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 лінійного пошуку в послідовностях»

Варіант 19

Виконав студент: ІП-15 Левченко Владислав В'ячеславович

(шифр, прізвище, ім'я, по батькові)

Перевірила: Вечерковська Анастасія Сергіївна

(прізвище, ім'я, по батькові)

Київ 2021

19	4 x 8	Дійсний	Із добутку значень елементів стовпців двовимірного масиву. Відсортувати методом Шела за зростанням.

Мета – дослідити алгоритми пошуку та сортування, набути практичних навичок використання цих алгоритмів під час складання програмних специфікацій.

Побудова математичної моделі

Постановка задачі. Для вирішення цієї задачі створюємо двовимірний масив a [4][8] і b [8]. Створюємо функцію matrixGen(), яка генерує випадкові значення для масиву a [][], далі за допомогою функції printMatrix() виводимо утворений масив. Фунція findValue() шукає добуток стовпців масиву a [][] і записує їх у масив b []. Остання функція findResult() впорядковує елементи масиву b [] методом Шела за зростанням.

Таблиця імен змінних

Змінна	Тип	Ім’я	Призначення
Масив a[4][8]	Дійсне	a	Вхідне дане
Масив b[8]	Дійсне	b	Проміжне, вихідне дане
storage - накопичує добуток стовпців	Дійсне	storage	Проміжне дане
counter - крок у методі Шела	Ціле	counter	Проміжне дане
accumulator - зберігає значення і-го елементу метода у методі Шела	Дійсне	accumulator	Проміжне дане
row - рядки масиву a[][]	Ціле	row	Проміжне дане
column - стовпці масиву a[][]	Ціле	column	Проміжне дане

Крок 1. Визначимо основні дії

Крок 2. Створення функції, що генерує значення для масиву

Крок 3. Створення функції, що виводить отриманий масив

Крок 4. Створення функції, що шукає добуток стовпців

Крок 5. Створення функції, що впорядковує методом Шела

Крок 6. Створення основної програми

Псевдокод

Крок 1

початок

Створення функції, що генерує значення для масиву

Створення функції, що виводить отриманий масив

Створення функції, що шукає добуток стовпців

Створення функції, що впорядковує методом Шела

Створення основної програми

кінець

Крок 2

початок

Підпрогами:

matrixGen(float a [4][8])

для i **від** 0 **до** 4

для j **від** 0 **до** 8

a[i][j]: = (float(rand()) / float((RAND_MAX)) * 8 - 5)

все повторити

все повторити

повернути a [4][8]

Створення функції, що виводить отриманий масив

Створення функції, що шукає добуток стовпців

Створення функції, що впорядковує методом Шела

Створення основної програми

кінець

Крок 3

початок

Підпрогами:

matrixGen(float a [4][8])

для i **від** 0 **до** 4

для j **від** 0 **до** 8

a[i][j]: = (float(rand()) / float((RAND_MAX)) * 8 - 5)

повернути a [4][8]

printMatrix(float a [4][8])

для i **від** 0 **до** 4

для j **від** 0 **до** 8

виведення a [i][j]

все повторити

все повторити

Створення функції, що шукає добуток стовпців

Створення функції, що впорядковує методом Шела

Створення основної програми

кінець

Крок 4

початок

Підпрогами:

matrixGen(float a [4][8])

для i **від** 0 **до** 4

для j **від** 0 **до** 8

a[i][j]: = (float(rand()) / float((RAND_MAX)) * 8 - 5)

повернути a [4][8]

printMatrix(float a [4][8])

для i **від** 0 **до** 4

для j **від** 0 **до** 8

виведення a [i][j]

все повторити

все повторити

findValue(float a [4][8], float b [8], int counter)

для col **від** 0 **до** 8

storage:= 1

для row **від** 0 **до** 4

storage *= a[row][col]

b[col] = storage

все повторити

все повторити

для i **від** 0 **до** 8

виведення b[i]

все повторити

повернення b [8]

Створення функції, що впорядковує методом Шела

Створення основної програми

кінець

Крок 4

початок

Підпрогами:

matrixGen(float a [4][8])

для i **від** 0 **до** 4

для j **від** 0 **до** 8

a[i][j]: = (float(rand()) / float((RAND_MAX)) * 8 - 5)

повернути a [4][8]

printMatrix(float a [4][8])

для i **від** 0 **до** 4

для j **від** 0 **до** 8

виведення a [i][j]

все повторити

все повторити

findValue(float a [4][8], float b [8], int counter)

для col **від** 0 **до** 8

storage:= 1

для row **від** 0 **до** 4

storage = storage * a[row][col]

b[col] = storage

все повторити

все повторити

для i **від** 0 **до** 8

виведення b[i]

все повторити

повернення b [8]

findResult(float b [8])

counter:= 8

для j **від** 0 **до** 6

i: = 0

counter: = counter / 2

якщо counter == 0

то counter = counter + 1

все якщо

повторити

якщо b[i] > b [i + counter]

то accumulator: = b[i]

b [i]: = b [i + counter]

b [i + counter]: = accumulator

все якщо

i: = i + 1

поки i + counter < 8

повернути b[8]

Створення основної програми

кінець

Крок 5

початок

Підпрогами:

matrixGen(float a [4][8])

для i **від** 0 **до** 4

для j **від** 0 **до** 8

a[i][j]: = (float(rand()) / float((RAND_MAX)) * 8 - 5)

все повторити

все повторити

повернути a [4][8]

printMatrix(float a [4][8])

для i **від** 0 **до** 4

для j **від** 0 **до** 8

виведення a [i][j]

все повторити

все повторити

findValue(float a [4][8], float b [8], int counter)

для col **від** 0 **до** 8

storage:= 1

для row **від** 0 **до** 4

storage: = storage * a[row][col]

b[col]: = storage

все повторити

все повторити

для i **від** 0 **до** 8

виведення b[i]

все повторити

повернення b [8]

findResult(float b [8])

counter:= 8

для j **від** 0 **до** 6

i: = 0

counter: = counter / 2

якщо counter == 0

то counter = counter + 1

все якщо

повторити

якщо b[i] > b [i + counter]

то accumulator: = b[i]

b [i]: = b [i + counter]

b [i + counter]: = accumulator

все якщо

i: = i + 1

поки i + counter < 8

повернути b[8]

Основна програма:

row: = 4

cloumn: = 8

a [row][column]: = {0}

b [column]: = {0}

matrixGen(a)

printMatrix(a)

findValue(a, b, column)

findResult(b)

для i **від** 0 **до** 8

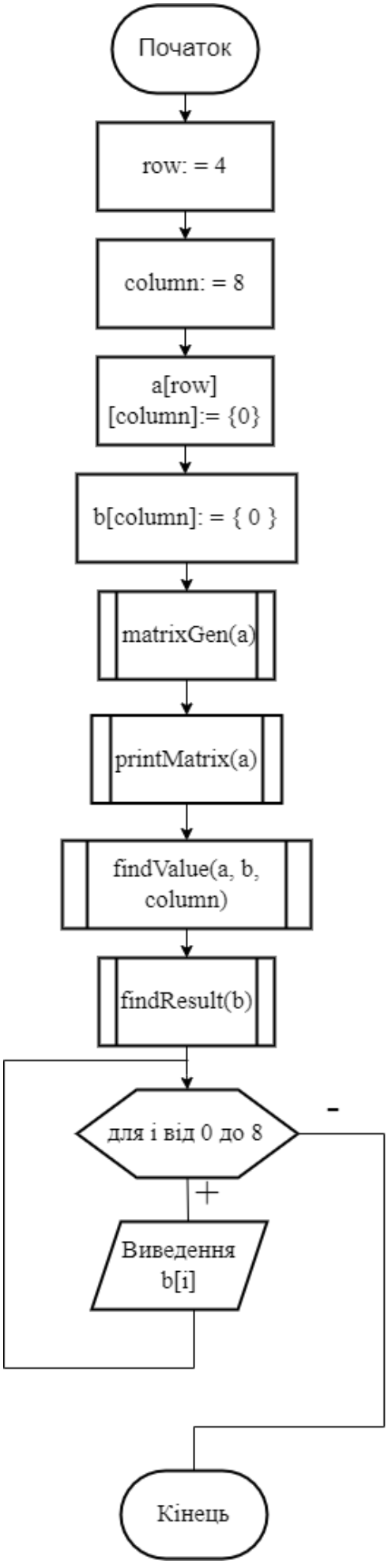
виведення b[i]

все повторити

кінець

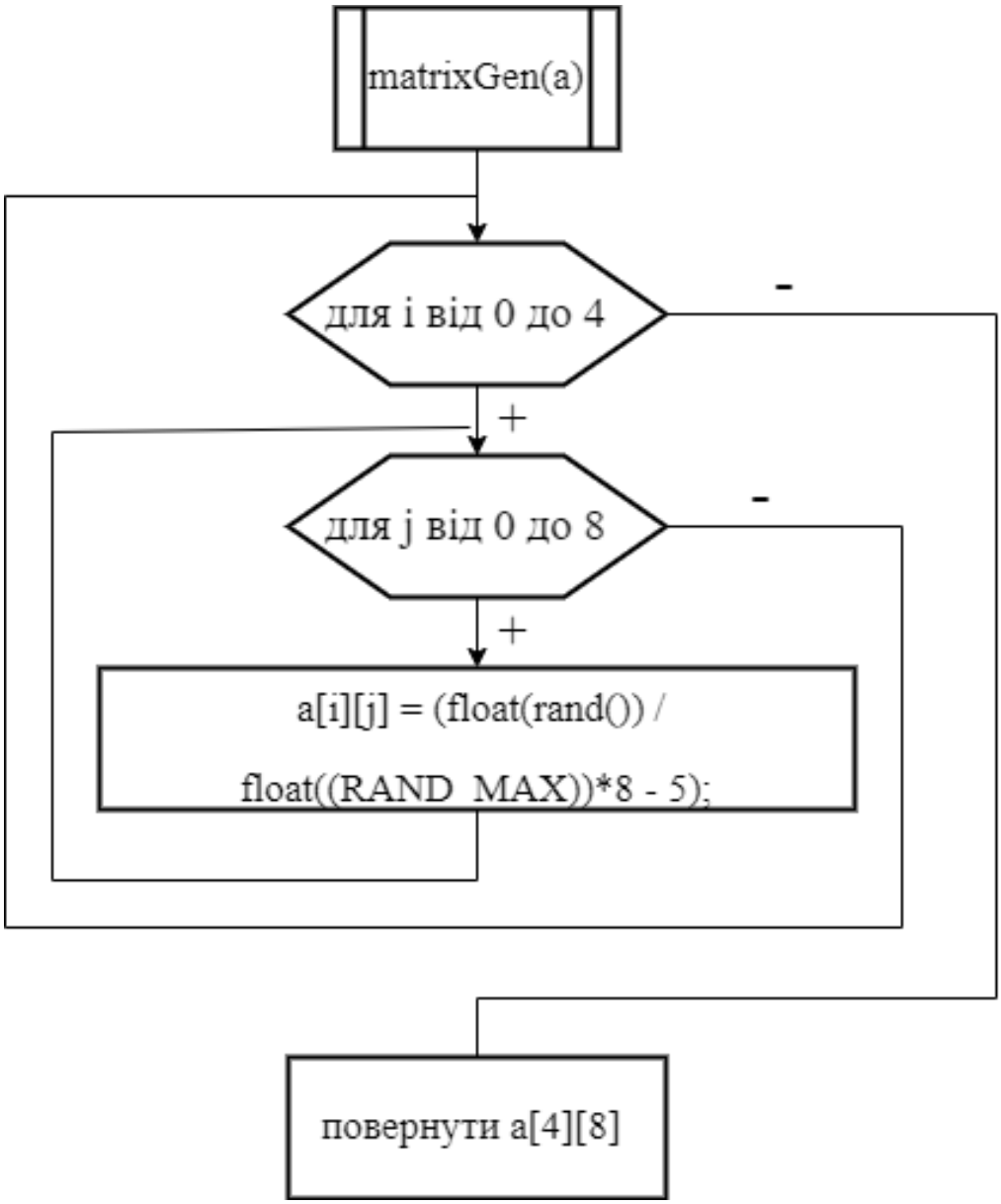
Блок схема

Основна програма:

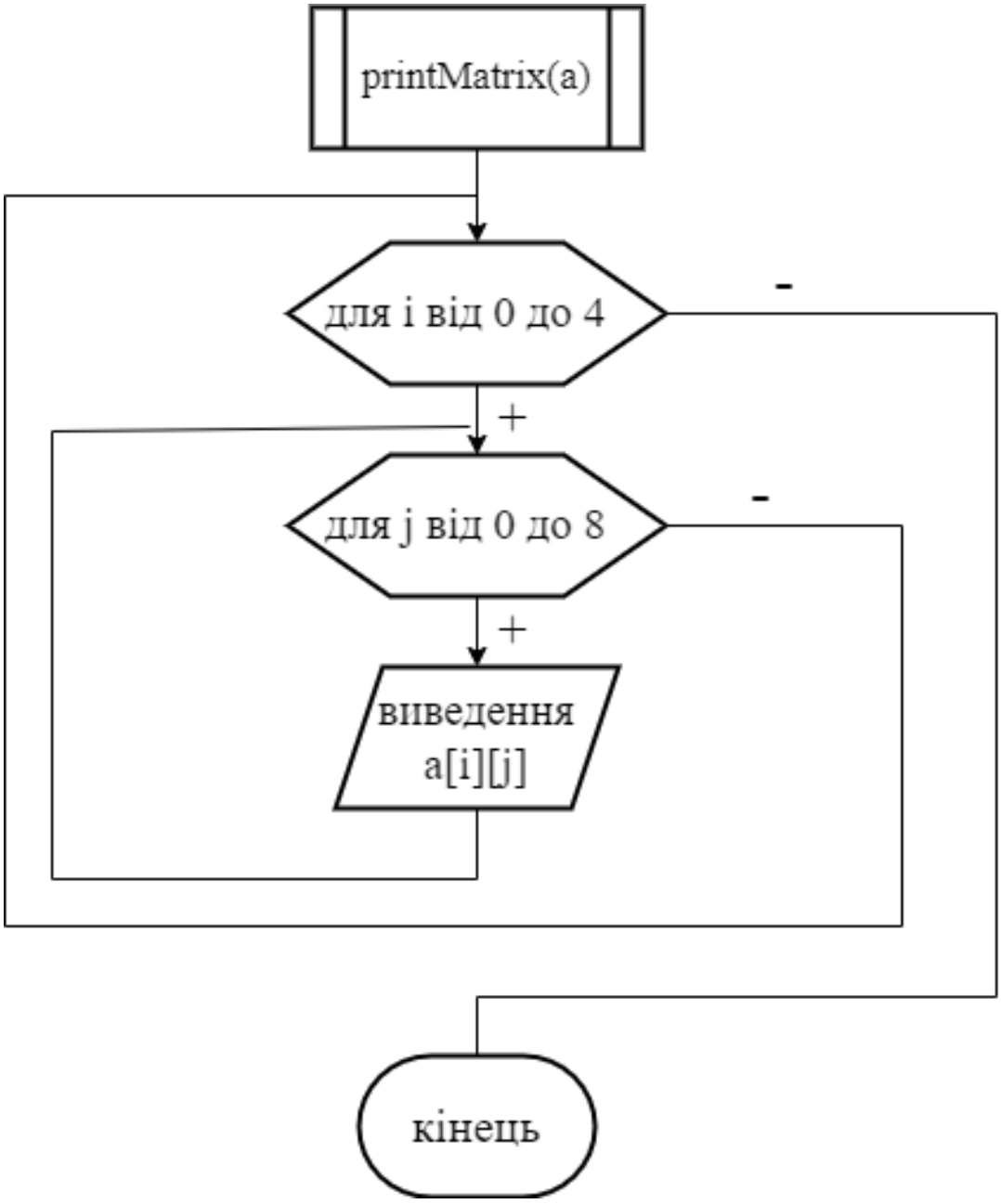

Початок
row: = 4
column: = 8
a[row]
[column]:= {0}
b[column]: = { 0 }
matrixGen(a)
printMatrix(a)
findValue(a, b,
column)
findResult(b)
для і від 0 до 8
-
+
Виведення
b[i]
Кінець

Підпрограми:

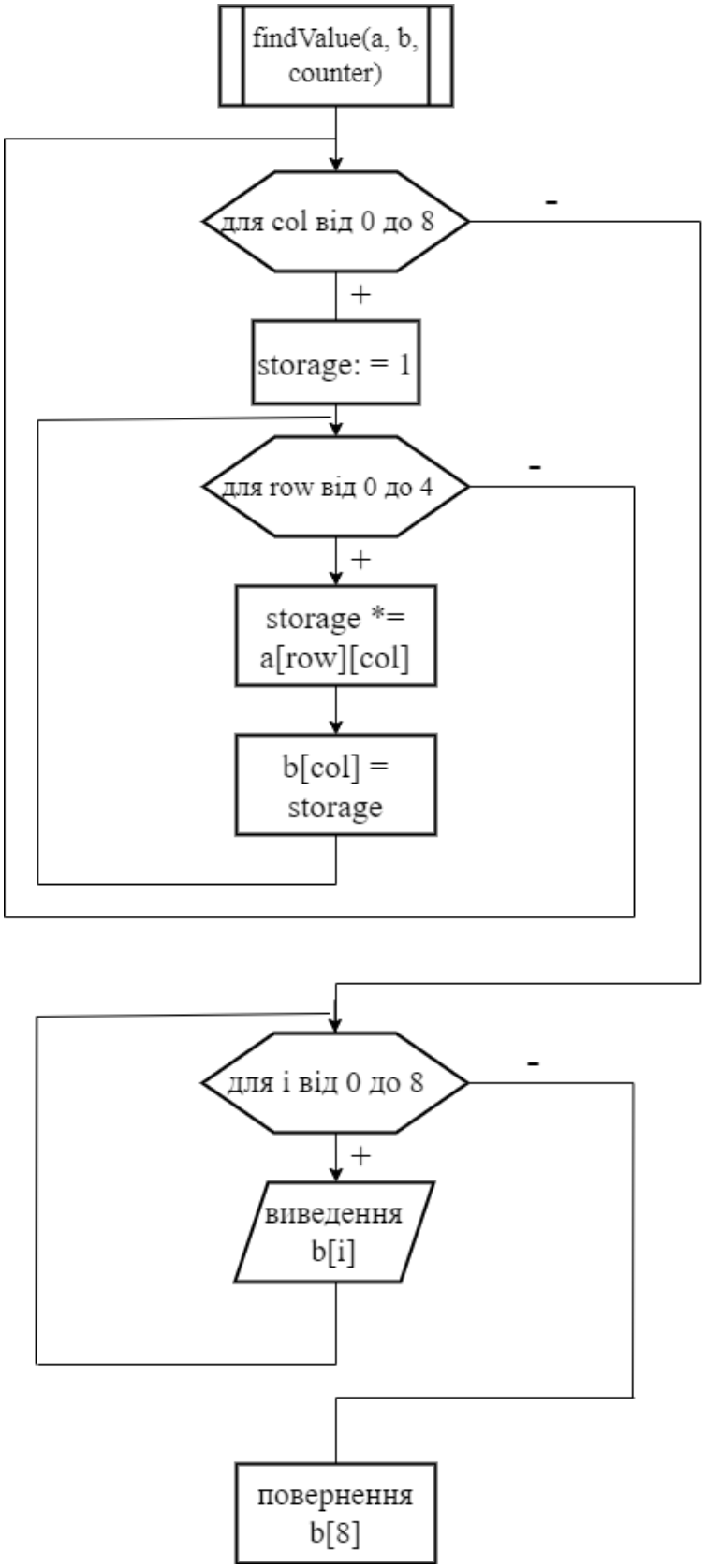
matrixGen() - генерує значення для масиву a [][]



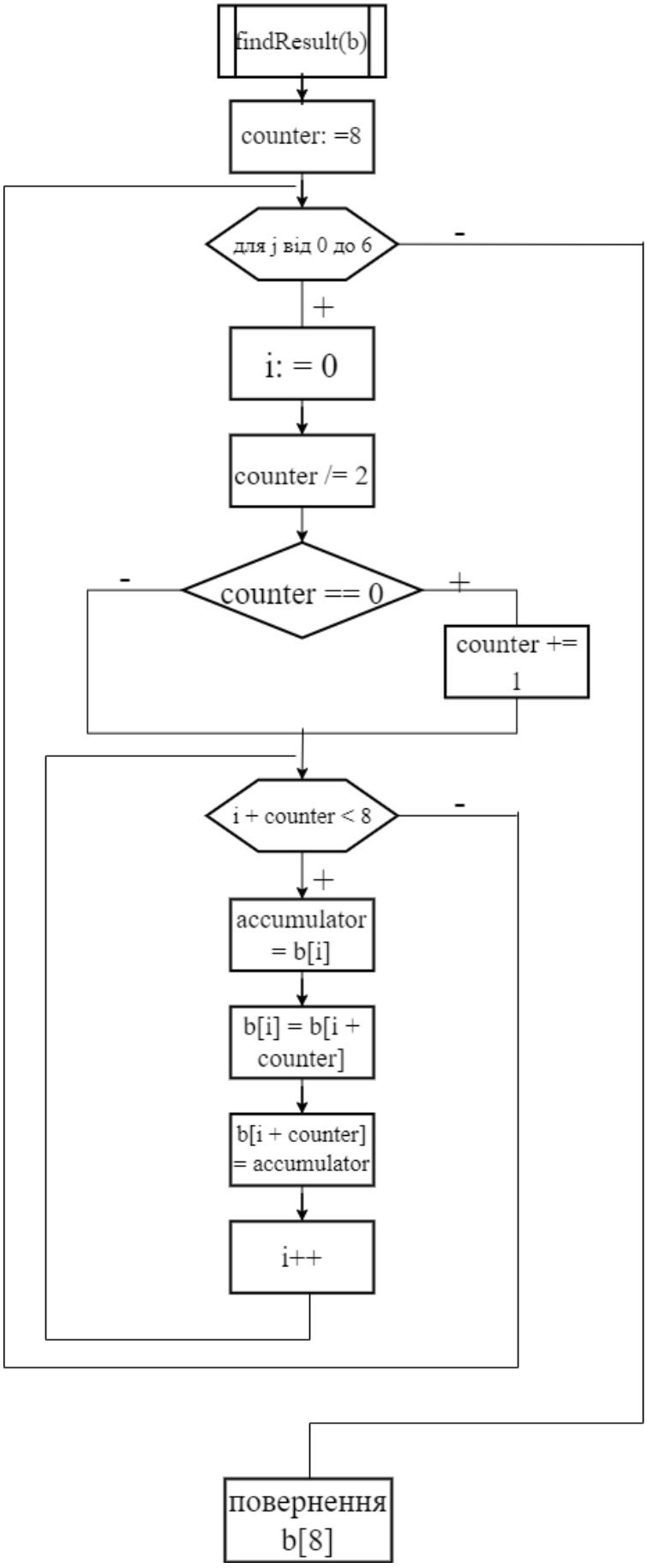
printMatrix() - виводить масив a [][]



findValue() - знаходить добуток стовпців матриці a [][] і зберігає його у поточний елемент масиву b[]



findResult() - сортує значення масиву b [] методом Шела


findResult(b)
counter: =8
для j від 0 до 6
-
+
i: = 0
counter /= 2
counter == 0
-
+
counter += 1
i + counter < 8
-
+
accumulator = b[i]
b[i] = b[i + counter]
b[i + counter] = accumulator
i++
повернення b[8]

Код програми

```cpp
#include <iostream>
#include <math.h>
#include <cstdio>
#include <cstdlib>

using namespace std;


float matrixGen(float a[4][8]) {

    srand(time(NULL));

    for (int i = 0; i < 4; i++) {
        for (int j = 0; j < 8; j++) {

            a[i][j] = (float(rand()) / float((RAND_MAX))*8 - 5);

        }
    }
    return a[4][8];
}

void printMatrix(float a[4][8]) {

    for (int i = 0; i < 4; i++) {
        std::cout << endl;
        for (int j = 0; j < 8; j++) {

            std::cout << a[i][j]<<" ";
        }
    }
    std::cout << endl << endl << endl;
}

float findValue(float a[4][8], float b[8], int counter) {

    float storage = 1;

    for (int col = 0; col < 8; col++) {

        storage = 1;

        for (int row = 0; row < 4; row++) {

            storage *= a[row][col];
            b[col] = storage;


        }
    }
```

```cpp
    std::cout << "Column multiplication result: " << endl;
    for (int i = 0; i < 8; i++) {
        std::cout<< b[i] << " ";
    }
    std::cout << endl;
    return b[8];
}

float findResult(float b[8]) {
    int counter = 8;
    int i;
    float accumulator;

    for (int j = 0; j < 6; j++) {

        i = 0;
        counter = counter / 2;
        if (counter == 0) { counter += 1; }
        do
        {
            if (b[i] > b[i + counter]) { accumulator = b[i]; b[i] = b[i + counter]; b[i + counter] = accumulator; }

          i++;

        } while (i + counter < 8);
    }
    return b[8];
}

int main()
{
    const int row = 4;
    const int column = 8;

    float a[row][column] = {0};

    float b[column] = { 0 };

    matrixGen(a);
    printMatrix(a);
    findValue(a, b, column);
    findResult(b);

    std::cout << "Array after Shell sort: " << endl;
    for (int i = 0; i < 8; i++) {
        std::cout << b[i] << " ";
    }

    cin.ignore(1, '\n');
    cin.get();
}
```

Тестування програми

```
-3.53066 1.24033 1.9639 -4.23435 -2.29865 -1.40461 -2.86541 0.378338
-3.01453 -3.11664 -1.66759 -0.847041 0.89938 -2.59795 -1.92886 0.823969
-0.0899382 -3.75774 -2.89724 -2.4388 -4.26832 1.08908 0.391156 -1.0784
1.84725 -4.46678 -4.4546 -0.575304 -1.84277 -1.00449 1.43196 1.59218


Column multiplication result:
-1.76826 -64.8854 -42.2671 5.03227 -16.2609 -3.99202 3.09577 -0.535259
Array after Shell sort:
-64.8854 -42.2671 -16.2609 -3.99202 -1.76826 -0.535259 3.09577 5.03227
```

Висновки. Ми навчилися використовувати алгоритми пошуку, сортування та працювати з двовимірними масивами. Створивши випадково

згенеровані числа, ми записали їх у одновимірний масив і відсортували методом Шела.