МІНІСТЕРСТВО НАУКИ І ОСВІТИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМ. ІГОРЯ СІКОРСЬКОГО»

КАФЕДРА АПЕПС

Звіт до лабораторної роботи №1

Об’єктно-орієнтоване програмування-2

Варіант 9

Виконав:

студент 2 курсу,

групи ТВ-51, ТЕФ

Іванів А.П.

Київ — 2017

**Код програми:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace OOP2\_Lab1

{

class aprox

{

protected double L, R, STEP;

public aprox()

{

L = 0;

R = 0.95;

STEP = 0.05;

}

public aprox(double L,double R,double STEP)

{

this.L = L;

this.R = R;

this.STEP = STEP;

}

static double f(double x)

{

return Math.Log((1 + x));

}

static double taylor(double x, int n)

{ return 2.0 \* (x + (n > 1 ? Math.Pow(x, 3) / 3 : 0) + (n > 2 ? Math.Pow(x, 5) / 5 : 0) + (n > 3 ? Math.Pow(x, 7) / 7 : 0)); }

double error(double a, double b)

{

return 100.0 \* Math.Abs(a - b);

}

public void showTable()

{

Console.ForegroundColor = ConsoleColor.Gray;

Console.WriteLine("╔═══════╦════════════╦═════════════════════════╦═════════════════════════╦═════════════════════════╦═════════════════════════╗");

Console.WriteLine("║ ║ ║ 1 Elem ║ 2 Elem ║ 3 Elem ║ 4 Elem ║");

Console.WriteLine("║ x ║ f(x) ╠════════════╦════════════╬════════════╦════════════╬════════════╦════════════╬════════════╦════════════╣");

Console.WriteLine("║ ║ ║ ~f(x) ║ err, % ║ ~f(x) ║ err, % ║ ~f(x) ║ err, % ║ ~f(x) ║ err, % ║");

for (double x = L + STEP; x < R + 1e-6; x += STEP)

{

double fun = f(x), t1 = taylor(x, 1), t2 = taylor(x, 2), t3 = taylor(x, 3), t4 = taylor(x, 4);

Console.WriteLine("╠═══════╬════════════╬════════════╬════════════╬════════════╬════════════╬════════════╬════════════╬════════════╬════════════╣");

Console.WriteLine("║ {0,5:0.00} ║ {1,10:0.000} ║ {2,10:0.000} ║ {3,10:0.000} ║ {4,10:0.000} ║ {5,10:0.000} ║ {6,10:0.000} ║ {7,10:0.000} ║ {8,10:0.000} ║ {9,10:0.000} ║",

x, fun, t1, error(t1, fun), t2, error(t2, fun), t3, error(t3, fun), t4, error(t4, fun));

}

Console.WriteLine("╚═══════╩════════════╩═════════════════════════╩═════════════════════════╩═════════════════════════╩═════════════════════════╝");

}

static void drawLine(int length, ConsoleColor color)

{

Console.ForegroundColor = color;

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

Console.Write("█");

Console.WriteLine();

}

public void showHistogram()

{

double average1 = 0, average2 = 0, average3 = 0, average4 = 0;

for (double x = L + STEP; x < R + 1e-6; x += STEP)

{

average1 += error(taylor(x, 1), f(x)) / (R - L) \* STEP;

average2 += error(taylor(x, 2), f(x)) / (R - L) \* STEP;

average3 += error(taylor(x, 3), f(x)) / (R - L) \* STEP;

average4 += error(taylor(x, 4), f(x)) / (R - L) \* STEP;

Console.Write(" "); drawLine((int)(error(taylor(x, 1), f(x))), ConsoleColor.Cyan);

Console.Write(" "); drawLine((int)(error(taylor(x, 2), f(x))), ConsoleColor.DarkMagenta);

Console.ForegroundColor = ConsoleColor.Gray;

Console.Write("{0,5:0.00} ", x); drawLine((int)(error(taylor(x, 3), f(x))), ConsoleColor.White);

Console.Write(" "); drawLine((int)(error(taylor(x, 4), f(x))), ConsoleColor.Gray);

}

Console.ForegroundColor = ConsoleColor.Gray;

Console.WriteLine("\nAverage errors:");

Console.WriteLine(" 1: {0,5:0.000}", average1);

Console.WriteLine(" 2: {0,5:0.000}", average2);

Console.WriteLine(" 3: {0,5:0.000}", average3);

Console.WriteLine(" 4: {0,5:0.000}", average4);

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace OOP2\_Lab1

{

class Program

{

static void Main(string[] args)

{

Console.SetWindowPosition(0, 0);

Console.SetWindowSize(150, 50);

Console.SetBufferSize(1800, 250);

aprox apr = new aprox(0, 0.95, 0.1);

apr.showTable();

apr.showHistogram();

aprox apr1 = new aprox(0,0.95,0.05);

apr1.showTable();

apr1.showHistogram();

Console.ReadKey();

}

}

}

Ось приклад використання конструктора ініціалізації в моїй лабораторній роботі.

public aprox()

{

L = 0;

R = 0.95;

STEP = 0.05;

}

public aprox(double L,double R,double STEP)

{

this.L = L;

this.R = R;

this.STEP = STEP;

}

В даному випадку у нас L i R – це межі апроксимування, а STEP – це крок апроксимування.

Далі ми обчислюємо задану функцію і потім обчислюємо ряд Тейлора. Це все ми робимо в конструкторі ініціалізації.

Приклад отримання даних з нашого класу і побудова залежності ( в даному випадку гістограми)

public void showHistogram()

{

double average1 = 0, average2 = 0, average3 = 0, average4 = 0;

for (double x = L + STEP; x < R + 1e-6; x += STEP)

{

average1 += error(taylor(x, 1), f(x)) / (R - L) \* STEP;

average2 += error(taylor(x, 2), f(x)) / (R - L) \* STEP;

average3 += error(taylor(x, 3), f(x)) / (R - L) \* STEP;

average4 += error(taylor(x, 4), f(x)) / (R - L) \* STEP;

Console.Write(" "); drawLine((int)(error(taylor(x, 1), f(x))), ConsoleColor.Cyan);

Console.Write(" "); drawLine((int)(error(taylor(x, 2), f(x))), ConsoleColor.DarkMagenta);

Console.ForegroundColor = ConsoleColor.Gray;

Console.Write("{0,5:0.00} ", x); drawLine((int)(error(taylor(x, 3), f(x))), ConsoleColor.White);

Console.Write(" "); drawLine((int)(error(taylor(x, 4), f(x))), ConsoleColor.Gray);

}

Console.ForegroundColor = ConsoleColor.Gray;

Console.WriteLine("\nAverage errors:");

Console.WriteLine(" 1: {0,5:0.000}", average1);

Console.WriteLine(" 2: {0,5:0.000}", average2);

Console.WriteLine(" 3: {0,5:0.000}", average3);

Console.WriteLine(" 4: {0,5:0.000}", average4);

}

**Теоретичні запитання:**

**1)** **Яку назву має мова прорамування С++, що може обслуговуватися у виконуваючому серидовищі .NET Framework?**

**Ця мова називається Common Intermediate Language (скорочено CIL) - «високорівневий асемблер» віртуальної машини .NET. Проміжна мова, розроблена фірмою «Microsoft» для платформи .NET Framework. JIT-компілятор CIL є частиною CLR (англ. Common language runtime) - загального середовища виконання програм, написаних на мовах .NET. Раніше мова називалася «Microsoft Intermediate Language (MSIL)», проте була перейменована для створення стандарту «ECMA-335».**

**Всі компілятори, що підтримують платформу .NET, повинні транслювати код з мов високого рівня платформи .NET на мову CIL. Зокрема, код на мові CIL генерують все компілятори .NET фірми «Microsoft», що входять в середу розробки «Microsoft Visual Studio» (C #, Managed C ++, Visual Basic .NET, Visual J # .NET).**

**За синтаксисом і мнемонікою мова CIL нагадує мову асемблера. Його можна розглядати як асемблер віртуальної машини .NET. У той же час мова CIL містить деякі досить високорівневі конструкції, що підвищують його рівень у порівнянні з асемблером для будь-якої реально існуючої машини, і писати код безпосередньо на CIL легше, ніж на асемблері для реальних машин. Тому CIL можна розглядати як своєрідний «високорівневий асемблер».**

**Мову CIL також нерідко називають просто IL від англ. intermediate language - буквально «проміжна мова».**

**Синтаксис і мнемоніка мови CIL описуються стандартом «ECMA-335». Специфікація CIL є складовою частиною більш загальної специфікації - специфікації CLI (англ. Common language infrastructure).**

**2) Який специфікатор формату на мові C# визначає грошовий формат виведення значень параметрів функції Write? Наведіть приклади.**

Це описаувач формата "C" або "c"

Приклад програми:

public static void Main(string[] args)

{

double value = 12345.6789;

Console.WriteLine(value.ToString("C", CultureInfo.CurrentCulture));

Console.WriteLine(value.ToString("C3", CultureInfo.CurrentCulture));

Console.WriteLine(value.ToString("C3",

CultureInfo.CreateSpecificCulture("da-DK"))); } } }

Результат отримаєм:

// $12,345.68

// $12,345.679

// kr 12.345,679