

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 № 7 з дисципліни
«Алгоритми та структури даних-1.
Основи алгоритмізації»

«Дослідження алгоритмів
перетворення матриць та
отримання агрегатних значень»

Варіант 8

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

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

Лабораторна робота 8

Дослідження алгоритмів перетворення матриць та отримання агрегатних значень

Мета – дослідити підходи до пошуку та перетворення на матрицях та набути практичних навичок використання укладених керувальних дій повторення і їх з'єднання під час складання програмних специфікацій.

Задача 8. Задати масив цілих чисел розміром 4x6. Заповнити одновимірний масив максимальними значеннями елементів стовпців двовимірного масиву. Відсортувати одновимірний масив обміном за спаданням.

Побудова математичної моделі.

Змінна	Тип	Ім'я	Призначення
Початковий масив	Ціле[4][6]	matrix	Тимчасовий масив
Вихідний масив	Ціле[6]	arr	Результат
Ітератор 1	Ціле	i	Тимчасова змінна
Ітератор 2	Ціле	j	Тимчасова змінна
Ітератор 3	Ціле	k	Тимчасова змінна
Змінна для обміну	Ціле	tmp	Тимчасова змінна

Створивши двовимірний масив `matrix`, його буде заповнено випадковими значеннями. Потім, за допомогою двох вкладених арифметичних циклів познаходити найбільше число в кожному стовпці і позаписувати їх в відповідні елементи масива `arr`. Потім відсортувати `arr` за спаданням способом алгоритмом сортування вставкою.

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 заповнення *arr* відповідно до *matrix*

Крок 4. Деталізуємо проходження по всім стовбцям матриці та знаходження макимального елемента у стовбці

Крок 5. Деталізуємо знаходження макимального елемента у стовбці

Крок 6. Деталізуємо сортування масиву *arr*

Крок 7. Деталізуємо проходження по всім елементам масива

Крок 8. Деталізуємо зсув елемента до потрібного місця

Крок 9. Деталізуємо обмін елементів

Псевдокод

крок 1

початок

Задання двовимірного масиву *matrix*[4][6],

та одновимірного *arr*[6]

Заповнення матриці *matrix* випадковими значеннями

Заповнення *arr* відповідно до *matrix*

Сортування масиву *arr*

Виведення *matrix* та *arr*

кінець

крок 2

початок

Задання двовимірного масиву *matrix*[4][6],

та одновимірного *arr*[6]

fillMatrix(*matrix*)

Заповнення *arr* відповідно до *matrix*

Сортування масиву *arr*

Виведення *matrix* та *arr*

кінець

початок *fillMatrix*(*matrix*)

повторити для *i* від 0 до 3 із кроком 1

повторити для *j* від 0 до 5 із кроком 1

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

все повторювати

все повторювати

кінець

крок 3

початок

Задання двовимірного масиву `matrix[4][6]`,

та одновимірного `arr[6]`

`fillMatrix(matrix)`

`findMaxInColumn(matrix, arr)`

Сортування масиву `arr`

Виведення `matrix` та `arr`

кінець

початок `fillMatrix(matrix)`

повторити для `i` від 0 до 3 із кроком 1

повторити для `j` від 0 до 5 із кроком 1

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

все повторювати

все повторювати

кінець

початок `findMaxInColumn(matrix, arr)`

Проходження по всім стовбцям матриці та знаходження максимального
елемента у стовбці

кінець

крок 4

початок

Задання двовимірного масиву matrix[4][6],

та одновимірного arr[6]

fillMatrix(matrix)

findMaxInColumn(matrix, arr)

Сортування масиву arr

Виведення matrix та arr

кінець

початок fillMatrix(matrix)

повторити для і від 0 до 3 із кроком 1

повторити для j від 0 до 5 із кроком 1

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

все повторювати

все повторювати

кінець

початок findMaxInColumn(matrix, arr)

повторити для j від 0 до 5 із кроком 1

max := 0

Знаходження макмимального елемента у стовбці

arr[j] := max

все повторювати

кінець

крок 5

початок

Задання двовимірного масиву matrix[4][6],

та одновимірного arr[6]

fillMatrix(matrix)

findMaxInColumn(matrix, arr)

Сортування масиву arr

Виведення matrix та arr

кінець

початок fillMatrix(matrix)

повторити для і від 0 до 3 із кроком 1

повторити для j від 0 до 5 із кроком 1

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

все повторювати

все повторювати

кінець

початок findMaxInColumn(matrix, arr)

повторити для j від 0 до 5 із кроком 1

max := 0

повторити для і від 0 до 3 із кроком 1

якщо matrix[i][j] > max

то

max := matrix[i][j]

все якщо

все повторювати

arr[j] := max

все повторювати

кінець

крок 6

початок

Задання двовимірного масиву `matrix[4][6]`,

та одновимірного `arr[6]`

`fillMatrix(matrix)`

`findMaxInColumn(matrix, arr)`

`sortArray(arr)`

Виведення `matrix` та `arr`

кінець

початок `fillMatrix(matrix)`

повторити для `i` від 0 до 3 із кроком 1

повторити для `j` від 0 до 5 із кроком 1

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

все повторювати

все повторювати

кінець

початок `findMaxInColumn(matrix, arr)`

повторити для `j` від 0 до 5 із кроком 1

`max := 0`

повторити для `i` від 0 до 3 із кроком 1

якщо `matrix[i][j] > max`

то

`max := matrix[i][j]`

все якщо

все повторювати

`arr[j] := max`

все повторювати

кінець

початок `sortArray(arr)`

Проходження по всім елементам масива

кінець

крок 7

початок

Задання двовимірного масиву matrix[4][6],

та одновимірного arr[6]

fillMatrix(matrix)

findMaxInColumn(matrix, arr)

sortArray(arr)

Виведення matrix та arr

кінець

початок fillMatrix(matrix)

повторити для і від 0 до 3 із кроком 1

повторити для j від 0 до 5 із кроком 1

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

все повторювати

все повторювати

кінець

початок findMaxInColumn(matrix, arr)

повторити для j від 0 до 5 із кроком 1

max := 0

повторити для і від 0 до 3 із кроком 1

якщо matrix[i][j] > max

то

max := matrix[i][j]

все якщо

все повторювати

arr[j] := max

все повторювати

кінець

початок sortArray(arr)

повторити для k від 0 до 5 із кроком 1

Зсув елемента до потрібного місця

все повторювати

кінець

крок 8

початок

Задання двовимірного масиву matrix[4][6],

та одновимірного arr[6]

fillMatrix(matrix)

findMaxInColumn(matrix, arr)

sortArray(arr)

Виведення matrix та arr

кінець

початок fillMatrix(matrix)

повторити для і від 0 до 3 із кроком 1

повторити для j від 0 до 5 із кроком 1

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

все повторювати

все повторювати

кінець

початок findMaxInColumn(matrix, arr)

повторити для j від 0 до 5 із кроком 1

max := 0

повторити для і від 0 до 3 із кроком 1

якщо matrix[i][j] > max

то

max := matrix[i][j]

все якщо

все повторювати

arr[j] := max

все повторювати

кінець

початок sortArray(arr)

повторити для k від 0 до 5 із кроком 1

j := k

повторити поки arr[k] > arr[k - 1] та k > 0

Обмін елементів

все повторювати

k := j

все повторювати

кінець

крок 9

початок

Задання двовимірного масиву matrix[4][6],

та одновимірного arr[6]

fillMatrix(matrix)

findMaxInColumn(matrix, arr)

sortArray(arr)

Виведення matrix та arr

кінець

початок fillMatrix(matrix)

повторити для і від 0 до 3 із кроком 1

повторити для j від 0 до 5 із кроком 1

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

все повторювати

все повторювати

кінець

початок findMaxInColumn(matrix, arr)

повторити для j від 0 до 5 із кроком 1

max := 0

повторити для і від 0 до 3 із кроком 1

якщо matrix[i][j] > max

то

max := matrix[i][j]

все якщо

все повторювати

arr[j] := max

все повторювати

кінець

початок sortArray(arr)

повторити для k від 0 до 5 із кроком 1

j := k

повторити поки arr[k] > arr[k - 1] **та** k > 0

tmp := arr[k]

arr[k] := arr[k-1]

arr[k-1] := tmp

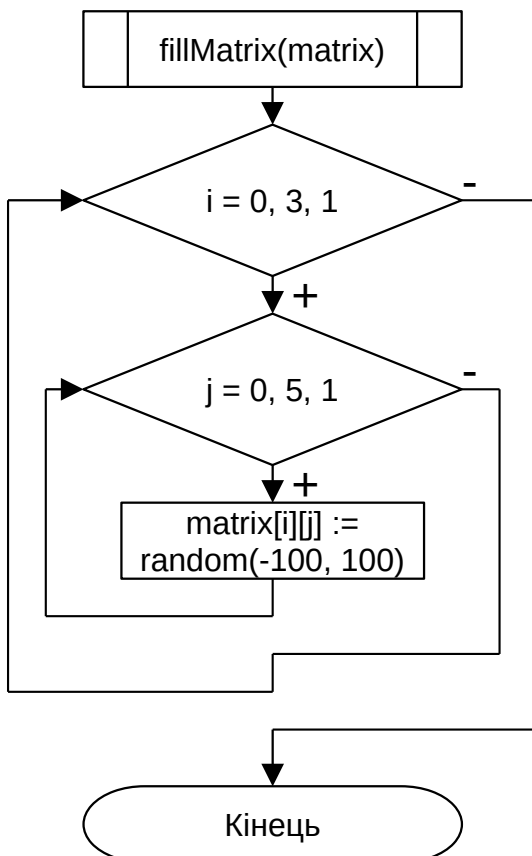
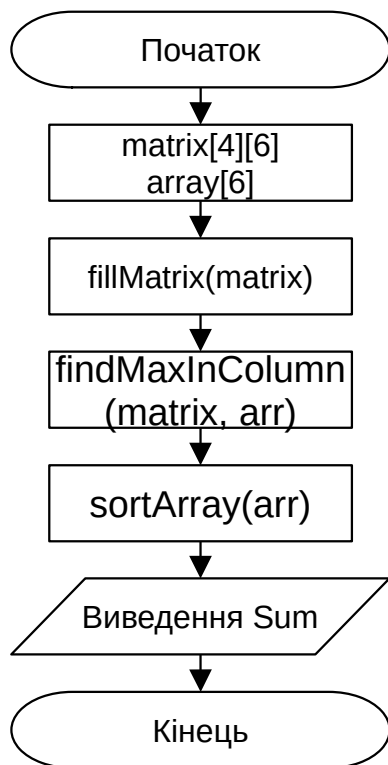
все повторювати

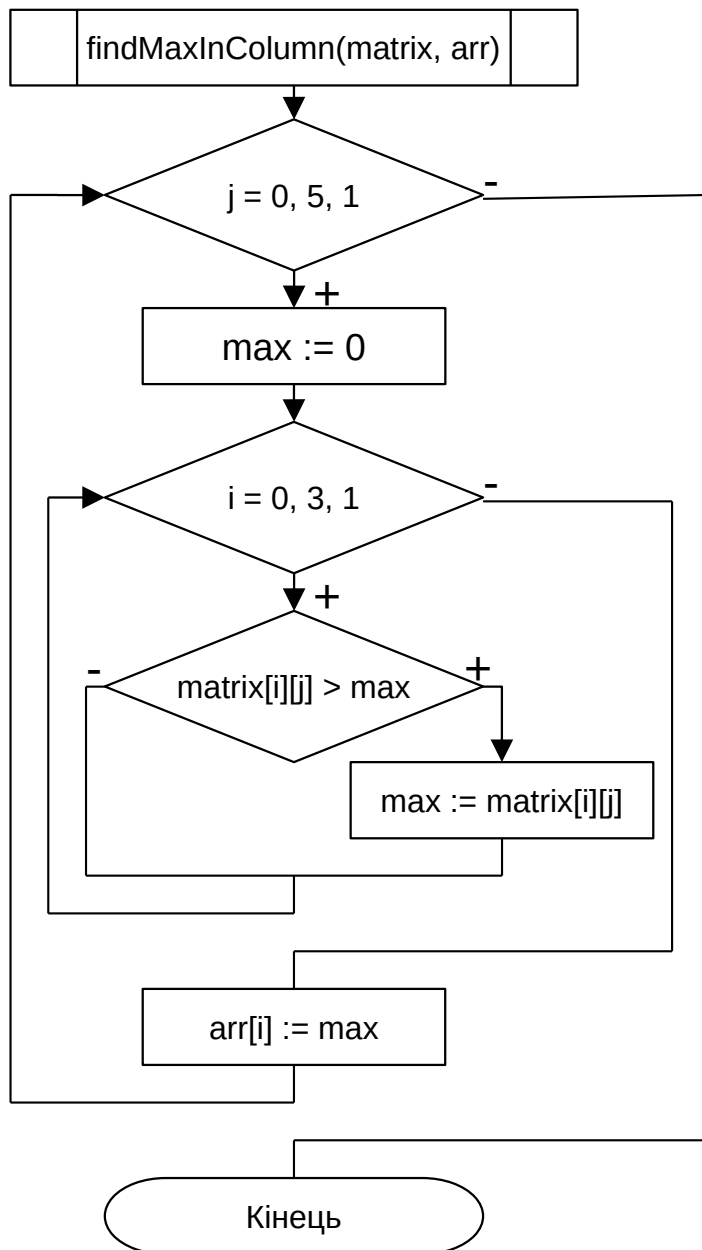
k := j

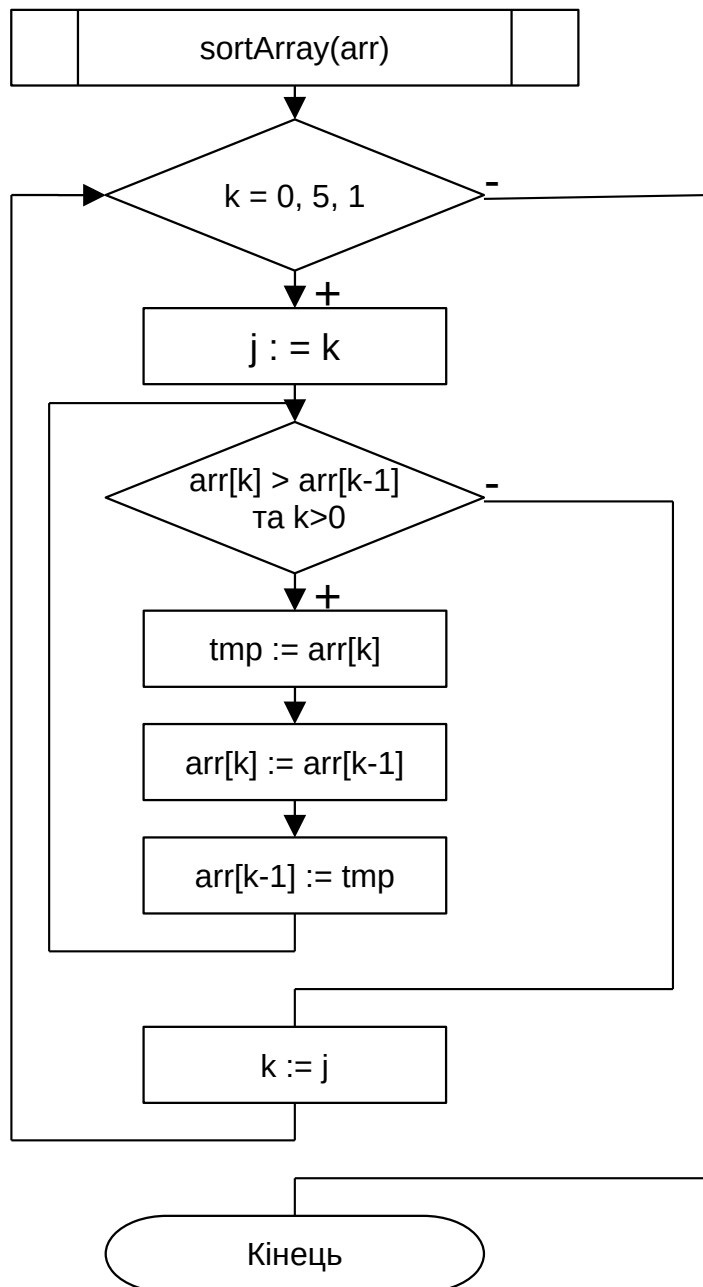
все повторювати

кінець

Блок схема







Код програми:

C++:

```
#include <iostream>

const int width = 6;
const int height = 4;

void fillMatrix(int[height][width]);
void findMaxInColumn(int[height][width], int[]);
void sortArray(int[]);
void printArray(int[]);
void printMatrix(int[height][width]);

int main() {
    int matrix[height][width];
    int arr[width];
    fillMatrix(matrix);
    findMaxInColumn(matrix, arr);
    sortArray(arr);
    printMatrix(matrix);
    printArray(arr);
}

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

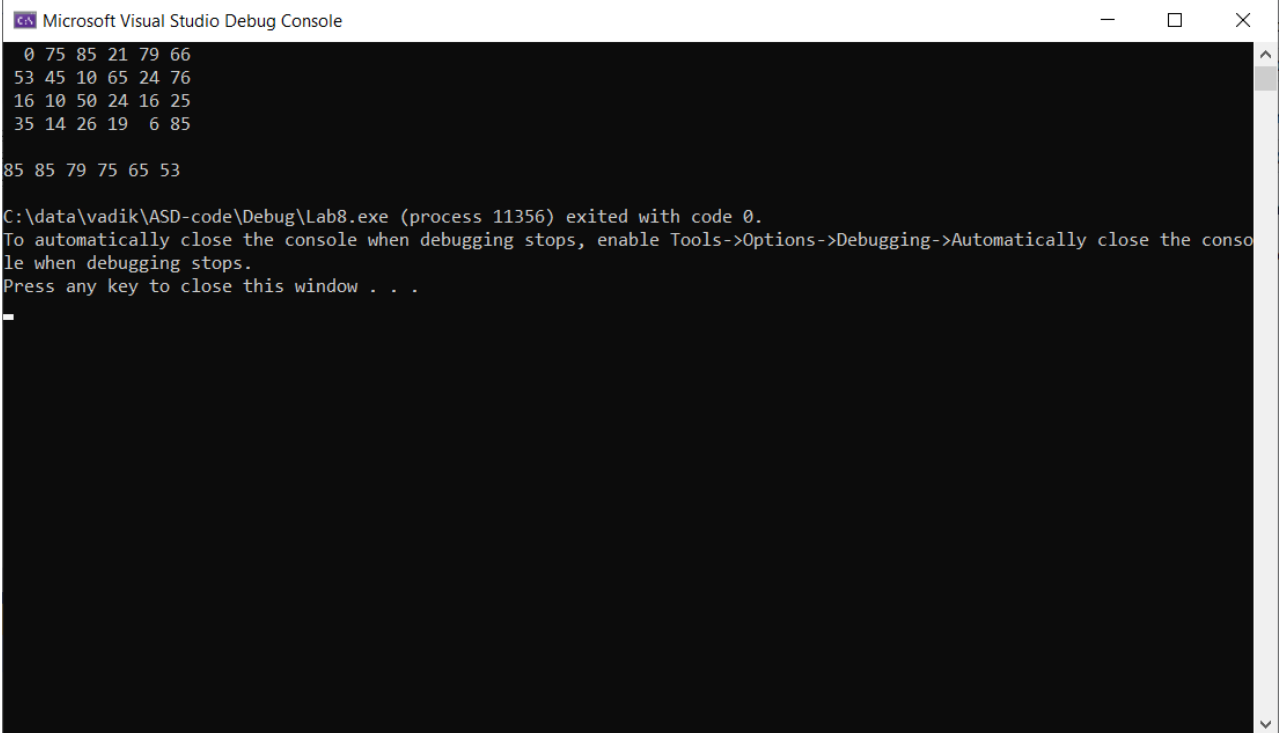
void findMaxInColumn(int matrix[height][width], int arr[]) {
    for (int j = 0; j < width; j++) {
        int max = 0;
        for (int i = 0; i < height; i++) {
            if (matrix[i][j] > max) max = matrix[i][j];
        }
        arr[j] = max;
    }
}

void sortArray(int arr[]) {
    for (int k = 1; k < width; k++) {
        int j = k;
        while (arr[k] > arr[k - 1] && k > 0) {
            int tmp = arr[k];
            arr[k] = arr[k - 1];
            arr[k - 1] = tmp;
            k--;
        }
        k = j;
    }
}

void printArray(int arr[]) {
    for (int i = 0; i < width; i++) {
        std::cout << arr[i] << " ";
    }
    std::cout << "\n";
}
```

```
void printMatrix(int matrix[height][width]) {  
    for (int i = 0; i < height; i++) {  
        for (int j = 0; j < width; j++) {  
            printf("%3i", matrix[i][j]);  
        }  
        std::cout << "\n";  
    }  
    std::cout << "\n";  
}
```

Перевіримо правильність програми запустивши її:



```
Microsoft Visual Studio Debug Console

0 75 85 21 79 66
53 45 10 65 24 76
16 10 50 24 16 25
35 14 26 19 6 85

85 85 79 75 65 53

C:\data\vadik\ASD-code\Debug\Lab8.exe (process 11356) 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 бробиав двовимірні масиви (матриці), шукаючи в них найбільший елемент кожного стопця, а потім сортуючи ці значення, було досліджено підходи до пошуку та перетворення на матрицях та набуто практичних навичок використання укладених керувальних дій повторення і їх з'єднання під час складання програмних специфікацій.