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 № 4**

по дисциплине:” Разработка приложений в визуальных средах”

# на тему:*” создание приложения, реализующего принцип полиморфизма”*

Вариант 4

Выполнил**:** студент группы 10701119 Сташкевич А.С.

Приняла**:** Горновская И.Д.

Минск 2020

# Лабораторная работа № 4. Создание приложения, реализующего принцип полиморфизма

Цель работы:

Изучить правила создания дочерних классов.

## Задание:

Разработать дочерний класс вычисления определенного интеграла с различными

Под интегральными функциями. Реализовать возможность арифметических операций над интегралами с выбранными функциями.

**Результаты выполнения:**

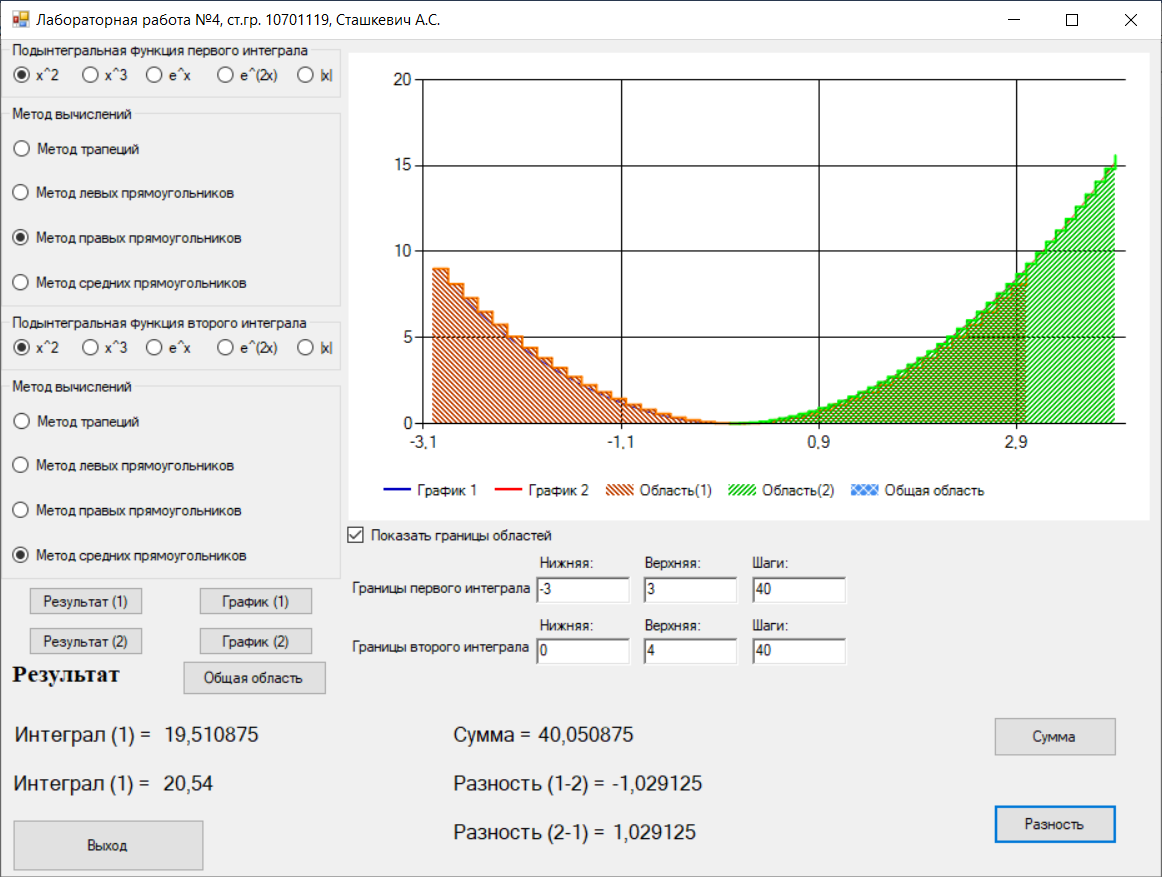


Рисунок 1 – Демонстрация работы программы

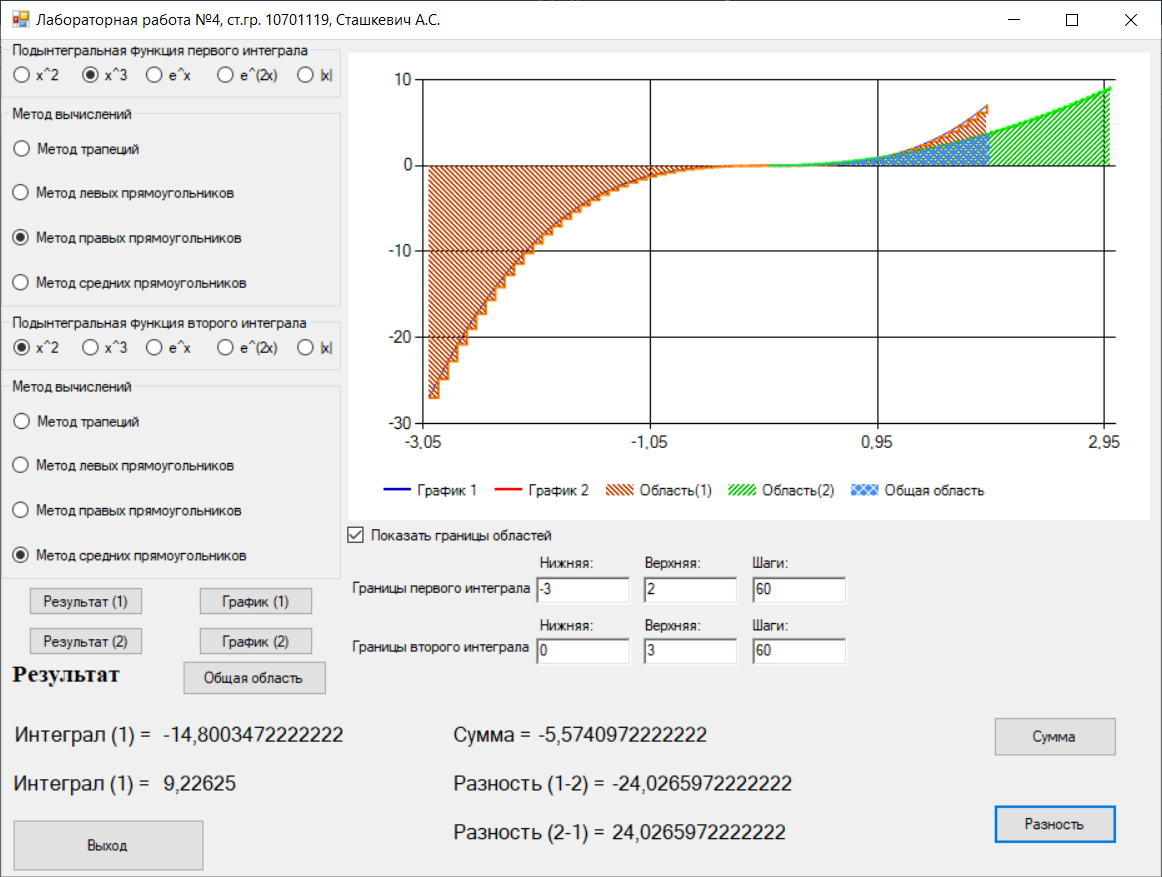


Рисунок 2 – Демонстрация работы программы

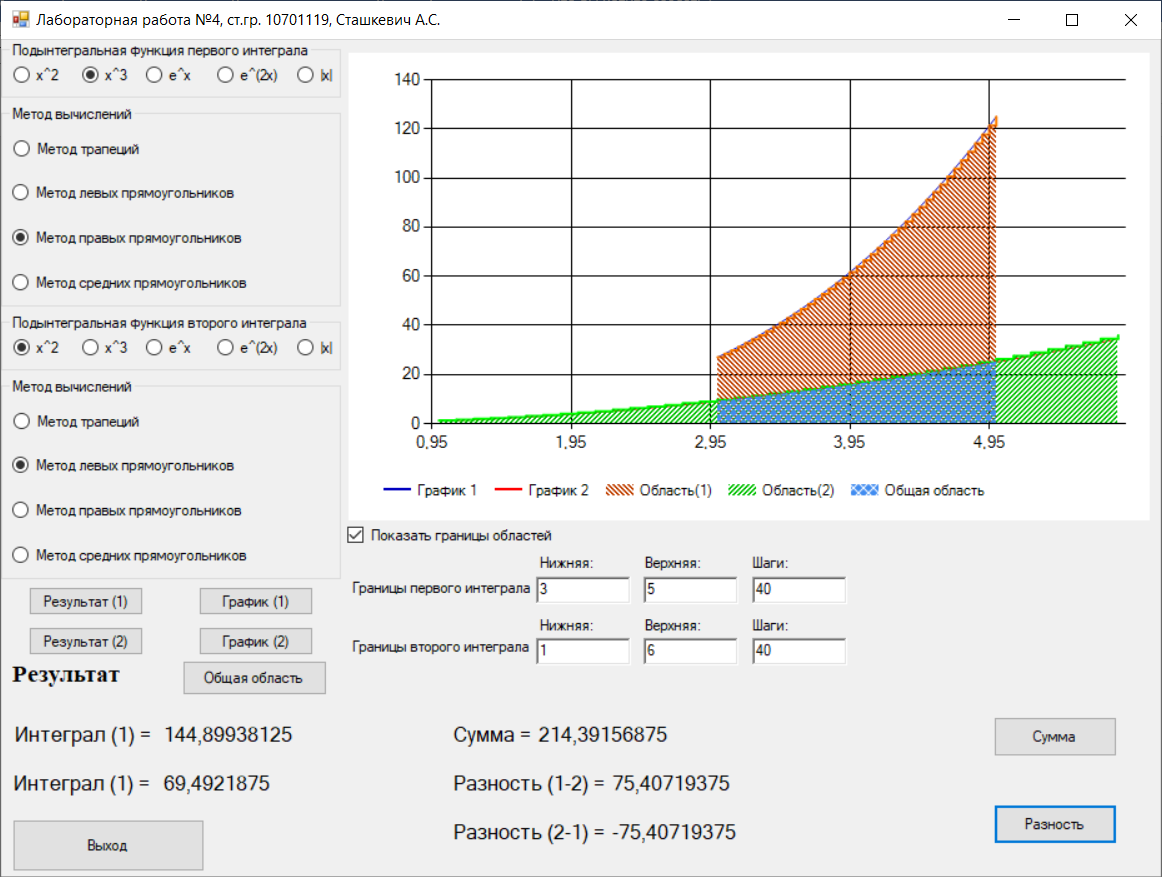


Рисунок 3 – Демонстрация работы программы

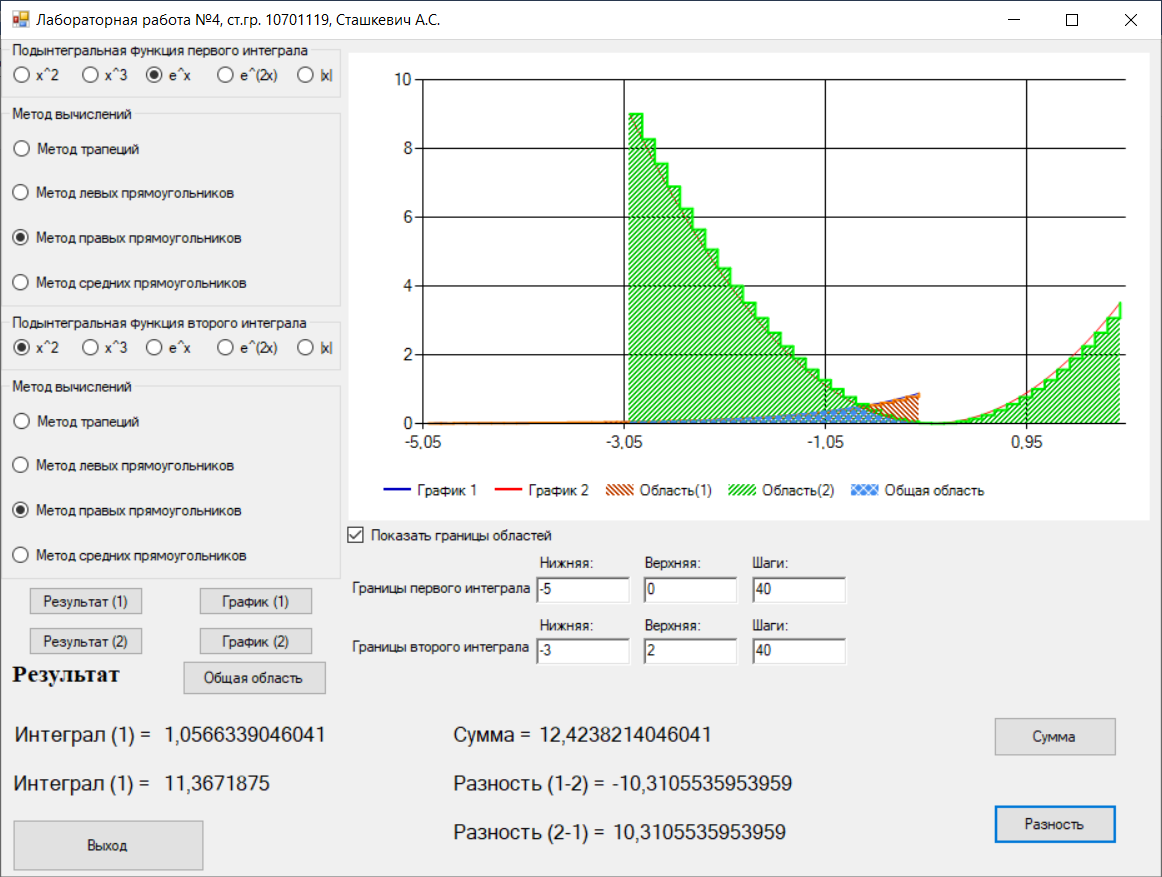


Рисунок 4 – Демонстрация работы программы

**Классы реализующие полиморфизм и их описание:**

class IntegralFunc1: Integral

{

public override double Func(double x)

{

return x \* x;

}

}

class IntegralFunc2: Integral

{

public override double Func(double x)

{

return Math.Pow(x,3);

}

}

class IntegralFunc3: Integral

{

public override double Func(double x)

{

return Math.Pow(Math.E,x);

}

}

class IntegralFunc4: Integral

{

public override double Func(double x)

{

return Math.Pow(Math.E,2\*x);

}

}

class IntegralFunc5: Integral

{

public override double Func(double x)

{

return Math.Abs(x);

}

}

А также реализация метода:

public virtual double Func(double x)

{

return 0;

}

**Методы реализации рисования графика:**

public void PrintGraphicAndOblLeft(System.Windows.Forms.DataVisualization.Charting.Chart chart, int numberGraphic)

{

double promezZnach;

double h = (b - a) / n;

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

chart.Series[numberGraphic+2].Points.Clear();

chart.Series[numberGraphic + 4].Points.Clear();

chart.Series[6].Points.Clear();

double x = a;

while (x < b)

{

promezZnach = Func(x);

double promezZnach2 = Func(x+h);

chart.Series[numberGraphic].Points.AddXY(x, promezZnach);

chart.Series[numberGraphic+2].Points.AddXY(x, promezZnach2);

chart.Series[numberGraphic + 4].Points.AddXY(x, promezZnach2);

if (x + h < b)

{

chart.Series[numberGraphic+2].Points.AddXY(x + h, promezZnach2);

chart.Series[numberGraphic + 4].Points.AddXY(x + h, promezZnach2);

}

x += h;

}

x = aMax;

double znachValue;

values.Clear();

while (x < bMin)

{

znachValue = Func(x);

values.Add(znachValue);

x += 0.05;

}

}

public void PrintGraphicAndOblRight(System.Windows.Forms.DataVisualization.Charting.Chart chart, int numberGraphic)

{

double promezZnach;

double h = (b - a) / n;

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

chart.Series[numberGraphic + 2].Points.Clear();

chart.Series[numberGraphic + 4].Points.Clear();

chart.Series[6].Points.Clear();

double x = a;

while (x < b)

{

promezZnach = Func(x);

chart.Series[numberGraphic].Points.AddXY(x, promezZnach);

chart.Series[numberGraphic+2].Points.AddXY(x, promezZnach);

chart.Series[numberGraphic + 4].Points.AddXY(x, promezZnach);

if (x + h < b)

{

chart.Series[numberGraphic+2].Points.AddXY(x + h, promezZnach);

chart.Series[numberGraphic+4].Points.AddXY(x + h, promezZnach);

}

x += h;

}

x = aMax;

double znachValue;

values.Clear();

while (x < bMin)

{

x += 0.05;

znachValue = Func(x);

values.Add(znachValue);

}

}

public void PrintGraphicAndOblSred(System.Windows.Forms.DataVisualization.Charting.Chart chart, int numberGraphic)

{

double promezZnach;

double h = (b - a) / n;

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

chart.Series[numberGraphic + 2].Points.Clear();

chart.Series[numberGraphic + 4].Points.Clear();

chart.Series[6].Points.Clear();

double x = a;

while (x < b)

{

promezZnach = Func(x);

chart.Series[numberGraphic].Points.AddXY(x, promezZnach);

double promezZnach2 = promezZnach = Func(x+(h/2));

chart.Series[numberGraphic+2].Points.AddXY(x, promezZnach2);

chart.Series[numberGraphic + 4].Points.AddXY(x, promezZnach2);

if (x + h < b)

{

chart.Series[numberGraphic+2].Points.AddXY(x + h, promezZnach2);

chart.Series[numberGraphic+4].Points.AddXY(x + h, promezZnach2);

}

x += h;

}

x = aMax;

double znachValue;

values.Clear();

while (x < bMin)

{

znachValue = Func(x);

values.Add(znachValue);

x += 0.05;

}

}

public void PrintGraphicAndOblTrap(System.Windows.Forms.DataVisualization.Charting.Chart chart, int numberGraphic)

{

double promezZnach;

double h = (b - a) / n;

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

chart.Series[numberGraphic + 2].Points.Clear();

chart.Series[numberGraphic + 4].Points.Clear();

chart.Series[6].Points.Clear();

double x = a;

while (x < b)

{

promezZnach = Func(x);

chart.Series[numberGraphic].Points.AddXY(x, promezZnach);

chart.Series[numberGraphic + 2].Points.AddXY(x, promezZnach);

chart.Series[numberGraphic + 4].Points.AddXY(x, promezZnach);

x += h;

}

x = aMax;

double znachValue;

values.Clear();

while (x < bMin)

{

znachValue = Func(x);

values.Add(znachValue);

x += 0.05;

}

}

**Выделение общей области:**

public void PrintSmezObl(System.Windows.Forms.DataVisualization.Charting.Chart chart, List<double> dopValues1, List<double> dopValues2)

{

double x = aMax;

chart.Series[6].Points.Clear();

int kol1 = dopValues1.Count();

int kol2 = dopValues2.Count();

int minKol;

double h = 0.05;

if(kol1<kol2)

{

minKol = kol1;

}

else

{

minKol = kol2;

}

for (int i = 0; i < minKol && x <= bMin; i++)

{

if (dopValues1[i] <= dopValues2[i])

{

chart.Series[6].Points.AddXY(x, dopValues1[i]);

}

else

{

chart.Series[6].Points.AddXY(x, dopValues2[i]);

}

x += 0.05;

}

}

**Методы вычисления интеграла:**

public double CalculateLeft()

{

double promezZnach, znachF = 0;

double h = (b - a) / n;

double x = a;

while (x < b)

{

promezZnach = Func(x);

znachF += promezZnach \* h;

x += h;

}

return znachF;

}

public double CalculateRight()

{

double promezZnach, znachF = 0;

double x = a;

double h = (b - a) / n;

while (x < b)

{

x += h;

promezZnach = Func(x);

znachF += promezZnach \* h;

}

return znachF;

}

public double CalculateMedium()

{

double promezZnach, znachF = 0;

double x = a;

double h = (b - a) / n;

while (x < b)

{

promezZnach = Func(x);

znachF += promezZnach \* h;

x += h;

}

return znachF;

}

public double CalculateTrap()

{

double promezZnach, znachF = 0;

double x = a;

double h = (b - a) / n;

promezZnach = Func(x);

double promezZnach2 = Func(b);

znachF += h \* (promezZnach + promezZnach2) / 2;

x += h;

while (x < b)

{

promezZnach = Func(x);

znachF += promezZnach \* h;

x += h;

}

return znachF;

}

**Пример доступа к полям класса:**

Integral integral = new Integral();

try

{

integral.A = Double.Parse(enterA2.Text);

integral.B = Double.Parse(enterB2.Text);

integral.N = Double.Parse(enterN2.Text);

}

catch

{

MessageBox.Show("Неккоректные данные!");

enterA2.Text = "0";

enterB2.Text = "0";

enterN2.Text = "0";

}

## Вывод:

Научился разрабатывать приложения, поддерживающие принцип полиморфизма. Так же не забыл про реализацию принципа инкапсуляции. Повторил основы реализации приложения на языке программирования C#.

ПРИЛОЖЕНИЕ А

**Файл Integral.cs**

using System.Collections.Generic;

using System.Linq;

using System.Windows.Forms;

namespace Lab.Rab4

{

public class Integral

{

public virtual double Func(double x)

{

return 0;

}

double a;

double b;

double n;

double aMax;

double bMin;

List<double> values = new List<double>();

public double A

{

set

{

a = value;

}

get

{

return a;

}

}

public double B

{

set

{

b = value;

}

get

{

return b;

}

}

public double N

{

set

{

if (value < 0)

{

MessageBox.Show("Неккоректное число шагов!");

}

n = value;

}

get

{

return n;

}

}

public double AMax

{

set

{

aMax = value;

}

get

{

return aMax;

}

}

public double BMin

{

set

{

bMin = value;

}

get

{

return bMin;

}

}

public List<double> Values

{

set

{

values = value;

}

get

{

return values;

}

}

public double CalculateLeft()

{

double promezZnach, znachF = 0;

double h = (b - a) / n;

double x = a;

while (x < b)

{

promezZnach = Func(x);

znachF += promezZnach \* h;

x += h;

}

return znachF;

}

public double CalculateRight()

{

double promezZnach, znachF = 0;

double x = a;

double h = (b - a) / n;

while (x < b)

{

x += h;

promezZnach = Func(x);

znachF += promezZnach \* h;

}

return znachF;

}

public double CalculateMedium()

{

double promezZnach, znachF = 0;

double x = a;

double h = (b - a) / n;

while (x < b)

{

promezZnach = Func(x);

znachF += promezZnach \* h;

x += h;

}

return znachF;

}

public double CalculateTrap()

{

double promezZnach, znachF = 0;

double x = a;

double h = (b - a) / n;

promezZnach = Func(x);

double promezZnach2 = Func(b);

znachF += h \* (promezZnach + promezZnach2) / 2;

x += h;

while (x < b)

{

promezZnach = Func(x);

znachF += promezZnach \* h;

x += h;

}

return znachF;

}

public void PrintGraphicAndOblLeft(System.Windows.Forms.DataVisualization.Charting.Chart chart, int numberGraphic)

{

double promezZnach;

double h = (b - a) / n;

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

chart.Series[numberGraphic+2].Points.Clear();

chart.Series[numberGraphic + 4].Points.Clear();

chart.Series[6].Points.Clear();

double x = a;

while (x < b)

{

promezZnach = Func(x);

double promezZnach2 = Func(x+h);

chart.Series[numberGraphic].Points.AddXY(x, promezZnach);

chart.Series[numberGraphic+2].Points.AddXY(x, promezZnach2);

chart.Series[numberGraphic + 4].Points.AddXY(x, promezZnach2);

if (x + h < b)

{

chart.Series[numberGraphic+2].Points.AddXY(x + h, promezZnach2);

chart.Series[numberGraphic + 4].Points.AddXY(x + h, promezZnach2);

}

x += h;

}

x = aMax;

double znachValue;

values.Clear();

while (x < bMin)

{

znachValue = Func(x);

values.Add(znachValue);

x += 0.05;

}

}

public void PrintGraphicAndOblRight(System.Windows.Forms.DataVisualization.Charting.Chart chart, int numberGraphic)

{

double promezZnach;

double h = (b - a) / n;

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

chart.Series[numberGraphic + 2].Points.Clear();

chart.Series[numberGraphic + 4].Points.Clear();

chart.Series[6].Points.Clear();

double x = a;

while (x < b)

{

promezZnach = Func(x);

chart.Series[numberGraphic].Points.AddXY(x, promezZnach);

chart.Series[numberGraphic+2].Points.AddXY(x, promezZnach);

chart.Series[numberGraphic + 4].Points.AddXY(x, promezZnach);

if (x + h < b)

{

chart.Series[numberGraphic+2].Points.AddXY(x + h, promezZnach);

chart.Series[numberGraphic+4].Points.AddXY(x + h, promezZnach);

}

x += h;

}

x = aMax;

double znachValue;

values.Clear();

while (x < bMin)

{

x += 0.05;

znachValue = Func(x);

values.Add(znachValue);

}

}

public void PrintGraphicAndOblSred(System.Windows.Forms.DataVisualization.Charting.Chart chart, int numberGraphic)

{

double promezZnach;

double h = (b - a) / n;

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

chart.Series[numberGraphic + 2].Points.Clear();

chart.Series[numberGraphic + 4].Points.Clear();

chart.Series[6].Points.Clear();

double x = a;

while (x < b)

{

promezZnach = Func(x);

chart.Series[numberGraphic].Points.AddXY(x, promezZnach);

double promezZnach2 = promezZnach = Func(x+(h/2));

chart.Series[numberGraphic+2].Points.AddXY(x, promezZnach2);

chart.Series[numberGraphic + 4].Points.AddXY(x, promezZnach2);

if (x + h < b)

{

chart.Series[numberGraphic+2].Points.AddXY(x + h, promezZnach2);

chart.Series[numberGraphic+4].Points.AddXY(x + h, promezZnach2);

}

x += h;

}

x = aMax;

double znachValue;

values.Clear();

while (x < bMin)

{

znachValue = Func(x);

values.Add(znachValue);

x += 0.05;

}

}

public void PrintGraphicAndOblTrap(System.Windows.Forms.DataVisualization.Charting.Chart chart, int numberGraphic)

{

double promezZnach;

double h = (b - a) / n;

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

chart.Series[numberGraphic + 2].Points.Clear();

chart.Series[numberGraphic + 4].Points.Clear();

chart.Series[6].Points.Clear();

double x = a;

while (x < b)

{

promezZnach = Func(x);

chart.Series[numberGraphic].Points.AddXY(x, promezZnach);

chart.Series[numberGraphic + 2].Points.AddXY(x, promezZnach);

chart.Series[numberGraphic + 4].Points.AddXY(x, promezZnach);

x += h;

}

x = aMax;

double znachValue;

values.Clear();

while (x < bMin)

{

znachValue = Func(x);

values.Add(znachValue);

x += 0.05;

}

}

public void PrintSmezObl(System.Windows.Forms.DataVisualization.Charting.Chart chart, List<double> dopValues1, List<double> dopValues2)

{

double x = aMax;

chart.Series[6].Points.Clear();

int kol1 = dopValues1.Count();

int kol2 = dopValues2.Count();

int minKol;

double h = 0.05;

if(kol1<kol2)

{

minKol = kol1;

}

else

{

minKol = kol2;

}

for (int i = 0; i < minKol && x <= bMin; i++)

{

if (dopValues1[i] <= dopValues2[i])

{

chart.Series[6].Points.AddXY(x, dopValues1[i]);

}

else

{

chart.Series[6].Points.AddXY(x, dopValues2[i]);

}

x += 0.05;

}

}

}

}

**Файл IntegralFunc1.cs**

namespace Lab.Rab4

{

class IntegralFunc1: Integral

{

public override double Func(double x)

{

return x \* x;

}

}

}

**Файл IntegralFunc2.cs**

using System;

namespace Lab.Rab4

{

class IntegralFunc2: Integral

{

public override double Func(double x)

{

return Math.Pow(x,3);

}

}

}

**Файл IntegralFunc3.cs**

using System;

namespace Lab.Rab4

{

class IntegralFunc3: Integral

{

public override double Func(double x)

{

return Math.Pow(Math.E,x);

}

}

}

**Файл IntegralFunc4.cs**

using System;

namespace Lab.Rab4

{

class IntegralFunc4: Integral

{

public override double Func(double x)

{

return Math.Pow(Math.E,2\*x);

}

}

}

**Файл IntegralFunc5.cs**

using System;

namespace Lab.Rab4

{

class IntegralFunc5: Integral

{

public override double Func(double x)

{

return Math.Abs(x);

}

}

}

**Файл Form1.cs**

using System;

using System.Collections.Generic;

using System.Windows.Forms;

namespace Lab.Rab4

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

answer1.Text = "0";

answer2.Text = "0";

enterA1.Text = "-3";

enterB1.Text = "3";

enterA2.Text = "0";

enterB2.Text = "5";

enterN1.Text = "500";

enterN2.Text = "500";

firstChoice1.Checked = true;

secondChoice1.Checked = true;

methodTrap1.Checked = true;

methodTrap2.Checked = true;

}

public Integral[] integrals = { new IntegralFunc1(), new IntegralFunc2(), new IntegralFunc3(), new IntegralFunc4(), new IntegralFunc5()};

List<double> dopValues1 = new List<double>();

List<double> dopValues2 = new List<double>();

private void calcRez1\_Click(object sender, EventArgs e)

{

Integral integral = null;

if(firstChoice1.Checked)

{

integral = integrals[0];

}

else if(firstChoice2.Checked)

{

integral = integrals[1];

}

else if(firstChoice3.Checked)

{

integral = integrals[2];

}

else if(firstChoice4.Checked)

{

integral = integrals[3];

}

else if (firstChoice5.Checked)

{

integral = integrals[4];

}

else

{

MessageBox.Show("Ошибка! Нужно выбрать f(x) первого интеграла!");

}

////////////////////////////////////////////////////////////////

DetermineValues(integral);

double result = 0;

if (methodTrap1.Checked)

{

result = integral.CalculateTrap();

}

else if (methodLeft1.Checked)

{

result = integral.CalculateLeft();

}

else if(methodRight1.Checked)

{

result = integral.CalculateRight();

}

else if(methodSred1.Checked)

{

result = integral.CalculateMedium();

}

else

{

MessageBox.Show("Ошибка! Нужно выбрать способ интегрирования первого интеграла!");

}

answer1.Text = result.ToString();

}

private void calcRez2\_Click(object sender, EventArgs e)

{

Integral integral = null;

if (secondChoice1.Checked)

{

integral = integrals[0];

}

else if (secondChoice2.Checked)

{

integral = integrals[1];

}

else if (secondChoice3.Checked)

{

integral = integrals[2];

}

else if (secondChoice4.Checked)

{

integral = integrals[3];

}

else if(secondChoice5.Checked)

{

integral = integrals[4];

}

else

{

MessageBox.Show("Ошибка! Нужно выбрать f(x) второго интеграла!");

}

////////////////////////////////////////////////////////////////

DetermineValues2(integral);

double result = 0;

if (methodTrap2.Checked)

{

result = integral.CalculateTrap();

}

else if (methodLeft2.Checked)

{

result = integral.CalculateLeft();

}

else if (methodRight2.Checked)

{

result = integral.CalculateRight();

}

else if (methodSred2.Checked)

{

result = integral.CalculateMedium();

}

else

{

MessageBox.Show("Ошибка! Нужно выбрать способ интегрирования второго интеграла!");

}

answer2.Text = result.ToString();

}

private void printGraf1\_Click(object sender, EventArgs e)

{

Integral integral = null;

int numberGraphic = 0;

double a1, a2, b1, b2;

a1 = Double.Parse(enterA1.Text);

a2 = Double.Parse(enterA2.Text);

b1 = Double.Parse(enterB1.Text);

b2 = Double.Parse(enterB2.Text);

if (firstChoice1.Checked)

{

integral = integrals[0];

}

else if (firstChoice2.Checked)

{

integral = integrals[1];

}

else if (firstChoice3.Checked)

{

integral = integrals[2];

}

else if (firstChoice4.Checked)

{

integral = integrals[3];

}

else if(firstChoice5.Checked)

{

integral = integrals[4];

}

else

{

MessageBox.Show("Ошибка! Нужно выбрать f(x) первого интеграла!");

}

DetermineValues(integral);

DetermineMinAndMaxValue(integral, a1, a2, b1, b2);

if (methodTrap1.Checked)

{

integral.PrintGraphicAndOblTrap(graphic, numberGraphic);

dopValues1 = integral.Values;

}

else if (methodLeft1.Checked)

{

integral.PrintGraphicAndOblLeft(graphic, numberGraphic);

dopValues1 = integral.Values;

}

else if (methodRight1.Checked)

{

integral.PrintGraphicAndOblRight(graphic, numberGraphic);

dopValues1 = integral.Values;

}

else if (methodSred1.Checked)

{

integral.PrintGraphicAndOblSred(graphic, numberGraphic);

dopValues1 = integral.Values;

}

else

{

MessageBox.Show("Ошибка! Нужно выбрать способ интегрирования первого интеграла!");

}

}

private void printGraf2\_Click(object sender, EventArgs e)

{

Integral integral = null;

int numberGraphic = 1;

double a1, a2, b1, b2;

a1 = Double.Parse(enterA1.Text);

a2 = Double.Parse(enterA2.Text);

b1 = Double.Parse(enterB1.Text);

b2 = Double.Parse(enterB2.Text);

if (secondChoice1.Checked)

{

integral = integrals[0];

}

else if (secondChoice2.Checked)

{

integral = integrals[1];

}

else if (secondChoice3.Checked)

{

integral = integrals[2];

}

else if (secondChoice4.Checked)

{

integral = integrals[3];

}

else if(secondChoice5.Checked)

{

integral = integrals[4];

}

else

{

MessageBox.Show("Ошибка! Нужно выбрать f(x) второго интеграла!");

}

DetermineValues2(integral);

DetermineMinAndMaxValue(integral, a1, a2, b1, b2);

if (methodTrap2.Checked)

{

integral.PrintGraphicAndOblTrap(graphic, numberGraphic);

dopValues2 = integral.Values;

}

else if (methodLeft2.Checked)

{

integral.PrintGraphicAndOblLeft(graphic, numberGraphic);

dopValues2 = integral.Values;

}

else if (methodRight2.Checked)

{

integral.PrintGraphicAndOblRight(graphic, numberGraphic);

dopValues2 = integral.Values;

}

else if (methodSred2.Checked)

{

integral.PrintGraphicAndOblSred(graphic, numberGraphic);

dopValues2 = integral.Values;

}

else

{

MessageBox.Show("Ошибка! Нужно выбрать способ интегрирования второго интеграла!");

}

}

protected void DetermineValues(Integral integral)

{

try

{

integral.A = Double.Parse(enterA1.Text);

integral.B = Double.Parse(enterB1.Text);

integral.N = Double.Parse(enterN1.Text);

}

catch

{

MessageBox.Show("Неккоректные данные!");

enterA1.Text = "0";

enterB1.Text = "0";

enterN1.Text = "0";

}

}

protected void DetermineMinAndMaxValue(Integral integral, double a1, double a2, double b1, double b2)

{

if (a1 > a2)

{

integral.AMax = a1;

}

else

{

integral.AMax = a2;

}

if (b2 < b1)

{

integral.BMin = b2;

}

else

{

integral.BMin = b1;

}

}

protected void DetermineValues2(Integral integral)

{

try

{

integral.A = Double.Parse(enterA2.Text);

integral.B = Double.Parse(enterB2.Text);

integral.N = Double.Parse(enterN2.Text);

}

catch

{

MessageBox.Show("Неккоректные данные!");

enterA2.Text = "0";

enterB2.Text = "0";

enterN2.Text = "0";

}

}

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

{

Close();

}

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

{

double result1 = Double.Parse(answer1.Text);

double result2 = Double.Parse(answer2.Text);

resultSum.Text = (result1 + result2).ToString();

}

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

{

double result1 = Double.Parse(answer1.Text);

double result2 = Double.Parse(answer2.Text);

resultRazn12.Text = (result1 - result2).ToString();

resultRazn21.Text = (result2 - result1).ToString();

}

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

{

Integral integral = new Integral();

double a1, a2, b1, b2;

a1 = Double.Parse(enterA1.Text);

a2 = Double.Parse(enterA2.Text);

b1 = Double.Parse(enterB1.Text);

b2 = Double.Parse(enterB2.Text);

DetermineMinAndMaxValue(integral, a1, a2, b1, b2);

integral.PrintSmezObl(graphic, dopValues1, dopValues2);

}

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

{

if (checkBoxObl.Checked)

{

graphic.Series[2].Enabled = true;

graphic.Series[3].Enabled = true;

}

else

{

graphic.Series[2].Enabled = false;

graphic.Series[3].Enabled = false;

}

}

}

}