

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

Задача. 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. Створення нової змінної індексованого типу (одновимірний масив) та її ініціювання значеннями, що обчислюються згідно з варіантом.

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

1. *Постановка задачі.* Початковими даними є розмірність двовимірного масиву $m \times n$, за умовою ця розмірність становить 5×5 . Оскільки $m=n$, то доцільно буде використовувати єдину змінну m на позначення кількості рядків та стовпців цієї матриці (масиву). Результатом розв'язку є інша змінна індексованого типу (одновимірний масив), яка складається із від'ємних значень елементів побічної діагоналі першого (двовимірного) масиву та відсортована за спаданням.
2. *Побудова математичної моделі.* Складемо таблицю імен змінних.

Змінна	Тип	Ім'я	Призначення
Розмірність двовимірного масиву	Цілий	m	Початкове дане
Двовимірний масив	Дійсний	A[m,m]	Допоміжна змінна
Розмір одновимірного масиву	Цілий	n	Допоміжна змінна
Одновимірний масив	Дійсний	B[n]	Допоміжна змінна
Параметр арифметичного циклу	Цілий, послідовний	i	Лічильник

Параметр арифметичного циклу	Цілий, послідовний	j	Лічильник
Формальний параметр для передачі розмірності двовимірного масиву у функцію	Цілий	m1	Допоміжна змінна
Формальний параметр для передачі розміру одновим. масиву у функцію	Цілий	n1	Допоміжна змінна
Формальний параметр; двовимірний масив, що передається у функцію	Дійсний	arr2[]	Допоміжна змінна
Форм. параметр; одновимірний масив, що передається у функцію	Дійсний	arr1[]	Допоміжна змінна
Змінна для перестановки елементів одновимірн. масиву при сортуванні	Дійсний	cop	Допоміжна змінна

Складемо таблицю імен допоміжних алгоритмів (функцій).

Функція	Тип результату	Ім'я
Генерація двовимірного масиву дійсних чисел	—	input_2()
Виведення двовимірного масиву	—	output_2()
Створення одновимірного масиву та визначення кількості його елементів	Цілий	creating_B()
Виведення одновимірного масиву	—	output_1()
Сортування одновимірного масиву за спаданням методом вставки	—	sorting()

Таким чином, математичне формулювання задачі зводиться до виконання наступних дій:

- 1) Ініціалізація змінної $m:=5$;
- 2) Ініціалізація двовимірного масиву $A[m,m]$ (розмірністю m на m) за допомогою виклику функції $input_2(A, m)$. Виведення згенерованого масиву на екран функцією $output_2(A, m)$.
- 3) Ініціалізація масиву $B[n]$ від'ємними елементами побічної діагоналі згенерованого масиву $A[m,m]$ за допомогою функції $creating_B()$, а також повернення з цієї функції значення кількості елементів утвореного масиву $B[n]$ за формулою $n:=creating_B(B, A, m)$.
- 4) У випадку якщо утворений масив B не містить жодного елемента, виводиться відповідне повідомлення, інакше відбувається виведення отриманого масиву $B[n]$ функцією $output_1(B, n)$, сортування масиву $B[n]$ за допомогою виклику функції $sorting(B, n)$ та виведення відсортованого масиву функцією $output_1(B, n)$.

- ✓ `input_2(arr2[],m1)` – функція, яка генерує двовимірний масив даної розмірності. Ця функція використовує арифметичний цикл з параметром i (i від 1 до $m1$ включно), з вкладеним у нього арифметичним циклом з параметром j (j від 1 до $m1$ включно), на кожній з ітерацій цього циклу генерується дійсний елемент двовимірного масиву з індексом i,j (`arr2[i,j]`) за формулою `arr2[i,j]=rand()/RAND_MAX*2*100-100`, яка означає, що буде згенероване дійсне число з діапазону від -100 до 100.
- ✓ `output_2(arr2[], m1)` – функція, яка виводить переданий через параметр двовимірний масив на екран, використовуючи арифметичний цикл з параметром i (i від 1 до $m1$ включно), з вкладеним у нього арифметичним циклом з параметром j (j від 1 до $m1$ включно), і виводячи на кожній ітерації змінну `arr2[i,j]`, яка відповідає індексу i,j (`arr2[i,j]`).
- ✓ `creating_B(arr1[],arr2[],m1)` – функція, яка створює одновимірний масив `arr1[]` з від’ємних елементів побічної діагоналі двовимірного масиву `arr2[]`.
 - 1) На початку ініціалізується змінна `n1:=1`.
 - 2) Далі за допомогою арифметичного циклу з параметром i (i від 1 до $m1$ включно) функція виконує лінійний пошук серед елементів побічної діагоналі двовимірного масиву (елементи, які належать до побічної діагоналі: `arr2[i,(m1-i+1)]`) і у разі, якщо даний елемент від’ємний, значення елемента присвоюється елементу одновимірного масиву з індексом `n1` (`arr1[n1]`) і `n1` збільшується на 1 (`n1++`).
 - 3) Як результат функції повертається вираз `(n1-1)`.
- ✓ `output_1(arr1[], n1)` – функція, яка виводить переданий через параметр одновимірний масив на екран, використовуючи арифметичний цикл з параметром i (i від 1 до $n1$ включно) і виводячи на кожній ітерації змінну `arr1[i]`, яка відповідає індексу i (`arr1[i]`).
- ✓ `sorting(arr1[],n1)` – функція, яка сортує елементи одновимірного масиву методом вставки за спаданням. Функція складається із арифметичного циклу з параметром i (i від 2 до $n1$ включно), у якому повторюються наступні дії:
 - 1) Ініціалізуються змінні `cop:=arr1[i]` та `j:=i-1`;
 - 2) Далі задається вкладений ітераційний цикл з передумовою (умова: `j>=1 && arr1[j]>cop`), на кожній ітерації якого виконується зсув відсортованої частини елементів масиву на одну позицію праворуч за формулою: `arr1[j+1]:=arr1[j]`;

- 3) Виконується вставка елементу $a[i]$ на місце елементу $a[j+1]$ за формулою: $a[j+1] := \text{cop}$.

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

- Крок 1.* Визначимо основні дії.
Крок 2. Деталізуємо ініціалізацію змінної m .
Крок 3. Деталізуємо ініціалізацію та виведення двовимірного масиву $A[m,m]$.
Крок 4. Деталізуємо дію створення одновимірного масиву $B[]$ та обчислення кількості його елементів n .
Крок 5. Деталізуємо перевірку чи масив $B[n]$ не порожній.
Крок 6. Деталізуємо дію сортування масиву $B[n]$ за спаданням.
Крок 7. Деталізуємо функцію $\text{input_2}()$.
Крок 8. Деталізуємо функцію $\text{output_2}()$.
Крок 9. Деталізуємо функцію $\text{output_1}()$.
Крок 10. Деталізуємо функцію $\text{creating_B}()$.
Крок 11. Деталізуємо функцію $\text{sorting}()$.

3. Псевдокод алгоритму.

Крок 1

початок

ініціалізація змінної m

ініціалізація та виведення $A[m,m]$

створення одновимірного масиву $B[n]$ і
обчислення кількості його елементів

перевірка чи масив $B[n]$ не порожній

кінець

Крок 2

початок

$m := 5$

ініціалізація та виведення $A[m,m]$

створення одновимірного масиву $B[n]$ і
обчислення кількості його елементів

перевірка чи масив $B[n]$ не порожній

кінець

Крок 3

початок

$m := 5$

$\text{input_2}(A, m)$

$\text{output_2}(A, m)$

створення одновимірного масиву $B[n]$ і
обчислення кількості його елементів

перевірка чи масив $B[n]$ не порожній

кінець

Крок 4

початок

$m := 5$

$\text{input_2}(A, m)$

$\text{output_2}(A, m)$

$n := \text{creating_B}(B, A, m)$

перевірка чи масив $B[n]$ не порожній

кінець

Крок 5

початок

$m := 5$

input_2(A, m)

output_2(A, m)

$n := \text{creating_B}(B, A, m)$

якщо ($n==0$)

то

виведення «Побічна діагональ
матриці A не містить від'ємних
елементів»

інакше

output_1(B, n)

сортування масиву B[n]

output_1(B, n)

все якщо

кінець

Крок 6

початок

$m := 5$

input_2(A, m)

output_2(A, m)

$n := \text{creating_B}(B, A, m)$

якщо ($n==0$)

то

виведення «Побічна діагональ
матриці A не містить від'ємних
елементів»

інакше

output_1(B, n)

sorting(B, n)

output_1(B, n)

все якщо

кінець

3.1. Псевдокод допоміжних алгоритмів (функцій).

Крок 7

початок

input_2(arr2[], m1)

для і від 1 до m1

повторити

для j від 1 до m1

повторити

$\text{arr2}[i,j] := \text{rand}() / \text{RAND_MAX} * 2 * 100 - 100$

все повторити

все повторити

кінець input_2()

Крок 8

початок

output_2(arr2[], m1)

для і від 1 до m1

повторити

для j від 1 до m1

повторити

виведення arr2[i,j]

все повторити

все повторити

кінець output_2()

Крок 9

початок

output_1(arr1[], n1)

для і від 1 до n1

повторити

виведення arr1[i]

все повторити

кінець output_1()

Крок 10

початок `creating_B(arr1[],arr2[],m1)`

`n1 := 1`

для `i` від 1 до `m1`

повторити

якщо `(arr2[i,(m1 - i + 1)] < 0)`

то

`arr1[n1] := arr2[i,(m1 - i + 1)]`

`n1++;`

все якщо

все повторити

повернути `(n1 - 1)`

кінець `creating_B()`

Крок 11

початок `sorting(arr1[], n1)`

для `i` від 2 до `n1`

повторити

`сру := arr1[i]`

`j := i - 1`

поки `(j >= 1 && arr1[j] > сру)`

повторити

`arr1[j + 1] = arr1[j]`

`j--`

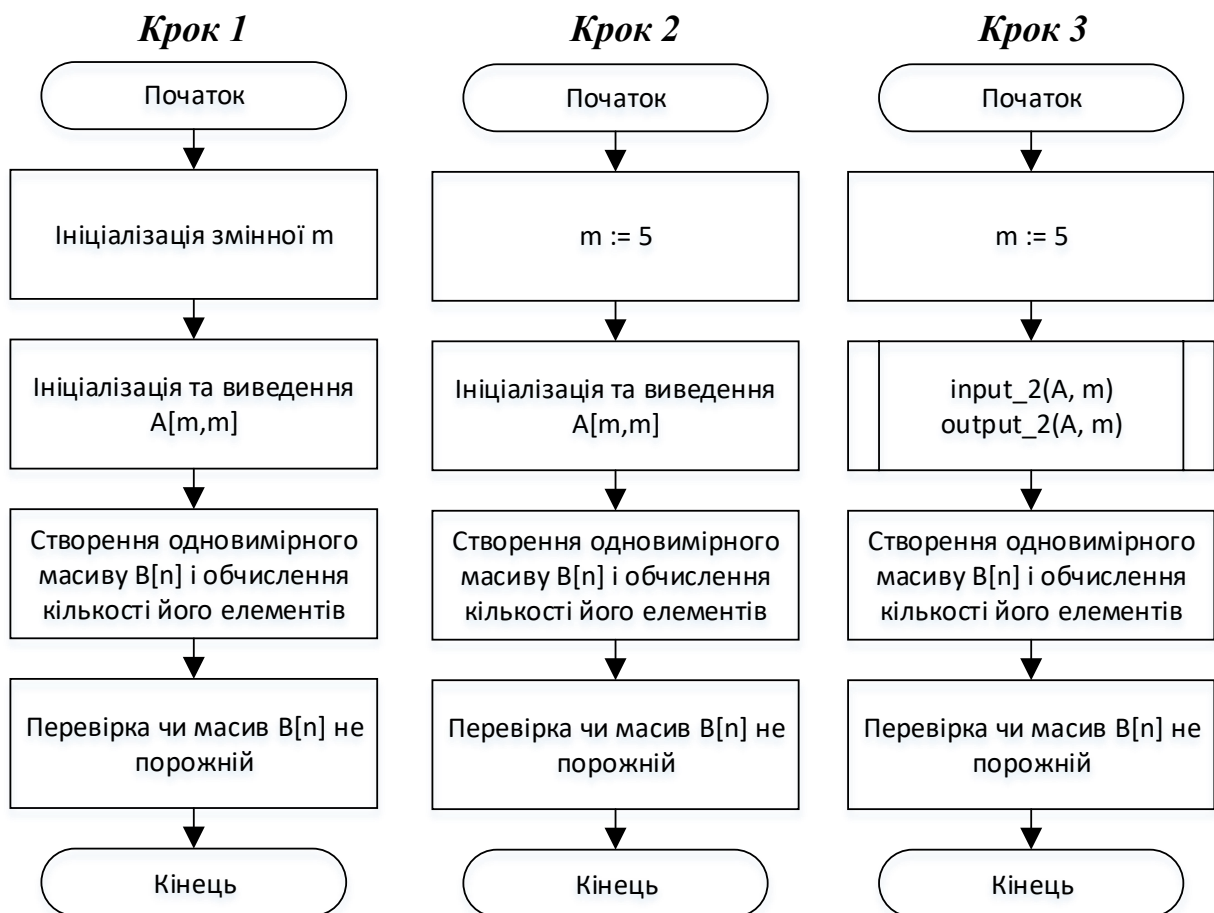
все повторити

`arr1[j + 1] = сру`

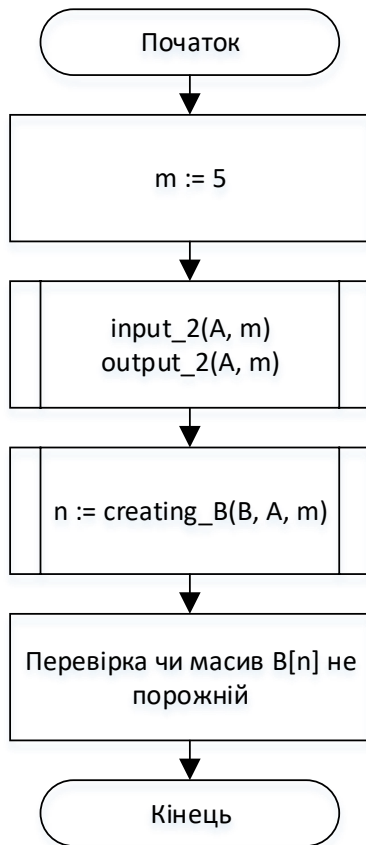
все повторити

кінець `sorting()`

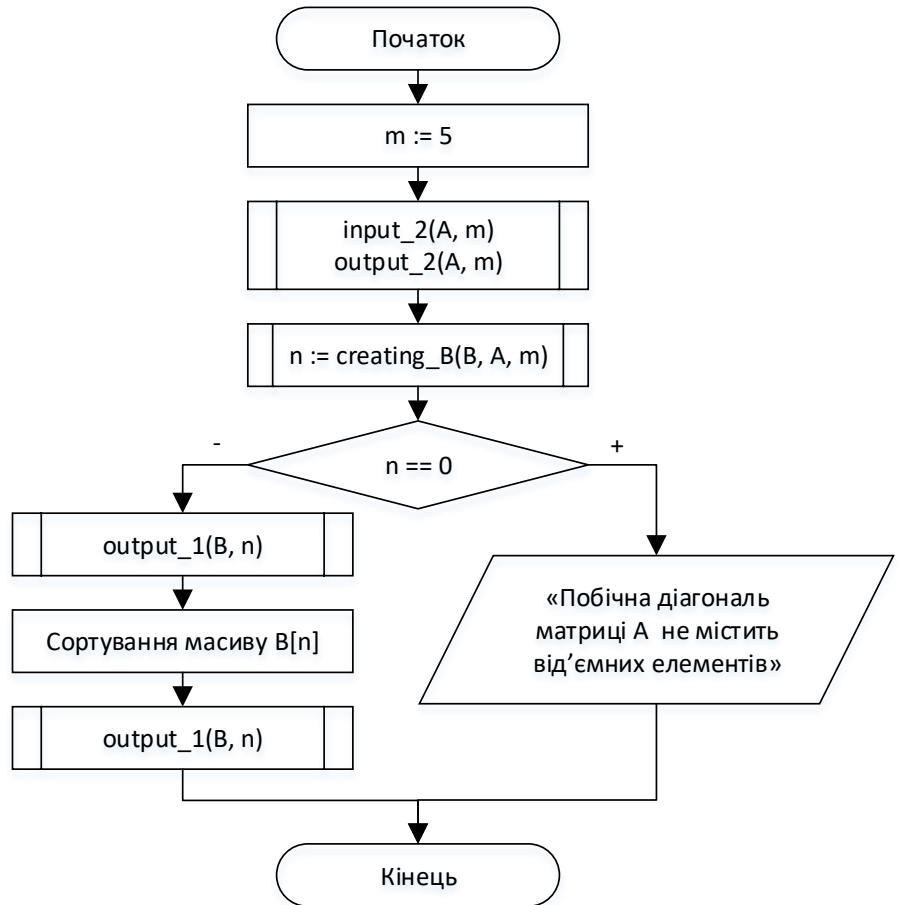
4. Блок-схема алгоритму.



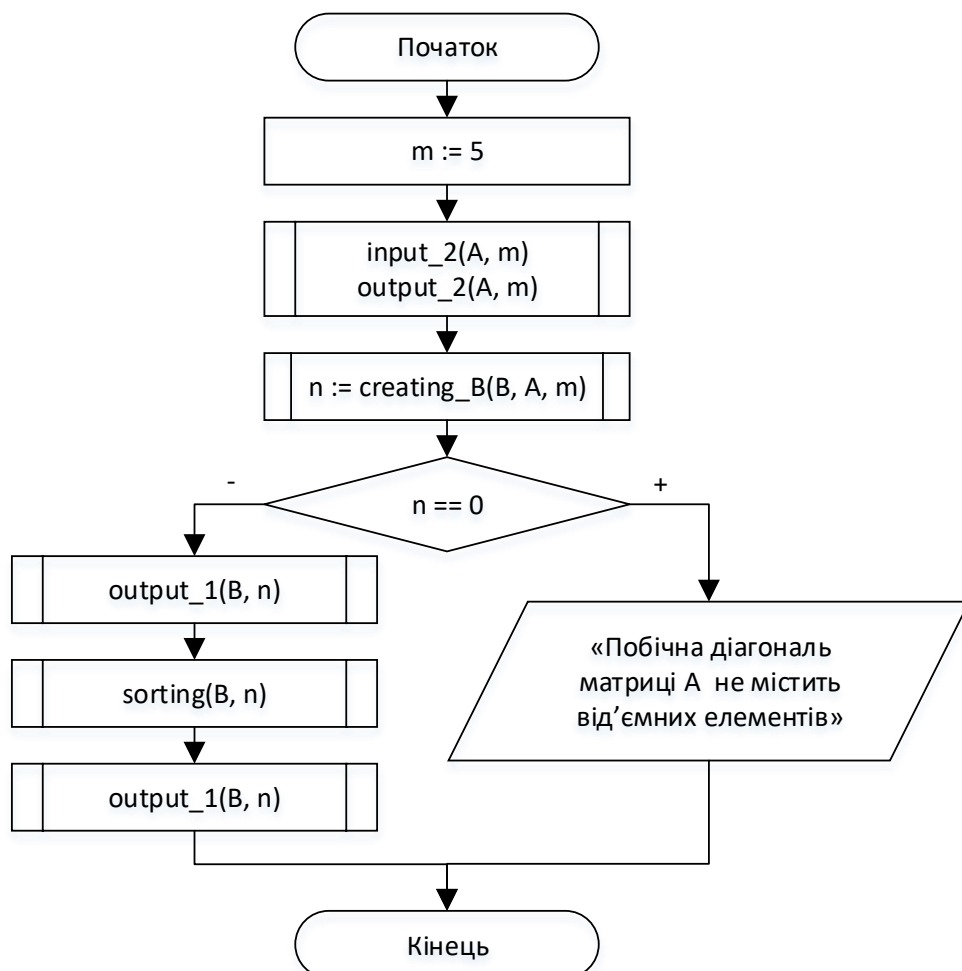
Крок 4



Крок 5

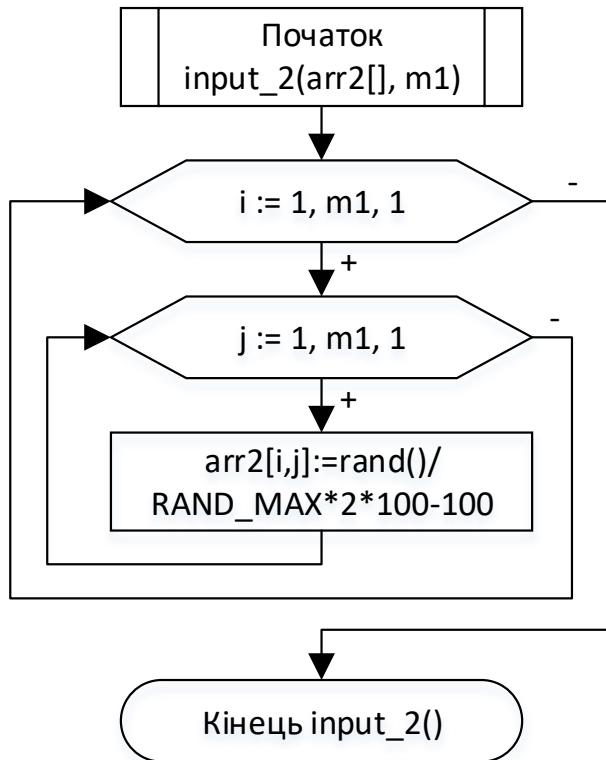


Крок 6

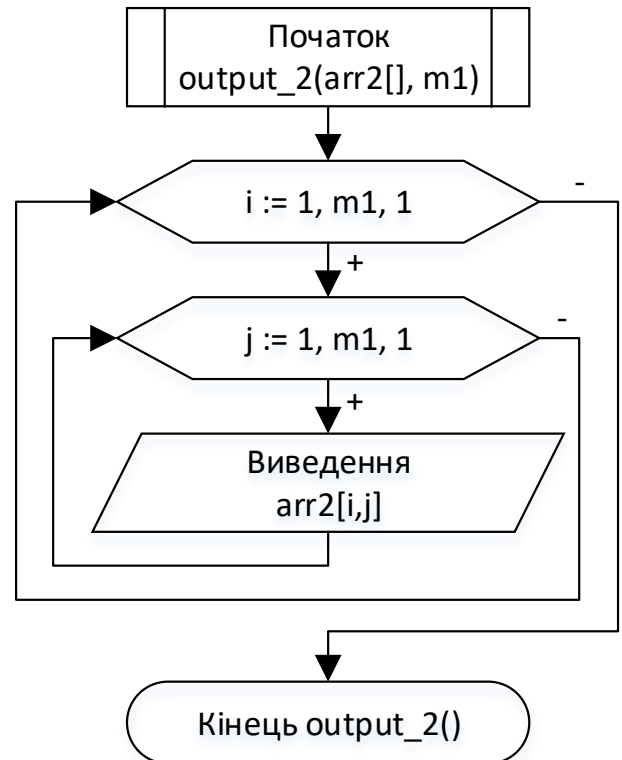


4.1. Блок-схеми допоміжних алгоритмів (функцій).

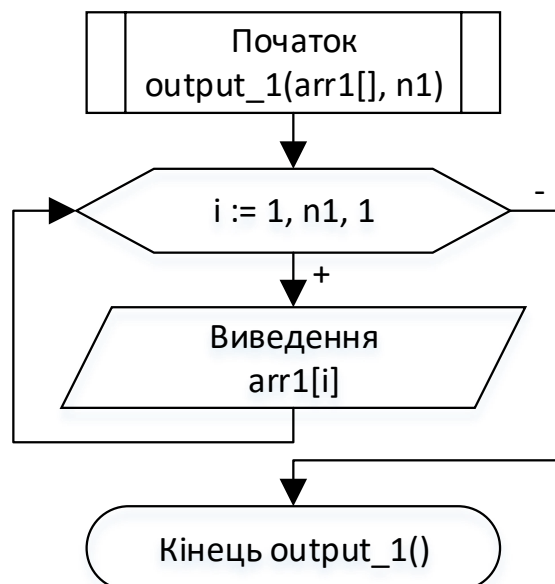
Крок 7



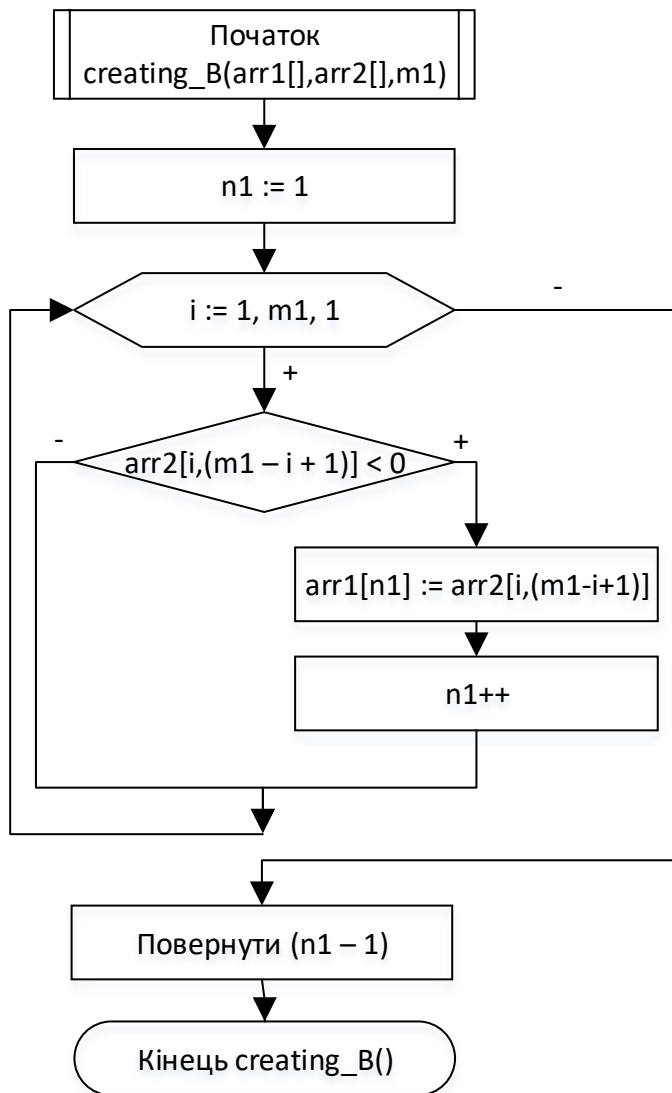
Крок 8



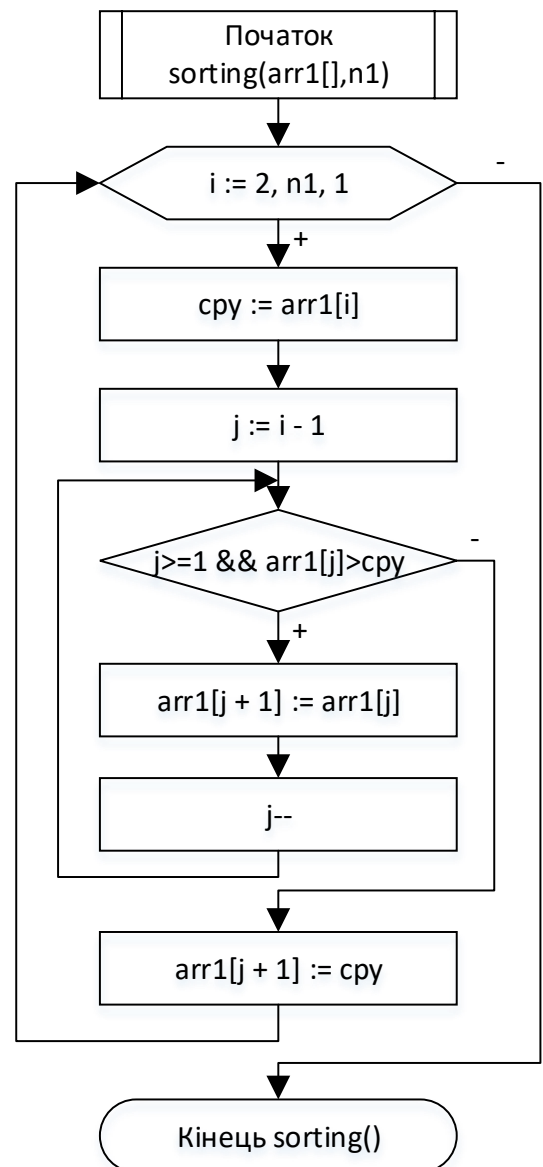
Крок 9



Крок 10



Крок 11



5. Код програми (на мові програмування C++).

```
#include <iostream>
#include <iomanip>
#include <ctime>
using namespace std;

void input_2(float**, int);
void output_2(float**, int);
int creating_B(float[], float**, int);
void output_1(float[], int);
void sorting(float[], int);

int main()
{
    int m = 5;
    int n;
    float** A;
    float* B;
    A = new float*[m];
```

```

        for (int i = 0; i < m; i++) {
            A[i] = new float[m];
        }
        B = new float[m]
        input_2(A, m);
        cout << "The matrix A:" << endl;
        output_2(A, m);
        n = creating_B(B, A, m);
        if (n == 0) {
            cout << "The side diagonal does not contain negative elements!" <<
endl;
        }
        else {
            cout << "The second array:" << endl;
            output_1(B, n);
            sorting(B, n);
            cout << "Sorted one-dimensional array:" << endl;
            output_1(B, n);
        }
        for (int i = 0; i < m; i++)
            delete[] A[i];
        delete[] A;
        delete[] B;
        system("pause");
        return 0;
    }
}

void input_2(float** arr2, int m1)
{
    srand(time(NULL));
    for (int i = 0; i < m1; i++) {
        for (int j = 0; j < m1; j++) {
            arr2[i][j] = (float)rand() / RAND_MAX * 2 * 100 - 100;
        }
    }
}

void output_2(float** arr2, int m1)
{
    for (int i = 0; i < m1; i++) {
        for (int j = 0; j < m1; j++) {
            printf("%10.2f", arr2[i][j]);
        }
        cout << endl;
    }
}

int creating_B(float arr1[], float** arr2, int m1) {
    int n1 = 0;
    for (int i = 0; i < m1; i++) {
        if (arr2[i][m1 - i - 1] < 0) {
            arr1[n1] = arr2[i][m1 - i - 1];
            n1++;
        }
    }
    return n1;
}

void output_1(float arr1[], int n1)
{
    for (int i = 0; i < n1; i++) {
        printf("%.2f", arr1[i]);
        cout << " ";
    }
}

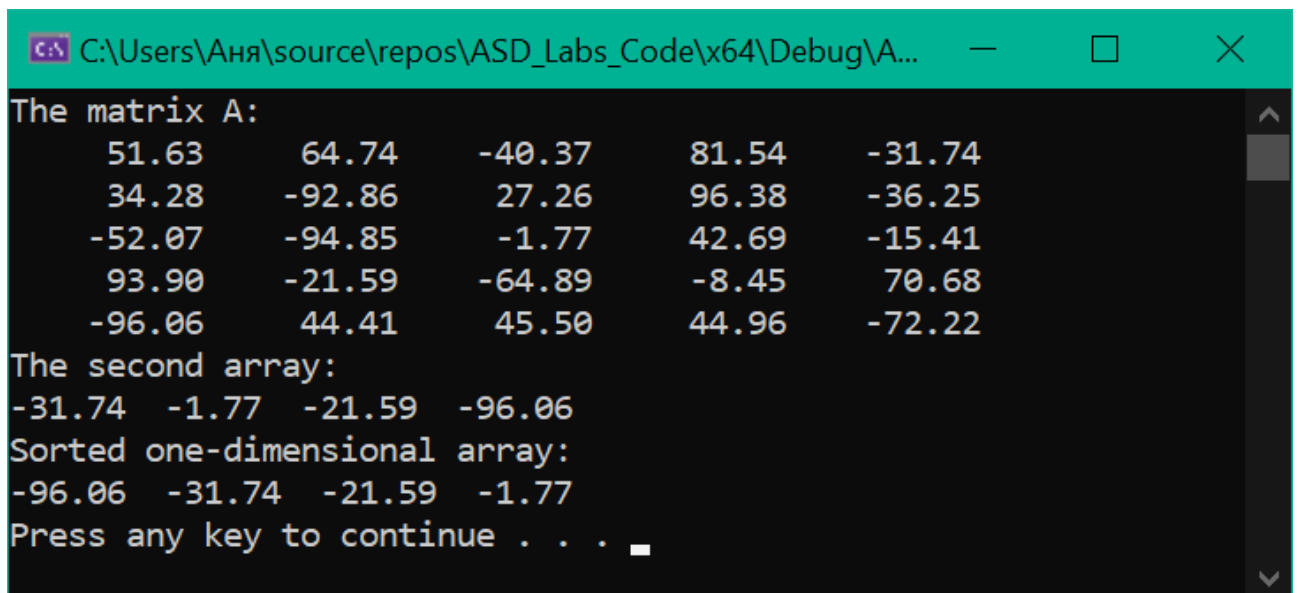
```

```

    }
    cout << endl;
}
void sorting(float arr1[], int n1)
{
    int i, j;
    float cop;
    for (i = 1; i < n1; i++) {
        cop = arr1[i];
        j = i - 1;
        while (j >= 0 && arr1[j] > cop) {
            arr1[j + 1] = arr1[j];
            j--;
        }
        arr1[j + 1] = cop;
    }
}
}

```

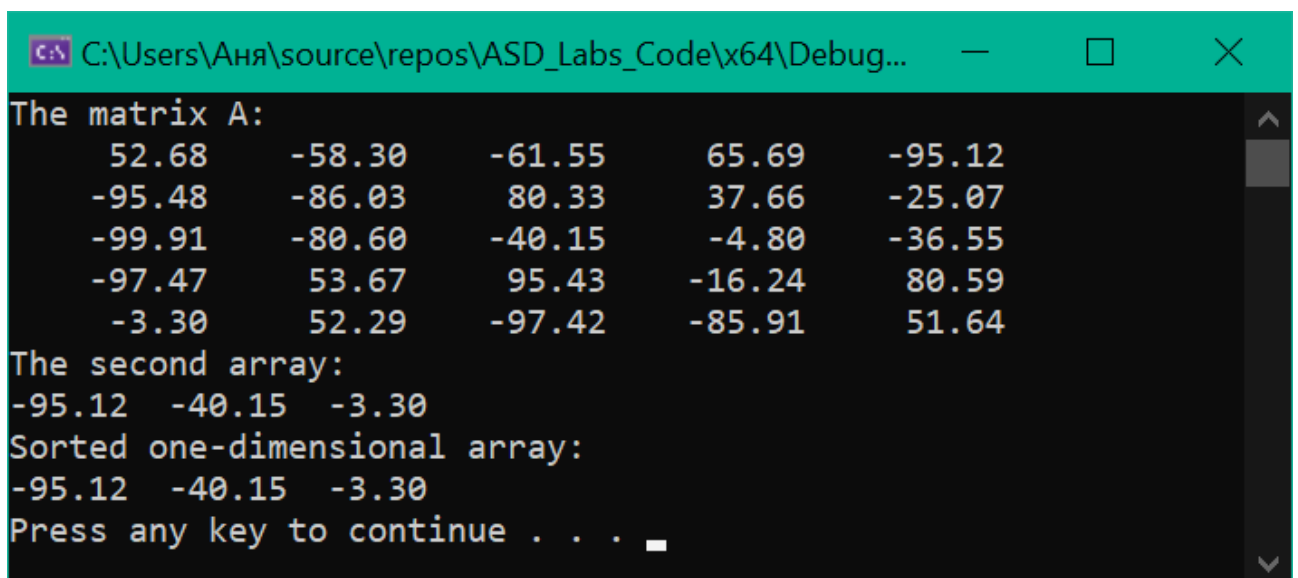
6. Тестування програми.



```

C:\Users\Аня\source\repos\ASD_Labs_Code\x64\Debug\A...
The matrix A:
  51.63    64.74   -40.37    81.54   -31.74
  34.28   -92.86    27.26    96.38   -36.25
 -52.07   -94.85    -1.77    42.69   -15.41
  93.90   -21.59   -64.89    -8.45    70.68
 -96.06    44.41    45.50    44.96   -72.22
The second array:
-31.74  -1.77  -21.59  -96.06
Sorted one-dimensional array:
-96.06  -31.74  -21.59  -1.77
Press any key to continue . . .

```



```

C:\Users\Аня\source\repos\ASD_Labs_Code\x64\Debug...
The matrix A:
  52.68   -58.30   -61.55    65.69   -95.12
 -95.48   -86.03    80.33    37.66   -25.07
 -99.91   -80.60   -40.15    -4.80   -36.55
 -97.47    53.67    95.43   -16.24    80.59
  -3.30    52.29   -97.42   -85.91    51.64
The second array:
-95.12  -40.15  -3.30
Sorted one-dimensional array:
-95.12  -40.15  -3.30
Press any key to continue . . .

```

```
C:\Users\Аня\source\repos\ASD_Labs_Code\x64\Debug...
The matrix A:
  54.20   -72.47    25.05    27.87    89.48
 -74.00   -42.27   -77.54    96.84   -76.96
  31.48   -60.17    29.36    70.50    57.67
 -54.90    93.66    42.28    32.97    -7.09
  91.98    33.38    67.47   -32.07    25.48
The side diagonal does not contain negative elements!
Press any key to continue . . .
```

7. *Висновки.* На цій лабораторній роботі було досліджено алгоритми пошуку та сортування, було набуто практичних навичок використання цих алгоритмів під час складання програмних специфікацій.

Побудований алгоритм було покладено на мову програмування C++ і написано програму, яку було випробувано три рази.

У першому із згенерованих двовимірних масивів побічна діагональ містила 4 від'ємні елементи, які були додані програмою до одновимірного масиву. Цей масив було виведено на екран. Далі було виконано сортування знайденого одновимірного масиву і виведено відсортований масив на екран.

У другому з утворених двовимірних масивів побічна діагональ містила 3 від'ємні елементи, які було додано до одновимірного масиву і виведено на екран. Під час сортування не було необхідності виконувати якісь дії з цим масивом, оскільки його елементи відразу були розташовані у порядку спадання. Тож його без змін було виведено на екран як відсортований.

У останньому із згенерованих масивів побічна діагональ не містила жодного від'ємного елементу, тож на екран було виведено відповідне повідомлення.

Отже, побудований алгоритм працює правильно.