**Виконав: Мельник Б. В.; КН-922в**

**Лабораторна робота №1**

**Тема:** Використання базових типів, консольного введення / виведення та операторів мови C#.

**Мета роботи:** набути навички в написанні програм з використання базових типів, операторів та засобів введення / виведення.

**Індивідуальні завдання**

**Варіант №12**

**Завдання 1**

Використовуючи завдання власного варіанта до лабораторної роботи 2 з електронних методичних вказівок з основ програмування та алгоритмічних мов створити проект з реалізації цих завдань засобами мови C#.***Текст програми:***

using System; // Підключення простору імен System для використання базових функцій

class Task1 // Оголошення класу Task1

{

static void Main() // Основний метод програми, точка входу

{

// Оголошення масивів для зберігання даних про транспорт

string[] transportType = new string[3]; // Масив для типу транспорту (трамвай, тролейбус, автобус)

string[] routeNumber = new string[3]; // Масив для номерів маршрутів

double[] routeLength = new double[3]; // Масив для довжини маршрутів (в кілометрах)

int[] travelTime = new int[3]; // Масив для часу в дорозі (в хвилинах)

// Цикл для введення даних від користувача

for (int i = 0; i < 3; i++) // Виконується 3 рази, для кожного транспорту

{

Console.WriteLine($"Введіть дані для транспорту {i + 1}:"); // Запит на введення даних для кожного транспорту

Console.Write("Вид транспорту (Тр, Тл, А): "); // Запит на введення типу транспорту

transportType[i] = Console.ReadLine(); // Зчитування введеного типу транспорту

Console.Write("Номер маршруту: "); // Запит на введення номера маршруту

routeNumber[i] = Console.ReadLine(); // Зчитування номера маршруту

Console.Write("Протяжність маршруту (км): "); // Запит на введення довжини маршруту

routeLength[i] = Convert.ToDouble(Console.ReadLine()); // Перетворення введеного значення на число типу double

Console.Write("Час в дорозі (хвилин): "); // Запит на введення часу в дорозі

travelTime[i] = Convert.ToInt32(Console.ReadLine()); // Перетворення введеного значення на ціле число

Console.WriteLine(); // Виведення порожнього рядка для візуального розділення введень

}

// Виведення заголовка таблиці

Console.WriteLine("-------------------------------------------------------------");

Console.WriteLine("| Відомості суспільного транспорту |");

Console.WriteLine("-------------------------------------------------------------");

// Виведення заголовків стовпців таблиці

Console.WriteLine("| Вид транспорту | № маршруту | Протяжність (км) | Час (хв) |");

Console.WriteLine("|----------------|------------|------------------|----------|");

// Цикл для виведення даних про кожен транспорт

for (int i = 0; i < transportType.Length; i++) // Проходить по всіх елементах масивів

{

// Виведення даних у форматі таблиці

Console.WriteLine($"| {transportType[i],-14} | {routeNumber[i],-10} | {routeLength[i],-16} | {travelTime[i],-8} |");

}

// Виведення примітки з роз'ясненнями про типи транспорту

Console.WriteLine("-------------------------------------------------------------");

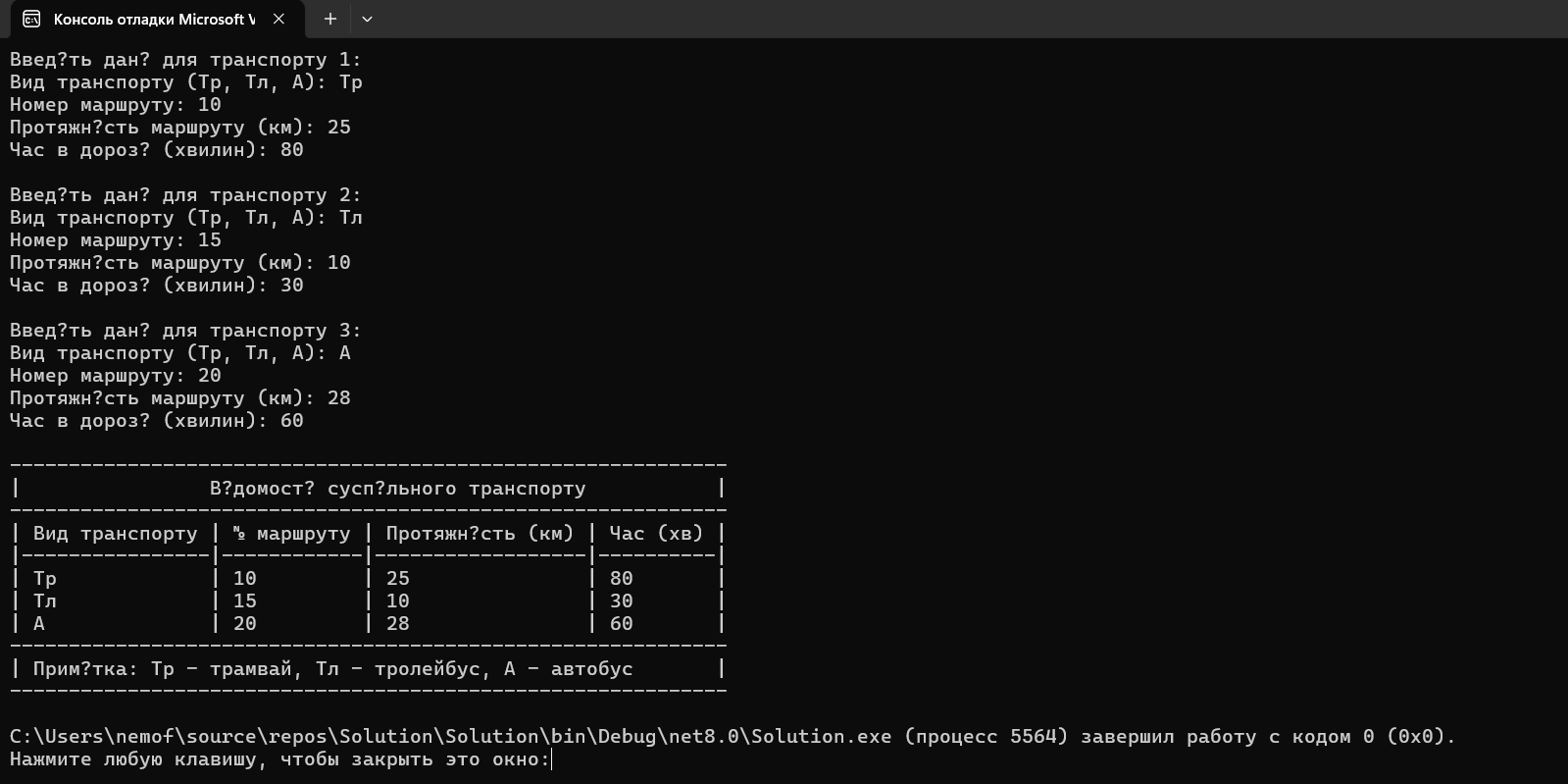
Console.WriteLine("| Примітка: Тр – трамвай, Тл – тролейбус, А – автобус |");

Console.WriteLine("-------------------------------------------------------------");

}

}

***Результат роботи програми:***



**Алгоритм побудови коду**

1. **Імпорт бібліотек**: Підключення простору імен System, що дозволяє використовувати базові функції для введення та виведення даних.
2. **Оголошення класу**: Створення класу Task1 для реалізації програми.
3. **Основний метод**:
   * **Оголошення масивів**: Створення масивів для зберігання типу транспорту, номерів маршрутів, довжини маршрутів та часу в дорозі.
   * **Введення даних**: Цикл, що запитує користувача про деталі кожного транспортного засобу, зчитує ці дані та зберігає їх у відповідних масивах.
   * **Виведення таблиці**: Виведення заголовків таблиці та даних про транспорти у вигляді відформатованої таблиці.
   * **Примітка**: Виведення примітки з поясненням типів транспорту.

**Функціональність**

1. **Збір даних**: Користувач вводить інформацію про три різні транспортні засоби, включаючи тип транспорту, номер маршруту, довжину маршруту в кілометрах і час в дорозі в хвилинах.
2. **Відображення інформації**: Програма форматує введені дані у вигляді таблиці для легкого перегляду. В таблиці відображаються тип транспорту, номер маршруту, довжина маршруту та час в дорозі.
3. **Примітка**: В кінці виводиться примітка, яка пояснює скорочення для типів транспорту.

**Висновок**

Програма дозволяє користувачеві ввести та переглянути інформацію про маршрути транспорту у зрозумілому та організованому вигляді. Завдяки використанню масивів і циклів для збирання та відображення даних, програма демонструє основи роботи з масивами та форматованим виведенням у мові C#. Це базове завдання допомагає закріпити знання про структурування даних та їх виведення.

**Завдання 2\***

Виконати минуле завдання, покращивши зовнішній вигляд таблиці із вихідними результатами за допомогою символів псевдографіки (┐, ┌, ┘, └, ├, ┤, ┬, ┴ та ін.) та налаштування кольорів консолі, наприклад, числа відобразити одним кольором, текст – іншим, таблицю – іншим від попередніх.

***Текст програми:***

using System;

using System.Text;

class Task2

{

static void Main()

{

// Встановлення кодування UTF-8 для правильної роботи з псевдографікою

Console.OutputEncoding = Encoding.UTF8;

// Оголошення масивів для зберігання даних про транспорт

string[] transportType = new string[3]; // Масив для виду транспорту

string[] routeNumber = new string[3]; // Масив для номерів маршрутів

double[] routeLength = new double[3]; // Масив для довжини маршруту

int[] travelTime = new int[3]; // Масив для часу в дорозі

// Введення даних користувачем

for (int i = 0; i < 3; i++)

{

// Запит даних для кожного виду транспорту

Console.WriteLine($"Введіть дані для транспорту {i + 1}:");

// Запит виду транспорту

Console.Write("Вид транспорту (Тр, Тл, А): ");

transportType[i] = Console.ReadLine();

// Запит номера маршруту

Console.Write("Номер маршруту: ");

routeNumber[i] = Console.ReadLine();

// Запит протяжності маршруту в кілометрах

Console.Write("Протяжність маршруту (км): ");

routeLength[i] = Convert.ToDouble(Console.ReadLine());

// Запит часу в дорозі в хвилинах

Console.Write("Час в дорозі (хвилин): ");

travelTime[i] = Convert.ToInt32(Console.ReadLine());

// Порожній рядок для відділення введень

Console.WriteLine();

}

// Встановлення кольорів для виведення таблиці

Console.ForegroundColor = ConsoleColor.Cyan;

// Виведення верхньої частини таблиці з псевдографікою

Console.WriteLine("┌───────────────────────────────────────────────────────────┐");

Console.WriteLine("│ Відомості суспільного транспорту │");

Console.WriteLine("├────────────────┬────────────┬──────────────────┬──────────┤");

// Виведення заголовків колонок таблиці

Console.WriteLine("│ Вид транспорту │ № маршруту │ Протяжність (км) │ Час (хв) │");

Console.WriteLine("├────────────────┼────────────┼──────────────────┼──────────┤");

// Цикл для виведення даних по кожному виду транспорту

for (int i = 0; i < transportType.Length; i++)

{

// Виведення даних з різними кольорами для кожної колонки

Console.Write("│ ");

Console.ForegroundColor = ConsoleColor.Yellow; // Жовтий для виду транспорту

Console.Write($"{transportType[i],-14}");

Console.ForegroundColor = ConsoleColor.Cyan; // Блакитний для розділення

Console.Write(" │ ");

Console.ForegroundColor = ConsoleColor.Green; // Зелений для номера маршруту

Console.Write($"{routeNumber[i],-10}");

Console.ForegroundColor = ConsoleColor.Cyan; // Блакитний для розділення

Console.Write(" │ ");

Console.ForegroundColor = ConsoleColor.Magenta;// Пурпурний для протяжності

Console.Write($"{routeLength[i],-16}");

Console.ForegroundColor = ConsoleColor.Cyan; // Блакитний для розділення

Console.Write(" │ ");

Console.ForegroundColor = ConsoleColor.Red; // Червоний для часу в дорозі

Console.Write($"{travelTime[i],-8}");

Console.ForegroundColor = ConsoleColor.Cyan; // Блакитний для завершення рядка

Console.WriteLine(" │");

}

// Завершення таблиці та примітки

Console.WriteLine("├────────────────┴────────────┴──────────────────┴──────────┤");

Console.WriteLine("│ Примітка: Тр – трамвай, Тл – тролейбус, А – автобус │");

Console.WriteLine("└───────────────────────────────────────────────────────────┘");

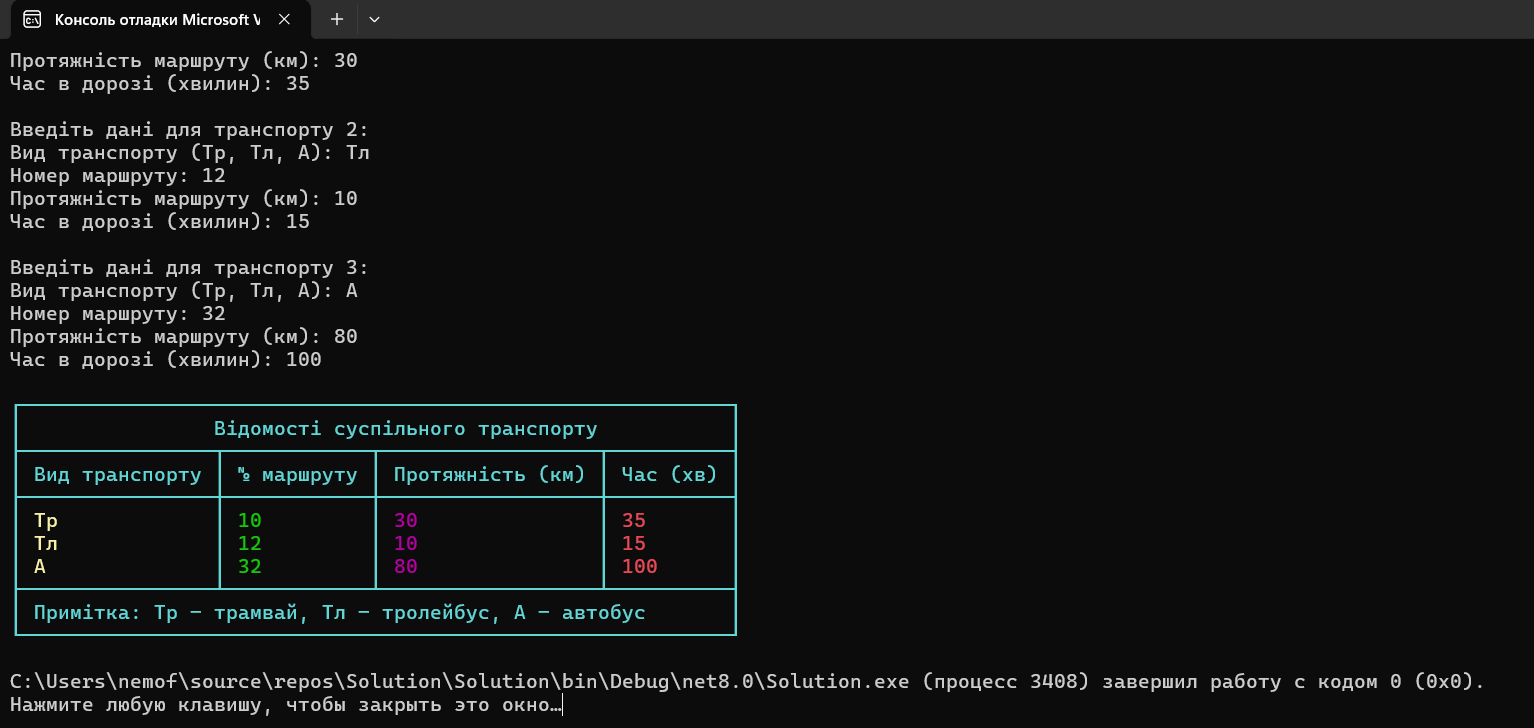
// Скидання кольорів консолі до значень за замовчуванням

Console.ResetColor();

}

}

***Результат роботи програми:***



**1. Алгоритм побудови коду**

1. **Ініціалізація консолі**:
   * Встановлюється кодування UTF-8 для підтримки псевдографіки.
2. **Оголошення масивів**:
   * Створюються масиви для зберігання даних про транспорт:
     + transportType — типи транспорту.
     + routeNumber — номери маршрутів.
     + routeLength — довжини маршрутів.
     + travelTime — час в дорозі.
3. **Введення даних**:
   * Виконується цикл для введення даних про кожний вид транспорту:
     + Тип транспорту.
     + Номер маршруту.
     + Протяжність маршруту в км.
     + Час в дорозі в хвилинах.
4. **Встановлення кольорів**:
   * Налаштовуються кольори консолі для різних частин таблиці:
     + Заголовки.
     + Дані про транспорт.
5. **Виведення таблиці**:
   * Форматується таблиця за допомогою псевдографічних символів.
   * Виводяться заголовки колонок.
   * Виводяться дані з використанням кольорів для кожної колонки.
   * Завершується таблиця примітками.
6. **Скидання кольорів**:
   * Після виведення таблиці кольори консолі скидаються до значень за замовчуванням.

**2. Функціональність**

* **Введення даних**: Користувач вводить дані для трьох видів транспорту, включаючи тип, номер маршруту, протяжність та час в дорозі.
* **Виведення таблиці**: Таблиця виводиться з використанням псевдографічних символів для створення чіткої та зрозумілої структури. Кожна колонка має свій колір, що полегшує сприйняття інформації.

**3. Висновок**

Програма успішно реалізує введення та відображення даних про транспорт у вигляді таблиці з псевдографічними символами та кольоровим форматуванням. Це покращує візуальне сприйняття інформації та робить дані легшими для читання і аналізу. Встановлені кольори допомагають чітко розділити різні типи даних, що підвищує зручність користування консоллю.

**Завдання 3**

Використовуючи завдання власного варіанта до лабораторної роботи 3 з електронних методичних вказівок з основ програмування та алгоритмічних мов створити проект з реалізації цих завдань засобами мови C#.

***Текст програми:***

using System;

class Task3

{

static void Main()

{

// Виведення повідомлення та введення значення x з клавіатури

Console.Write("Введіть x: ");

// Читання рядка, конвертація у тип double і присвоєння значення змінній x

double x = Convert.ToDouble(Console.ReadLine());

// Виведення повідомлення та введення значення y з клавіатури

Console.Write("Введіть y: ");

// Читання рядка, конвертація у тип double і присвоєння значення змінній y

double y = Convert.ToDouble(Console.ReadLine());

// Ініціалізація змінних a та b значеннями 12.5 та 1.3 відповідно

double a = 12.5, b = 1.3;

// Обчислення значення t1 за формулою з використанням логарифма та степеневих функцій

double t1 = (1 / Math.Pow(b, 3)) \* (Math.Pow(a, 2) \* Math.Log(y / x) + (2 \* a \* x / y) + (Math.Pow(y, 2) / (2 \* Math.Pow(x, 2))));

// Обчислення проміжного значення ax, яке дорівнює добутку a і x

double ax = a \* x;

// Обчислення значення t2 за іншою формулою з використанням синусоїдальних функцій

double t2 = (1 / (2 \* Math.Sqrt(2) \* a)) + ((3 \* Math.Pow(Math.Sin(ax), 2) - 1) / (Math.Pow(Math.Sin(ax), 2) - 1));

// Виведення результатів t1 та t2 на екран

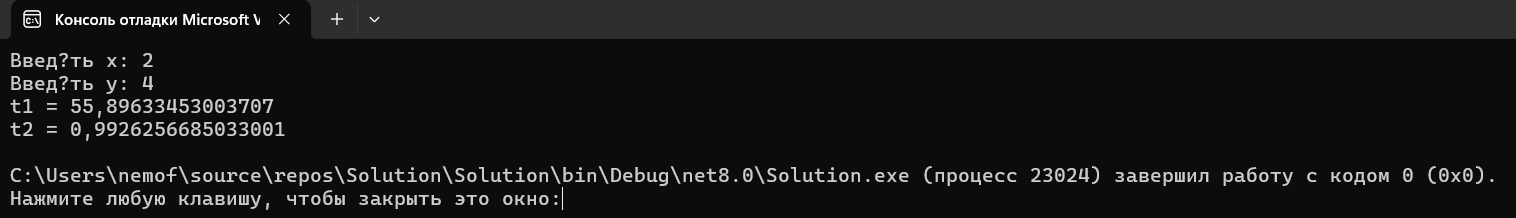
Console.WriteLine($"t1 = {t1}");

Console.WriteLine($"t2 = {t2}");

}

}

***Результат роботи програми:***



**Алгоритм побудови коду**

1. **Введення даних:**
   * Запросити у користувача ввести значення змінних x та y.
   * Зчитати ці значення з клавіатури та конвертувати їх у тип double.
2. **Ініціалізація констант:**
   * Встановити значення констант a та b як 12.5 та 1.3 відповідно.
3. **Обчислення t1:**
   * Використати формулу для обчислення t1, яка включає логарифми, степеневі функції та арифметичні операції: t1 = \*
4. **Обчислення проміжного значення ax:**
   * Вирахувати ax як добуток a і x.
5. **Обчислення t2:**
   * Використати формулу для обчислення t2, яка включає синусоїдальні функції: t2 = +
6. **Виведення результатів:**
   * Вивести на екран результати обчислень t1 і t2.

**Функціональність**

Програма реалізує наступний функціонал:

* **Введення даних:** Програма запитує у користувача два значення (x та y) та зчитує їх.
* **Обчислення:** На основі введених даних та констант, програма обчислює два значення t1 та t2 згідно з наведеними формулами.
* **Виведення результатів:** Програма виводить результати обчислень t1 та t2 на екран.

**Висновок**

Програма демонструє базові можливості мови C# для обробки математичних обчислень, включаючи використання математичних функцій та арифметичних операцій. Вона ефективно реалізує алгоритми для обчислення складних виразів і виводить результати в зрозумілому форматі. Користувач отримує можливість ввести значення змінних та отримати результати на основі формул, що забезпечує інтерактивність програми.

**Завдання 4\***

Виконати минуле завдання, покращивши гнучкість програми обчислення виразу, якщо вхідні дані є некоректними із точки зору математичних правил (нуль в знаменнику, від’ємне значення під знаком квадратного кореня та ін.). Для цього використати механізм обробки відповідних виключень (try … catch … finally)

***Текст програми:***

using System;

class Task4

{

static void Main()

{

try

{

// Введення значень x і y з клавіатури

Console.Write("Введіть x: ");

double x = Convert.ToDouble(Console.ReadLine()); // Читання значення x з клавіатури і перетворення його на тип double

Console.Write("Введіть y: ");

double y = Convert.ToDouble(Console.ReadLine()); // Читання значення y з клавіатури і перетворення його на тип double

// Початкові значення змінних

double a = 12.5, b = 1.3; // Ініціалізація змінних a і b значеннями 12.5 і 1.3 відповідно

// Перевірка на некоректні значення

if (x == 0 || y == 0) // Перевіряє, чи не дорівнюють x або y нулю

{

throw new ArgumentException("Значення x і y не можуть бути нульовими."); // Викидання помилки, якщо x або y дорівнює нулю

}

// Обчислення t1

double t1 = (1 / Math.Pow(b, 3)) \* (Math.Pow(a, 2) \* Math.Log(y / x) + (2 \* a \* x / y) + (Math.Pow(y, 2) / (2 \* Math.Pow(x, 2))));

// Обчислює значення t1 за допомогою заданої формули

// Обчислення t2

double ax = a \* x; // Обчислює значення ax як добуток a і x

if (Math.Sin(ax) == 0) // Перевіряє, чи не дорівнює sin(ax) нулю

{

throw new ArgumentException("Значення ax не може бути таким, щоб sin(ax) дорівнював нулю."); // Викидає помилку, якщо sin(ax) дорівнює нулю

}

double t2 = (1 / (2 \* Math.Sqrt(2) \* a)) + ((3 \* Math.Pow(Math.Sin(ax), 2) - 1) / (Math.Pow(Math.Sin(ax), 2) - 1));

// Обчислює значення t2 за іншою формулою

// Виведення результатів

Console.WriteLine($"t1 = {t1}"); // Виводить значення t1

Console.WriteLine($"t2 = {t2}"); // Виводить значення t2

}

catch (ArgumentException ex) // Ловить помилки, пов'язані з некоректними аргументами

{

Console.WriteLine($"Помилка: {ex.Message}"); // Виводить повідомлення про помилку

}

catch (Exception ex) // Ловить інші непередбачені помилки

{

Console.WriteLine($"Непередбачена помилка: {ex.Message}"); // Виводить повідомлення про непередбачену помилку

}

finally

{

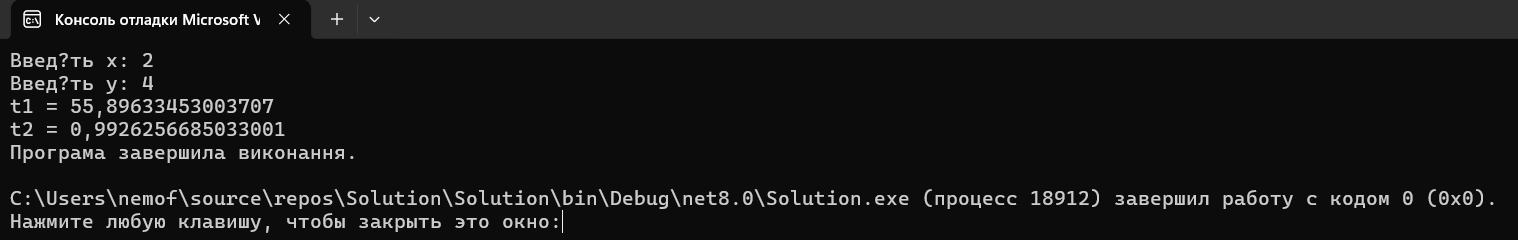
Console.WriteLine("Програма завершила виконання."); // Повідомлення про завершення роботи програми

}

}

}

***Результат роботи програми:***



**Алгоритм побудови коду**

1. **Введення даних:**
   * Програма запитує у користувача введення значень x і y.
   * Значення конвертуються в тип double.
2. **Перевірка на некоректні значення:**
   * Перевіряється, чи не дорівнюють x або y нулю. Якщо так, генерується виняток ArgumentException з відповідним повідомленням.
3. **Обчислення виразів:**
   * Розраховується значення t1 за формулою: t1 = \*
   * Перевіряється, чи не дорівнює Math.Sin(ax) нулю. Якщо так, генерується виняток ArgumentException.
   * Розраховується значення t2 за формулою: t2 = +
4. **Виведення результатів:**
   * Виводяться результати обчислень t1 і t2.
5. **Обробка винятків:**
   * Ловляться винятки ArgumentException і загальні Exception, і виводяться відповідні повідомлення про помилки.
6. **Завершення роботи:**
   * Виводиться повідомлення про завершення програми в блок finally.

**Функціональність**

* **Введення і перетворення даних:** Програма забезпечує коректне перетворення введених даних у числовий формат, що дозволяє уникнути помилок формату.
* **Обробка виключень:** За допомогою try-catch обробляються випадки некоректних вхідних значень (наприклад, нуль у знаменнику або від'ємне значення під квадратним коренем).
* **Обчислення:** Виконуються обчислення відповідно до наданих формул з урахуванням перевірок на можливі помилки.
* **Виведення результатів:** Результати обчислень виводяться на екран, що дозволяє користувачу бачити результати програми.
* **Завершення роботи:** Завжди виконується блок finally, що гарантує повідомлення про завершення виконання програми, незалежно від того, чи сталися помилки.

**Висновок**

Програма реалізована з використанням механізму обробки виключень для покращення гнучкості обчислень виразу. Вона коректно обробляє ситуації, коли вхідні дані є некоректними, завдяки перевіркам і виключенням. Включення блоків try, catch, і finally забезпечує стабільність і надійність виконання програми, а також надає користувачу зрозумілі повідомлення про помилки і завершення роботи програми.

**Завдання 5**

Використовуючи завдання власного варіанта до лабораторної роботи 5 з електронних методичних вказівок з основ програмування та алгоритмічних мов створити проект з реалізації цих завдань засобами мови C#.

***Текст програми:***

using System; // Підключення базових класів.

class Task5 // Оголошення класу.

{

static void Main() // Головний метод.

{

double x, y; // Змінні для координат.

try // Обробка помилок.

{

// Введення координат

Console.Write("Введіть координату x > ");

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

Console.Write("Введіть координату y > ");

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

// Виведення значень

Console.WriteLine($"x={x:0.000}; y={y:0.000}");

// Перевірка умови

if ((y >= 1 - x) && (x \* x + y \* y <= 1)) // Перевірка двох умов: точка повинна бути в межах лінії y >= 1 - x і в межах кола радіусом 1.

{

Console.WriteLine("Точка в межах області");

}

else

{

Console.WriteLine("Точка не в межах області");

}

}

catch (FormatException) // Помилка формату.

{

Console.WriteLine("Помилка: введено не число.");

}

catch (Exception ex) // Інші помилки.

{

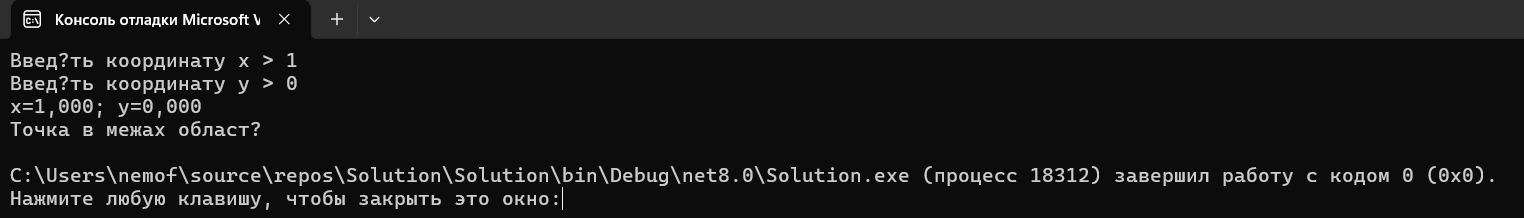
Console.WriteLine($"Помилка: {ex.Message}");

}

}

}

***Результат роботи програми:***



**Алгоритм побудови коду**

1. **Оголошення класу та методів:**
   * Оголошено клас Task5, який містить метод Main(), що є точкою входу програми.
2. **Введення даних:**
   * Користувачеві пропонується ввести координати точки x та y. Ці значення зчитуються з консолі і перетворюються в тип double.
3. **Перевірка умов:**
   * Програма перевіряє, чи знаходиться точка всередині області, яка обмежена двома умовами:
     + Точка повинна знаходитися вище лінії y = 1 - x.
     + Точка повинна бути всередині кола з радіусом 1, центром в початку координат.
   * Якщо обидві умови виконуються, виводиться повідомлення "Точка в межах області", інакше – "Точка не в межах області".
4. **Обробка помилок:**
   * Використовуються блоки try-catch для обробки помилок, таких як неправильний формат введених даних або інші непередбачені помилки.

**Функціональність**

1. **Введення координат:**
   * Користувач вводить значення для x і y, які перевіряються на коректність.
2. **Перевірка умов:**
   * Програма визначає, чи точка лежить у заданій області, використовуючи дві математичні умови:
     + y >= 1 - x перевіряє, чи точка знаходиться над або на лінії y = 1 - x.
     + x \* x + y \* y <= 1 перевіряє, чи точка всередині кола з радіусом 1.
3. **Обробка помилок:**
   * Програма коректно обробляє помилки формату і інші можливі виключення, надаючи зрозумілі повідомлення.

**Висновок**

Програма на C# реалізує перевірку розташування точки в двовимірній площині щодо заданих умов. Вона забезпечує ввід координат, перевіряє їх відповідність до області, і обробляє можливі помилки вводу. Код ефективний для вирішення поставленого завдання, легко зрозумілий і забезпечує коректний результат за умови правильного вводу даних.

**Завдання 6\***Виконати минуле завдання без використання оператору if у будь-яких формах

***Текст програми:***

using System; // Підключення бібліотеки System.

class Task6 // Оголошення класу Task6.

{

static void Main() // Головний метод.

{

double x, y; // Оголошення змінних x і y.

try // Початок блоку обробки помилок.

{

Console.Write("Введіть координату x > "); // Запит на введення x.

x = Convert.ToDouble(Console.ReadLine()); // Читання та конвертація x.

Console.Write("Введіть координату y > "); // Запит на введення y.

y = Convert.ToDouble(Console.ReadLine()); // Читання та конвертація y.

Console.WriteLine($"x={x:0.000}; y={y:0.000}"); // Виведення x і y.

string result = (y >= 1 - x) && (x \* x + y \* y <= 1) // Перевірка умов.

? "Точка в межах області" // Якщо умови істинні.

: "Точка не в межах області"; // Якщо умови хибні.

Console.WriteLine(result); // Виведення результату.

}

catch (FormatException) // Обробка помилок форматування.

{

Console.WriteLine("Помилка: введено некоректне значення."); // Повідомлення про помилку.

}

catch (Exception ex) // Обробка інших помилок.

{

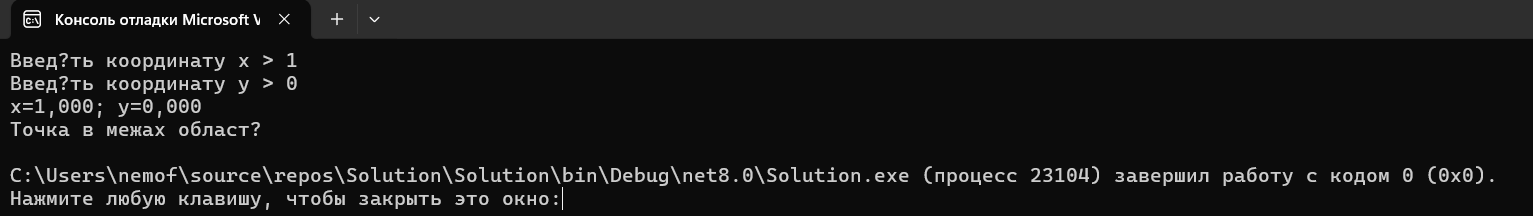
Console.WriteLine($"Сталася помилка: {ex.Message}"); // Повідомлення про помилку.

}

}

}

***Результат роботи програми:***



**Алгоритм побудови коду**

1. **Оголошення змінних:**
   * Задаємо змінні x та y для зберігання координат точки.
2. **Обробка виключень:**
   * Використовуємо блок try-catch для обробки помилок введення та конвертації чисел.
3. **Введення даних:**
   * Запитуємо у користувача координати x і y, конвертуємо введені значення в тип double.
4. **Перевірка умови:**
   * Використовуємо тернарний оператор для перевірки умови без if:
     + y >= 1 - x перевіряє, чи точка лежить вище або на лінії y = 1 - x.
     + x \* x + y \* y <= 1 перевіряє, чи точка лежить всередині або на межі кола з центром у (0,0) і радіусом 1.
   * Відповідно до результату перевірки, присвоюємо змінній result відповідне повідомлення.
5. **Виведення результатів:**
   * Виводимо координати точки та результат перевірки.

**Функціональність**

Програма виконує перевірку, чи точка з заданими координатами x і y знаходиться в межах заданої області. Область визначається двома умовами:

* Точка повинна бути розташована вище або на прямій y = 1 - x.
* Точка повинна бути всередині або на межі кола радіусом 1, що має центр у точці (0,0).

**Висновок**

У цьому завданні використано тернарний оператор для реалізації умовної логіки без використання оператора if. Це дозволяє перевіряти умови і призначати значення змінній result в залежності від результатів перевірки. Блок try-catch забезпечує обробку помилок вводу та конвертації, що робить програму більш стійкою до некоректного вводу.

**Завдання 7**

Використовуючи завдання власного варіанта до лабораторної роботи 6 з електронних методичних вказівок з основ програмування та алгоритмічних мов створити проект з реалізації цих завдань засобами мови C#.

***Текст програми:***

using System; // Підключення базових типів

class Task7

{

static void Main()

{

long n = 0; // Індекс ряду

double sum = 0; // Сума всіх членів

double term; // Поточний член

const double eps = 0.000001; // Точність

long k2 = 1; // 2^n

short k1 = 1; // (-1)^n

double sumFirst10 = 0; // Сума перших 10 членів

for (n = 0; ; n++, k2 \*= 2, k1 = (short)-k1)

{

term = k1 \* (1.0 / (n \* n + k2)); // Обчислення члена

if (Math.Abs(term) >= eps)

{

sum += term; // Додавання до суми

}

else

{

break; // Вихід з циклу

}

if (n < 10)

{

sumFirst10 += term; // Сума перших 10 членів

}

}

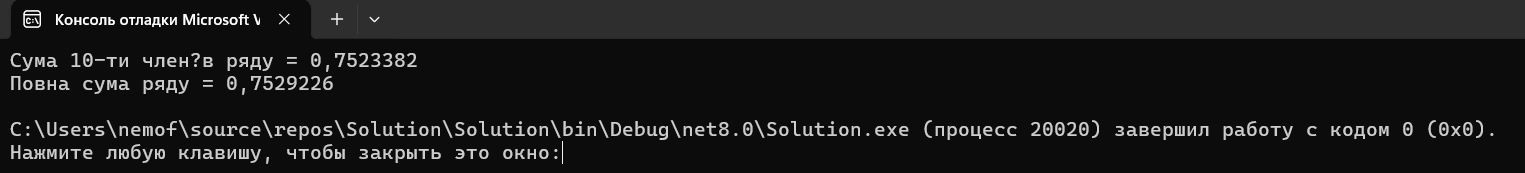
Console.WriteLine("Сума 10-ти членів ряду = {0:F7}", sumFirst10); // Виведення суми 10 членів

Console.WriteLine("Повна сума ряду = {0:F7}", sum); // Виведення повної суми

}

}

***Результат роботи програми:***



**Алгоритм побудови коду**

1. **Оголошення змінних**:
   * n: Індекс ряду (довгий тип).
   * sum: Сума всіх членів ряду (подвійний тип).
   * term: Поточний член ряду (подвійний тип).
   * eps: Точність для припинення обчислень (постійна).
   * k2: Змінна для обчислення степеня 2 (короткий тип).
   * k1: Змінна для обчислення знака (короткий тип).
   * sumFirst10: Сума перших 10 членів (подвійний тип).
2. **Цикл for**:
   * Цикл без явного умови завершення. Змінні n, k2 і k1 оновлюються на кожній ітерації.
   * term обчислюється як k1 \* (1.0 / (n \* n + k2)), де k1 змінюється між 1 і -1, а k2 є степенем 2.
3. **Умови**:
   * Якщо модуль term менший ніж eps, цикл переривається.
   * Сума перших 10 членів накопичується у sumFirst10.
   * Після завершення циклу виводяться значення суми перших 10 членів та повної суми ряду.

**Функціональність**

* **Ініціалізація змінних**: Встановлює початкові значення для всіх змінних.
* **Обчислення члена ряду**: Вираховує значення кожного члена ряду згідно з формулою.
* **Сума членів ряду**: Накопичує значення членів ряду до тих пір, поки модуль члена перевищує точність eps.
* **Завершення обчислень**: Перериває цикл, коли модуль члена менший ніж eps.
* **Виведення результатів**: Виводить суму перших 10 членів і загальну суму ряду з точністю до 7 знаків після коми.

**Висновок**

Програма коректно реалізує алгоритм для обчислення суми членів ряду з урахуванням точності. Вона забезпечує правильний розрахунок як часткової суми перших 10 членів, так і повної суми ряду. Використання циклу for дозволяє динамічно оновлювати змінні та переривати обчислення при досягненні заданої точності. Це дозволяє уникнути надмірних обчислень і забезпечити ефективність програми.

**Завдання 8\***

Виконати минуле завдання, переробивши алгоритм вираховування суми ряду таким чином:

* похибка та кількість членів ряду мають вводитися як вхідні параметри у відповідні змінні;
* сума ряду вираховується або при досягненні заданої похибки, або при досягненні заданої кількості членів ряду із видачою відповідного повідомлення.

***Текст програми:***

using System; // Підключення простору імен System

class Task8 // Оголошення класу

{

static void Main() // Основний метод

{

// Введення даних

Console.Write("Введіть похибку: "); // Запит похибки

double eps = Convert.ToDouble(Console.ReadLine()); // Зчитування похибки

Console.Write("Введіть кількість членів ряду: "); // Запит кількості членів

int maxTerms = Convert.ToInt32(Console.ReadLine()); // Зчитування кількості членів

double sum = 0; // Сума членів ряду

double term; // Поточний член ряду

double sumFirst10 = 0; // Сума перших 10 членів

long k2 = 1; // Значення pow(2,n)

short k1 = 1; // Значення pow(-1,n)

bool epsReached = false; // Флаг для перевірки досягнення похибки

for (long n = 0; n < maxTerms; n++, k2 \*= 2, k1 = (short)-k1) // Обчислення членів ряду

{

term = k1 \* (1.0 / (n \* n + k2)); // Обчислення поточного члена

sum += term; // Додавання до суми

if (Math.Abs(term) < eps) // Перевірка точності

{

epsReached = true; // Встановлення флагу досягнення похибки

break; // Вихід з циклу

}

if (n < 10) // Перевірка перших 10 членів

{

sumFirst10 += term; // Додавання до суми перших 10 членів

}

}

if (epsReached)

{

Console.WriteLine("Досягнута задана похибка."); // Повідомлення про похибку

}

else

{

Console.WriteLine("Досягнуто максимальну кількість членів ряду."); // Повідомлення про досягнення максимальної кількості членів

}

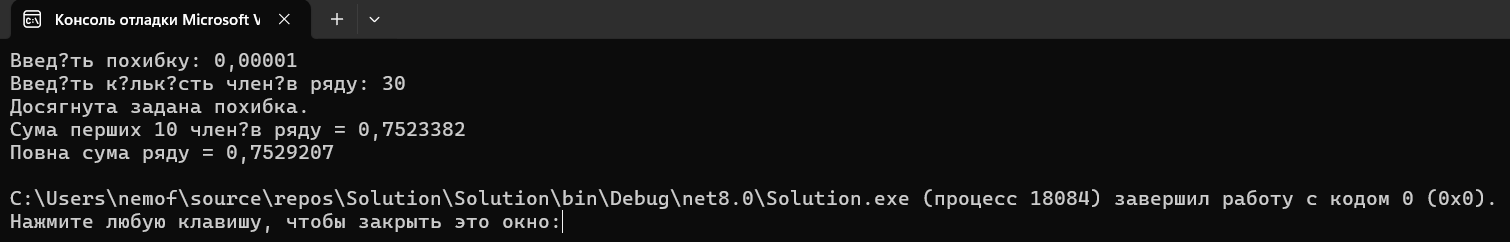
Console.WriteLine("Сума перших 10 членів ряду = {0:F7}", sumFirst10); // Виведення суми перших 10 членів

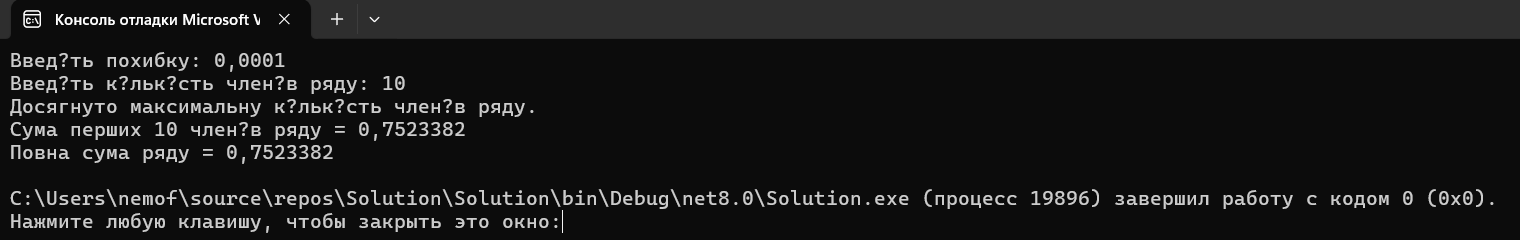
Console.WriteLine("Повна сума ряду = {0:F7}", sum); // Виведення загальної суми

}

}

***Результат роботи програми:***





**Алгоритм побудови коду:**

1. **Ініціалізація параметрів**:
   * Користувач вводить два параметри: похибку (eps) та максимальну кількість членів ряду (maxTerms).
2. **Оголошення змінних**:
   * sum: змінна для накопичення суми членів ряду.
   * term: змінна для зберігання значення поточного члена ряду.
   * sumFirst10: змінна для зберігання суми перших 10 членів ряду.
   * k2 і k1: змінні для зберігання значень pow(2,n) і pow(-1,n) відповідно.
   * epsReached: логічна змінна для перевірки досягнення заданої похибки.
3. **Цикл обчислення членів ряду**:
   * Запускається цикл для обчислення членів ряду до maxTerms або до досягнення заданої похибки.
   * У кожній ітерації обчислюється значення поточного члена ряду, яке додається до загальної суми.
   * Перевіряється, чи поточний член ряду менший за похибку (eps). Якщо так, встановлюється флаг epsReached і цикл переривається.
   * Якщо кількість членів менша за 10, додається до суми перших 10 членів ряду.
4. **Виведення результатів**:
   * Виводиться повідомлення про досягнення або похибки, або максимальної кількості членів ряду.
   * Виводяться суми перших 10 членів та загальна сума ряду.

**Функціональність:**

* **Введення параметрів**: Користувач має можливість ввести похибку та максимальну кількість членів ряду.
* **Обчислення ряду**: Програма обчислює члени ряду, поки не досягне заданої похибки або максимальної кількості членів.
* **Агрегування результатів**: Рахує суму перших 10 членів та загальну суму ряду.
* **Виведення результатів**: Інформує користувача про те, чи досягнуто задану похибку або максимальну кількість членів, і виводить результати обчислень.

**Висновок:**

Код ефективно реалізує алгоритм обчислення суми ряду з можливістю зупинки обчислень при досягненні заданої похибки або при досягненні максимальної кількості членів. Використання флагів і циклів дозволяє точно контролювати процес обчислень і виводити результати у зручному для користувача форматі. Програма коректно обробляє введені дані і надає зрозумілі результати.

**Завдання 9**

Використовуючи завдання власного варіанта до лабораторної роботи 7 з електронних методичних вказівок з основ програмування та алгоритмічних мов створити проект з реалізації цих завдань засобами мови C#.

***Текст програми:***

using System; // Підключення System.

class Task9 // Клас Task9.

{

static void Main() // Головний метод.

{

for (int n = 0; n < 5; n++) // Цикл 5 разів.

{

Console.WriteLine("| x | y |"); // Заголовок таблиці.

Console.WriteLine("|-------|------------|"); // Роздільник таблиці.

for (double x = 0; x < 4; x += 0.25) // Цикл для x.

{

double y; // Значення y.

if (x < 1) // Якщо x < 1.

y = 2 \* x - 1; // y = 2x - 1.

else if (x < 3) // Якщо x < 3.

y = 1 - Math.Sqrt(1 - (x - 2) \* (x - 2)); // y = 1 - sqrt(1 - (x - 2)²).

else // Якщо x >= 3.

y = 7 - 2 \* x; // y = 7 - 2x.

Console.Write($"| {x + n \* 4,5:0.00} | {y,10:0.0000000} |"); // Вивід x і y.

int h = (int)((y + 1) \* 10); // Обчислення h.

if (y - 1 - h \* 10 > 0.5) h++; // Коригування h.

for (int i = 0; i < h; i++) Console.Write(" "); // Пробіли для графіку.

Console.WriteLine("\*"); // Символ '\*' для графіку.

}

Console.WriteLine("Натисніть клавішу Enter..."); // Запрос на Enter.

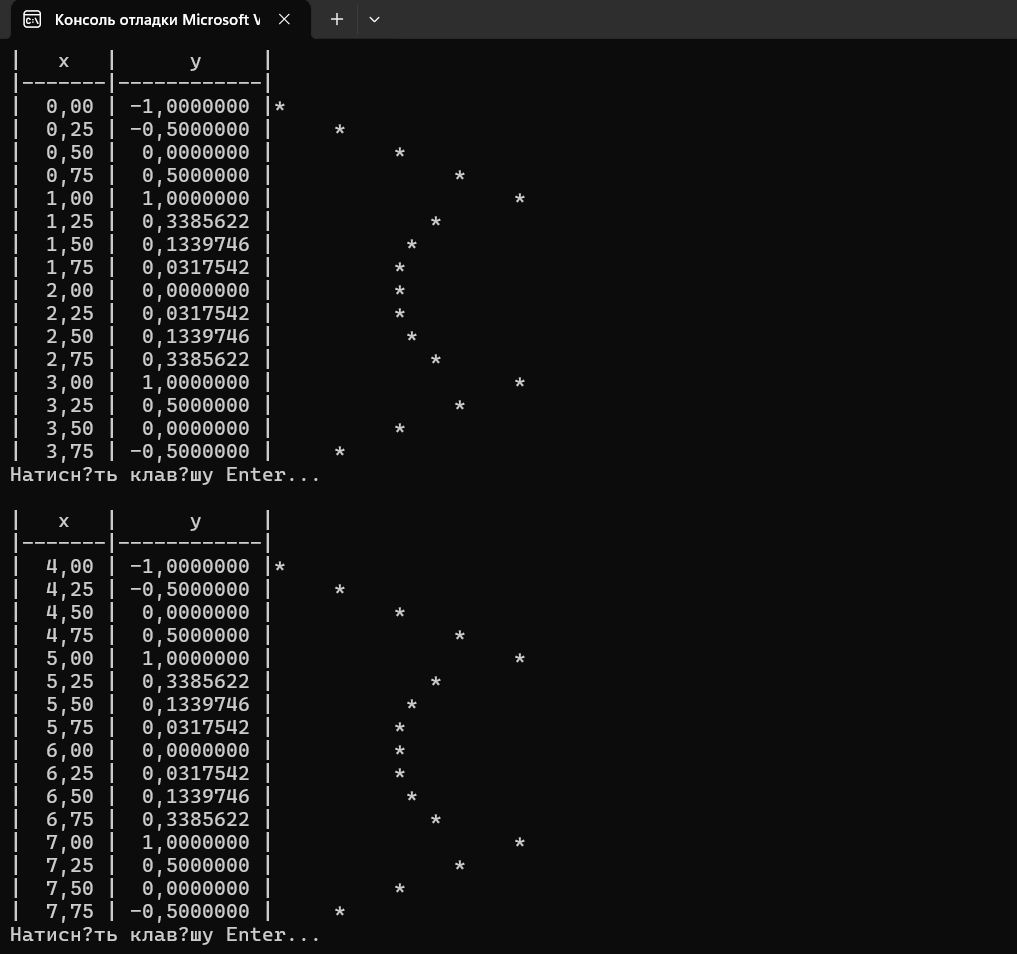
Console.ReadLine(); // Очікування Enter.

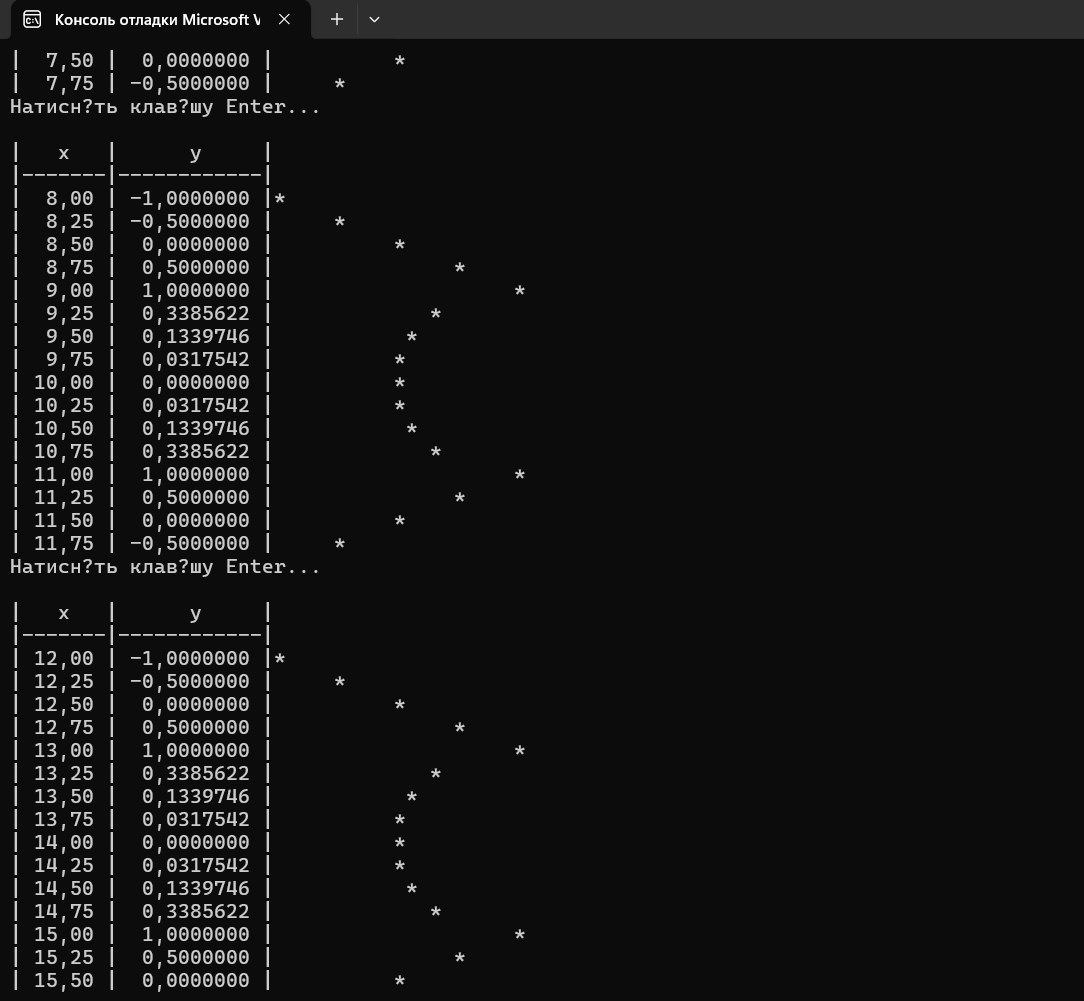
}

}

}

***Результат роботи програми:***







**Опис алгоритму**

1. **Ініціалізація**: Програма починає з підключення необхідного простору імен (System) та створення класу Task9 з основним методом Main.
2. **Цикл виводу**: Основний цикл for виконується 5 разів, щоб надати можливість повторювати виведення таблиці з графіком.
3. **Заголовок таблиці**: Виводиться заголовок таблиці з стовпцями x та y, а також роздільник таблиці.
4. **Обчислення значень**:
   * Для кожного значення x (починаючи з 0 до 4 з кроком 0.25) обчислюється відповідне значення y за допомогою умов:
     + Якщо x < 1, то y = 2 \* x - 1.
     + Якщо x між 1 і 3, то y = 1 - .
     + Якщо x 3, то y = 7 - 2 \* x.
5. **Вивід значень**: Виводяться значення x та y у таблиці, при цьому x зсувається на n \* 4 для зміни діапазону x.
6. **Графічний вивід**:
   * Обчислюється значення h, яке визначає кількість пробілів перед символом \* для графічного відображення значення y.
   * Якщо значення y перевищує поріг, h коригується.
   * Виводиться рядок з пробілами і символом \*, що представляє графік.
7. **Очікування**: Після виводу кожної таблиці програма чекає, поки користувач натисне клавішу Enter, щоб продовжити до наступної ітерації циклу.

**Функціональність**

* **Виведення таблиці**: Програма виводить таблицю значень x і y для кожної ітерації циклу.
* **Графічне відображення**: Програма генерує простий графік для кожної таблиці, де висота графіка відповідає значенню y.
* **Ітерації**: Таблиця та графік повторюються 5 разів, дозволяючи спостерігати результати для різних діапазонів значень x.

**Висновок**

Програма на мові C# успішно реалізує функціональність для обчислення і відображення таблиць значень і графіків. Вона дозволяє користувачеві спостерігати зміну значень функцій y у залежності від x, а також наочно представляє результати у вигляді графіків. Використання циклів та умов у коді забезпечує коректність обчислень і наглядність виводу даних.

**Завдання 10\***

Виконати минуле завдання, покращивши зовнішній вигляд побудованого графіку за допомогою символів псевдографіки ⸜, ⸝, ⸺, \, /, ­ та ін.) та налаштування кольорів консолі, наприклад, числа відобразити одним кольором, пояснюючий текст – іншим, графік – іншим від попередніх.

***Текст програми:***

using System;

class Task10

{

static void Main()

{

Console.OutputEncoding = System.Text.Encoding.UTF8; // Встановлення кодування UTF-8

Console.ForegroundColor = ConsoleColor.Yellow; // Жовтий текст

Console.WriteLine("| x | y |"); // Заголовок таблиці

Console.WriteLine("|-------|------------|"); // Розділювач таблиці

Console.ResetColor(); // Скидання кольору

for (int n = 0; n < 5; n++) // Цикл для рядків

{

for (double x = 0; x < 4; x += 0.25) // Цикл для x

{

double y; // Значення y

if (x < 1) // Перша формула

y = 2 \* x - 1;

else if (x < 3) // Друга формула

y = 1 - Math.Sqrt(1 - (x - 2) \* (x - 2));

else // Третя формула

y = 7 - 2 \* x;

Console.ForegroundColor = ConsoleColor.Cyan; // Блакитний текст

Console.Write($"| {x + n \* 4,5:0.00} | {y,10:0.0000000} |"); // Виведення x і y

Console.ResetColor(); // Скидання кольору

int h = (int)((y + 1) \* 10 / 2); // Висота графіка

Console.ForegroundColor = ConsoleColor.Green; // Зелений текст для графіка

// Виведення графіка з псевдографікою

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

{

Console.Write(" ");

}

if (y > 0) // Позитивний y

Console.WriteLine("/");

else if (y < 0) // Негативний y

Console.WriteLine("\\");

else // Нульовий y

Console.WriteLine("|");

Console.ResetColor(); // Скидання кольору

}

Console.ForegroundColor = ConsoleColor.Magenta; // Магентовий текст

Console.WriteLine("Натисніть клавішу Enter..."); // Запрошення натиснути Enter

Console.ResetColor(); // Скидання кольору

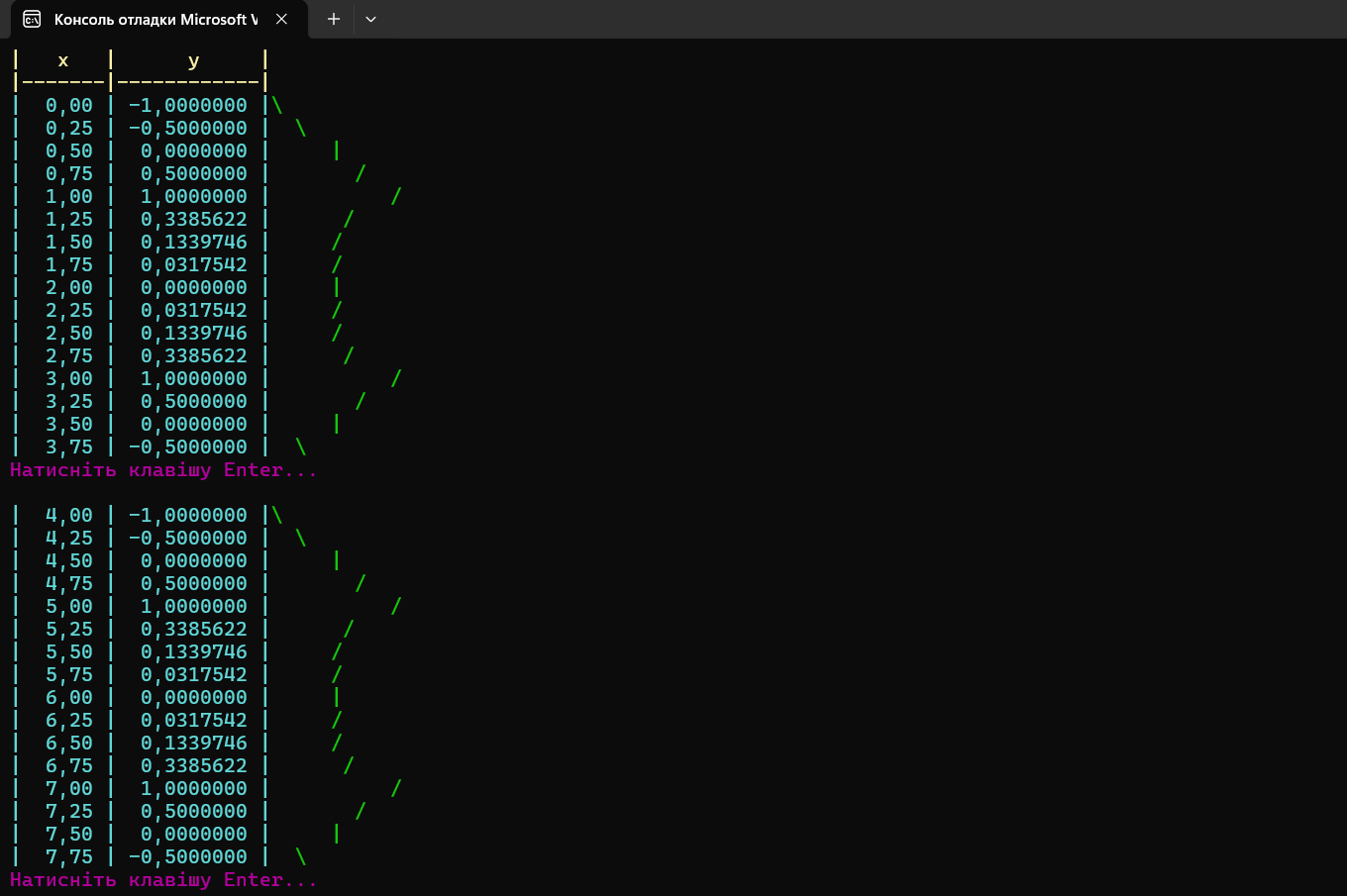
Console.ReadLine(); // Очікування Enter

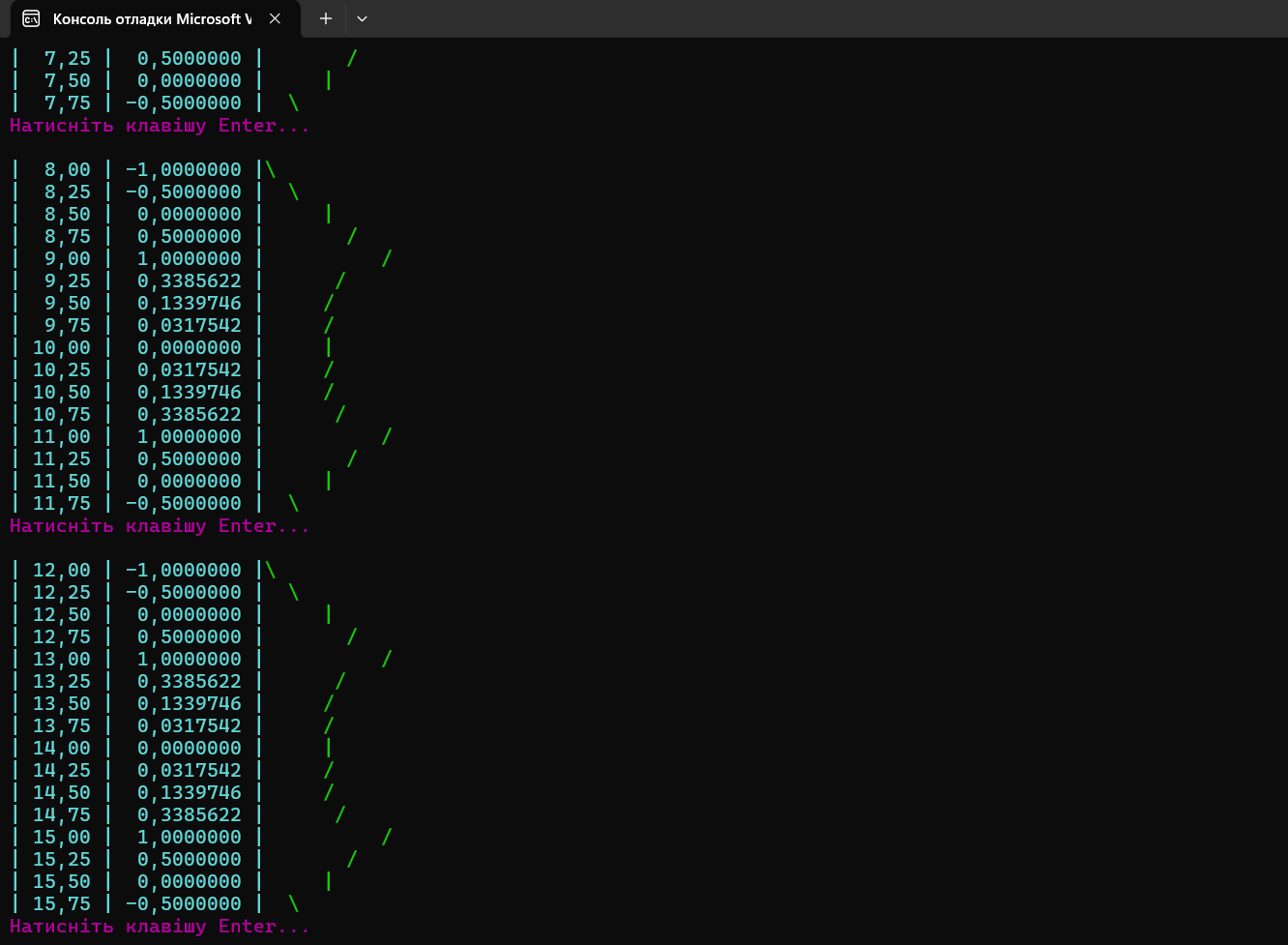
}

}

}

***Результат роботи програми:***







**Алгоритм побудови коду**

1. **Налаштування кодування консолі:**
   * Встановлено кодування UTF-8 для коректного відображення символів псевдографіки.
2. **Виведення заголовка таблиці:**
   * Використовується жовтий колір для заголовка та розділювача таблиці.
   * Виконано скидання кольору після виведення заголовка.
3. **Цикл для обчислення та виведення даних:**
   * Задано два цикли: один для рядків, інший для значень x з кроком 0.25.
   * Для кожного значення x обчислюється відповідне значення y за допомогою трьох умовних формул:
     + Перша формула: y=2x−1, якщо x < 1.
     + Друга формула: y = 1 - , якщо 1 x 3
     + Третя формула: y=7−2x, якщо x≥3.
4. **Виведення даних у таблиці:**
   * Для кожної пари значень x і y встановлюється блакитний колір.
   * Форматування чисел для чіткого відображення.
5. **Виведення графіка:**
   * Розраховується висота графіка h на основі значення y.
   * Використовується зелений колір для відображення псевдографіки.
   * Для позитивних значень y виводиться символ /, для негативних — символ \, для нульових — символ |.
6. **Очікування вводу користувача:**
   * Після кожного циклу виводиться запрошення натиснути Enter, використовується магентовий колір.

**Функціональність**

* **Виведення таблиці з результатами:** Програма обчислює значення y для різних x і виводить їх у таблицю з кольоровим форматуванням.
* **Графічне відображення:** На основі обчислених значень y виводиться графік, що використовує символи псевдографіки для візуалізації.
* **Інтерактивність:** Програма запрошує користувача натиснути клавішу Enter перед завершенням виконання.

**Висновок**

Програма реалізує можливість візуалізації математичних функцій у консолі з використанням кольорового форматування та псевдографіки. Вона дозволяє не лише обчислювати та відображати числові результати, але й представляти їх графічно, що робить результати більш наочними. Застосування кольорів та символів псевдографіки значно покращує зовнішній вигляд графіка, роблячи його легшим для сприйняття.

**Загальний висновок**

У ході виконання лабораторної роботи було отримано практичні навички використання базових типів даних, операторів та засобів введення/виведення мови C#. Було розглянуто різні аспекти програмування на C#, включаючи:

1. **Робота з масивами та циклами**: Завдання 1 та 2 продемонстрували використання масивів для зберігання та обробки даних про транспортні засоби. Цикли були використані для ефективного введення та виведення даних, а також для форматування таблиці з результатами.
2. **Математичні обчислення та обробка винятків**: Завдання 3 та 4 охоплювали виконання математичних обчислень за заданими формулами. У завданні 4 було реалізовано механізм обробки винятків для забезпечення коректності роботи програми при некоректних вхідних даних.
3. **Перевірка умов та тернарний оператор**: У завданнях 5 та 6 було розглянуто перевірку умов для визначення належності точки до заданої області. Завдання 6 продемонструвало використання тернарного оператора як альтернативи традиційному оператору if.
4. **Обчислення сум рядів**: Завдання 7 та 8 були присвячені обчисленню сум рядів з заданою точністю. У завданні 8 було додано можливість обмеження кількості членів ряду та виведення відповідних повідомлень.
5. **Візуалізація даних**: Завдання 9 та 10 розглядали побудову графіків функцій у консолі. Завдання 10 продемонструвало використання символів псевдографіки та кольорового форматування для покращення візуального представлення даних.

Виконання цих завдань дозволило поглибити розуміння основних концепцій програмування на C#, таких як типи даних, оператори, введення/виведення, обробка винятків, цикли та умовні конструкції. Крім того, було отримано досвід роботи з масивами, математичними функціями, форматуванням виведення та візуалізацією даних. Ці знання та навички є важливими для подальшого вивчення мови C# та розробки більш складних програм.