

Міністерство освіти і науки України
Національний технічний університет України «Київський політехнічний
інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт

з лабораторної роботи № 9 з дисципліни
«Алгоритми та структури даних-1. Основи
алгоритмізації»

«Дослідження алгоритмів обходу масивів»

Варіант 5

Виконав студент

ІП-13 Вальчишен Ярослав Олександрович

(шифр, прізвище, ім'я, по батькові)

Перевірів

Вечерковська А.С.

(прізвище, ім'я, по батькові)

Лабораторна робота 9

Дослідження алгоритмів обходу масивів

Мета – дослідити алгоритми обходу масивів, набути практичних навичок використання цих алгоритмів під час складання програмних специфікацій.

Варіант 5.

Розробити алгоритм та написати програму, яка складається з наступних дій:

1. Опису змінної індексованого типу (двовимірний масив) згідно з варіантом
2. Ініціювання змінної, що описана в п.1 даного завдання
3. Обчислення змінної, що описана в п.1, згідно з варіантом

5 Задано матрицю дійсних чисел $A[n,n]$, ініціалізувати матрицю обходом по рядках. Знайти суму елементів, розташованих нижче головній діагоналі матриці.

1. Постановка задачі

Результатом розв'язку є сума елементів, розташованих нижче головній діагоналі матриці.

2. Побудова математичної моделі

Таблиця змінних

Змінна	Тип	Ім'я	Призначення
Довжина квадратної матриці	Натуральний	n	Вхідні дані
Двовимірний масив	Цілий	matrix	Проміжне дане
Лічильник циклу	Натуральний	i	Проміжне дане
Лічильник вкладеного циклу	Натуральний	j	Проміжне дане
Знак ітератора	Цілий	direction	Проміжне дане
Елемент матриці	Цілий	counter	Проміжне дане
Сума елементів	Цілий	sum	Вихідні дані

Таблиця функцій

Назва	Синтаксис	Призначення
Отримання довжини виміру	GetLength(matrix, n)	Повертає довжину виміру n матриці matrix

3. Розв'язання

Програмні специфікації запишемо у псевдокодi та графічній формi у вигляді блок-схеми.

Крок 1. Визначимо основні дії

Крок 2. Деталізуємо дію оголошення змінних

Крок 3. Деталізуємо дію створення матриці та її заповнення по рядках

Крок 4. Деталізуємо дію знаходження суми елементів матриці, розташованих нижче головної діагоналі

Псевдокод

Крок 1

Підпрограма GenerateMatrix(n)

Створення матриці довжиною n та її заповнення по рядках

Все підпрограма

Підпрограма GetMatrixSum(matrix)

Знаходження суми елементів матриці, розташованих нижче головної діагоналі

Все підпрограма

Початок

Введення **n**

Оголошення змінних

Виведення **sum**

Кінець

Крок 2

Підпрограма GenerateMatrix(n)

Створення матриці довжиною n та її заповнення по рядках

Все підпрограма

Підпрограма GetMatrixSum(matrix)

Знаходження суми елементів матриці, розташованих нижче головної діагоналі

Все підпрограма

Початок

Введення **n**

int[,] matrix = GenerateMatrix(n)

sum = GetMatrixSum(matrix)

Виведення **sum**

Кінець

Крок 3

Підпрограма GenerateMatrix(n)

int[,] matrix = int[n, n]

direction = 1

counter = 0

для j від 0 до GetLength(matrix, 1) **повторити**

якщо direction > 0

то

i = 0

інакше

i = matrix.GetLength(matrix, 0) - 1

все якщо

для i поки GetLength(matrix, 0) && i >= 0 **повторити**

i = i + direction

counter = counter + 1

matrix[i, j] = counter

все повторити

direction = direction * -1

все потворити

Повернути matrix

Все підпрограма

Підпрограм GetMatrixSum(matrix)

Знаходження суми елементів матриці, розташованих нижче головної діагоналі

Все підпрограма

Початок

Введення **n**

int[,] matrix = GenerateMatrix(n)

sum = GetMatrixSum(matrix)

Виведення **sum**

Кінець

Крок 4

Підпрограма GenerateMatrix(n)

int[,] matrix = int[n, n]

direction = 1

counter = 0

для j від 0 до GetLength(matrix, 1) **повторити**

якщо direction > 0

то

i = 0

інакше

i = matrix.GetLength(matrix, 0) - 1

все якщо

для i поки GetLength(matrix, 0) && i >= 0 **повторити**

i = i + direction

counter = counter + 1

matrix[i, j] = counter

все повторити

direction = direction * -1

все потворити
Повернути matrix

Все підпрограма

Підпрограма GetMatrixSum(matrix)

sum = 0
для i від 0 до Length(matrix, 0) **повторити**
 для j від 0 до Length(matrix, 1) **повторити**
 якщо i < j
 то
 sum = sum + matrix[i,j]
 все якщо
 все повторити
все потворити
Повернути sum

Все підпрограма

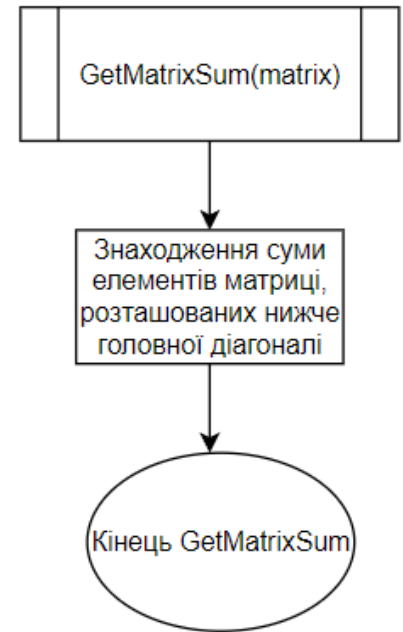
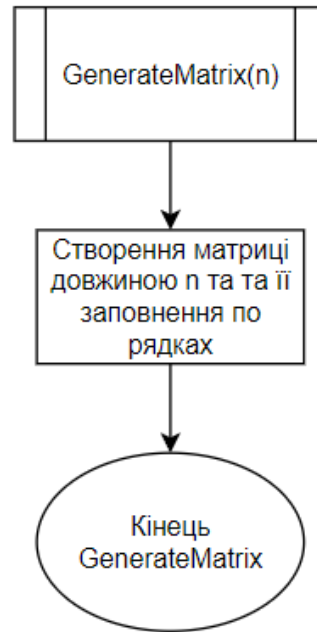
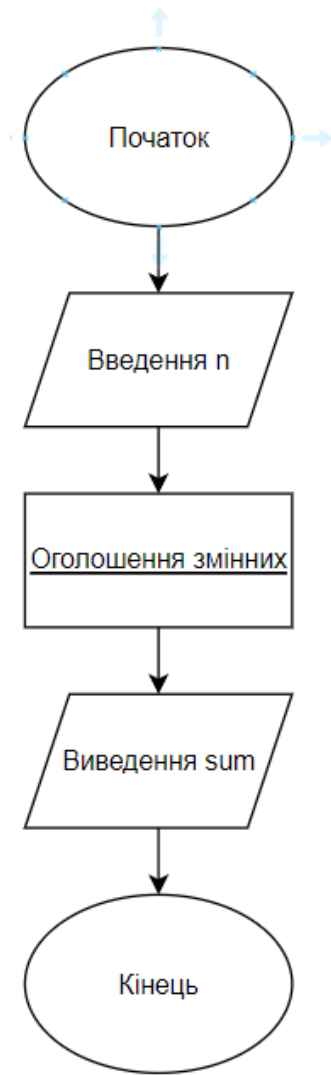
Початок

Введення **n**
int[,] matrix = GenerateMatrix(n)
sum = GetMatrixSum(matrix)
Виведення **sum**

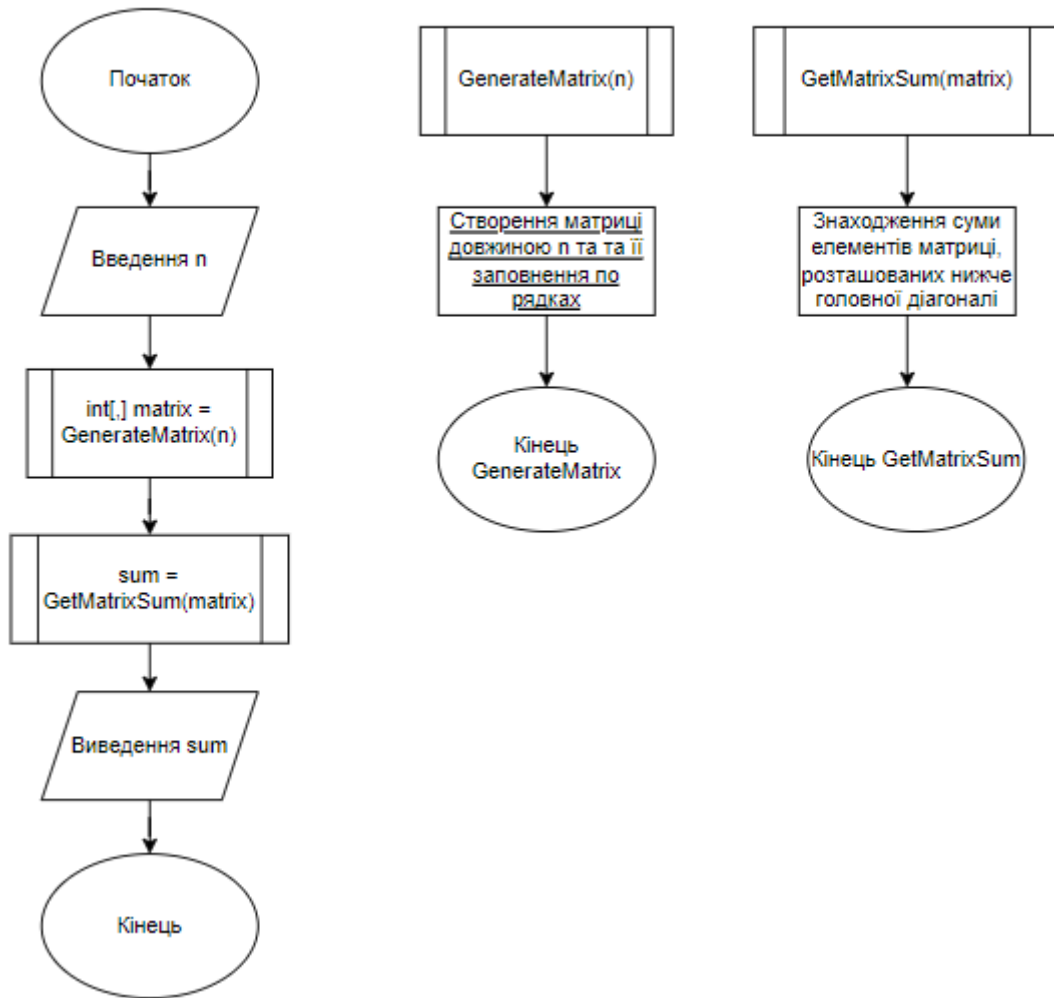
Кінець

Блок-схема

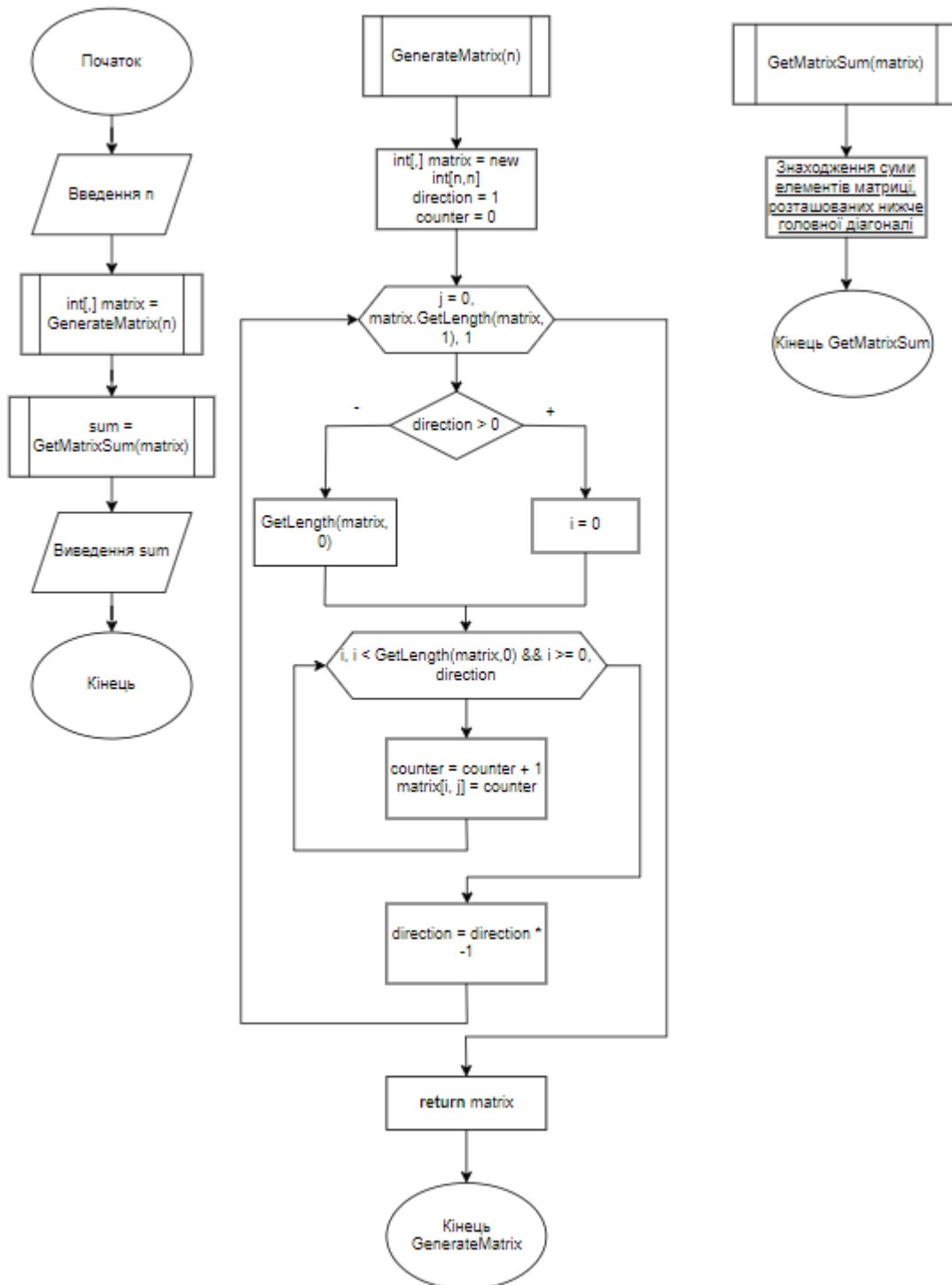
Крок 1

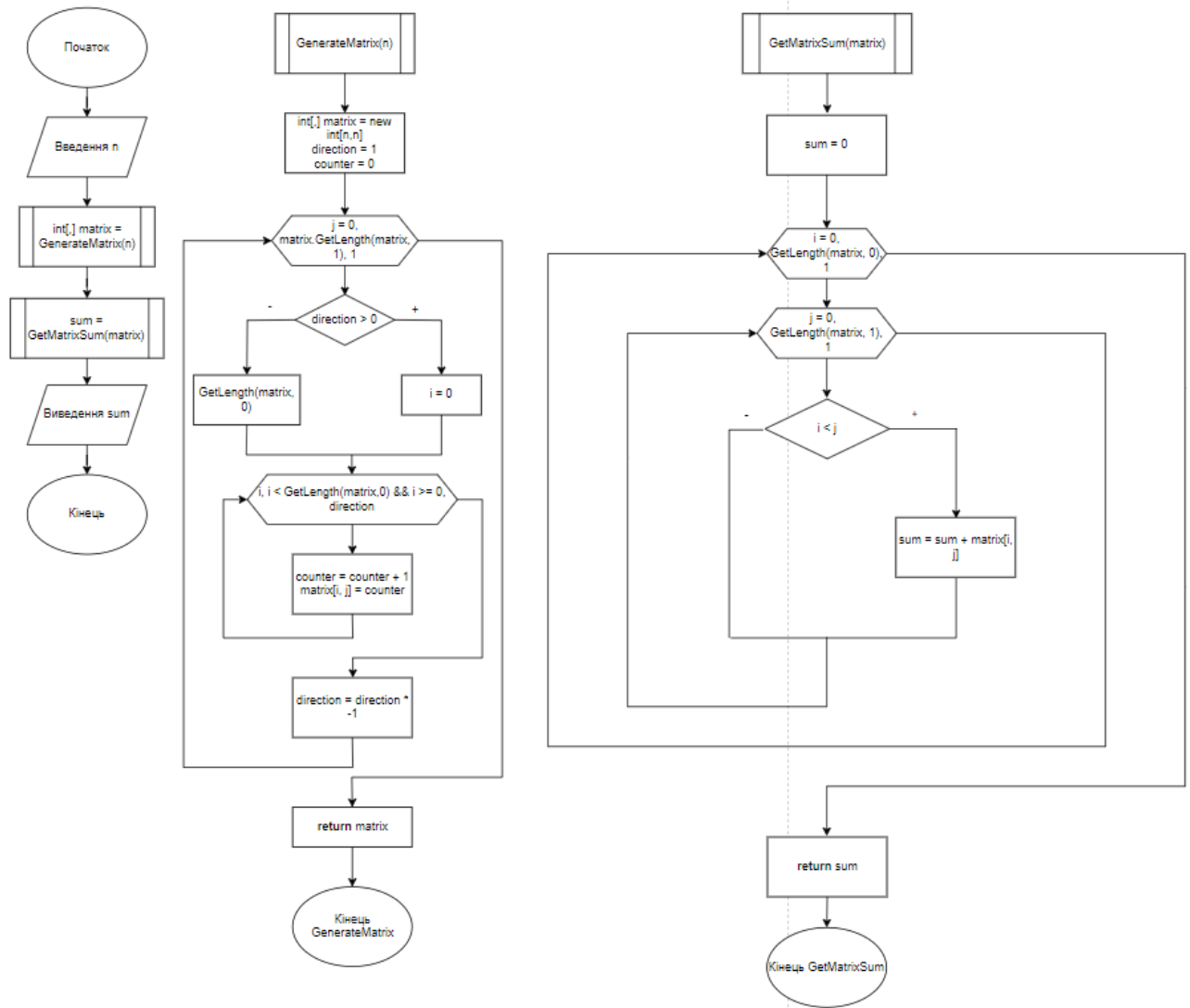


Крок 2



Крок 3





Код програми

```

class Program
{
    ссылка: 0
    static void Main(string[] args)
    {
        int n;
        Console.WriteLine("Enter matrix length: ");
        n = Convert.ToInt32(Console.ReadLine());

        int[,] matrix = GenerateMatrix(n);
        Console.WriteLine("Sum: " + GetMatrixSum(matrix));
    }

    ссылка: 1
    static int[,] GenerateMatrix(int n)
    {
        int[,] matrix = new int[n, n];
        int direction = 1;
        int counter = 0;


        for (int j = 0; j < matrix.GetLength(1); j++)
        {
            int i = (direction > 0) ? 0 : matrix.GetLength(0) - 1;
            for (; i < matrix.GetLength(0) && i >= 0; i += direction)
            {
                counter++;
                matrix[i, j] = counter;
            }

            direction *= -1;
        }
        return matrix;
    }
}
  
```

```

static int GetMatrixSum(int[,] matrix)
{
    int sum = 0;
    for (int i = 0; i < matrix.GetLength(0); i++)
    {
        for (int j = 0; j < matrix.GetLength(1); j++)
        {
            if (i < j)
                sum += matrix[i, j];
        }
    }
    return sum;
}
  
```

Результат роботи програми

 Консоль отладки Microsoft Visual Studio

```
Enter matrix length:  
5  
Sum: 180
```

Висновок

Виконуючи лабораторну роботу, я дослідив особливості алгоритмів обходу масивів, набув практичних навичок їх використання під час складання програмних специфікацій.