МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «Гомельский государственный технический

университет имени П. О. Сухого»

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

Лабораторная работа №2

По дисциплине «Объектно-ориентированное программирование»

**«Основы синтаксиса объектно-ориентированного языка программирования»**

Выполнил: студент группы ИТИ-21

Говядкова П. Ю.

Принял: преподаватель

Карабчикова Е. А.

Гомель 2020

**Цель работы:** изучить основы синтаксиса объектно-ориентированного языка программирования, перегрузку операций.

**Задание:**

1. Необходимо разработать тип и выполнить для него перегрузку операций. согласно варианту (Рисунок 1).
2. Тип должен быть размещён в библиотеке классов.
3. Весь од должен быть снабжен элементами документирования.
4. Разработать не менее 15 модульных тестов для тестирования созданного класса.
5. Создать консольное приложение с интерфейсом пользователя.
6. Подключить созданную библиотеку в консольное приложение.
7. Сгенерировать XML-файл с документацией по проекту.
8. Верифицировать разработанное приложение.

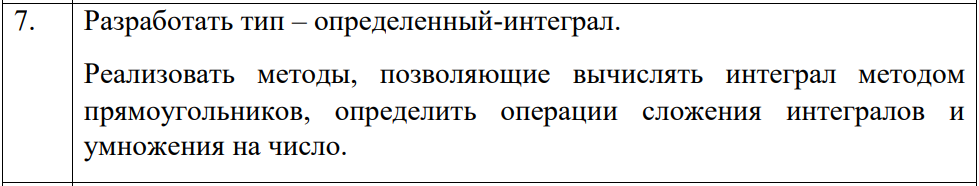


Рисунок 1 – Вариант задания

На рисунке 2 изображена структура решения.

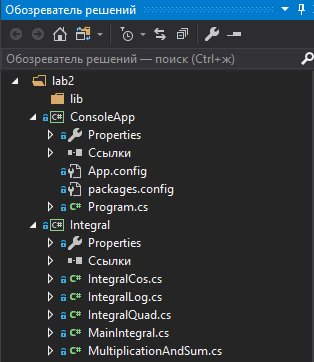


Рисунок 2 – Структура решения

На рисунке 3 изображено начало выполнения программы, вывод меню.

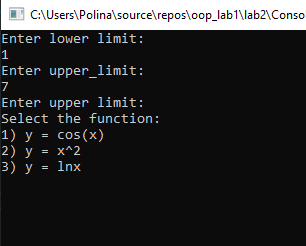


Рисунок 3 – Начало выполнения программы, вывод меню

На рисунке 4 изображено выполнение 1-ого пункта меню: выбор функции для нахождения интеграла.

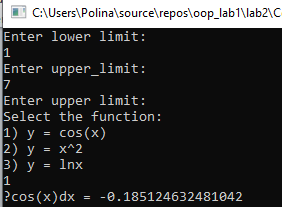


Рисунок 4 – Выполнение первого пункта меню: выбор функции cos(x) для нахождения интеграла

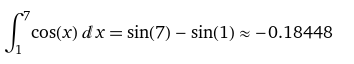


Рисунок 5 – Проверка ответа

На рисунке 6 изображено выполнение следующего пункта задания: ввод цифры для нахождения произведения интеграла на число.

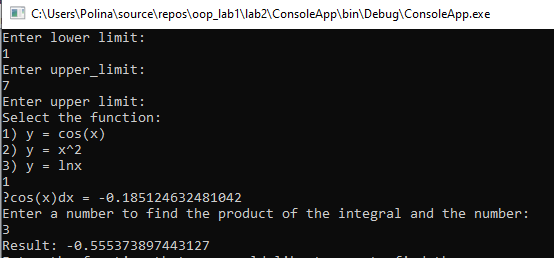


Рисунок 6 – Ввод числа 3 для нахождения произведения интеграла cos(x) на число

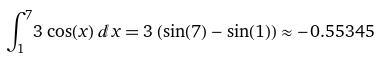


Рисунок 7 – Проверка ответа

На рисунке 8 изображено выполнение следующего пункта задания: выбор следующей функции для нахождения интеграла с тем же верхним и нижним пределом для нахождения суммы интегралов.

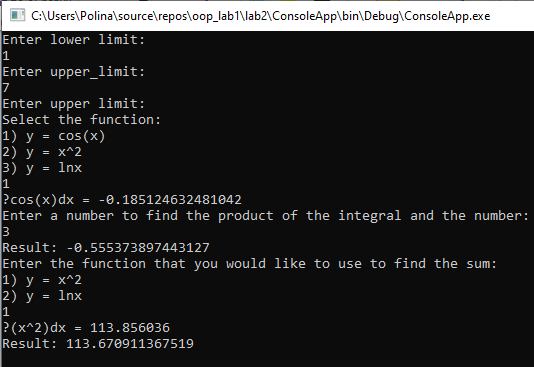


Рисунок 8 – Нахождение суммы интегралов х^2 и cos(x)



Рисунок 9 – Проверка ответа



Рисунок 10 – Проверка ответа

На рисунке 11 изображено выполнение тех же действия для интеграла от функции log10.

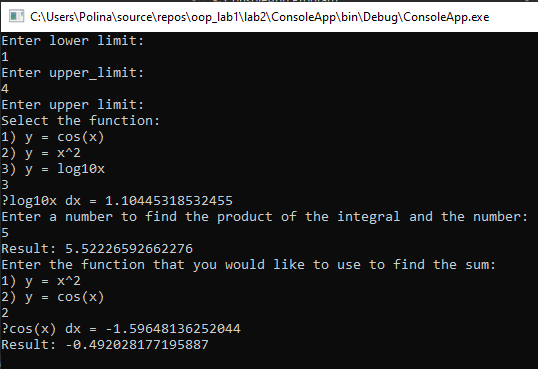


Рисунок 11 – Нахождение суммы, произведения для интеграла log10(x)

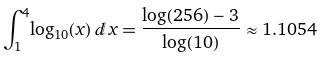


Рисунок 12 – Проверка ответа



Рисунок 13 – Проверка ответа



Рисунок 14 – Проверка ответа



Рисунок 15 – Проверка ответа

На рисунке 16 изображено выполнение вычисления интеграла с одинаковыми пределами.

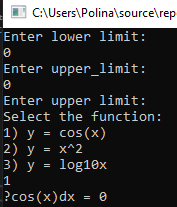


Рисунок 16 – Вычисление функции cos(x)

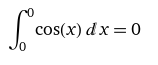


Рисунок 17 – Проверка ответа

На рисунке 18 изображено выполнение вычисления интеграла с отрицательными пределами.

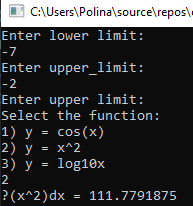


Рисунок 16 – Вычисление функции x^2

Результаты прохождения тестов:

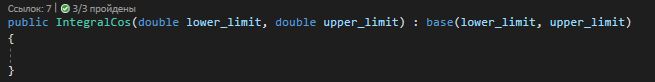


Рисунок 17 – Результат выполнения теста для cos

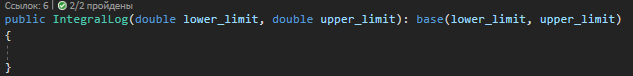


Рисунок 18 – Результат выполнения теста для log

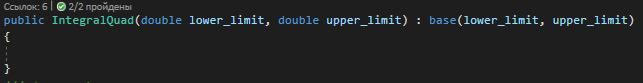


Рисунок 19 – Результат выполнения теста для quad

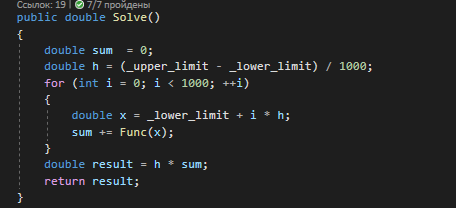


Рисунок 20 – Результат выполнения теста для quad

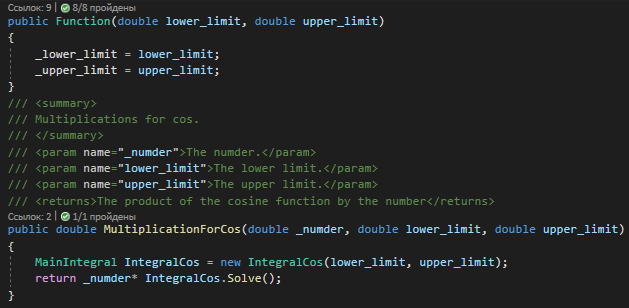


Рисунок 21 – Результат выполнения тестов для вычисления произведения числа на функцию cos

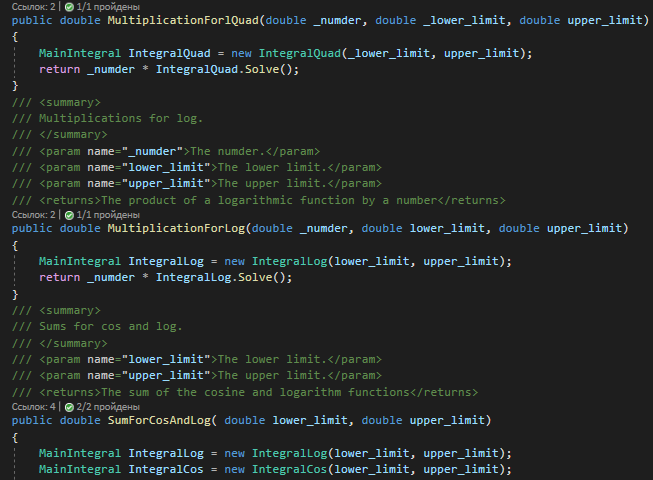


Рисунок 22 – Результат выполнения тестов для вычисления произведения числа на функцию log и суммы функций cos и log

**Вывод:** изучены основы синтаксиса объектно-ориентированного языка программирования. Реализовано наследование для работы с определённым интегралом: вычисление суммы двух интегралом чисел, вычисление произведения интеграла на число.

**ПРИЛОЖЕНИЕ А**

**Листинг программы**

*MainIntegral*:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Integral

{

/// <summary>

///

/// </summary>

public class MainIntegral

{

/// <summary>

/// The lower limit

/// </summary>

double \_lower\_limit;

/// <summary>

/// The upper limit

/// </summary>

double \_upper\_limit;

/// <summary>

/// Initializes a new instance of the <see cref="MainIntegral"/> class.

/// </summary>

/// <param name="lower\_limit">The lower limit.</param>

/// <param name="upper\_limit">The upper limit.</param>

public MainIntegral(double lower\_limit, double upper\_limit)

{

\_lower\_limit = lower\_limit;

\_upper\_limit = upper\_limit;

}

/// <summary>

/// Functions the specified x.

/// </summary>

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

/// <returns> Returns a linear function</returns>

public virtual double Func(double x)

{

return x;

}

/// <summary>

/// Solves this instance.

/// </summary>

/// <returns>Returns the result of evaluating an integral</returns>

public double Solve()

{

double sum = 0;

double h = (\_upper\_limit - \_lower\_limit) / 1000;

for (int i = 0; i < 1000; ++i)

{

double x = \_lower\_limit + i \* h;

sum += Func(x);

}

double result = h \* sum;

return result;

}

}

}

IntegralCos:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Integral

{

/// <summary>

///

/// </summary>

/// <seealso cref="Integral.MainIntegral" />

public class IntegralCos : MainIntegral

{

/// <summary>

/// Initializes a new instance of the <see cref="IntegralCos"/> class.

/// </summary>

/// <param name="lower\_limit">The lower limit.</param>

/// <param name="upper\_limit">The upper limit.</param>

public IntegralCos(double lower\_limit, double upper\_limit) : base(lower\_limit, upper\_limit)

{

}

/// <summary>

/// Functions the specified x.

/// </summary>

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

/// <returns>

/// Returns a cos function

/// </returns>

public override double Func(double x)

{

return Math.Cos(x);

}

}

}

IntegralLog:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Integral

{

/// <summary>

///

/// </summary>

/// <seealso cref="Integral.MainIntegral" />

public class IntegralLog : MainIntegral

{

/// <summary>

/// Initializes a new instance of the <see cref="IntegralLog"/> class.

/// </summary>

/// <param name="lower\_limit">The lower limit.</param>

/// <param name="upper\_limit">The upper limit.</param>

public IntegralLog(double lower\_limit, double upper\_limit): base(lower\_limit, upper\_limit)

{

}

/// <summary>

/// Functions the specified x.

/// </summary>

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

/// <returns>

/// Returns a log function

/// </returns>

public override double Func(double x)

{

return Math.Log10(x);

}

}

}

IntegralQuad:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Integral

{

/// <summary>

///

/// </summary>

/// <seealso cref="Integral.MainIntegral" />

public class IntegralQuad : MainIntegral

{

/// <summary>

/// Initializes a new instance of the <see cref="IntegralQuad"/> class.

/// </summary>

/// <param name="lower\_limit">The lower limit.</param>

/// <param name="upper\_limit">The upper limit.</param>

public IntegralQuad(double lower\_limit, double upper\_limit) : base(lower\_limit, upper\_limit)

{

}

/// <summary>

/// Functions the specified x.

/// </summary>

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

/// <returns>Returns a quadratic function</returns>

public override double Func(double x)

{

return Math.Pow(x,2);

}

}

}

MultiplicationAndSum:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.Remoting.Messaging;

using System.Text;

using System.Threading.Tasks;

namespace Integral

{

/// <summary>

///

/// </summary>

public class Function

{

/// <summary>

/// The x

/// </summary>

private double \_x, \_x1, \_x2, \_lower\_limit, \_upper\_limit, \_a, \_b, \_c, \_number;

/// <summary>

/// Initializes a new instance of the <see cref="Function"/> class.

/// </summary>

/// <param name="lower\_limit">The lower limit.</param>

/// <param name="upper\_limit">The upper limit.</param>

public Function(double lower\_limit, double upper\_limit)

{

\_lower\_limit = lower\_limit;

\_upper\_limit = upper\_limit;

}

/// <summary>

/// Multiplications for cos.

/// </summary>

/// <param name="\_numder">The numder.</param>

/// <param name="lower\_limit">The lower limit.</param>

/// <param name="upper\_limit">The upper limit.</param>

/// <returns>The product of the cosine function by the number</returns>

public double MultiplicationForCos(double \_numder, double lower\_limit, double upper\_limit)

{

MainIntegral IntegralCos = new IntegralCos(lower\_limit, upper\_limit);

return \_numder\* IntegralCos.Solve();

}

/// <summary>

/// Multiplications the forl quad.

/// </summary>

/// <param name="\_numder">The numder.</param>

/// <param name="\_lower\_limit">The lower limit.</param>

/// <param name="upper\_limit">The upper limit.</param>

/// <returns>The product of the quadratic function function by the number</returns>

public double MultiplicationForlQuad(double \_numder, double \_lower\_limit, double upper\_limit)

{

MainIntegral IntegralQuad = new IntegralQuad(\_lower\_limit, upper\_limit);

return \_numder \* IntegralQuad.Solve();

}

/// <summary>

/// Multiplications for log.

/// </summary>

/// <param name="\_numder">The numder.</param>

/// <param name="lower\_limit">The lower limit.</param>

/// <param name="upper\_limit">The upper limit.</param>

/// <returns>The product of a logarithmic function by a number</returns>

public double MultiplicationForLog(double \_numder, double lower\_limit, double upper\_limit)

{

MainIntegral IntegralLog = new IntegralLog(lower\_limit, upper\_limit);

return \_numder \* IntegralLog.Solve();

}

/// <summary>

/// Sums for cos and log.

/// </summary>

/// <param name="lower\_limit">The lower limit.</param>

/// <param name="upper\_limit">The upper limit.</param>

/// <returns>The sum of the cosine and logarithm functions</returns>

public double SumForCosAndLog( double lower\_limit, double upper\_limit)

{

MainIntegral IntegralLog = new IntegralLog(lower\_limit, upper\_limit);

MainIntegral IntegralCos = new IntegralCos(lower\_limit, upper\_limit);

return IntegralCos.Solve() + IntegralLog.Solve();

}

/// <summary>

/// Sums for cos and quad.

/// </summary>

/// <param name="lower\_limit">The lower limit.</param>

/// <param name="upper\_limit">The upper limit.</param>

/// <returns>The sum of the cosine and quadratic functions</returns>

public double SumForCosAndQuad( double lower\_limit, double upper\_limit)

{

MainIntegral IntegralQuad = new IntegralQuad(lower\_limit, upper\_limit);

MainIntegral IntegralCos = new IntegralCos(lower\_limit, upper\_limit);

return IntegralCos.Solve() + IntegralQuad.Solve();

}

/// <summary>

/// Sums for log and quad.

/// </summary>

/// <param name="lower\_limit">The lower limit.</param>

/// <param name="upper\_limit">The upper limit.</param>

/// <returns>The sum of the logarithm and quadratic functions</returns>

public double SumForLogAndQuad(double lower\_limit, double upper\_limit)

{

MainIntegral IntegralQuad = new IntegralQuad(lower\_limit, upper\_limit);

MainIntegral IntegralLog = new IntegralLog(lower\_limit, upper\_limit);

return IntegralLog.Solve() + IntegralQuad.Solve();

}

}

}

Program:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Integral;

namespace ConsoleApp

{

/// <summary>

/// <br />

/// </summary>

class Program

{

/// <summary>

/// Defines the entry point of the application.

/// </summary>

/// <param name="args">The arguments.</param>

static void Main(string[] args)

{

double y, number, lower\_limit, upper\_limit;

Console.WriteLine("Enter lower limit:");

lower\_limit = Convert.ToDouble(Console.ReadLine());

do

{

Console.WriteLine("Enter upper\_limit:");

upper\_limit = Convert.ToDouble(Console.ReadLine());

} while (upper\_limit < lower\_limit);

Console.WriteLine("Enter upper limit:");

Function Integral = new Function(lower\_limit, upper\_limit);

MainIntegral IntegralCos = new IntegralCos(lower\_limit, upper\_limit);

MainIntegral IntegralLog = new IntegralLog(lower\_limit, upper\_limit);

MainIntegral IntegralQuad = new IntegralQuad(lower\_limit, upper\_limit);

var resultCos = IntegralCos.Solve();

var resultLog = IntegralLog.Solve();

var resultQuad = IntegralQuad.Solve();

Console.WriteLine("Select the function:");

Console.WriteLine("1) y = cos(x)");

Console.WriteLine("2) y = x^2");

Console.WriteLine("3) y = log10x");

y = Convert.ToInt64(Console.ReadLine());

switch (y)

{

case 1:

{

Console.WriteLine($"∫cos(x)dx = {resultCos}");

Console.WriteLine("Enter a number to find the product of the integral and the number:");

number = Convert.ToDouble(Console.ReadLine());

Console.WriteLine($"Result: {Integral.MultiplicationForCos(number, lower\_limit, upper\_limit)}");

Console.WriteLine("Enter the function that you would like to use to find the sum:");

Console.WriteLine("1) y = x^2");

Console.WriteLine("2) y = log10x");

y = Convert.ToInt64(Console.ReadLine());

switch (y)

{

case 1:

{

Console.WriteLine($"∫(x^2)dx = {resultQuad}");

Console.WriteLine($"Result: { Integral.SumForCosAndQuad(lower\_limit, upper\_limit)} ");

break;

}

case 2:

{

Console.WriteLine($"∫log10x dx = {resultLog}");

Console.WriteLine($"Result: { Integral.SumForCosAndLog(lower\_limit, upper\_limit)} ");

break;

}

}

break;

}

case 2:

{

Console.WriteLine($"∫(x^2)dx = {resultQuad}");

Console.WriteLine("Enter a number to find the product of the integral and the number:");

number = Convert.ToDouble(Console.ReadLine());

Console.WriteLine($"Result: {Integral.MultiplicationForlQuad(number, lower\_limit, upper\_limit)}");

Console.WriteLine("Enter the function that you would like to use to find the sum:");

Console.WriteLine("1) y = cos(x)");

Console.WriteLine("2) y = log10x");

y = Convert.ToInt64(Console.ReadLine());

switch (y)

{

case 1:

{

Console.WriteLine($"∫cos(x)dx = {resultCos}");

Console.WriteLine($"Result: { Integral.SumForCosAndQuad(lower\_limit, upper\_limit)}");

break;

}

case 2:

{

Console.WriteLine($"∫log10x dx = {resultLog}");

Console.WriteLine($"Result: { Integral.SumForLogAndQuad(lower\_limit, upper\_limit)} ");

break;

}

}

break;

}

case 3:

{

Console.WriteLine($"∫log10x dx = {resultLog}");

Console.WriteLine("Enter a number to find the product of the integral and the number:");

number = Convert.ToDouble(Console.ReadLine());

Console.WriteLine($"Result: {Integral.MultiplicationForLog(number, lower\_limit, upper\_limit)}");

Console.WriteLine("Enter the function that you would like to use to find the sum:");

Console.WriteLine("1) y = x^2");

Console.WriteLine("2) y = cos(x)");

y = Convert.ToInt64(Console.ReadLine());

switch (y)

{

case 1:

{

Console.WriteLine($"∫(x^2)dx = {resultQuad}");

Console.WriteLine($"Result: { Integral.SumForLogAndQuad(lower\_limit, upper\_limit)} ");

break;

}

case 2:

{

Console.WriteLine($"∫cos(x) dx = {resultCos}");

Console.WriteLine($"Result: { Integral.SumForCosAndLog(lower\_limit, upper\_limit)} ");

break;

}

}

break;

}

}

Console.ReadKey();

}

}

}