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 № 6**

по дисциплине: ”Разработка приложений в визуальных средах ”

# **на тему: *”* Создание DLL*”***

Вариант 1

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

Принял**:** доц. Гурский Н. Н.

Минск 2024

**Лабораторная работа № 6.**

**Цель лабораторной работы:** изучить принципы создания и использования DLL.

**Задание:** разработать библиотеку DLL, содержащую класс вычисления определенного интеграла, экспортируемую функцию и форму выбора цвета, которым будет выведено вычисленное значение интеграла. Головная программа должна содержать вызов формы выбора цвета, задания исходных данных и обращение к DLL для получения результата и вывода его выбранным цветом.

**Текст программы:**

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Windows.Forms.DataVisualization.Charting;

namespace OprIntegral

{

public abstract class OprIntegral

{

public abstract double Calculate();

protected double rez; // результат

protected List<double> X, Y;

// Геттер и сеттер для поля a

public double GetSet\_A { get; set; }

// Геттер и сеттер для поля b

public double GetSet\_B { get; set; }

// Геттер и сеттер для поля h

public double GetSet\_H { get; set; }

// Геттер и сеттер для поля mychart

public Chart Mychart { get; set; }

// Геттер и сеттер для поля mycolor1

public Color Mycolor { get; set; }

// Геттер и сеттер для поля area(площадь)

public Color Ploshad { get; set; }

//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 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 = GetSet\_A; x <= GetSet\_B; x += GetSet\_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 = GetSet\_A; x <= GetSet\_B; x += GetSet\_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 = GetSet\_A; x <= GetSet\_B; x += GetSet\_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 = GetSet\_A; x <= GetSet\_B; x += GetSet\_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;

}

}

//производный класс для трапеций

public class Trap : OprIntegral

{

public override double Calculate()

{

X = new List<double>();

Y = new List<double>();

double sum = (f(GetSet\_A) + f(GetSet\_B)) / 2.0;

X.Add(GetSet\_A);

Y.Add(f(GetSet\_A));

for (double x = GetSet\_A + GetSet\_H; x < GetSet\_B; x += GetSet\_H)

{

X.Add(x);

Y.Add(f(x));

sum += f(x);

}

rez = sum \* GetSet\_H;

return rez;

}

}

//производный класс для левых прямоугольников

public class Left : OprIntegral

{

public override double Calculate()

{

X = new List<double>();

Y = new List<double>();

double sum = 0.0;

for (double x = GetSet\_A; x < GetSet\_B; x += GetSet\_H)

{

sum += f(x);

X.Add(x);

Y.Add(f(x));

}

rez = sum \* GetSet\_H;

return rez;

}

}

//производный класс для правых прямоугольников

public class Right : OprIntegral

{

public override double Calculate()

{

X = new List<double>();

Y = new List<double>();

double sum = 0.0;

for (double x = GetSet\_A + GetSet\_H; x <= GetSet\_B; x += GetSet\_H)

{

sum += f(x);

X.Add(x);

Y.Add(f(x));

}

rez = sum \* GetSet\_H;

return rez;

}

}

//производный класс для средних прямоугольников

public class Middle : OprIntegral

{

public override double Calculate()

{

double sum = 0.0;

X = new List<double>();

Y = new List<double>();

for (double x = GetSet\_A + GetSet\_H / 2; x <= GetSet\_B; x += GetSet\_H)

{

sum += f(x);

X.Add(x);

Y.Add(f(x));

}

rez = sum \* GetSet\_H;

return rez;

}

}

**Результаты выполнения программы:**

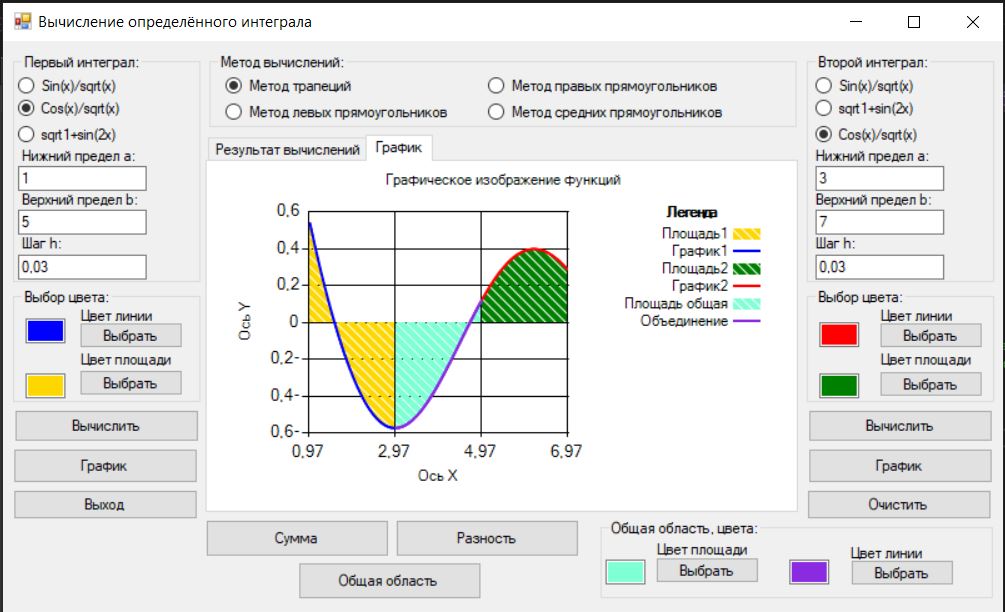


Рисунок 1. Результат работы программы.

**Вывод:** изучил принципы создания и использования DLL.