Министерство образования Республики Беларусь

Белорусский Национальный Технический Университет

Факультет информационных технологий и робототехники

Кафедра «Программное обеспечение вычислительной техники

и автоматизированных систем»

**Отчёт**

по лабораторной работе № 3

по дисциплине ***«РАЗРАБОТКА ПРИЛОЖЕНИЙ В ВИЗУАЛЬНЫХ СРЕДАХ»***

тема: **«СОЗДАНИЕ ПРИЛОЖЕНИЯ, ПОДДЕРЖИВАЮЩЕГО ПРИНЦИП**

**ИНКАПСУЛЯЦИИ»**

|  |  |  |
| --- | --- | --- |
| Исполнитель: |  | студент группы 10701217  Голованов Павел Андреевич |
| Преподаватель: |  | Гурский Николай Николаевич |

2019 учебный год

**ЛАБОРАТОРНАЯ РАБОТА № 3**

**СОЗДАНИЕ ПРИЛОЖЕНИЯ, ПОДДЕРЖИВАЮЩЕГО ПРИНЦИП**

**ИНКАПСУЛЯЦИИ**

Цель 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. Примерный вид

приложения показан на рис.1.



Разработать приложение в соответствии с л.р. №1 и п.2.

Результаты выполнения задания:

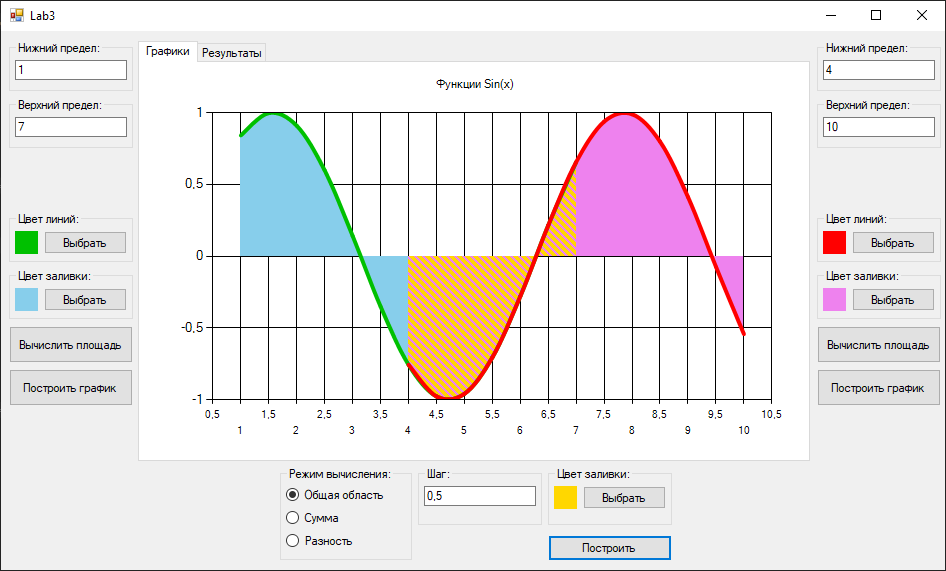


Рисунок 1 – Результат вычисления.

*ПРИЛОЖЕНИЕ A*

Листинг исходных кодов программ

**Файл Lab3Form.cs**

using System;

using System.Windows.Forms;

using lab\_3.Integrals;

namespace lab\_3

{

public partial class Lab3Form : Form

{

//eto hren

SinIntegral Integral1;

SinIntegral func1Area;

SinIntegral Integral2;

SinIntegral func2Area;

SinIntegral mutualArea;

public Lab3Form()

{

InitializeComponent();

}

private void Lab3Form\_Load(object sender, EventArgs e)

{

pictureBox1.BackColor = chart1.Series["Func1"].Color; //серия графика 1й функции

pictureBox4.BackColor = chart1.Series["Func2"].Color; //серия графика 2й функции

pictureBox2.BackColor = chart1.Series["Func1Area"].Color; //серия графика площади 1й функции

pictureBox3.BackColor = chart1.Series["Func2Area"].Color; //серия графика площади 2й фунцкии

pictureBox5.BackColor = chart1.Series["MutualArea1"].Color; //серия графика общей площади

}

#region Выбор цветов

private void SetFunc1Color\_Button\_Click(object sender, EventArgs e)

{

colorDialog1.ShowDialog();

chart1.Series["Func1"].Color = colorDialog1.Color;

pictureBox1.BackColor = colorDialog1.Color;

}

private void SetFunc2Color\_Button\_Click(object sender, EventArgs e)

{

colorDialog1.ShowDialog();

chart1.Series["Func2"].Color = colorDialog1.Color;

pictureBox4.BackColor = colorDialog1.Color;

}

private void SetFunc1AreaColor\_Button\_Click(object sender, EventArgs e)

{

colorDialog1.ShowDialog();

chart1.Series["Func1Area"].Color = colorDialog1.Color;

pictureBox2.BackColor = colorDialog1.Color;

}

private void SetFunc2AreaColor\_Button\_Click(object sender, EventArgs e)

{

colorDialog1.ShowDialog();

chart1.Series["Func2Area"].Color = colorDialog1.Color;

pictureBox3.BackColor = colorDialog1.Color;

}

private void MutualAreaColor\_Button\_Click(object sender, EventArgs e)

{

colorDialog1.ShowDialog();

chart1.Series["MutualArea1"].Color = colorDialog1.Color;

pictureBox5.BackColor = colorDialog1.Color;

}

#endregion

#region Рисование графиков

private void DrawFunc1\_Button\_Click(object sender, EventArgs e)

{

bool flag1 = double.TryParse(textBox1.Text, out double bottom);

bool flag2 = double.TryParse(textBox2.Text, out double top);

bool flag3 = double.TryParse(textBox4.Text, out double step);

if (flag1 && flag2 && flag3)

{

Integral1 = new SinIntegral(ref chart1, "Func1", bottom, top, step);

Integral1.DrawFunc();

func1Area = new SinIntegral(ref chart1, "Func1Area", bottom, top, step);

func1Area.DrawFunc();

}

}

private void DrawFunc2\_Button\_Click(object sender, EventArgs e)

{

bool flag1 = double.TryParse(textBox6.Text, out double bottom);

bool flag2 = double.TryParse(textBox5.Text, out double top);

bool flag3 = double.TryParse(textBox4.Text, out double step);

if (flag1 && flag2 && flag3)

{

Integral2 = new SinIntegral(ref chart1, "Func2", bottom, top, step);

Integral2.DrawFunc();

func2Area = new SinIntegral(ref chart1, "Func2Area", bottom, top, step);

func2Area.DrawFunc();

}

}

private void DrawMutualArea\_Button\_Click(object sender, EventArgs e)

{

if (Integral2.Bottom > Integral1.Bottom)

{

if (Integral2.Top > Integral1.Top)

{

mutualArea = new SinIntegral(ref chart1, "MutualArea1", Integral2.Bottom, Integral1.Top, Integral1.Step);

mutualArea.DrawFunc();

}

}

}

#endregion

private void RadioButton1\_CheckedChanged(object sender, EventArgs e)

{

calcName\_Label.Text = "Общая площадь:";

if (mutualArea != null) { calcValue\_Label.Text = mutualArea.CalcLeftRectanglesSquare().ToString(); }

}

private void RadioButton2\_CheckedChanged(object sender, EventArgs e)

{

calcName\_Label.Text = "Сумма:";

if (Integral1 != null && Integral2 != null) { calcValue\_Label.Text = (Integral1.CalcLeftRectanglesSquare() + Integral2.CalcLeftRectanglesSquare()).ToString(); }

}

private void RadioButton3\_CheckedChanged(object sender, EventArgs e)

{

calcName\_Label.Text = "Разность:";

if (Integral1 != null && Integral2 != null) { calcValue\_Label.Text = (Math.Abs(Integral1.CalcLeftRectanglesSquare()) - Math.Abs(Integral2.CalcLeftRectanglesSquare())).ToString(); }

}

private void CalcF1Area\_Button\_Click(object sender, EventArgs e)

{

if (Integral1 != null) { func1Area\_Label.Text = Integral1.CalcLeftRectanglesSquare().ToString(); }

}

private void CalcF2Area\_Button\_Click(object sender, EventArgs e)

{

if (Integral2 != null) { func2Area\_Label.Text = Integral2.CalcLeftRectanglesSquare().ToString(); }

}

}

}

**Файл FatherOfIntegrals.cs**

using System;

using System.Collections.Generic;

using System.Windows.Forms.DataVisualization.Charting;

namespace lab\_3

{

public class FatherOfIntegrals

{

/// <summary>

/// Переччисление доступных режимов работы

/// </summary>

private enum Mods { USUAL, LEFT\_RECTANGLE\_METHOD, MIDDLE\_RECTANGLE\_METHOD, RIGHT\_RECTANGLE\_METHOD, TRAPEZOID\_METHOD }

private Mods Mod = Mods.USUAL;

#region Выбор режима рисования

/// <summary>

/// Устанавливает отображение функции в обычный режим.

/// </summary>

public void SetModUSUAL() => Mod = Mods.USUAL;

/// <summary>

/// Устанавливает отображение функции в режим левых прямоугольников.

/// </summary>

public void SetModLEFT\_RECTANGLE\_METHOD() => Mod = Mods.LEFT\_RECTANGLE\_METHOD;

/// <summary>

/// Устанавливает отображение функции в режим средних прямоугольников.

/// </summary>

public void SetModMIDDLE\_RECTANGLE\_METHOD() => Mod = Mods.MIDDLE\_RECTANGLE\_METHOD;

/// <summary>

/// Устанавливает отображение функции в режим правых прямоугольников.

/// </summary>

public void SetModRIGHT\_RECTANGLE\_METHOD() => Mod = Mods.RIGHT\_RECTANGLE\_METHOD;

/// <summary>

/// Устанавливает отображение функции в трапеций.

/// </summary>

public void SetModTRAPEZOID\_METHOD() => Mod = Mods.TRAPEZOID\_METHOD;

#endregion

#region Поля класса

private double bottom; //нижний предел интегрированя

private double top; //верхний предел интергирования

private double step; //шаг интегрированя

/// <summary>

/// Поле нижнего предела функции.

/// </summary>

public double Bottom { get; set; }

/// <summary>

/// Поле верхнего предела функции.

/// </summary>

public double Top { get; set; }

/// <summary>

/// Поле шага функции.

/// </summary>

public double Step { get; set; }

private Chart chart; //указатель на редактируемый chart

private string nameOfSeries; //указатель на редактируемый chart

#endregion

public FatherOfIntegrals(ref Chart chartRef, string nameOfSeries)

{

chart = chartRef;

chart.Series[nameOfSeries].Points.Clear();

this.nameOfSeries = nameOfSeries;

}

public FatherOfIntegrals(ref Chart chartRef, string nameOfSeries, double Bottom, double Top, double Step)

{

chart = chartRef;

chart.Series[nameOfSeries].Points.Clear();

this.nameOfSeries = nameOfSeries;

this.Bottom = Bottom;

this.Top = Top;

this.Step = Step;

}

/// <summary>

/// Функция по которой рисуется график

/// </summary>

/// <param name="x"></param>

/// <returns></returns>

public virtual double Function(double x) => x;

#region Вычисление точек

private List<DataPoint> GetFuncDots()

{

List<DataPoint> points = new List<DataPoint>();

switch (Mod)

{

case Mods.USUAL:

{

for (double x = Bottom; x <= Top; x += Step)

{

points.Add(new DataPoint(x, Function(x)));

}

break;

}

case Mods.LEFT\_RECTANGLE\_METHOD:

{

for (double x = Bottom; x <= Top; x += Step)

{

double y = Function(x);

points.Add(new DataPoint(x, y));

if (x + Step <= Top)

{

points.Add(new DataPoint(x + Step, y));

}

}

break;

}

case Mods.MIDDLE\_RECTANGLE\_METHOD:

{

for (double x = Bottom; x <= Top; x += Step)

{

double y = Function(x + Step / 2);

points.Add(new DataPoint(x, y));

if (x + Step <= Top)

{

points.Add(new DataPoint(x + Step, y));

}

}

break;

}

case Mods.RIGHT\_RECTANGLE\_METHOD:

{

for (double x = Bottom; x <= Top; x += Step)

{

double y = Function(x + Step);

points.Add(new DataPoint(x, y));

if (x + Step <= Top)

{

points.Add(new DataPoint(x + Step, y));

}

}

break;

}

case Mods.TRAPEZOID\_METHOD:

{

for (double x = Bottom; x <= Top; x += Step)

{

double y = Function(x);

points.Add(new DataPoint(x, y));

}

break;

}

default:

break;

}

return points;

}

#endregion

#region Рисование функции

/// <summary>

/// Отрисовывает Функцию на заданной серии

/// </summary>

/// <returns></returns>

public bool DrawFunc()

{

if (Bottom < Top && Step != 0) //защита от зацикливания

{

chart.Series[nameOfSeries].Points.Clear();

List<DataPoint> points = GetFuncDots();

foreach (DataPoint point in points)

{

chart.Series[nameOfSeries].Points.AddXY(point.XValue, point.YValues[0]);

}

return true;

}

else

{

return false;

}

}

#endregion

#region Вычисление площади разными методами

/// <summary>

/// Вычисляет площадь функции, вычисленную методом левых прямоугольников.

/// </summary>

/// <returns></returns>

public double CalcLeftRectanglesSquare()

{

double area = 0;

for (double x = Bottom; x <= Top; x += Step)

{

double y = Math.Abs(Function(x));

area += y \* Step;

}

return area;

}

/// <summary>

/// Вычисляет площадь функции, вычисленную методом средних прямоугольников.

/// </summary>

/// <returns></returns>

public double CalcMiddleRectanglesSquare()

{

double area = 0;

for (double x = Bottom; x <= Top; x += Step)

{

double y = Function(x + Step / 2);

area += y \* Step;

}

return area;

}

/// <summary>

/// Вычисляет площадь функции, вычисленную методом правых прямоугольников.

/// </summary>

/// <returns></returns>

public double CalcRightRectanglesSquare()

{

double area = 0;

for (double x = Bottom; x <= Top; x += Step)

{

double y = Function(x + Step);

area += y \* Step;

}

return area;

}

/// <summary>

/// Вычисляет площадь функции, вычисленную методом трапеций.

/// </summary>

/// <returns></returns>

public double CalcTrapezoidSquare()

{

double area = 0;

for (double x = Bottom; x <= Top; x += Step)

{

double y = Function(x);

area += Step \* (Function(x) + Function(x + Step)) / 2;

}

return area;

}

#endregion

}

}

**Файл MutualAreaIntegals.cs**

using System;

using System.Windows.Forms.DataVisualization.Charting;

namespace lab\_3

{

public class MutualAreaIntegals : FatherOfIntegrals

{

public MutualAreaIntegals(Chart chart, string nameOfsereies, double f1bottom, double f1top,

double f2bottom, double f2top, double step) : base(ref chart, nameOfsereies)

{

if (f1bottom > f2bottom)

{

Bottom = f1bottom;

}

else

{

Bottom = f2bottom;

}

if (f1top < f2top)

{

Top = f1top;

}

else

{

Top = f2top;

}

Step = step;

}

public override double Function(double x) => Math.Sin(x);

}

}