МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

***Факультет информационных технологий и робототехники***

Кафедра программного обеспечения информационных систем и технологий

**Отчет по лабораторной работе № 9**

по дисциплине: ”Разработка приложений в визуальных средах ”

# **на тему: *”* Разработка многопоточных приложений *”***

Вариант 1

Выполнил**:** студент группы 10701322 Бородкин Д.В.

Принял**:** доц. Гурский Н. Н.

Минск 2024

**Лабораторная работа № 9.**

**Цель лабораторной работы:** изучить принципы организации параллельных вычислений в отдельных классах.

**Задание:** разработать приложение, содержащее два потока. В первом потоке случайным образом формировать параметры вычисления определенного интеграла, выбор подынтегральной функции с последующим ее отображением и выводом результатов (для работы с определенным интегралом использовать класс, разработанный в лабораторной работе 3). Второй поток выполняет рисование геометрических фигур. Для построения геометрических фигур разработать класс в отдельном Unit. Предусмотреть изменение приоритетов в потоках.

**Текст программы:**

using System;

using System.Drawing;

using System.Drawing.Drawing2D;

using System.Windows.Forms;

using System.Threading;

using OprIntegral;

using System.Windows.Forms.DataVisualization.Charting;

namespace Lab\_8

{

public partial class Form1 : Form

{

Random rand; // Переменная для объекта рандом

Thread ThreadRectangle; // Переменная для объекта потока рисование прямоугольника

Thread ThreadFigure; // Переменная для объекта потока рисование фигур

Thread Integral; // Переменная для объекта потока расчёт интеграла

private OprIntegral.OprIntegral ITrap; // Переменная для объекта графика

private OprIntegral.OprIntegral ILeft; // Переменная для объекта графика

private OprIntegral.OprIntegral IRight; // Переменная для объекта графика

private OprIntegral.OprIntegral IMiddle; // Переменная для объекта графика

// Конструктор формы

[Obsolete]

public Form1()

{

InitializeComponent();

textBox1.Text = "1";

textBox3.Text = "5";

textBox2.Text = "0,1";

rand = new Random();

radioButton1.Checked = true;

radioButton8.Checked = true;

comboBox1.SelectedIndex = 4;

comboBox2.SelectedIndex = 3;

comboBox3.SelectedIndex = 2;

}

// Метод для установки цвета текстового поля

private void SetTextBoxColor(System.Windows.Forms.TextBox textBox)

{

//colorDialog1.AllowFullOpen = false;

//colorDialog1.ShowHelp = true;

//colorDialog1.Color = textBox.BackColor;

//if (colorDialog1.ShowDialog() == DialogResult.OK)

//{

// textBox.BackColor = colorDialog1.Color;

//}

// Генерация случайного цвета

Color randomColor = Color.FromArgb(rand.Next(256), rand.Next(256), rand.Next(256));

// Установка цвета текстового поля

textBox.BackColor = randomColor;

}

// Выбрать цвет линии графика

private void Button12\_Click(object sender, EventArgs e)

{

SetTextBoxColor(textBox9);

}

// Выбрать цвет площади графика

private void Button13\_Click(object sender, EventArgs e)

{

SetTextBoxColor(textBox10);

}

// Функция для рисования фигуры

private void Graph()

{

int x = 0, y = 0;

int angle = 0;

Graphics graph = panel1.CreateGraphics();

Pen pen = new Pen(Color.Black);

while (true)

{

Rectangle myRectangle = new Rectangle(x, y, 10, 10);

// Задание градиентной заливки

LinearGradientBrush myBrush = new LinearGradientBrush(myRectangle, Color.Red, Color.Yellow, angle);

//Генерация x,y

x = rand.Next(0, 579);

y = rand.Next(0, 579);

graph.FillRectangle(myBrush, x, y, 10, 10);

graph.DrawRectangle(pen, x,y, 10, 10);

angle += 10;

}

}

// Старт первого потока

private void Button9\_Click(object sender, EventArgs e)

{

ThreadRectangle = new Thread(Graph);

switch (comboBox3.SelectedIndex)

{

case 0:

ThreadRectangle.Priority = ThreadPriority.Highest;

break;

case 1:

ThreadRectangle.Priority = ThreadPriority.AboveNormal;

break;

case 2:

ThreadRectangle.Priority = ThreadPriority.Normal;

break;

case 3:

ThreadRectangle.Priority = ThreadPriority.BelowNormal;

break;

case 4:

ThreadRectangle.Priority = ThreadPriority.Lowest;

break;

}

ThreadRectangle.Start();

}

// Пауза первого потока

[Obsolete] //пометить элемент программы как устаревший

private void Button8\_Click(object sender, EventArgs e)

{

ThreadRectangle.Suspend();

}

// Возообновить работу первого потока

[Obsolete] //пометить элемент программы как устаревший

private void Button7\_Click(object sender, EventArgs e)

{

ThreadRectangle.Resume();

}

// Остановить работу первого потока

private void Button10\_Click(object sender, EventArgs e)

{

ThreadRectangle.Abort();

}

//Рисование фигуры

private void DrawFigure()

{

pictureBox1.BackColor = Color.Black;

Shapes shapes = new Shapes(pictureBox1.CreateGraphics(), pictureBox1.Height, pictureBox1.Width);

while (true)

{

while (shapes.DrawNextPoint())

{

Thread.Sleep(10);

}

Thread.Sleep(100);

shapes.Reset();

}

}

// Старт второго потока

private void Button11\_Click(object sender, EventArgs e)

{

ThreadFigure = new Thread(DrawFigure);

switch (comboBox2.SelectedIndex)

{

case 0:

ThreadFigure.Priority = ThreadPriority.Highest;

break;

case 1:

ThreadFigure.Priority = ThreadPriority.AboveNormal;

break;

case 2:

ThreadFigure.Priority = ThreadPriority.Normal;

break;

case 3:

ThreadFigure.Priority = ThreadPriority.BelowNormal;

break;

case 4:

ThreadFigure.Priority = ThreadPriority.Lowest;

break;

}

ThreadFigure.Start();

}

// Пауза второго потока

[Obsolete] //пометить элемент программы как устаревший

private void Button3\_Click(object sender, EventArgs e)

{

ThreadFigure.Suspend();

}

// Возообновить работу второго потока

[Obsolete] //пометить элемент программы как устаревший

private void Button2\_Click(object sender, EventArgs e)

{

ThreadFigure.Resume();

}

// Остановить работу второго потока

private void Button1\_Click(object sender, EventArgs e)

{

ThreadFigure.Abort();

}

// Подсчёт Интеграла

private void IntegralFunc()

{

while (true)

{

Invoke(new Action(() =>

{

chart1.Series.Clear();

chart1.ResetAutoValues();

}));

// Инициализация переменных

OprIntegral.OprIntegral I = null;

ITrap = new Trap();

ILeft = new Left();

IRight = new Right();

IMiddle = new Middle();

SeriesChartType tp = SeriesChartType.Column;

double rez;

int radioButtonFunc = rand.Next(1, 4); // Случайный выбор между 1 и 3

// С использованием метода Invoke для установки радио кнопки

Invoke(new Action(() =>

{

switch (radioButtonFunc)

{

case 1:

radioButton10.Checked = true;

break;

case 2:

radioButton9.Checked = true;

break;

case 3:

radioButton8.Checked = true;

break;

}

}));

if (radioButton10.Checked)

{

ILeft.f = ILeft.F1;

IRight.f = IRight.F1;

IMiddle.f = IMiddle.F1;

ITrap.f = ITrap.F1;

}

else if (radioButton9.Checked)

{

ILeft.f = ILeft.F2;

IRight.f = IRight.F2;

IMiddle.f = IMiddle.F2;

ITrap.f = ITrap.F2;

}

else if (radioButton8.Checked)

{

ILeft.f = ILeft.F3;

IRight.f = IRight.F3;

IMiddle.f = IMiddle.F3;

ITrap.f = ITrap.F3;

}

int radioButtonMethod = rand.Next(1, 5); // Случайный выбор между 1 и 4

// С использованием метода Invoke для установки выбранного метода подсчёта

Invoke(new Action(() =>

{

switch (radioButtonMethod)

{

case 1:

radioButton1.Checked = true;

break;

case 2:

radioButton2.Checked = true;

break;

case 3:

radioButton3.Checked = true;

break;

case 4:

radioButton4.Checked = true;

break;

}

}));

if (radioButton1.Checked)

{

I = ITrap;

tp = SeriesChartType.Area;

}

else if (radioButton2.Checked)

{

I = ILeft;

}

else if (radioButton3.Checked)

{

I = IRight;

}

else if (radioButton4.Checked)

{

I = IMiddle;

}

// Установка значений полей случайным образом

int A = rand.Next(-5, 5); // Случайное число от -5 до 5

int B = rand.Next(5, 15); // Случайное число от 5 до 15

double H = rand.Next(1, 51) \* 0.01; // Случайное число от 0.01 до 0.5 с точностью до сотых

// С использованием метода Invoke для установки значений полей в текстовых полях

Invoke(new Action(() =>

{

textBox1.Text = A.ToString();

textBox3.Text = B.ToString();

textBox2.Text = H.ToString();

}));

// Эмулируем нажатие кнопки, чтобы выбрался цвет для графика автоматически

Invoke(new Action(() =>

{

button12.PerformClick();

button13.PerformClick();

}));

// Установка цветов графика

ILeft.Mycolor = textBox9.BackColor; // Цвет линии первого интеграла

ILeft.Ploshad = textBox10.BackColor; // Цвет площади

IRight.Mycolor = textBox9.BackColor; // Цвет линии первого интеграла

IRight.Ploshad = textBox10.BackColor; // Цвет площади

IMiddle.Mycolor = textBox9.BackColor; // Цвет линии первого интеграла

IMiddle.Ploshad = textBox10.BackColor; // Цвет площади

ITrap.Mycolor = textBox9.BackColor; // Цвет линии первого интеграла

ITrap.Ploshad = textBox10.BackColor; // Цвет площади

Invoke(new Action(() =>

{

// Установка ссылки на график

I.Mychart = chart1;

// Расчет и отрисовка

I.GetSet\_A = double.TryParse(textBox1.Text, out double aValue) ? aValue : 1;

I.GetSet\_B = double.TryParse(textBox3.Text, out double bValue) ? bValue : 5;

I.GetSet\_H = double.TryParse(textBox2.Text, out double hValue) ? hValue : 10;

rez = I.Calculate();

// Проверка на бесконечность или NaN

if (double.IsInfinity(rez) || double.IsNaN(rez))

{

// Установка значений полей случайным образом

int A1 = 2;

int B1 = 5;

double H1 = 0.1;

textBox1.Text = A1.ToString();

textBox3.Text = B1.ToString();

textBox2.Text = H1.ToString();

// Пересчет интеграла с этими данными

I.GetSet\_A = double.TryParse(textBox1.Text, out double aV) ? aV : 1;

I.GetSet\_B = double.TryParse(textBox3.Text, out double bV) ? bV : 5;

I.GetSet\_H = double.TryParse(textBox2.Text, out double hV) ? hV : 10;

rez = I.Calculate();

}

// Отрисовка результата

I.DrawPl1(tp);

I.DrawF();

// Вывод результата

richTextBox1.Clear();

richTextBox1.Text = rez.ToString("F5") + Environment.NewLine;

}));

// Задержка на 5 секунд

Thread.Sleep(5000);

}

}

// Старт третьего потока

private void Button14\_Click(object sender, EventArgs e)

{

Integral = new Thread(IntegralFunc);

switch (comboBox1.SelectedIndex)

{

case 0:

Integral.Priority = ThreadPriority.Highest;

break;

case 1:

Integral.Priority = ThreadPriority.AboveNormal;

break;

case 2:

Integral.Priority = ThreadPriority.Normal;

break;

case 3:

Integral.Priority = ThreadPriority.BelowNormal;

break;

case 4:

Integral.Priority = ThreadPriority.Lowest;

break;

}

Integral.Start();

}

// Пауза третьего потока

[Obsolete] //пометить элемент программы как устаревший

private void Button6\_Click(object sender, EventArgs e)

{

Integral.Suspend();

}

// Возобновить работу третьего потока

[Obsolete] //пометить элемент программы как устаревший

private void Button5\_Click(object sender, EventArgs e)

{

Integral.Resume();

}

// Остановить работу третьего потока

private void Button4\_Click(object sender, EventArgs e)

{

Integral.Abort();

}

}

}

**Результаты выполнения программы:**

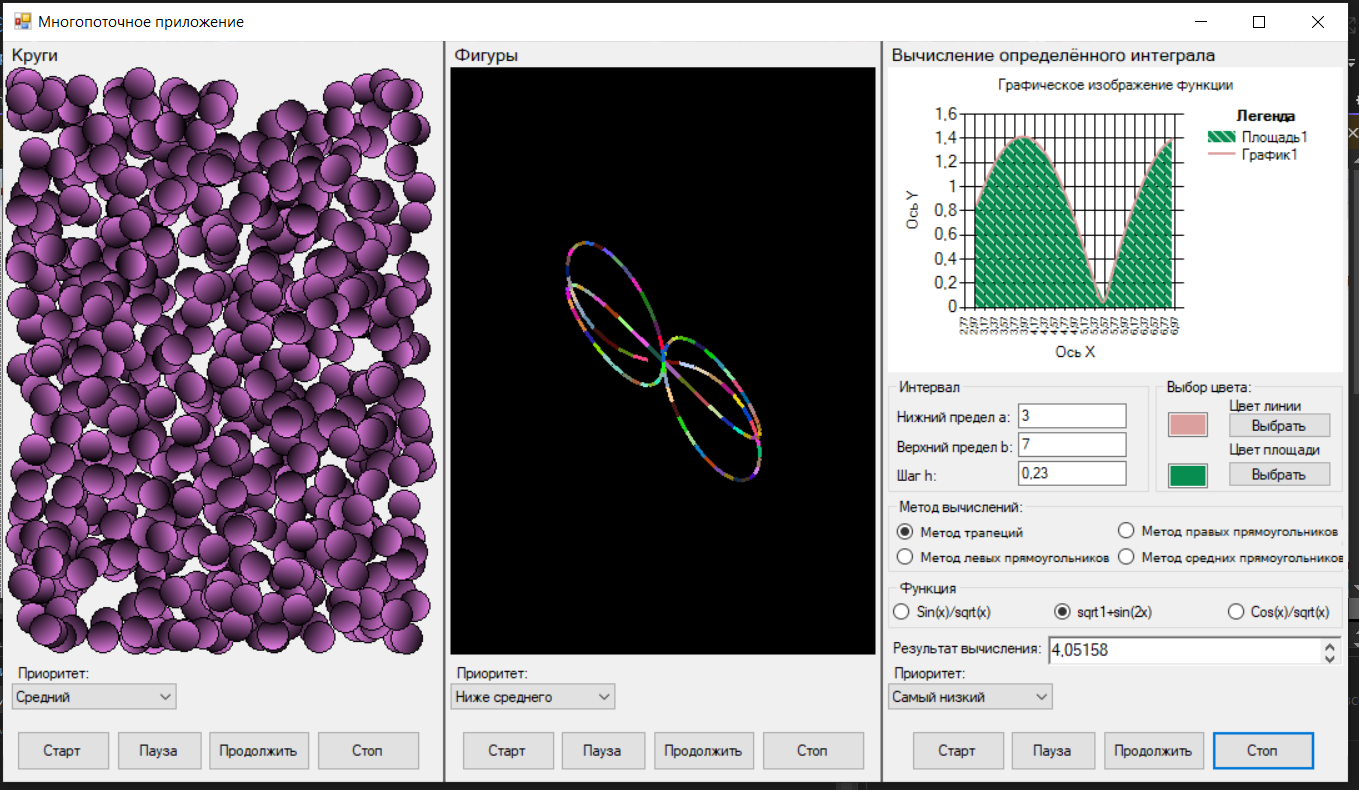


Рисунок 1. Результат работы программы.

**Вывод:** изучил принципы организации параллельных вычислений в отдельных классах.