen();

}

}

public partial class Form2 : Form

{

private int type = 1;

private int r = 0;

private int g = 0;

private int b = 0;

public int Type { get { return type; } }

public int R { get { return r; } }

public int G { get { return g; } }

public int B { get { return b; } }

public Form2(int type, RGBColor color)

{

InitializeComponent();

this.type = type;

this.r = color.R;

this.g = color.G;

this.b = color.B;

}

private void OnlyDigit(object sender, KeyPressEventArgs e)

{

if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar))

{

e.Handled = true;

}

}

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

{

type = (int)TypeNumericUpDown.Value;

r = (int)RNumericUpDown.Value;

g = (int)GNumericUpDown.Value;

b = (int)BNumericUpDown.Value;

}

private void Form2\_Load(object sender, EventArgs e)

{

TypeNumericUpDown.Value = type;

RNumericUpDown.Value = r;

GNumericUpDown.Value = g;

BNumericUpDown.Value = b;

TypeNumericUpDown.Select();

}

private void TypeNumericUpDown\_ValueChanged(object sender, EventArgs e)

{

this.type = (int)TypeNumericUpDown.Value;

}

private void RNumericUpDown\_ValueChanged(object sender, EventArgs e)

{

this.r = (int)RNumericUpDown.Value;

}

private void GNumericUpDown\_ValueChanged(object sender, EventArgs e)

{

this.g = (int)GNumericUpDown.Value;

}

private void BNumericUpDown\_ValueChanged(object sender, EventArgs e)

{

this.b = (int)BNumericUpDown.Value;

}

}

**Вывод:** в ходе выполнения лабораторной работы были изучены способы рисования в C#, способы работы с файлами и данными внутри программы, а также взаимодействие с диалоговыми окнами. Была создана программа, отображающая разные виды графиков по заданным в файле точкам.