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 № 3**

по дисциплине: ”Разработка приложений в визуальных средах ”

# **на тему: *”* Создание приложения, поддерживающего принцип инкапсуляции*”***

Вариант 1

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

Принял**:** доц. Гурский Н. Н.

Минск 2024

**Лабораторная работа № 3.**

**Цель лабораторной работы:** изучить принципы реализации инкапсуляции, секции доступа, свойства, работу с несколькими объектами класса.

**Задание:** на примере класса вычисления определенного интеграла разработать свойства доступа к защищенным полям. Посредством свойств организовать проверку корректности вводимых данных и рисование пределов интегрирования. Программа должна вычислять сумму и разность двух интегралов и визуально показывать их площади. Общую область интегралов необходимо выделить отдельным цветом.

**Текст программы:**

using System;

using System.Windows.Forms;

using System.Windows.Forms.DataVisualization.Charting;

namespace Lab\_3

{

public partial class Form1 : Form

{

private OprIntegral I; // Переменная для объекта графика 1

private OprIntegral I1; // Переменная для объекта графика 2

private OprIntegral I2; // Переменная для объекта общего графика

private OprIntegral I3; // Переменная для объекта общего графика

//Конструктор

public Form1()

{

InitializeComponent();

textBox1.Text = "1";

textBox2.Text = "5";

textBox3.Text = "0,03";

textBox4.Text = "3";

textBox5.Text = "7";

textBox6.Text = "0,03";

textBox7.BackColor = System.Drawing.Color.Blue;

textBox8.BackColor = System.Drawing.Color.Gold;

textBox9.BackColor = System.Drawing.Color.Red;

textBox10.BackColor = System.Drawing.Color.Green;

textBox11.BackColor = System.Drawing.Color.Yellow;

textBox12.BackColor = System.Drawing.Color.BlueViolet;

radioButton6.Checked = true;

radioButton9.Checked = true;

radioButton1.Checked = true;

I = new OprIntegral();

I1 = new OprIntegral();

}

//Метод для выборки цветов на графике

private void SetTextBoxColor(TextBox textBox)

{

colorDialog1.AllowFullOpen = false;

colorDialog1.ShowHelp = true;

colorDialog1.Color = textBox.BackColor;

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

{

textBox.BackColor = colorDialog1.Color;

}

}

//Цвет линии первого интеграла

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

{

SetTextBoxColor(textBox7);

}

//Цвет площади первого интеграла

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

{

SetTextBoxColor(textBox8);

}

//Цвет линии второго интеграла

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

{

SetTextBoxColor(textBox9);

}

//Цвет линии для общей

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

{

SetTextBoxColor(textBox12);

}

//Цвет площади второго интеграла

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

{

SetTextBoxColor(textBox10);

}

//Цвет площади объединения интегралов

private void Button15\_Click(object sender, EventArgs e)

{

SetTextBoxColor(textBox11);

}

//Проверка на ввод, чтобы только цифры были

private void TextBox1\_KeyPress(object sender, KeyPressEventArgs e)

{

// Разрешаем ввод цифр, знака минус и клавиши Backspace

if (char.IsDigit(e.KeyChar) || e.KeyChar == '-' || e.KeyChar == '\b' || e.KeyChar == '.')

{

// Разрешаем ввод

e.Handled = false;

}

else

{

// Запрещаем ввод

e.Handled = true;

}

}

//Вычислить интеграл первый

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

{

if (radioButton5.Checked)

{

I.f = I.F1;

RezCalc();

}

else if (radioButton6.Checked)

{

I.f = I.F2;

RezCalc();

}

else if (radioButton7.Checked)

{

I.f = I.F3;

RezCalc();

}

}

//Построить график первого интеграла

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

{

chart1.ResetAutoValues();

chart1.Series.Clear();

I.Mycolor = textBox7.BackColor; // Цвет линии первого интеграла

I.Ploshad = textBox8.BackColor; // Цвет площади

if (radioButton5.Checked)

{

I.f = I.F1;

Draw1();

}

else if (radioButton6.Checked)

{

I.f = I.F2;

Draw1();

}

else if (radioButton7.Checked)

{

I.f = I.F3;

Draw1();

}

}

//Вычислить второй интеграл

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

{

if (radioButton10.Checked)

{

I1.f = I1.F1;

RezCalc2();

}

else if (radioButton9.Checked)

{

I1.f = I1.F2;

RezCalc2();

}

else if (radioButton8.Checked)

{

I1.f = I1.F3;

RezCalc2();

}

}

//Построить график второго интеграла

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

{

chart1.ResetAutoValues();

chart1.Series.Clear();

I1.Mycolor = textBox9.BackColor; // Цвет линии второго интеграла

I1.Ploshad = textBox10.BackColor; // Цвет площади

if (radioButton10.Checked)

{

I1.f = I.F1;

Draw2();

}

else if (radioButton9.Checked)

{

I1.f = I.F2;

Draw2();

}

else if (radioButton8.Checked)

{

I1.f = I.F3;

Draw2();

}

}

// Метод подсчёта функций для первого интеграла

private double RezCalc()

{

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

I.GetSet\_B = double.TryParse(textBox2.Text, out double bValue) ? bValue : 5;

I.GetSet\_H = double.TryParse(textBox3.Text, out double hValue) ? hValue : 10;

double rez1 = 0;

if (radioButton1.Checked)

{

rez1 = I.Trap();

}

else if (radioButton2.Checked)

{

rez1 = I.Left();

}

else if (radioButton3.Checked)

{

rez1 = I.Right();

}

else if (radioButton4.Checked)

{

rez1 = I.Middle();

}

// Результирующая строка для отображения пользовательских данных

string rezText = "Ваша выбранная функция: " + Environment.NewLine;

rezText += label1.Text + textBox1.Text + Environment.NewLine;

rezText += label2.Text + textBox2.Text + Environment.NewLine;

rezText += label3.Text + textBox3.Text + Environment.NewLine;

rezText += "Результаты вычисления : " + rez1.ToString("F5") + Environment.NewLine;

richTextBox1.Text = rezText.ToString() + Environment.NewLine;

return rez1;

}

//Метод для рисования графика 1

private void Draw1()

{

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

I.GetSet\_B = double.TryParse(textBox2.Text, out double bValue) ? bValue : 5;

I.GetSet\_H = double.TryParse(textBox3.Text, out double hValue) ? hValue : 10;

I.Mychart = chart1;

if (radioButton1.Checked)

{

I.Trap();

I.DrawPl1(SeriesChartType.Area);

}

else if (radioButton2.Checked)

{

I.Left();

I.DrawPl1(SeriesChartType.Column);

}

else if (radioButton3.Checked)

{

I.Right();

I.DrawPl1(SeriesChartType.Column);

}

else if (radioButton4.Checked)

{

I.Middle();

I.DrawPl1(SeriesChartType.Column);

}

I.DrawF();

}

// Метод подсчёта функций для второго интеграла

private double RezCalc2()

{

I1.GetSet\_A = double.TryParse(textBox4.Text, out double aValue) ? aValue : 3;

I1.GetSet\_B = double.TryParse(textBox5.Text, out double bValue) ? bValue : 7;

I1.GetSet\_H = double.TryParse(textBox6.Text, out double hValue) ? hValue : 10;

double rez2 = 0;

if (radioButton1.Checked)

{

rez2 = I1.Trap();

}

else if (radioButton2.Checked)

{

rez2 = I1.Left();

}

else if (radioButton3.Checked)

{

rez2 = I1.Right();

}

else if (radioButton4.Checked)

{

rez2 = I1.Middle();

}

// Результирующая строка для отображения пользовательских данных

string rezText = "Ваши введённые данные: " + Environment.NewLine;

rezText += label4.Text + textBox4.Text + Environment.NewLine;

rezText += label5.Text + textBox5.Text + Environment.NewLine;

rezText += label6.Text + textBox6.Text + Environment.NewLine;

rezText += "Результаты вычисления : " + rez2.ToString("F5") + Environment.NewLine;

richTextBox1.Text = rezText.ToString() + Environment.NewLine;

return rez2;

}

//Метод для рисования графика 2

private void Draw2()

{

I1.GetSet\_A = double.TryParse(textBox4.Text, out double aValue) ? aValue : 3;

I1.GetSet\_B = double.TryParse(textBox5.Text, out double bValue) ? bValue : 7;

I1.GetSet\_H = double.TryParse(textBox6.Text, out double hValue) ? hValue : 10;

I1.Mychart = chart1;

if (radioButton1.Checked)

{

I1.Trap();

I1.DrawPl2(SeriesChartType.Area);

}

else if (radioButton2.Checked)

{

I1.Left();

I1.DrawPl2(SeriesChartType.Column);

}

else if (radioButton3.Checked)

{

I1.Right();

I1.DrawPl2(SeriesChartType.Column);

}

else if (radioButton4.Checked)

{

I1.Middle();

I1.DrawPl2(SeriesChartType.Column);

}

I1.DrawF2();

}

//Выход из приложения

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

{

Application.Exit();

}

//Очистить текстовые поля

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

{

textBox1.Clear();

textBox2.Clear();

textBox3.Clear();

textBox4.Clear();

textBox5.Clear();

textBox6.Clear();

richTextBox1.Clear();

}

//Сумма двух интегралов

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

{

if (radioButton5.Checked)

{

I.f = I.F1;

RezCalc();

}

else if (radioButton6.Checked)

{

I.f = I.F2;

RezCalc();

}

else if (radioButton7.Checked)

{

I.f = I.F3;

RezCalc();

}

if (radioButton10.Checked)

{

I1.f = I1.F1;

RezCalc2();

}

else if (radioButton9.Checked)

{

I1.f = I1.F2;

RezCalc2();

}

else if (radioButton8.Checked)

{

I1.f = I1.F3;

RezCalc2();

}

double integral1 = RezCalc();

double integral2 = RezCalc2();

double rezultTwo = integral1 + integral2;

// Результирующая строка для отображения пользовательских данных

string rezText = "Ваши введённые данные первого интеграла: " + Environment.NewLine;

rezText += label1.Text + textBox1.Text + Environment.NewLine;

rezText += label2.Text + textBox2.Text + Environment.NewLine;

rezText += label3.Text + textBox3.Text + Environment.NewLine;

rezText += "Первый интеграл: " + integral1.ToString("F5") + Environment.NewLine;

rezText += "Ваши введённые данные второго интеграла: " + Environment.NewLine;

rezText += label4.Text + textBox4.Text + Environment.NewLine;

rezText += label5.Text + textBox5.Text + Environment.NewLine;

rezText += label6.Text + textBox6.Text + Environment.NewLine;

rezText += "Второй интеграл: " + integral2.ToString("F5") + Environment.NewLine;

rezText += "Сумма двух интегралов: " + rezultTwo.ToString("F5") + Environment.NewLine;

richTextBox1.Text = rezText.ToString() + Environment.NewLine;

}

//Разность двух интегралов

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

{

if (radioButton5.Checked)

{

I.f = I.F1;

RezCalc();

}

else if (radioButton6.Checked)

{

I.f = I.F2;

RezCalc();

}

else if (radioButton7.Checked)

{

I.f = I.F3;

RezCalc();

}

if (radioButton10.Checked)

{

I1.f = I1.F1;

RezCalc2();

}

else if (radioButton9.Checked)

{

I1.f = I1.F2;

RezCalc2();

}

else if (radioButton8.Checked)

{

I1.f = I1.F3;

RezCalc2();

}

double integral1 = RezCalc();

double integral2 = RezCalc2();

double rezultTwo1 = integral1 - integral2;

double rezultTwo2 = integral2 - integral1;

// Результирующая строка для отображения пользовательских данных

string rezText = "Ваши введённые данные первого интеграла: " + Environment.NewLine;

rezText += label1.Text + textBox1.Text + Environment.NewLine;

rezText += label2.Text + textBox2.Text + Environment.NewLine;

rezText += label3.Text + textBox3.Text + Environment.NewLine;

rezText += "Первый интеграл: " + integral1.ToString("F5") + Environment.NewLine;

rezText += "Ваши введённые данные второго интеграла: " + Environment.NewLine;

rezText += label4.Text + textBox4.Text + Environment.NewLine;

rezText += label5.Text + textBox5.Text + Environment.NewLine;

rezText += label6.Text + textBox6.Text + Environment.NewLine;

rezText += "Второй интеграл: " + integral2.ToString("F5") + Environment.NewLine;

rezText += "Разность двух интегралов (1 - 2): " + rezultTwo1.ToString("F5") + Environment.NewLine;

rezText += "Разность двух интегралов (2 - 1): " + rezultTwo2.ToString("F5") + Environment.NewLine;

richTextBox1.Text = rezText.ToString() + Environment.NewLine;

}

//Общая область двух интегралов

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

{

chart1.ResetAutoValues();

chart1.Series.Clear();

Pl1();

Pl2();

G1();

G();

}

//График объединения 1

private void G()

{

if ( I.GetSet\_B < I1.GetSet\_A)

{

I2 = new OprIntegral();

I2.GetSet\_A = double.TryParse(textBox4.Text, out double aValue) ? aValue : 3;

I2.GetSet\_B = double.TryParse(textBox2.Text, out double bValue) ? bValue : 5;

I2.GetSet\_H = double.TryParse(textBox6.Text, out double hValue) ? hValue : 0.3;

I2.Mychart = chart1;

I2.Mycolor = textBox12.BackColor; // Цвет линии

I2.Ploshad = textBox11.BackColor; // Цвет площади

if (radioButton5.Checked)

{

I2.f = I2.F1;

}

else if (radioButton6.Checked)

{

I2.f = I2.F2;

}

else if (radioButton7.Checked)

{

I2.f = I2.F3;

}

if (radioButton1.Checked)

{

I2.Trap();

I2.UfPl(SeriesChartType.Area);

}

else if (radioButton2.Checked)

{

I2.Left();

I2.UfPl(SeriesChartType.Column);

}

else if (radioButton3.Checked)

{

I2.Right();

I2.UfPl(SeriesChartType.Column);

}

else if (radioButton4.Checked)

{

I2.Middle();

I2.UfPl(SeriesChartType.Column);

}

I2.Unification();

}

}

//График объединения 2

private void G1()

{

if (I.GetSet\_B < I1.GetSet\_A)

{

I3 = new OprIntegral();

I3.GetSet\_A = double.TryParse(textBox4.Text, out double aValue) ? aValue : 3;

I3.GetSet\_B = double.TryParse(textBox2.Text, out double bValue) ? bValue : 5;

I3.GetSet\_H = double.TryParse(textBox6.Text, out double hValue) ? hValue : 0.3;

I3.Mychart = chart1;

I3.Mycolor = textBox12.BackColor; // Цвет линии

I3.Ploshad = textBox11.BackColor; // Цвет площади

if (radioButton10.Checked)

{

I3.f = I3.F1;

}

else if (radioButton9.Checked)

{

I3.f = I3.F2;

}

else if (radioButton8.Checked)

{

I3.f = I3.F3;

}

if (radioButton1.Checked)

{

I3.Trap();

I3.UfPl1(SeriesChartType.Area);

}

else if (radioButton2.Checked)

{

I3.Left();

I3.UfPl1(SeriesChartType.Column);

}

else if (radioButton3.Checked)

{

I3.Right();

I3.UfPl1(SeriesChartType.Column);

}

else if (radioButton4.Checked)

{

I3.Middle();

I3.UfPl1(SeriesChartType.Column);

}

I3.Unification1();

}

}

//Для объединения графиков 1

private void Pl1()

{

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

I.GetSet\_B = double.TryParse(textBox4.Text, out double bValue) ? bValue : 5;

I.GetSet\_H = double.TryParse(textBox3.Text, out double hValue) ? hValue : 0.3;

I.Mychart = chart1;

I.Mycolor = textBox7.BackColor; // Цвет линии первого интеграла

I.Ploshad = textBox8.BackColor; // Цвет площади

if (radioButton5.Checked)

{

I.f = I.F1;

}

else if (radioButton6.Checked)

{

I.f = I.F2;

}

else if (radioButton7.Checked)

{

I.f = I.F3;

}

if (radioButton1.Checked)

{

I.Trap();

I.DrawPl1(SeriesChartType.Area);

}

else if (radioButton2.Checked)

{

I.Left();

I.DrawPl1(SeriesChartType.Column);

}

else if (radioButton3.Checked)

{

I.Right();

I.DrawPl1(SeriesChartType.Column);

}

else if (radioButton4.Checked)

{

I.Middle();

I.DrawPl1(SeriesChartType.Column);

}

I.DrawF();

}

//Для объединения графиков 2

private void Pl2()

{

I1.GetSet\_A = double.TryParse(textBox2.Text, out double aValue) ? aValue : 3;

I1.GetSet\_B = double.TryParse(textBox5.Text, out double bValue) ? bValue : 7;

I1.GetSet\_H = double.TryParse(textBox6.Text, out double hValue) ? hValue : 0.3;

I1.Mychart = chart1;

I1.Mycolor = textBox9.BackColor; // Цвет линии второго интеграла

I1.Ploshad = textBox10.BackColor; // Цвет площади

if (radioButton10.Checked)

{

I1.f = I1.F1;

}

else if (radioButton9.Checked)

{

I1.f = I1.F2;

}

else if (radioButton8.Checked)

{

I1.f = I1.F3;

}

if (radioButton1.Checked)

{

I1.Trap();

I1.DrawPl2(SeriesChartType.Area);

}

else if (radioButton2.Checked)

{

I1.Left();

I1.DrawPl2(SeriesChartType.Column);

}

else if (radioButton3.Checked)

{

I1.Right();

I1.DrawPl2(SeriesChartType.Column);

}

else if (radioButton4.Checked)

{

I1.Middle();

I1.DrawPl2(SeriesChartType.Column);

}

I1.DrawF2();

}

}

}

Отдельный класс:

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Windows.Forms.DataVisualization.Charting;

namespace Lab\_3

{

internal class OprIntegral

{

private double a; // верхний

private double b; // нижний

private double h; // шаг

private Chart mychart; // чарт

private Color mycolor, ploshad; // цвета

private double rez; // результат

private List<double> X, Y;

// Геттер и сеттер для поля a

public double GetSet\_A

{

get { return a; }

set { a = value; }

}

// Геттер и сеттер для поля b

public double GetSet\_B

{

get { return b; }

set { b = value; }

}

// Геттер и сеттер для поля h

public double GetSet\_H

{

get { return h; }

set { h = value; }

}

// Геттер и сеттер для поля mychart

public Chart Mychart

{

get { return mychart; }

set { mychart = value; }

}

// Геттер и сеттер для поля mycolor1

public Color Mycolor

{

get { return mycolor; }

set { mycolor = value; }

}

// Геттер и сеттер для поля area(площадь)

public Color Ploshad

{

get { return ploshad; }

set { ploshad = value; }

}

//Метод sin

public double F1(double x)

{

return Math.Sin(x) / Math.Sqrt(x);

}

//Метод для cos

public double F2(double x)

{

return Math.Cos(x) / Math.Sqrt(x);

}

//Метод для sqrt 1 + sin(2x)

public double F3(double x)

{

return Math.Sqrt(1 + Math.Sin(2 \* x));

}

//Делегат для функции

public delegate double myFunc(double x);

public myFunc f;

//Метод для трапеций

public double Trap()

{

X = new List<double>();

Y = new List<double>();

double sum = (f(a) + f(b)) / 2.0;

X.Add(a);

Y.Add(f(a));

for (double x = a + h; x < b; x += h)

{

X.Add(x);

Y.Add(f(x));

sum += f(x);

}

rez = sum \* h;

return rez;

}

//Метод для левых прямоугольников

public double Left()

{

X = new List<double>();

Y = new List<double>();

double sum = 0.0;

for (double x = a; x < b; x += h)

{

sum += f(x);

X.Add(x);

Y.Add(f(x));

}

rez = sum \* h;

return rez;

}

//Метод для правых прямоугольников

public double Right()

{

X = new List<double>();

Y = new List<double>();

double sum = 0.0;

for (double x = a + h; x <= b; x += h)

{

sum += f(x);

X.Add(x);

Y.Add(f(x));

}

rez = sum \* h;

return rez;

}

//Метод для средних прямоугольников

public double Middle()

{

double sum = 0.0;

X = new List<double>();

Y = new List<double>();

for (double x = a + h / 2; x <= b; x += h)

{

sum += f(x);

X.Add(x);

Y.Add(f(x));

}

rez = sum \* h;

return rez;

}

// Метод для рисования функции

public void DrawF()

{

// Установка свойств графика (например, подписей осей)

mychart.ChartAreas[0].AxisX.Title = "Ось X";

mychart.ChartAreas[0].AxisY.Title = "Ось Y";

// Создание нового объекта класса Series для графика линии

Series series = new Series

{

Name = "График1",

ChartType = SeriesChartType.Spline,

};

// Расчет значения функции для каждой точки

for (double x = a; x <= b; x += h)

{

double y = f(x);

series.Points.AddXY(x, y);

}

//// Добавление созданного объекта Series на график

mychart.Series.Add(series);

mychart.Series["График1"].Color = mycolor;

mychart.Series["График1"].BorderWidth = 2;

// Легенда

mychart.Legends["Legend1"].Title = "Легенда";

mychart.Legends["Legend1"].Docking = Docking.Right;

// Заголовок

mychart.Titles["Title1"].Text = "Графическое изображение функции";

}

// Метод для рисования функции

public void DrawF2()

{

// Установка свойств графика (например, подписей осей)

mychart.ChartAreas[0].AxisX.Title = "Ось X";

mychart.ChartAreas[0].AxisY.Title = "Ось Y";

// Создание нового объекта класса Series для графика линии

Series series1 = new Series

{

Name = "График2",

ChartType = SeriesChartType.Spline,

};

// Расчет значения функции для каждой точки

for (double x = a; x <= b; x += h)

{

double y = f(x);

series1.Points.AddXY(x, y);

}

// Добавление созданного объекта Series на график

mychart.Series.Add(series1);

mychart.Series["График2"].Color = mycolor;

mychart.Series["График2"].BorderWidth = 2;

// Легенда

mychart.Legends["Legend1"].Title = "Легенда";

mychart.Legends["Legend1"].Docking = Docking.Right;

// Заголовок

mychart.Titles["Title1"].Text = "Графическое изображение функции";

}

//Площадь 1

public void DrawPl1(SeriesChartType tp)

{

// Создание нового объекта класса Series для графика площади

Series series2 = new Series

{

Name = "Площадь1",

ChartType = tp

};

// Заполнение значения функции для каждой точки, которые были рассчитаны при интегрировании

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

{

series2.Points.AddXY(X[i], Y[i]);

}

// Добавление созданного объекта Series на график

mychart.Series.Add(series2);

mychart.Series["Площадь1"].BackHatchStyle = ChartHatchStyle.ForwardDiagonal; // Косая штриховка

mychart.Series["Площадь1"].BorderWidth = 0; // Убрать границы

mychart.Series["Площадь1"].Color = ploshad;

mychart.Series["Площадь1"]["PointWidth"] = "1.1"; // Установка ширины столбцов

}

//Площадь 2

public void DrawPl2(SeriesChartType tp)

{

// Создание нового объекта класса Series для графика площади

Series series3 = new Series

{

Name = "Площадь2",

ChartType = tp

};

// Заполнение значения функции для каждой точки, которые были рассчитаны при интегрировании

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

{

series3.Points.AddXY(X[i], Y[i]);

}

// Добавление созданного объекта Series на график

mychart.Series.Add(series3);

mychart.Series["Площадь2"].BackHatchStyle = ChartHatchStyle.ForwardDiagonal; // Косая штриховка

mychart.Series["Площадь2"].BorderWidth = 0; // Убрать границы

mychart.Series["Площадь2"].Color = ploshad;

mychart.Series["Площадь2"]["PointWidth"] = "1.1"; // Установка ширины столбцов

}

// График объединения

public void Unification()

{

// Установка свойств графика (например, подписей осей)

mychart.ChartAreas[0].AxisX.Title = "Ось X";

mychart.ChartAreas[0].AxisY.Title = "Ось Y";

// Создание нового объекта класса Series для графика линии

Series series4 = new Series

{

Name = "Объединение",

ChartType = SeriesChartType.Spline,

};

// Расчет значения функции для каждой точки

for (double x = a; x <= b; x += h)

{

double y = f(x);

series4.Points.AddXY(x, y);

}

// Добавление созданного объекта Series на график

mychart.Series.Add(series4);

mychart.Series["Объединение"].Color = mycolor;

mychart.Series["Объединение"].BorderWidth = 2;

// Легенда

mychart.Legends["Legend1"].Title = "Легенда";

mychart.Legends["Legend1"].Docking = Docking.Right;

// Заголовок

mychart.Titles["Title1"].Text = "Графическое изображение функций";

}

//Площадь общая

public void UfPl(SeriesChartType tp)

{

// Создание нового объекта класса Series для графика площади

Series series5 = new Series

{

Name = "Площадь общая",

ChartType = tp

};

// Заполнение значения функции для каждой точки, которые были рассчитаны при интегрировании

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

{

series5.Points.AddXY(X[i], Y[i]);

}

// Добавление созданного объекта Series на график

mychart.Series.Add(series5);

mychart.Series["Площадь общая"].BackHatchStyle = ChartHatchStyle.ForwardDiagonal; // Косая штриховка

mychart.Series["Площадь общая"].BorderWidth = 0; // Убрать границы

mychart.Series["Площадь общая"].Color = ploshad;

mychart.Series["Площадь общая"]["PointWidth"] = "1.1"; // Установка ширины столбцов

}

// График объединения 2

public void Unification1()

{

// Создание нового объекта класса Series для графика линии

Series series6 = new Series

{

Name = "Объединение1",

ChartType = SeriesChartType.Spline,

};

// Расчет значения функции для каждой точки

for (double x = a; x <= b; x += h)

{

double y = f(x);

series6.Points.AddXY(x, y);

}

// Добавление созданного объекта Series на график

mychart.Series.Add(series6);

mychart.Series["Объединение1"].Color = mycolor;

mychart.Series["Объединение1"].BorderWidth = 2;

mychart.Series["Объединение1"].IsVisibleInLegend = false;

}

//Площадь общая 2

public void UfPl1(SeriesChartType tp)

{

// Создание нового объекта класса Series для графика площади

Series series7 = new Series

{

Name = "Площадь общая 1",

ChartType = tp

};

// Заполнение значения функции для каждой точки, которые были рассчитаны при интегрировании

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

{

series7.Points.AddXY(X[i], Y[i]);

}

// Добавление созданного объекта Series на график

mychart.Series.Add(series7);

mychart.Series["Площадь общая 1"].BackHatchStyle = ChartHatchStyle.ForwardDiagonal; // Косая штриховка

mychart.Series["Площадь общая 1"].BorderWidth = 0; // Убрать границы

mychart.Series["Площадь общая 1"].Color = ploshad;

mychart.Series["Площадь общая 1"]["PointWidth"] = "1.1"; // Установка ширины столбцов

mychart.Series["Площадь общая 1"].IsVisibleInLegend = false;

}

}

}

**Результаты выполнения программы:**

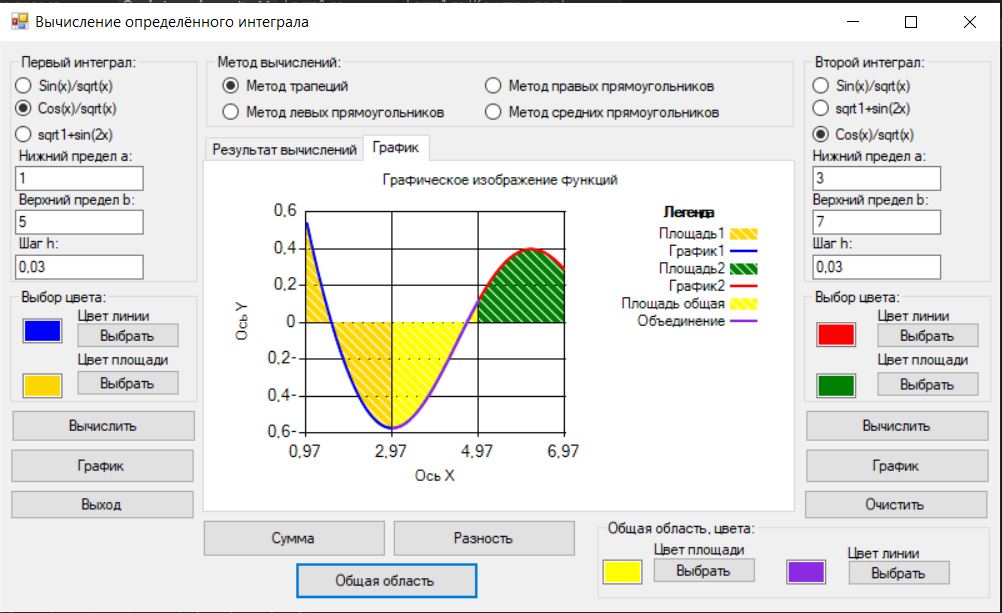


Рисунок 1. Результат работы программы.

**Вывод:** изучила принципы реализации инкапсуляции, секции доступа, свойства, работу с несколькими объектами класса.