Буниф Мажда

18-КБ-ПР2

**Отчет КПО Lab 6**

4 Задание

4.1 Численное интегрирование

1) Написать программу (метод класса) для приближенного вычисления значения интеграла ∫ b a f (x)dx с заранее заданной точностью ε > 0 , т.е. вычисления продолжаются до тех пор, пока Δ > ε 2n , и прекращаются в тот момент, когда значение погрешности Δ2n из формулы Рунге становится меньше либо равно ε . (В лабораторной работе считать ε = 0.0001 ).

2) Для вычислений использовать формулу приближенного вычисления в соответствии с номером в журнале по модулю 3 (N%3 + 1): 1) Обобщенная формула прямоугольников. 2) Обобщенная формула Симпсона. 3) Обобщенная формула трапеций.

3) При реализации метода интегрирования использовать делегаты для передачи функций через механизм параметров.

4) Выбрать три функции для использования в качестве параметров из вариантов в разделе 5.1 с номерами: N, N+3, N+5, где N - с номер в журнале. 5) Оценить погрешность вычислений с помощью правила Рунге.

*5.2 Обработка списков*

1) Написать программу для обработки списков (массивов) в соответствии с вариантом в пункте 5.2.

2) При функции обработки map выбрать варианты трех функций из пункта 5.3. с номерами: N, N+4, N+8, где N - с номер в журнале.

3) При функции обработки fold\* выбрать варианты двух функций из пункта 5.4. с номерами: N, N+3, N+5, где N - с номер в журнале.

**Листинг :**

class Class

{

//Functions 4 integrals 5.1 N=1

static double function1(double x) => Exp(Pow(-x, 2) + 0.38) / (2 + Sin(1 / (1.5 + Pow(x, 2))));

static double function2(double x) => Exp(-Tan(0.84 \* x)) / (0.35 + Cos(x));

static double function3(double x) => Log(1 + x) / x;

static int Factorial(int x)

{

int factorial = 1;

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

{

factorial = i == 0 ? 1 : factorial \* i;

}

return factorial;

}

static void Map(double[] array, Func<double, double> func)

{

int j = 0;

foreach (var item in array)

{

array.SetValue(func(item), j++);

}

}

static double Fold(double[] array, Func<double, double, double> func)

{

double item = 0;

foreach (var i in array)

item = func(item, i);

return item;

}

private static double IntegralFirstFun(double a1, double b1, double n)

{

double h1 = (b1 - a1) / n, s1 = 0.0;

for (double x = a1, y = x + h1; x < b1 && y <= b1; x += h1, y += h1)

s1 += (function1(x) + function1(y)) / 2 \* h1;

return s1;

}

private static double IntegralSecondFun(double a2, double b2, double n)

{

double h2 = (b2 - a2) / n, s2 = 0.0;

for (double x = a2, y = x + h2; x < b2 && y <= b2; x += h2, y += h2)

s2 += (function2(x) + function2(y)) / 2 \* h2;

return s2;

}

private static double IntegralThirdFun(double a3, double b3, double n)

{

double h3 = (b3 - a3) / n, s3 = 0.0;

for (double x = a3, y = x + h3; x < b3 && y <= b3; x += h3, y += h3)

s3 += (function3(x) + function3(y)) / 2 \* h3;

return s3;

}

static void Main()

{

//this 4 Map function

double[] array = { 7, 2, 1, 5 };

Map(array, x => x \* x);

foreach (var x in array) Console.Write(x + " ");

Console.WriteLine();

double[] array1 = { 7, 2, 1, 5, -3, -1.5 };

Map(array1, x => Cos(x));

foreach (var x in array1) Console.Write(x + " ");

Console.WriteLine();

double[] array2 = { 7, 2, 1, 5, 0, 1.5 };

Map(array2, x => Factorial((int)(x)));

foreach (var x in array1) Console.Write(x + " ");

Console.WriteLine();

//this 4 Fold Function

//N

double[] arr = { 7, 2, 1, 5, -3 };

var fold = Fold(arr, (x, y) => x + y);

Console.WriteLine(fold);

//N+3

double[] arr1 = { -3.7, -5, -0.5, 9, 5.6 };

var fold1 = Fold(arr1, (x, y) => Min(x, y));

Console.WriteLine(fold1);

double a1 = 0.4, b1 = 1.0, a2 = 0.0, b2 = PI / 2, a3 = 0.1, b3 = 1.0, e = 0.0001;

double n = 50000;

double s1 = IntegralFirstFun(a1, b1, n); //n step

double s2 = IntegralFirstFun(a2, b2, n); //n step

double s3 = IntegralFirstFun(a3, b3, n); //n step

double runge\_func1 = 1 / 3 \* Abs(s1 - IntegralFirstFun(a1, b1, 2 \* n));

double runge\_func2 = 1 / 3 \* Abs(s2 - IntegralSecondFun(a2, b2, 2 \* n));

double runge\_func3 = 1 / 3 \* Abs(s3 - IntegralThirdFun(a3, b3, 2 \* n));

if (runge\_func1 > e)

Console.WriteLine("The small accuracy for the 1st integral is :{0} \nThe accuracy and formula runge is :{1}" + "\t" + e, s1, runge\_func1);

else Console.WriteLine("the accuracy for the 1st integral is {0} \nThe accuracy and formula runge is :{1} " + "\t" + e, s1, runge\_func1);

if (runge\_func2 > e)

Console.WriteLine("The small accuracy for the 2nd integral is :{0} \nThe accuracy and formula runge is :{1}" + "\t" + e, s2, runge\_func2);

else Console.WriteLine("the accuracy for the 2st integral is {0} \nThe accuracy and formula runge is :{1} " + "\t" + e, s2, runge\_func2);

if (runge\_func3 > e)

Console.WriteLine("The small accuracy for the 3rd integral is :{0} \nThe accuracy and formula runge is :{1}" + "\t" + e, s3, runge\_func3);

else Console.WriteLine("the accuracy for the 3rd integral is {0} \nThe accuracy and formula runge is :{1} " + "\t" + e, s3, runge\_func3);

Console.ReadKey();

}

}

}

**Результат Работы**

