

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 8

Виконав студент ПІ-12 Волков Вадим Всеволодович
(шифр, прізвище, ім'я, по батькові)

Перевірів _____
(прізвище, ім'я, по батькові)

Київ 202 1

Лабораторна робота 9

Дослідження алгоритмів обходу масивів

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

Задача 8. Задано матрицю дійсних чисел $A[n,n]$, ініціалізувати матрицю обходом по рядках. На головній діагоналі матриці знайти перший від'ємний і останній додатний елементи, та поміняти їх місцями з елементами побічної діагоналі.

Побудова математичної моделі.

Змінна	Тип	Ім'я	Призначення
Початковий масив	Ціле[4][6]	matrix	Тимчасовий масив
Ітератор 1	Ціле	i	Тимчасова змінна
Ітератор 2	Ціле	j	Тимчасова змінна
Індех знайденого елемента	Ціле	found	Тимчасова змінна
Змінна для обміну	Ціле	tmp	Тимчасова змінна

Створивши квадратну матрицю (двовимірний масив) з ім'ям matrix, її буде заповнено випадковими значеннями обходом за рядками реалізованого за допомогою двох вкладених арифметичних циклів. Потім за допомогою арифметичних циклів буде знайдено індекси першого від'ємного та останнього додатного елементів і потім за допомогою цих індексів буде змінено значення цих елементів з елементами на відповідних їм місцях на головній діагоналі.

Розв'язання

Програмні специфікації запишемо у псевдокоді та графічній формі у вигляді блоксхеми.

Крок 1. Визначимо основні дії.

Крок 2. Деталізуємо заповнення матриці matrix випадковими значеннями

Крок 3. Деталізуємо обробку першого від'ємного елемента згідно умови

Крок 4. Деталізуємо знаходження першого від'ємного елемента

Крок 5. Деталізуємо обмін елементів основної та побічної діагоналі

Крок 6. Деталізуємо обробку останнього додатнього елемента згідно умови

Крок 7. Деталізуємо знаходження останнього додатнього елемента

Крок 8. Деталізуємо обмін елементів основної та побічної діагоналі

Псевдокод

крок 1

початок

Задання двовимірному масиву matrix[n][n]

Заповнення матриці matrix випадковими значеннями

Виведення matrix

Обробка першого від'ємного елемента згідно умови

Обробка останнього додатнього елемента згідно умови

Виведення matrix

кінець

крок 2

початок

Задання двовимірному масиву matrix[n][n]

виклик fillMatrix(matrix)

Виведення matrix

Обробка першого від'ємного елемента згідно умови

Обробка останнього додатнього елемента згідно умови

Виведення matrix

кінець

початок fillMatrix(matrix)

повторити для i від 1 до n із кроком 1

повторити для j від 1 до n із кроком 1

matrix[i][j] := random(-100, 100)

все повторювати

все повторювати

кінець

крок 3

початок

Задання двовимірного масиву `matrix[n][n]`

виклик `fillMatrix(matrix)`

Виведення `matrix`

виклик `findAndSwapFirstNeg(matrix)`

Обробка останнього додатнього елемента згідно умови

Виведення `matrix`

кінець

початок `fillMatrix(matrix)`

повторити для `i` від 1 до `n` із кроком 1

повторити для `j` від 1 до `n` із кроком 1

`matrix[i][j] := random(-100, 100)`

все повторювати

все повторювати

кінець

початок `findAndSwapFirstNeg(matrix)`

Знаходження першого від'ємного елемента

Обмін елементів основної та побічної діагоналі

кінець

крок 4

початок

Задання двовимірного масиву $matrix[n][n]$

виклик $fillMatrix(matrix)$

Виведення $matrix$

виклик $findAndSwapFirstNeg(matrix)$

Обробка останнього додатнього елемента згідно умови

Виведення $matrix$

кінець

початок $fillMatrix(matrix)$

повторити для i від 1 до n із кроком 1

повторити для j від 1 до n із кроком 1

$matrix[i][j] := random(-100, 100)$

все повторювати

все повторювати

кінець

початок $findAndSwapFirstNeg(matrix)$

$found := -1$

повторити для i від 1 до n із кроком 1

якщо $matrix[n][n] < 0$ та $found = -1$

то

$found := n$

все якщо

все повторювати

Обмін елементів основної та побічної діагоналі

кінець

крок 5

початок

Задання двовимірного масиву $matrix[n][n]$

виклик $fillMatrix(matrix)$

Виведення $matrix$

виклик $findAndSwapFirstNeg(matrix)$

Обробка останнього додатнього елемента згідно умови

Виведення $matrix$

кінець

початок $fillMatrix(matrix)$

повторити для i від 1 до n із кроком 1

повторити для j від 1 до n із кроком 1

$matrix[i][j] := random(-100, 100)$

все повторювати

все повторювати

кінець

початок $findAndSwapFirstNeg(matrix)$

$found := -1$

повторити для i від 1 до n із кроком 1

якщо $matrix[n][n] < 0$ та $found = -1$

то

$found := n$

все якщо

все повторювати

$tmp := matrix[found][found]$

$matrix[found][found] := matrix[found][n - 1 - found]$

$matrix[found][n - 1 - found] := tmp$

кінець

крок 6

початок

Задання двовимірного масиву $matrix[n][n]$

виклик $fillMatrix(matrix)$

Виведення $matrix$

виклик $findAndSwapFirstNeg(matrix)$

виклик $findAndSwapLastPos(matrix)$

Виведення $matrix$

кінець

початок $fillMatrix(matrix)$

повторити для i від 1 до n із кроком 1

повторити для j від 1 до n із кроком 1

$matrix[i][j] := random(-100, 100)$

все повторювати

все повторювати

кінець

початок $findAndSwapFirstNeg(matrix)$

$found := -1$

повторити для i від 1 до n із кроком 1

якщо $matrix[n][n] < 0$ та $found = -1$

то

$found := n$

все якщо

все повторювати

$tmp := matrix[found][found]$

$matrix[found][found] := matrix[found][n - 1 - found]$

$matrix[found][n - 1 - found] := tmp$

кінець

початок $findAndSwapLastPos(matrix)$

Знаходження останнього додатнього елемента

Обмін елементів основної та побічної діагоналі

кінець

крок 7

початок

Задання двовимірного масиву $matrix[n][n]$

виклик $fillMatrix(matrix)$

Виведення $matrix$

виклик $findAndSwapFirstNeg(matrix)$

виклик $findAndSwapLastPos(matrix)$

Виведення $matrix$

кінець

початок $fillMatrix(matrix)$

повторити для i **від** 1 **до** n **із кроком** 1

повторити для j **від** 1 **до** n **із кроком** 1

$matrix[i][j] := random(-100, 100)$

все повторювати

все повторювати

кінець

початок $findAndSwapFirstNeg(matrix)$

$found := -1$

повторити для i **від** 1 **до** n **із кроком** 1

якщо $matrix[n][n] < 0$ **та** $found = -1$

то

$found := n$

все якщо

все повторювати

$tmp := matrix[found][found]$

$matrix[found][found] := matrix[found][n - 1 - found]$

$matrix[found][n - 1 - found] := tmp$

кінець

початок $findAndSwapLastPos(matrix)$

$found := -1$

повторити для i **від** 1 **до** n **із кроком** 1

якщо $matrix[n][n] > 0$

то

$found := n$

все якщо

все повторювати

Обмін елементів основної та побічної діагоналі

кінець

крок 8

початок

Задання двовимірного масиву `matrix[n][n]`

виклик `fillMatrix(matrix)`

Виведення `matrix`

виклик `findAndSwapFirstNeg(matrix)`

виклик `findAndSwapLastPos(matrix)`

Виведення `matrix`

кінець

початок `fillMatrix(matrix)`

повторити для `i` **від** 1 **до** `n` **із кроком** 1

повторити для `j` **від** 1 **до** `n` **із кроком** 1

`matrix[i][j] := random(-100, 100)`

все повторювати

все повторювати

кінець

початок `findAndSwapFirstNeg(matrix)`

`found := -1`

повторити для `i` **від** 1 **до** `n` **із кроком** 1

якщо `matrix[n][n] < 0` **та** `found = -1`

то

`found := n`

все якщо

все повторювати

`tmp := matrix[found][found]`

`matrix[found][found] := matrix[found][n - 1 - found]`

`matrix[found][n - 1 - found] := tmp`

кінець

початок `findAndSwapLastPos(matrix)`

`found := -1`

повторити для `i` **від** 1 **до** `n` **із кроком** 1

якщо `matrix[n][n] > 0`

то

`found := n`

все якщо

все повторювати

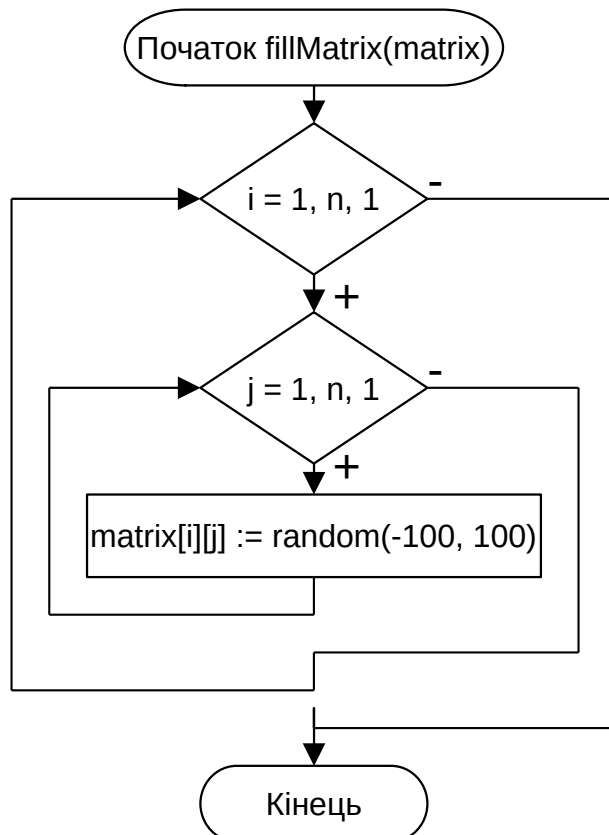
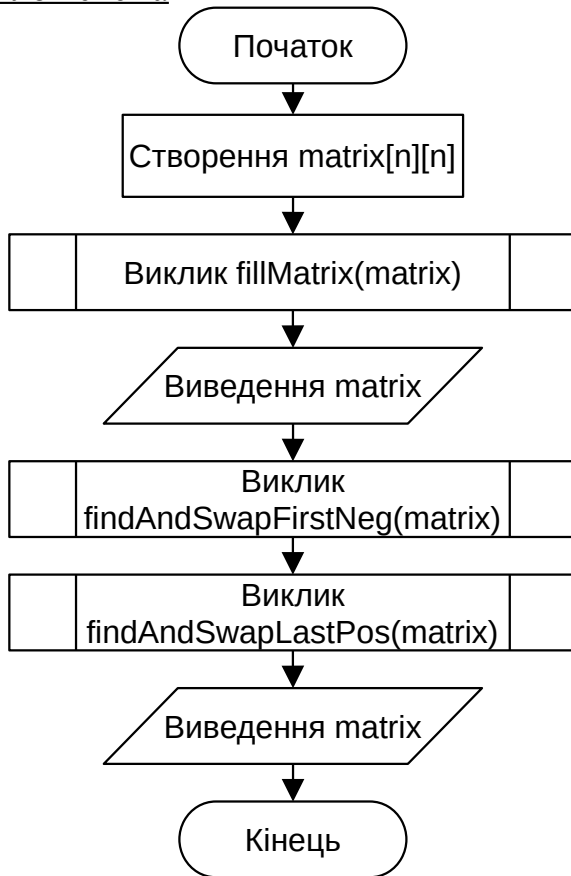
`tmp := matrix[found][found]`

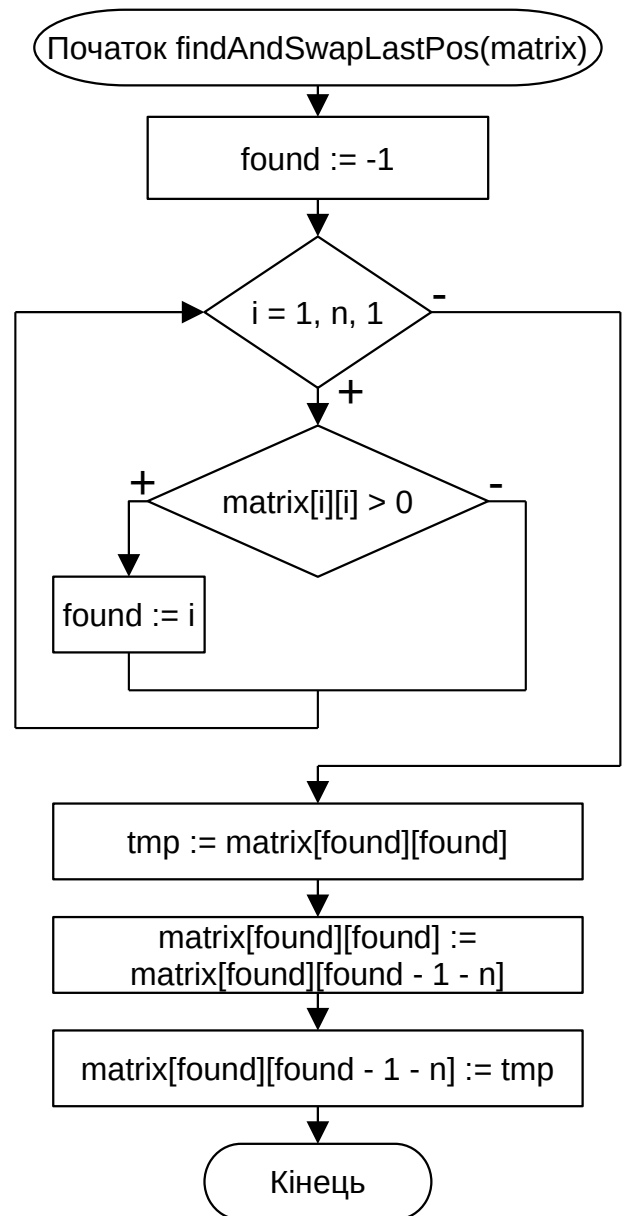
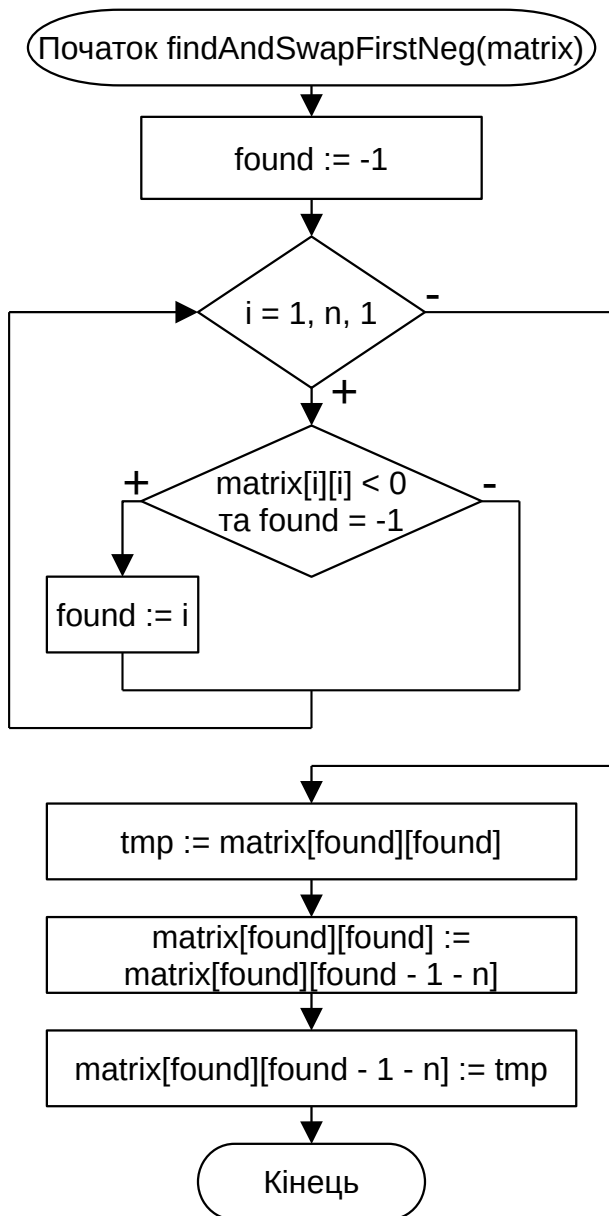
`matrix[found][found] := matrix[found][n - 1 - found]`

`matrix[found][n - 1 - found] := tmp`

кінець

Блок схема





Код програми:

C++:

```
#include <iostream>
```

```
const int n = 5;
```

```
void fillMatrix(int[n][n]);  
void findAndSwapFirstNeg(int[n][n]);  
void findAndSwapLastPos(int[n][n]);  
void printMatrix(int[n][n]);
```

```
int main() {  
    int matrix[n][n];  
    fillMatrix(matrix);  
    printMatrix(matrix);  
    findAndSwapFirstNeg(matrix);  
    findAndSwapLastPos(matrix);  
    printMatrix(matrix);  
}
```

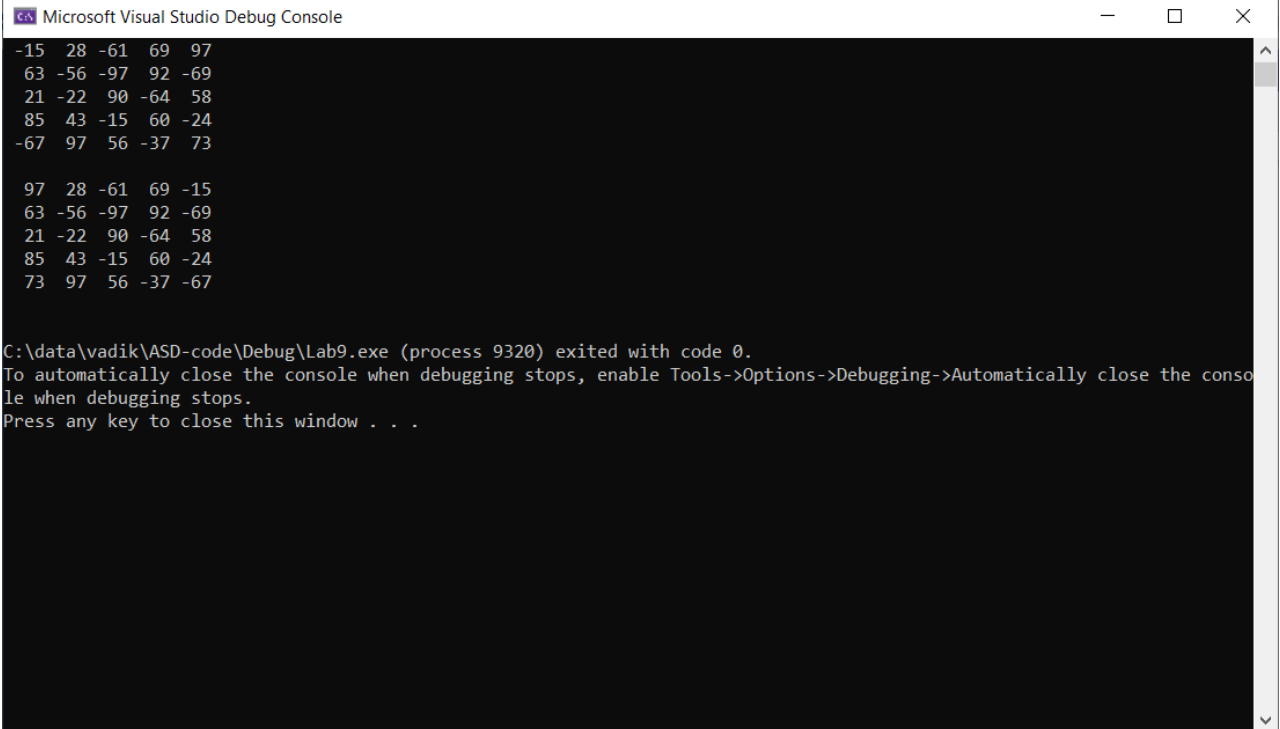
```
void fillMatrix(int matrix[n][n]) {  
    std::srand(std::time(nullptr));  
    for (int i = 0; i < n; i++) {  
        for (int j = 0; j < n; j++) {  
            matrix[i][j] = std::rand() % 201 - 100;  
        }  
    }  
}
```

```
void findAndSwapFirstNeg(int matrix[n][n]) {  
    int found = -1;  
    for (int i = 0; i < n; i++) {  
        if (matrix[i][i] < 0 && found == -1) found = i;  
    }  
    int tmp = matrix[found][found];  
    matrix[found][found] = matrix[found][n - 1 - found];  
    matrix[found][n - 1 - found] = tmp;  
}
```

```
void findAndSwapLastPos(int matrix[n][n]) {  
    int found = -1;  
    for (int i = 0; i < n; i++) {  
        if (matrix[i][i] > 0) found = i;  
    }  
    int tmp = matrix[found][found];  
    matrix[found][found] = matrix[found][n - 1 - found];  
    matrix[found][n - 1 - found] = tmp;  
}
```

```
void printMatrix(int matrix[n][n]) {  
    for (int i = 0; i < n; i++) {  
        for (int j = 0; j < n; j++) {  
            printf("%4i", matrix[i][j]);  
        }  
        std::cout << "\n";  
    }  
    std::cout << "\n";  
}
```

Перевіримо правильність програми запустивши її:



```
Microsoft Visual Studio Debug Console

-15  28 -61  69  97
 63 -56 -97  92 -69
 21 -22  90 -64  58
 85  43 -15  60 -24
-67  97  56 -37  73

 97  28 -61  69 -15
 63 -56 -97  92 -69
 21 -22  90 -64  58
 85  43 -15  60 -24
 73  97  56 -37 -67

C:\data\vadik\ASD-code\Debug\Lab9.exe (process 9320) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

Висновок:

На основі цього алгоритму, що обробляв двовимірні масиви (матриці), шукаючи в них перший від'ємний та останній додатний елемент на головній діагоналі та обмінює їх з відповідними елементами на побічній діагоналі було досліджено алгоритми обходу масивів та набуто практичних навичок використання цих алгоритмів під час складання програмних специфікацій.