Міністерство освіти України

Національний технічний університет "ХПІ"

кафедра "Інформатики та інтелектуальної власності"

**Звіт**

**Лабораторна робота 2**

з дисципліни "Кросплатформне програмування"

Виконав: студент групи КН-1224С

Мульков М. В.

Перевірив:

Івашко А.В.

Харків 2025

Зміст

[Завдання 3](#_Toc193178889)

[Реалізація коду 4](#_Toc193178890)

[Алгоритм коду 4](#_Toc193178891)

[Тестування коду 6](#_Toc193178892)

[Висновок 8](#_Toc193178893)

[Додаток А 10](#_Toc193178894)

**Робота з рваними масивами у .NET Framework**

**Мета роботи**: Дослідити роботу з рваними масивами, навчитися створювати, обробляти та сортувати їх.

# Завдання

За власним варіантом, створити за завданням алгоритм, код програми, та протестувати його.

Мій варіант 7.

Ввести цілий двовимірний рваний масив ( jagged array ), що складається з рядків довільної довжини. Упорядкувати рядки цього масиву за зростанням суми рядків. Реалізувати функцію обчислення суми рядка. Видалити всі нульові рядки. Для перевірки рядків на рівність всіх елементів 0 реалізувати окрему функцію.

# Реалізація коду

## Алгоритм коду

Повний код програми відображено в додатку А.

1. Виведення меню

Програма запускається в безкінечному циклі, доки користувач не вибере вихід. На екран виводиться меню з трьома варіантами:

* Створити випадковий зубчастий масив.
* Ввести масив вручну.
* Вийти з програми.

Користувач вводить вибір, і залежно від введених даних виконується відповідний блок коду.

2. Створення випадкового масиву

Генерується випадкова кількість рядків від 1 до 7. Для кожного рядка генерується випадкова довжина від 0 до 7. Заповнюється випадковими числами в діапазоні від -7 до 7.

У результаті створюється зубчастий масив, у якому кожен рядок має різну довжину.

3. Ручне введення масиву

Користувач вводить кількість рядків. Для кожного рядка вводиться кількість елементів. Користувач вводить елементи, які розбиваються на окремі числа та записуються в масив.

4. Виведення початкового масиву

Викликається функція PrintArray, яка проходить по всіх рядках масиву та виводить їх на екран. Якщо рядок порожній, виводиться (empty). Якщо всі елементи рядка рівні нулю, вони виділяються червоним кольором і виводиться їх сума.

5. Видалення рядків, що складаються тільки з нулів

Викликається функція DeleteZeroRows. Спочатку підраховується кількість рядків, у яких є хоча б один ненульовий елемент. Потім створюється новий масив, у який записуються тільки ненульові рядки. Повертається новий масив без рядків, що містять лише нулі.

6. Виведення масиву після видалення нульових рядків

Викликається PrintArray, щоб показати змінений масив після видалення рядків із нулями.

7. Сортування рядків за зростанням суми елементів

Проходиться двома вкладеними циклами по всіх рядках масиву. Якщо сума елементів поточного рядка більша, ніж у наступного, вони міняються місцями. У результаті рядки масиву впорядковуються за зростанням їх суми.

8. Виведення відсортованого масиву

Знову викликається PrintArray, щоб показати відсортований масив.

9. Повернення до меню або вихід із програми

Після виведення масиву користувач повертається до головного меню. Якщо користувач вибирає "3", програма завершує виконання.

## Тестування коду

Тестування коду з автоматичним створенням масиву відображено на рис. 1.

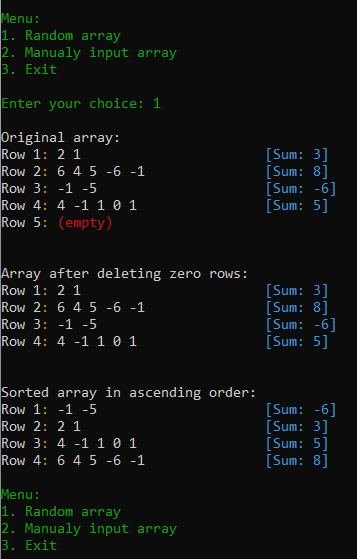


Рис. 1. Тестування коду з автоматичним створенням масиву

Тестування коду з ручним введенням масиву відображено на рис. 2.

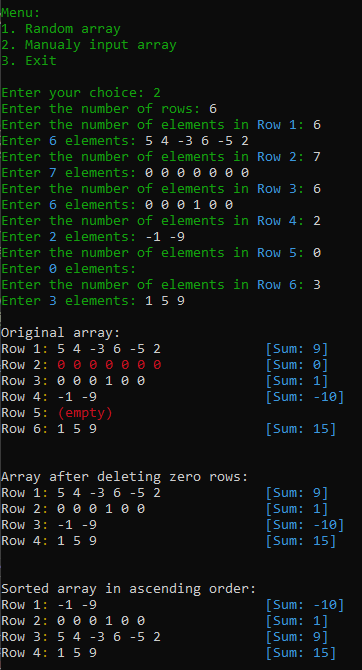


Рис. 2. Тестування коду з ручним введенням масиву

# Висновок

У ході виконання цієї роботи були вивчені та повторені матеріали, пов’язані з роботою зубчастих масивів у мові програмування C#. Було детально розглянуто принципи їх створення, ініціалізації, заповнення випадковими значеннями та ручного введення користувачем. Окрему увагу приділено методам обробки таких масивів, зокрема перевірці наявності нульових рядків, їх видаленню та сортуванню за певним критерієм.

У даній програмі реалізовано алгоритм, що дозволяє працювати з зубчастими масивами динамічної структури. Було розглянуто три основні етапи: генерацію випадкових даних, ручне введення масиву та подальшу обробку введених значень. Програма дозволяє користувачеві вибирати між автоматичним створенням масиву та можливістю самостійного введення даних, що робить її гнучкою та адаптивною до різних умов використання.

Одним із важливих елементів програми є видалення рядків, що містять лише нулі. Це здійснюється за допомогою спеціальної функції, яка перевіряє кожен рядок масиву та створює новий масив, що не містить таких рядків. У результаті цього етапу масив стає компактнішим, а подальші обчислення виконуються ефективніше.

Ще одним важливим аспектом є сортування рядків масиву за сумою їхніх елементів. Було реалізовано метод сортування, що порівнює суми елементів у кожному рядку та виконує обмін місцями для впорядкування даних у зростаючому порядку. Це дозволяє аналізувати масив та працювати з відсортованими значеннями, що є корисним для багатьох практичних застосувань.

Окремо варто відзначити реалізацію виведення масиву в зручному для користувача форматі. Використано кольорове форматування за допомогою ANSI-кодів, що дозволяє візуально виділити ключові елементи, такі як сума рядка, наявність нульових значень або порожніх рядків. Це значно покращує зручність взаємодії користувача з програмою та робить інтерфейс більш інформативним.

Таким чином, у ході виконання роботи були засвоєні важливі навички роботи із зубчастими масивами, їх динамічним створенням, обробкою та сортуванням. Також було вдосконалено навички оптимізації коду, зокрема використання окремих функцій для перевірки умов, обчислення сум елементів та сортування масиву.

# Додаток А

namespace LB\_2

{

class LB\_2

{

const string GREEN = "\u001b[32m";

const string RED = "\u001b[31m";

const string CYAN = "\u001b[36m";

const string YELLOW = "\u001b[33m";

const string RESET = "\u001b[0m";

static Random rand = new Random();

static bool IsZeroRow(int[] row)

{

foreach (int num in row)

{

if (num != 0)

return false;

}

return true;

}

static int[][] DeleteZeroRows(int[][] array)

{

int count = 0;

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

{

if (!IsZeroRow(array[i]))

{

count++;

}

}

int[][] jagged = new int[count][];

int index = 0;

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

{

if (!IsZeroRow(array[i]))

{

jagged[index] = array[i];

index++;

}

}

return jagged;

}

static int Sum(int[] row)

{

int sum = 0;

foreach (int x in row)

sum += x;

return sum;

}

static void PrintArray(int[][] array, string title)

{

Console.WriteLine(RESET + $"\n{title}:");

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

{

Console.Write($"Row {i + 1}{YELLOW}:{RESET} ");

if (array[i].Length == 0)

Console.WriteLine(RED + "(empty)" + RESET);

else if (IsZeroRow(array[i]))

Console.WriteLine(RED + string.Join(" ", array[i]).PadRight(25) + $" {CYAN}[Sum: {Sum(array[i])}]" + RESET);

else

Console.WriteLine(string.Join(" ", array[i]).PadRight(25) + $" {CYAN}[Sum: {Sum(array[i])}]" + RESET);

}

}

static int[][] ManualInput()

{

Console.Write("Enter the number of rows:{RESET} ");

int rows = int.Parse(Console.ReadLine());

int[][] arr = new int[rows][];

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

{

Console.Write($"{GREEN}Enter the number of elements in {CYAN}Row {i + 1}{GREEN}:{RESET} ");

int columns = int.Parse(Console.ReadLine());

arr[i] = new int[columns];

Console.Write($"{GREEN}Enter {CYAN}{columns} {GREEN}elements: {RESET}");

string[] elements = Console.ReadLine().Split(' ');

for (int j = 0; j < columns; j++)

{

arr[i][j] = int.Parse(elements[j]);

}

}

return arr;

}

static void Main()

{

while (true)

{

Console.WriteLine(

$"\n{GREEN}Menu:" +

$"\n1. Random array" +

$"\n2. Manualy input array" +

$"\n3. Exit\n"

);

Console.Write("Enter your choice: ");

string choice = Console.ReadLine();

int[][] jagged;

switch (choice)

{

case "1":

int rows = rand.Next(1, 7);

jagged = new int[rows][];

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

{

int len = rand.Next(0, 7);

jagged[i] = new int[len];

for (int j = 0; j < len; j++)

jagged[i][j] = rand.Next(-7, 7);

}

break;

case "2":

jagged = ManualInput();

break;

case "3":

return;

default:

Console.WriteLine(RED + "Try agian\n" + RESET);

continue;

}

PrintArray(jagged, "Original array");

jagged = DeleteZeroRows(jagged);

PrintArray(jagged, "\nArray after deleting zero rows");

for (int i = 0; i < jagged.Length - 1; i++)

{

for (int j = i + 1; j < jagged.Length; j++)

{

if (Sum(jagged[i]) > Sum(jagged[j]))

{

(jagged[i], jagged[j]) = (jagged[j], jagged[i]);

}

}

}

PrintArray(jagged, "\nSorted array in ascending order");

}

}

}

}