Міністерство освіти і науки України

Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"

Факультет інформатики та обчислювальної техніки

Катедра інформатики та програмної інженерії

Звіт

з лабораторної роботи № 8 з дисципліни

«Алгоритми та структури даних-1.

Основи алгоритмізації»

«Дослідження алгоритмів пошуку та сортування»

Варіант 18

Виконав студент ІП-11 Лесів Владислав Ігорович

Перевірив Мартинова О.П.

Київ 2021

Лабораторна робота 8

Дослідження алгоритмів пошуку та сортування

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

Варіант №18.

Завдання

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

1. Опису змінної індексованого типу (двовимірний масив) згідно з варіантом (табл. 1).
2. Ініціювання змінної, що описана в п.1 даного завдання.
3. Створення нової змінної індексованого типу (одновимірний масив) та її ініціювання значеннями, що обчислюються згідно з варіантом (табл. 1).

№ варіанта	Розмірність	Тип даних	Обчислення значень елементів одновимірного масиву
18	5 x 5	Дійсний	Із від'ємних значень елементів побічної діагоналі двовимірного масиву. Відсортувати методом вставки за спаданням.

Постановка задачі. Результатом розв'язку є одновимірний масив, утворений з від'ємних значень елементів побічної діагоналі заданої матриці, та відсортований методом вставки за спаданням. Для визначення результату повинна бути задана матриця 5*5 з дійсних елементів . Інших початкових даних для розв'язку не потрібно.

Побудова математичної моделі. Складемо таблицю імен змінних.

Змінна	Тип	Ім'я	Призначення
Матриця	Дійсний	arr1	Початкове дане
Розмір утвореного масиву	Цілий	size	Проміжний результат
Розмір масиву (у функції diagArr)	Цілий	sizeArr	Проміжний результат

Поточний елемент сортування (у функції sortAndOutput)	Дійсний	elem	Проміжний результат
Перевірка відсортованості на поточній позиції	Логічний	k	Проміжний результат
Утворений масив	Дійсний	arr2	Кінцевий результат

Математичне формулювання задачі зводиться до знаходження від'ємних елементів на побічній діагоналі матриці, внесення їх до масиву та сортування методом вставки за спаданням: якщо поточний елемент більший за попередній, то місце поточного елемента заповнюємо попереднім, а попереднє – поточним; інакше переходимо до наступної ітерації.

Програмні специфікації запишемо у псевдокоді та графічній формі у вигляді блок-схеми.

Крок 1. Визначимо основні дії.

Крок 2. Деталізуємо дію ініціалізації заданої в умові задачі матриці за допомогою підпрограми.

Крок 3. Деталізуємо дію ініціалізації одновимірного масиву за умовою і знаходження розміру послідовності за допомогою підпрограми.

Крок 4. Деталізуємо дію виведення матриці за допомогою підпрограми.

Крок 5. Деталізуємо дію сортування і виведення масиву згідно з умовою задачі.

Псевдокод

крок 1

початок

ініціалізація матриці

ініціалізація одновимірного масиву і знаходження розміру

виведення матриці

сортування і виведення масиву

кінець

крок 2

початок

inputMatrix(arr1)

ініціалізація одновимірного масиву і знаходження розміру

виведення матриці

сортування і виведення масиву

кінець

підпрограма inputMatrix(arr1[5][5])

```
srand(time(NULL))
повторити
  для i від 0 до 4
  повторити
    для j від 0 до 4
      arr1[i][j] = rand() % 30 - 15;
  все повторити
все повторити
```

кінець підпрограми

крок 3

початок

inputMatrix(arr1)

size = diagArr(arr1, arr2)

виведення матриці

сортування і виведення масиву

кінець

підпрограма inputMatrix(arr1[5][5])

srand(time(NULL))

повторити

для i **від** 0 **до** 4

повторити

для i **від** 0 **до** 4

arr1[i][j] = rand() % 30 - 15;

все повторити

все повторити

кінець підпрограми

підпрограма diagArr(arr1[5][5], arr2[])

sizeArr:=0

повторити

для i **від** 0 **до** 4

якщо arr1[i][4-i] < 0

то

arr2[sizeArr] = arr1[i][4-i]

sizeArr:=sizeArr+1

все якщо

все повторити

повернути sizeArr

кінець підпрограми

крок 4

початок

inputMatrix(arr1)

size = diagArr(arr1, arr2)

outputMatrix(arr1)

сортування і виведення масиву

кінець

підпрограма inputMatrix(arr1[5][5])

srand(time(NULL))

повторити

для i **від** 0 **до** 4

повторити

для i **від** 0 **до** 4

arr1[i][j] = rand() % 30 - 15;

все повторити

все повторити

кінець підпрограми

підпрограма diagArr(arr1[5][5], arr2[])

sizeArr:=0

повторити

для i **від** 0 **до** 4

якщо arr1[i][4-i] < 0

то

arr2[sizeArr] = arr1[i][4-i]

sizeArr:=sizeArr+1

все якщо

все повторити

повернути sizeArr

кінець підпрограми

підпрограма outputMatrix(arr1[5][5])

повторити

для i **від** 0 **до** 4

повторити

для j **від** 0 **до** 4

виведення arr1[i][j]

все повторити

виведення “\n”

все повторити

кінець підпрограми

крок 5

початок

inputMatrix(arr1)

size = diagArr(arr1, arr2)

outputMatrix(arr1)

sortAndOutput(arr2, size)

кінець

```
підпрограма inputMatrix(arr1[5][5])
  srand(time(NULL))
  повторити
    для i від 0 до 4
    повторити
      для i від 0 до 4
        arr1[i][j] = rand() % 30 - 15;
    все повторити
  все повторити
кінець підпрограми

підпрограма diagArr(arr1[5][5], arr2[])
  sizeArr:=0
  повторити
    для i від 0 до 4
      якщо arr1[i][4-i] < 0
          то
            arr2[sizeArr] = arr1[i][4-i]
            sizeArr:=sizeArr+1
      все якщо
  все повторити
  повернути sizeArr
кінець підпрограми

підпрограма outputMatrix(arr1[5][5])
  повторити
    для i від 0 до 4
          повторити
```

```
        для j від 0 до 4
            виведення arr1[i][j]
        все повторити
        виведення “\n”
  все повторити
кінець підпрограми

підпрограма sortAndOutput(arr2, size);
  якщо (size == 0)
        то
          виведення “Відсутні від'ємні елементи, масив не утворюється”
        інакше
          якщо (size == 1)
                то
                  виведення arr2[0]
                інакше
                  повторити
                    для i від 1 до size-1
                    elem := arr2[i]
                    k :=0
                    повторити
                      для j від i-1 до 0
                      якщо (arr2[j] < elem && !k)
                          то
                            arr2[j + 1] := arr2[j]
                            arr2[j] := elem
                          інакше
                            k := 1
```

все якщо

все повторити

все повторити

повторити

для i **від** 0 **до** size-1

виведення arr2[i]

все повторити

все якщо

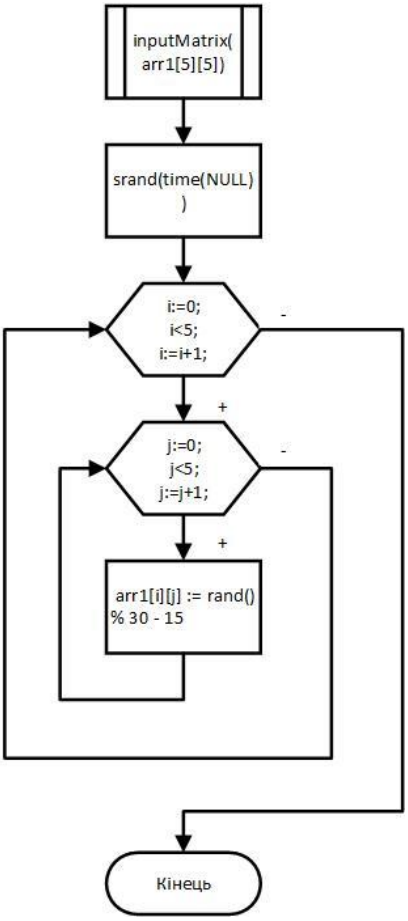
все якщо

Блок-схема

Крок 1

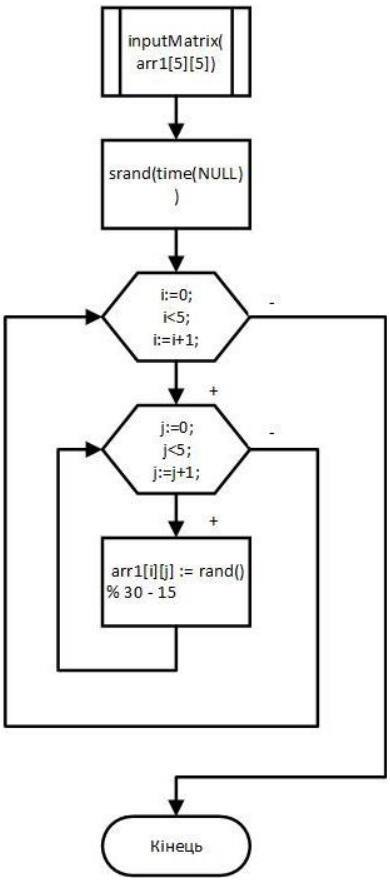
Крок 2

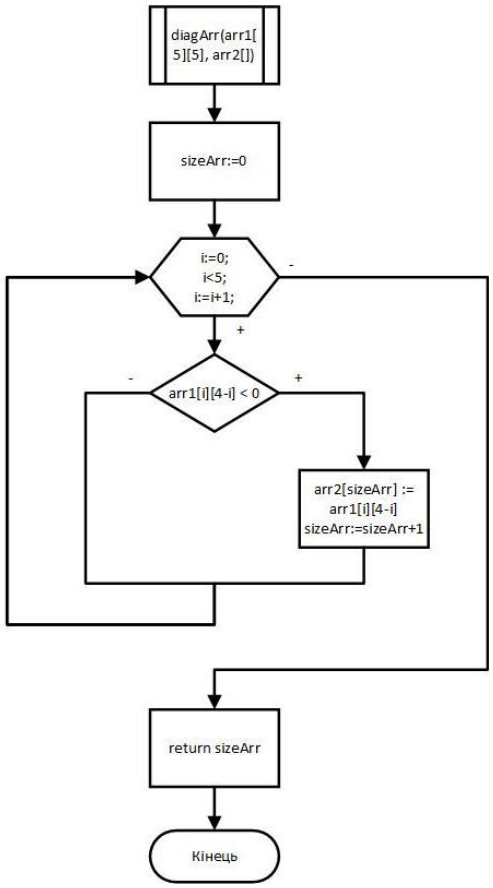

Початок
inputMatrix(arr1)
ініціалізація одновимірного масиву
виведення матриці
сортування й вивід масиву
Кінець


inputMatrix(arr1[5][5])
srand(time(NULL))
i:=0; i<5; i:=i+1;
j:=0; j<5; j:=j+1;
arr1[i][j] := rand() % 30 - 15
Кінець

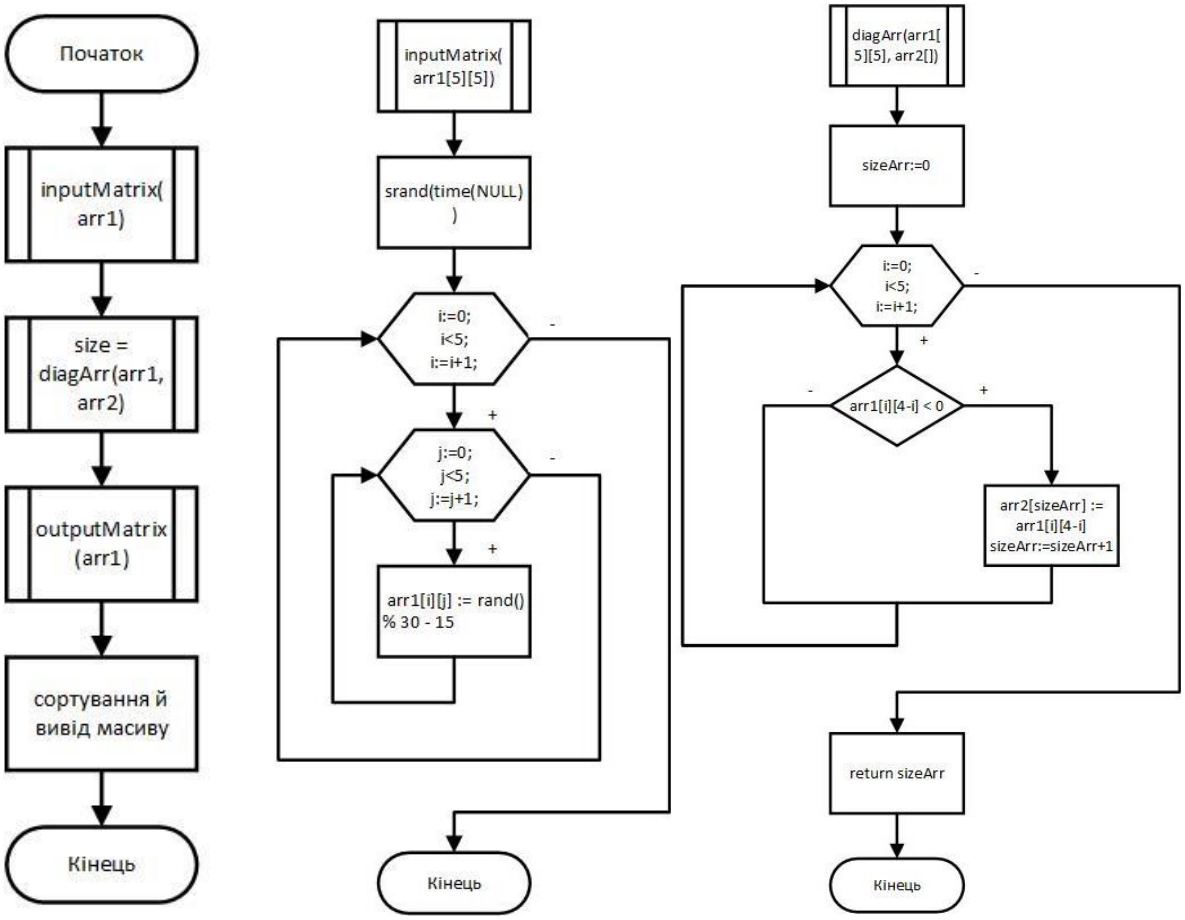
Крок 3

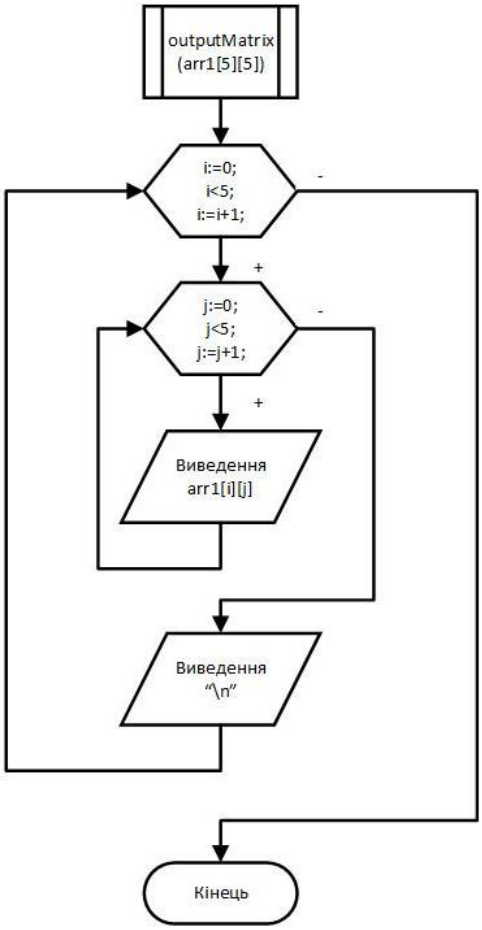

Початок
inputMatrix(arr1)
size = diagArr(arr1, arr2)
виведення матриці
сортування й вивід масиву
Кінець


inputMatrix(arr1[5][5])
srand(time(NULL))
i:=0; i<5; i:=i+1;
j:=0; j<5; j:=j+1;
arr1[i][j] := rand() % 30 - 15
Кінець

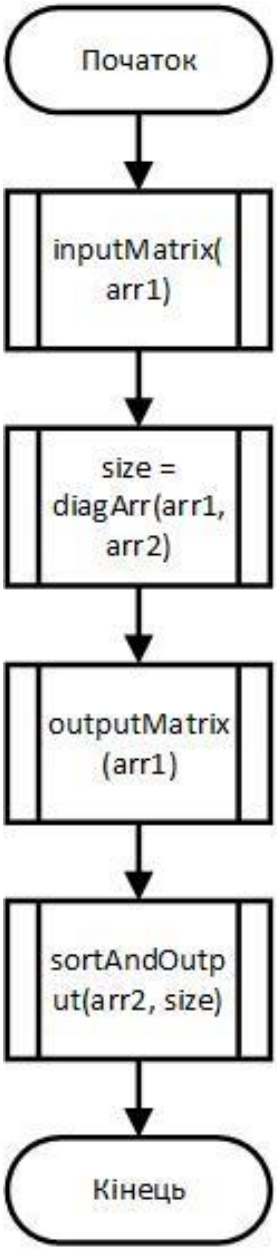

diagArr(arr1[5][5], arr2[])
sizeArr:=0
i:=0; i<5; i:=i+1;
arr1[i][4-i] < 0
arr2[sizeArr] := arr1[i][4-i] sizeArr:=sizeArr+1
return sizeArr
Кінець

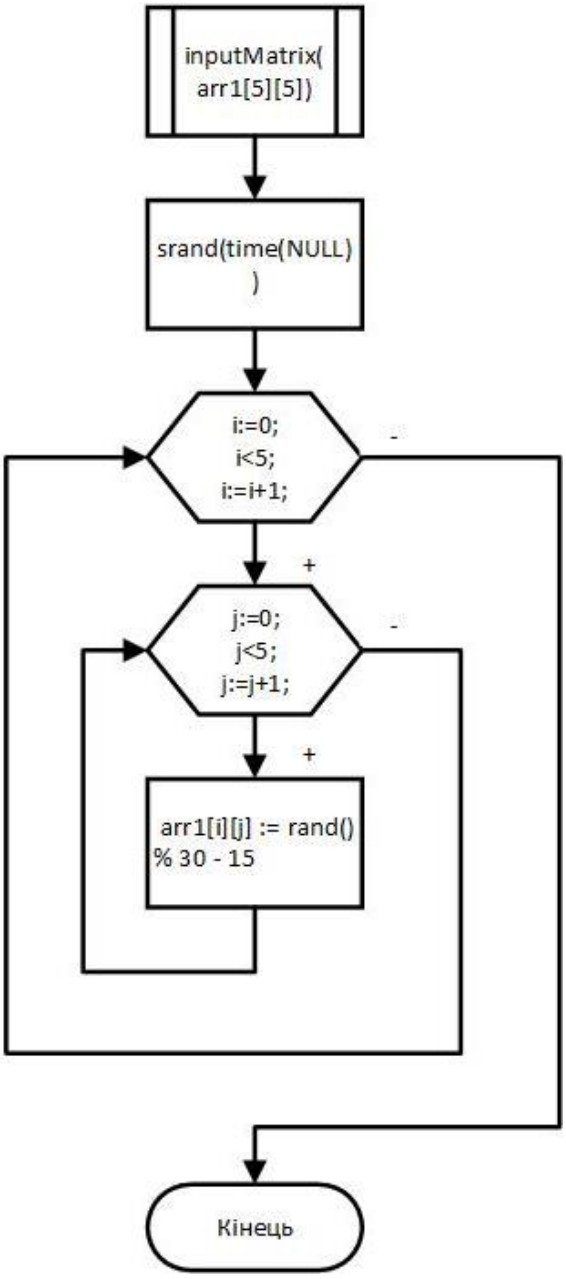
Крок 4

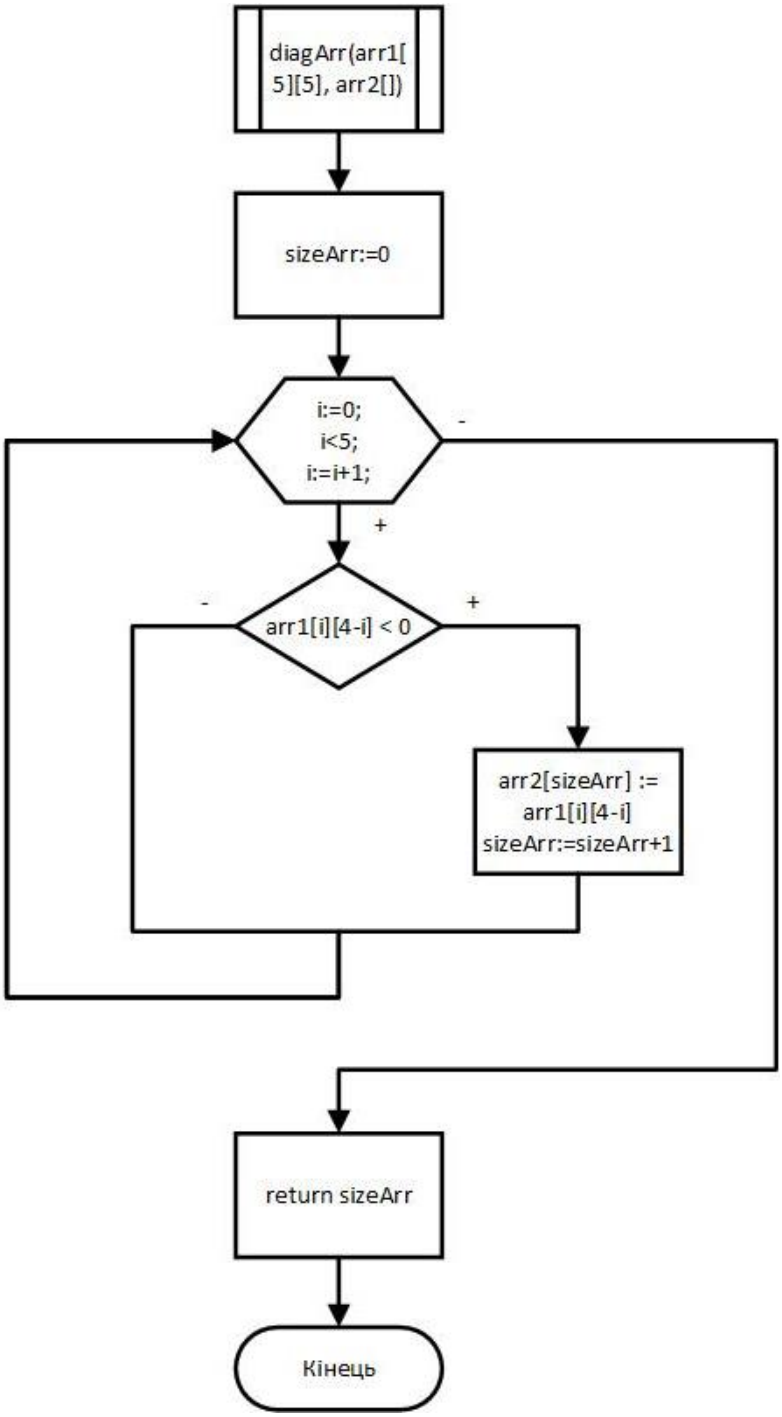

Початок
inputMatrix(arr1)
size = diagArr(arr1, arr2)
outputMatrix (arr1)
сортування й вивід масиву
Кінець
inputMatrix(arr1[5][5])
srand(time(NULL))
i:=0; i<5; i:=i+1;
j:=0; j<5; j:=j+1;
arr1[i][j] := rand() % 30 - 15
Кінець
diagArr(arr1[5][5], arr2[])
sizeArr:=0
i:=0; i<5; i:=i+1;
arr1[i][4-i] < 0
arr2[sizeArr] := arr1[i][4-i] sizeArr:=sizeArr+1
return sizeArr
Кінець

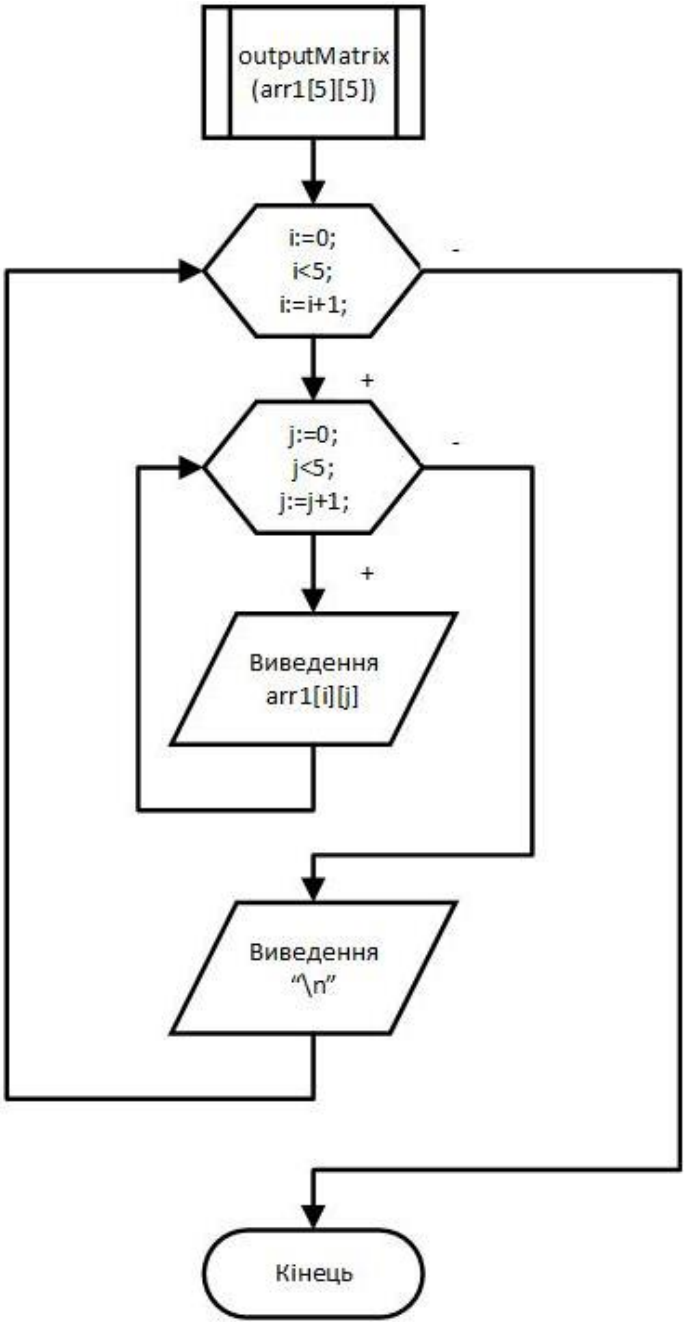

outputMatrix (arr1[5][5])
i:=0; i<5; i:=i+1;
j:=0; j<5; j:=j+1;
Виведення arr1[i][j]
Виведення "\n"
Кінець

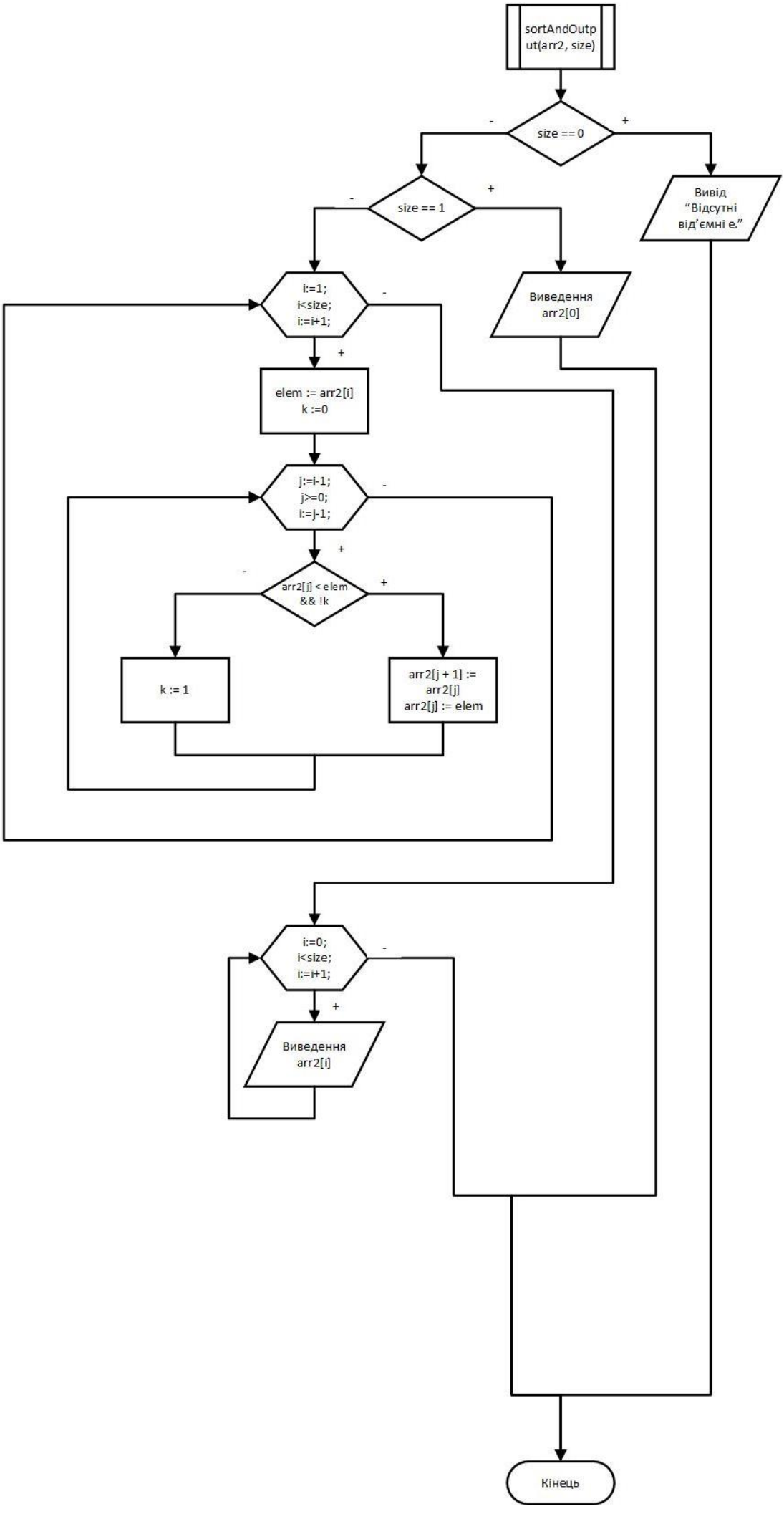
Крок 5


Початок
inputMatrix(
arr1)
size =
diagArr(arr1,
arr2)
outputMatrix
(arr1)
sortAndOutp
ut(arr2, size)
Кінець


inputMatrix(
arr1[5][5])
srand(time(NULL)
)
i:=0;
i<5;
i:=i+1;
-
+
j:=0;
j<5;
j:=j+1;
-
+
arr1[i][j] := rand()
% 30 - 15
Кінець


diagArr(arr1[5][5], arr2[])
sizeArr:=0
i:=0;
i<5;
i:=i+1;
-
+
-
arr1[i][4-i] < 0
+
arr2[sizeArr] := arr1[i][4-i]
sizeArr:=sizeArr+1
return sizeArr
Кінець


outputMatrix
(arr1[5][5])
i:=0;
i<5;
i:=i+1;
-
+
j:=0;
j<5;
j:=j+1;
-
+
Виведення
arr1[i][j]
Виведення
“\n”
Кінець


sortAndOutput(arr2, size)
size == 0
-
+
Вивід
“Відсутні
від’ємні е.”
size == 1
-
+
i:=1;
i<size;
i:=i+1;
-
+
Виведення
arr2[0]
elem := arr2[i]
k :=0
j:=i-1;
j>=0;
i:=j-1;
-
+
arr2[j] < elem
&& !k
-
+
k := 1
arr2[j + 1] :=
arr2[j]
arr2[j] := elem
i:=0;
i<size;
i:=i+1;
-
+
Виведення
arr2[i]
Кінець

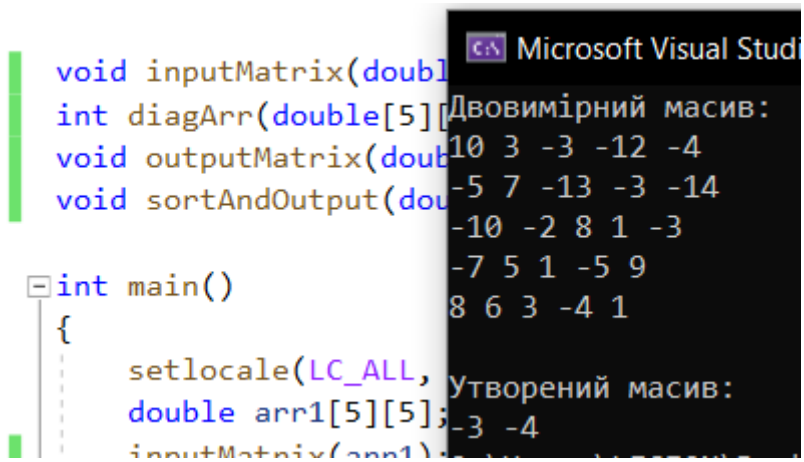
Виконання мовою C++.

Код програми:

```
#include <iostream>
#include<ctime>

using namespace std;

void inputMatrix(double[5][5]);                         //Прототипи: введення матриці
int diagArr(double[5][5],double[]);                     //Утворення одновимірного масиву
void outputMatrix(double[5][5]);                        //Виведення матриці
void sortAndOutput(double[], int);                      //Сортування вставками та виведення

int main()
{
	setlocale(LC_ALL, "ukr");
	double arr1[5][5];
	inputMatrix(arr1);
	double arr2[5];
	int size = diagArr(arr1, arr2);                 //Знаходження масиву і його розміру
	cout << "Двовимірний масив:\n";
	outputMatrix(arr1);
	sortAndOutput(arr2, size);

}

void inputMatrix(double arr1[5][5]) {
	srand(time(NULL));
	for (int i = 0;i < 5;i++)
		for (int j = 0;j < 5;j++) {
			arr1[i][j] = rand() % 30 - 15;    //Заповнення матриці від -15 до 15
		}
}

int diagArr(double arr1[5][5], double arr2[]) {
	int sizeArr = 0;
	for (int i = 0;i < 5;i++) {
		if (arr1[i][4-i] < 0) {
			arr2[sizeArr] = arr1[i][4-i];     //Шукаємо від'ємні на побічній
діагоналі
			sizeArr++;
		}
	}
	return sizeArr;
}

void outputMatrix(double arr1[5][5]) {
	for (int i = 0;i < 5;i++) {
		for (int j = 0;j < 5;j++) {
			cout << arr1[i][j] << " ";
		}
		cout << endl;
	}
}
void sortAndOutput(double arr2[], int size) {
	if (size == 0)
		cout << "\nВідсутні від'ємні елементи, масив не утворюється";
	else {
		cout << "\nУтворений масив:\n";
		if (size == 1)
			cout << arr2[0];
		else {
			bool k;
			double elem;
			for (int i = 1;i < size;i++) {          //Сортування вставками
```

```
                        elem = arr2[i];
                        k = 0;
                        for (int j = i - 1; j >= 0;j--) {
                                if (arr2[j] < elem && !k) {      //Якщо поточний більший за попередній
                                        arr2[j + 1] = arr2[j];   //Переставляємо їх місцями
                                        arr2[j] = elem;
                                }
                                else
                                        k = 1;
                        }
                }
                for (int i = 0;i < size;i++)
                        cout << arr2[i] << " ";
        }
    }
}
```

Випробування алгоритму.



Перевірка алгоритму.

Блок	Дія
	Початок
1	inputMatrix(arr1) – заповнення випадковим чином (на скриншоті)
2	size:=diagArr(arr1, arr2): результат – arr2=[-4,-3]; size=2;
3	outputMatrix(arr1) – виведення матриці (на скриншоті)
4	sortAndOutput(arr2, size=2): результат сортування за спаданням – arr2=[-3,-4]; Виведення arr2 – “-3 -4”
	Кінець

Висновок. Отже, у цій роботі я дослідив алгоритми пошуку та сортування, набув практичних навичок використання цих алгоритмів під час складання програмних специфікацій. У результаті лабораторної роботи було розроблено математичну модель, що відповідає постановці задачі; псевдокод та блок-схеми, які пояснюють логіку алгоритму, а також програму, що виконує задачу відповідно до постановки. Використовуючи чотири підпрограми для роботи з масивами – одна для введення матриці, одна – для її виведення, одна – для пошуку в масиві, остання – для сортування вставками за спаданням - отримуємо коректний результат.