Міністерство освіти і науки України

Державний вищий навчальний заклад

«Приазовський державний технічний університет»

Факультет інформаційних технологій

Кафедра автоматизації та комп'ютерних технологій

**Звіт**

**з лабораторної роботи №5**

**ЗАСОБИ ВВЕДЕННЯ-ВИВЕДЕННЯ В МОВІ C#**

дисципліни «Комп’ютерні технології та програмування»

|  |  |
| --- | --- |
| Виконав студент групи МА-24 | Деркач Андрій Сергійович |
| Перевірив доцент | Щербаков Сергій Володимирович |

ЛАБОРАТОРНА РОБОТА №5

Засоби введення-виведення

Мета роботи: вивчити роботу функцій та прийоми реалізації форматованого введення-виведення в середовищі програмування C#.

Завдання на роботу

1. У вигляді, аналогічному таблицям тригонометричних функцій (напр., таблиць Брадіса), вивести (із заданою кількістю знаків) значення функції в заданому діапазоні та з заданим кроком. По горизонталі крок повинен змінюватися на величину (0.02 ... 0.1) · h. Видати результат у вигляді таблиці, для побудови якої використати графічні символи кодової сторінки операційної системи (наприклад, символи ASCII). Вид функції вибрати з лабораторної роботи No3, з іншого джерела за вказівкою викладача, або самостійно.. Варіант завдання №2 (формула 1.1)

Y=

(1.1)

Хід виконання роботи

Завдання 1

* блок-схема виконаного завдання наведена у додатку А.
* лістинг програми для обчислення математичних виразів у вигляді таблиці тригонометричних функцій наведена у додатку B

Додаток А

Изображение выглядит как диаграмма, текст, чек

Автоматически созданное описание

Изображение выглядит как зарисовка, черно-белый, диаграмма, чек

Автоматически созданное описание

Рисунок А.1 – блок-схема

Додаток B

using System;

using System.Text;

using static System.Math;

namespace Lab5Part1

{

internal class Program

{

static void Main()

{

Console.OutputEncoding = System.Text.Encoding.UTF8;

double

start = -5,

end = 5,

rowStep = 0.5;

int colls = 10,

visibleRow = 5;

double collStep = rowStep / colls;

printHeader();

int rowCounter = 0;

for (double row = start; row <= end; row += rowStep)

{

rowCounter++;

decorationFont(true);

Console.Write("──────────");

decorationFont(false);

Console.WriteLine(buildBorder("middle", colls));

decorationFont(true);

Console.Write($"{row,10:F}");

decorationFont(false);

double

y, x, numerator, denominator;

for (int col = 0; col < colls; col++)

{

x = row + col \* collStep;

numerator = Math.Log(6.0 \* x, 3.0) - Math.Exp(x);

denominator = x \* x - 0.8;

y = (numerator / denominator) + Math.Pow(x, 9.0);

y = Math.Sign((int)y) \* Math.Pow(Math.Abs(y), 1.0 / 3.0);

y = x;

if (Math.Abs(y) > 999.00)

{

Console.Write($"│{y,10:E2}");

}

else

{

Console.Write($"│{y,10:F}");

}

}

Console.WriteLine("│");

if (rowCounter >= visibleRow)

{

Console.WriteLine("Натисніть Enter для продовження...");

Console.ReadLine();

Console.Clear();

printHeader();

rowCounter = 0;

}

}

Console.WriteLine("  " + buildBorder("bottom", colls));

Console.Read();

void printHeader()

{

decorationFont(true);

Console.WriteLine("  " + buildBorder("top", colls));

Console.Write("  ");

for (int col = 0; col < colls; col++)

{

Console.Write($"│{col \* collStep,10:F}");

}

Console.WriteLine("│");

}

void decorationFont(bool value)

{

if (value)

{

Console.ForegroundColor = ConsoleColor.Black;

Console.BackgroundColor = ConsoleColor.Green;

}

else

{

Console.ForegroundColor = ConsoleColor.White;

Console.BackgroundColor = ConsoleColor.Black;

}

}

String buildBorder(String borderType, int cellQuantity, int cellSize = 10)

{

String[,] ch = {

{ "┌", "┬", "┐" },

{ "├", "┼", "┤" },

{ "└", "┴", "┘" }

};

byte borderTypeIndex = 0;

if (borderType == "top") borderTypeIndex = 0;

else if (borderType == "middle") borderTypeIndex = 1;

else if (borderType == "bottom") borderTypeIndex = 2;

StringBuilder str = new StringBuilder("");

str.Append("");

for (int cell = 0; cell < cellQuantity; cell++)

{

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

{

str.Append("─");

}

str.Append(ch[borderTypeIndex, 1]);

}

str.Insert(0, ch[borderTypeIndex, 0]);

str.Remove(str.Length - 1, 1);

str.Insert(str.Length, ch[borderTypeIndex, 2]);

return str.ToString();

}

}

}

}