

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 № 8 з дисципліни
«Алгоритми та структури даних-1.
Основи алгоритмізації»
«Дослідження алгоритмів пошуку та сортування»

Варіант 34

Виконав студент ІІІ-15, Чінь Хоанг Вьет
Перевірив Вечерковська Анастасія Сергіївна

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 34

Постановка задачі

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
34	5 x 7	Дійсний	Із середнього арифметичного додатних значень елементів стовпців двовимірного масиву. Відсортувати обміном за спаданням.

Розв’язання: Для початку, створимо матрицю, яка складається із дійсних чисел, за допомогою функції, яка складається із двох арифметичних циклів. Заповнювати ми їх будемо різними числами. Потім створимо іншу функцію, яка буде створювати одновимірний масив із середнього арифметичного додатних значень елементів стовпців цієї матриці і відсортуємо його за спаданням. І для того, щоб побачити ці значення, виведемо їх в консоль.

Математична модель:

Змінна	Тип	Ім'я	Призначення
Рядки	Дійсний	row	Початкові дані
Стовпці	Дійсний	col	Початкові дані
Двовимірний	Дійсний	Arr	Початкові

масив			дані
Одновимірний масив	Дійсний	NewArr	Початкові дані
Лічильник 1	Цілочисельний та натуральний	i	Параметр циклу
Лічильник 2	Цілочисельний та натуральний	j	Параметр циклу
Сума	Дійсний	sum	Проміжні дані
Доповнення	Дійсний	add	Проміжні дані
Лічильник додатних чисел	Цілочисельний та натуральний	counter	Проміжні дані
Створення матриці	Процедура	TwoDimensional	Початкові дані
Вивід матриці	Процедура	output	Початкові дані
Створення масиву	Процедура	newArr	Початкові дані
Сортування масиву	Процедура	SortArray	Початкові дані
Видалення масиву	Процедура	remove	Початкові дані

1. Визначаємо основні дії
2. Вводимо значення row та col та ініціалізація початкових даних
3. Створення підпрограми TwoDimensional для створення масиву
4. Створення підпрограми output для виводу масиву
5. Створення підпрограми newArr для створення нового масиву
6. Створення підпрограми SortArray для сортування нового масиву
7. Створення підпрограми remove для видалення масиву

TwoDimensional

Псевдокод:

Головна програма:

Крок 1

Вводимо значення row та col та ініціалізація початкових даних

Підпрограми:

Створення підпрограми TwoDimensional для створення масиву

Створення підпрограми output для виводу масиву

Створення підпрограми newArr для створення нового масиву

Створення підпрограми SortArray для сортування нового масиву

Створення підпрограми remove для видалення масиву TwoDimensional

Крок 2

Головна програма:

Ввід row та col

```
**Arr = TwoDimensional(row, col);
```

```
output(Arr, row, col);
```

```
*NewArr = newArr(Arr, row, col);
```

```
SortArray(NewArr, col);
```

```
remove(Arr, row)
```

Підпрограми:

Створення підпрограми TwoDimensional для створення масиву

Створення підпрограми output для виводу масиву

Створення підпрограми newArr для створення нового масиву

Створення підпрограми SortArray для сортування нового масиву

Створення підпрограми remove для видалення масиву TwoDimensional

Крок 3

Головна програма:

Ввід row та col

```
**Arr = TwoDimensional(row, col);
```

```
output(Arr, row, col);
```

```
*NewArr = newArr(Arr, row, col);
```

```
SortArray(NewArr, col);
```

remove(Arr, row)

Підпрограми:

Початок **TwoDimensional**

float **Arr = new float*[row];

Для **i** від **0** до **n** повторити

Arr[i] = new float[m];

Все повторити

Для **i** від **0** до **n** повторити

Для **j** від **0** до **m** повторити

Arr[i][j] = float(rand())/RAND_MAX*200-100;

Все повторити

Все повторити

return Arr;

Кінець **TwoDimensional**

Створення підпрограми output для виводу масиву

Створення підпрограми newArr для створення нового масиву

Створення підпрограми SortArray для сортування нового масиву

Створення підпрограми remove для видалення масиву TwoDimensional

Крок 4

Головна програма:

Ввід row та col

**Arr = TwoDimensional(row, col);

output(Arr, row, col);

*NewArr = newArr(Arr, row, col);

SortArray(NewArr, col);

remove(Arr, row)

Підпрограми:

Початок **TwoDimensional**

float **Arr = new float*[row];

Для **i** від **0** до **n** повторити

Arr[i] = new float[m];

Все повторити

Для **i** від **0** до **n** повторити

Для **j** від **0** до **m** повторити

Arr[i][j] = float(rand())/RAND_MAX*200-100;

Все повторити

Все повторити

return Arr;

Кінець **TwoDimensional**

Початок **output**

Для **i** від **0** до **n** повторити

Для **j** від **0** до **m** повторити

Вивід Arr[i][j];

Все повторити

Все повторити

Кінець **output**

Створення підпрограми newArr для створення нового масиву

Створення підпрограми SortArray для сортування нового масиву

Створення підпрограми remove для видалення масиву TwoDimensional

Крок 5

Головна програма:

Ввід row та col

```
**Arr = TwoDimensional(row, col);
```

```
output(Arr, row, col);
```

```
*NewArr = newArr(Arr, row, col);
```

```
SortArray(NewArr, col);
```

```
remove(Arr, row)
```

Підпрограми:

Початок **TwoDimensional**

```
float **Arr = new float*[row];
```

Для **i** від **0** до **n** повторити

```
Arr[i] = new float[m];
```

Все повторити

Для **i** від **0** до **n** повторити

Для **j** від **0** до **m** повторити

```
Arr[i][j] = float(rand())/RAND_MAX*200-100;
```

Все повторити

Все повторити

```
return Arr;
```

Кінець **TwoDimensional**

Початок **output**

Для **i** від **0** до **n** повторити

Для **j** від **0** до **m** повторити

Вивід `Arr[i][j]`;

Все повторити

Все повторити

Кінець **output**

Початок **newArr**

`float* newArr = new float[m];`

Для **j** від **0** до **m** повторити

`sum = 0;`

`counter = 0;`

Для **i** від **0** до **n** повторити

Якщо (`Arr[i][j] > 0`)

`sum = sum + arr[i][j];`

`Counter++;`

Все якщо

Все повторити

Якщо (`sum == 0`)

`NewArr[j] = 0`

Все якщо

Інакше

`sum = sum/counter`

NewArr[j] = sum;

Все інакше

Все повторити

Для **j** від **0** до **m** повторити

Вивід NewArr[j];

Все повторити

return NewArr;

Кінець **newArr**

Створення підпрограми SortArray для сортування нового масиву

Створення підпрограми remove для видалення масиву TwoDimensional

Крок 6

Головна програма:

Ввід row та col

****Arr** = TwoDimensional(row, col);

output(Arr, row, col);

***NewArr** = newArr(Arr, row, col);

SortArray(NewArr, col);

remove(Arr, row)

Підпрограми:

Початок **TwoDimensional**

float ****Arr** = new float*[row];

Для **i** від **0** до **n** повторити

Arr[i] = new float[m];

Все повторити

Для **i** від **0** до **n** повторити

Для **j** від **0** до **m** повторити

$\text{Arr}[i][j] = \text{float}(\text{rand}()) / \text{RAND_MAX} * 200 - 100;$

Все повторити

Все повторити

return Arr;

Кінець **TwoDimensional**

Початок **output**

Для **i** від **0** до **n** повторити

Для **j** від **0** до **m** повторити

Вивід $\text{Arr}[i][j];$

Все повторити

Все повторити

Кінець **output**

Початок **newArr**

$\text{float}^* \text{NewArr} = \text{new float}[m];$

Для **j** від **0** до **m** повторити

$\text{sum} = 0;$

$\text{counter} = 0;$

Для **i** від **0** до **n** повторити

Якщо $(\text{Arr}[i][j] > 0)$

$\text{sum} = \text{sum} + \text{arr}[i][j];$

Counter++;

Все якщо

Все повторити

Якщо (sum == 0)

NewArr[j] = 0

Все якщо

Інакше

sum = sum/counter

NewArr[j] = sum;

Все інакше

Все повторити

Для j від 0 до m повторити

Вивід NewArr[j];

Все повторити

return NewArr;

Кінець newArr

Початок SortArray

add = 0;

Для i від 0 до m повторити

Для j від i+1 до m повторити

Якщо (NewArr[i] < NewArr[j])

add = NewArr[i];

NewArr[i] = NewArr[j];

NewArr[j] = add;

Все якщо

Все повторити

Вивід NewArr[i];

Все повторити

delete[]NewArr;

Кінець SortArray

Створення підпрограми remove для видалення масиву TwoDimensional

Крок 7

Головна програма:

Ввід row та col

**Arr = TwoDimensional(row, col);

output(Arr, row, col);

*NewArr = newArr(Arr, row, col);

SortArray(NewArr, col);

remove(Arr, row)

Підпрограми:

Початок **TwoDimensional**

float **Arr = new float*[n];

Для **i** від **0** до **n** повторити

Arr[i] = new float[m];

Все повторити

Для **i** від **0** до **n** повторити

Для **j** від **0** до **m** повторити

Arr[i][j] = float(rand())/RAND_MAX*200-100;

Все повторити

Все повторити

return Arr;

Кінець TwoDimensional

Початок output

Для i від 0 до n повторити

Для j від 0 до m повторити

Вивід Arr[i][j];

Все повторити

Все повторити

Кінець output

Початок newArr

float* newArr = new float[m];

Для j від 0 до m повторити

sum = 0;

counter = 0;

Для i від 0 до n повторити

Якщо (Arr[i][j] > 0)

sum = sum + Arr[i][j];

counter++;

Все якщо

Все повторити

Якщо (sum == 0)

NewArr[j] = 0

Все якщо

Інакше

sum = sum/counter

NewArr[j] = sum;

Все інакше

Все повторити

Для j від 0 до m повторити

Вивід NewArr[j];

Все повторити

return NewArr;

Кінець newArr

Початок SortArray

add = 0;

Для i від 0 до m повторити

Для j від i+1 до m повторити

Якщо (NewArr[i] < NewArr[j])

add = NewArr[i];

NewArr[i] = NewArr[j];

NewArr[j] = add;

Все якщо

Все повторити

Вивід NewArr[i];

Все повторити

delete[]NewArr;

Кінець SortArray

Початок remove

Для i від 0 до n повторити

delete[]Arr[i];

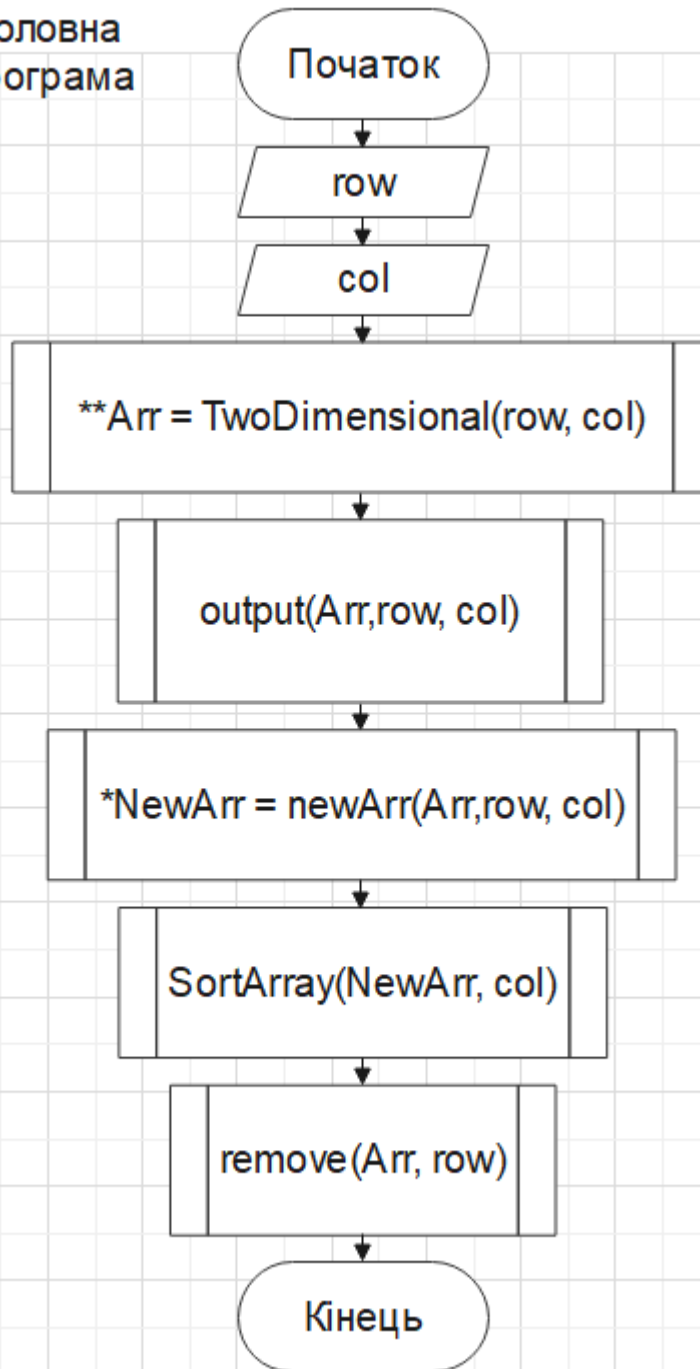
Все повторити

delete[]Arr;

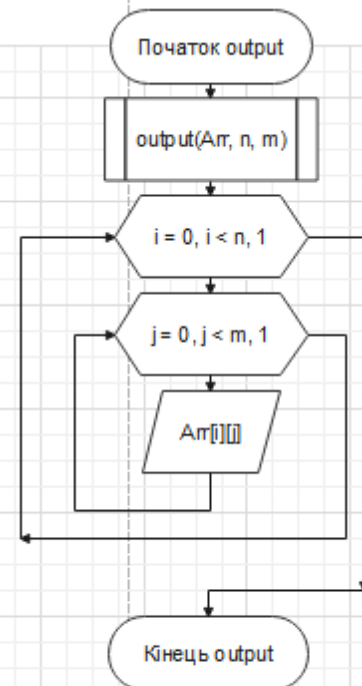
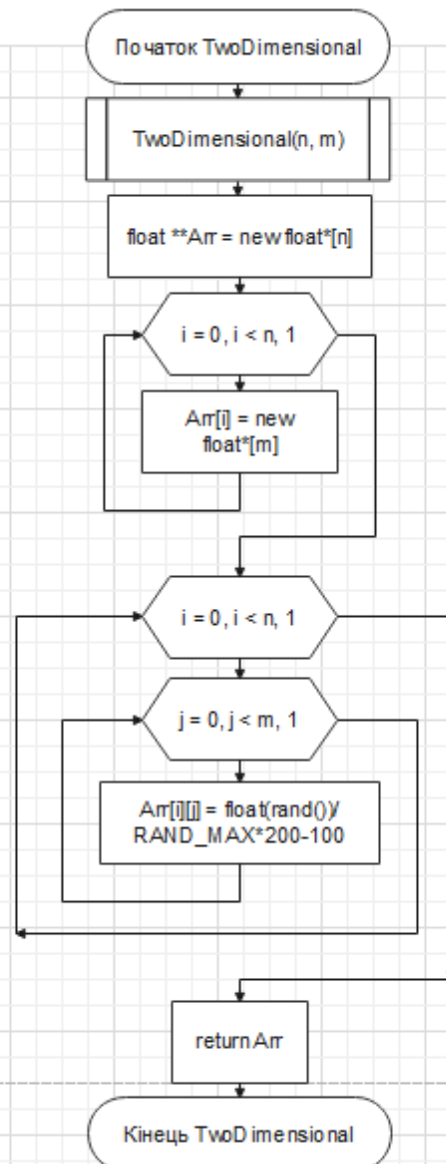
Кінець remove

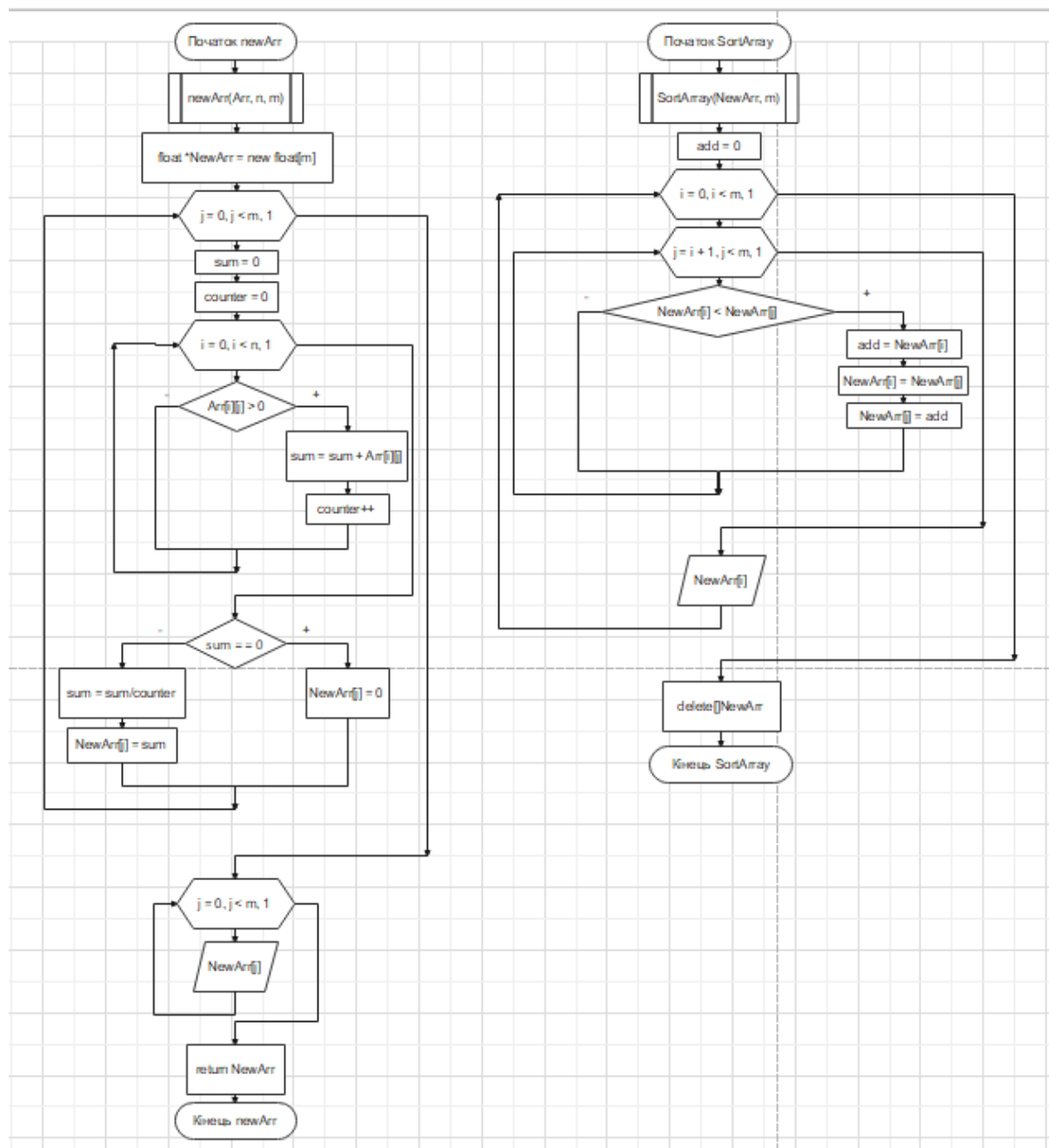
Блок-схема:

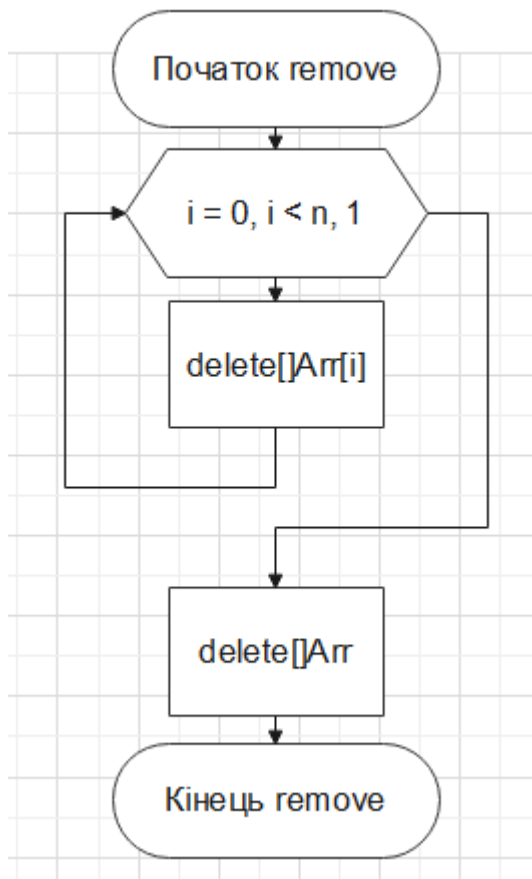
Головна
програма



Підпрограми







Код програми:

```
1  #include <iostream>
2  #include <iomanip>
3  #include <ctime>
4
5  using namespace std;
6
7  float** TwoDimensional(int, int);
8  void output(float**, int, int);
9  float* newArr(float**, int, int);
10 void SortArray(float*, int);
11 void remove(float**, int);
12
13 int main() {
14     srand(time(NULL));
15     int row, col;
16     cout << "Enter the Array[5,7]" << endl;
17     cin >> row >> col;
18     float** Arr; //показчик на двовимірний масив
19     Arr = TwoDimensional(row, col);
20 }
```

```

21     cout << "Two dimensional array: " << endl; output(Arr, row, col);
22     cout << endl;
23     cout << "New array:";
24
25     float* NewArr = newArr(Arr, row, col);
26
27     cout << "New sorted array: "; SortArray(NewArr, col);
28     remove(Arr, row);
29 }
30 //Створення матриці
31 float** TwoDimensional(int n, int m) {
32     float** Arr = new float* [n];
33     for (int i = 0; i < n; i++) {
34         Arr[i] = new float[m];
35     }
36     for (int i = 0; i < n; i++) {
37         for (int j = 0; j < m; j++) {
38             Arr[i][j] = float(rand()) / RAND_MAX * 200 - 100;
39         }
40     }

```

```

41     return Arr;
42 }
43 //Вивід матриці
44 void output(float** arr, int n, int m) {
45     for (int i = 0; i < n; i++) {
46         for (int j = 0; j < m; j++) {
47             cout << setw(7) << fixed << setprecision(1) << arr[i][j];
48         }
49         cout << endl;
50     }
51 }
52 // Створення масиву, за умовою задачі
53 float* newArr(float** arr, int n, int m) {
54     float* NewArr = new float[m];
55     for (int j = 0; j < m; j++) {
56         float sum = 0;
57         int counter = 0;
58         for (int i = 0; i < n; i++) {
59             if (arr[i][j] > 0) {
60                 sum += arr[i][j];

```

```

61                 counter++;
62             }
63         }
64         if (sum == 0) {
65             NewArr[j] = 0;
66         }
67         else {
68             sum /= counter;
69             NewArr[j] = sum;
70         }
71     }
72     for (int j = 0; j < m; j++) {
73         cout << setw(7) << NewArr[j];
74     }
75     cout << endl << endl;
76     return NewArr;
77 }
78 //Сортування масиву
79 void SortArray(float* NewArr, int m) {
80     float add = 0;

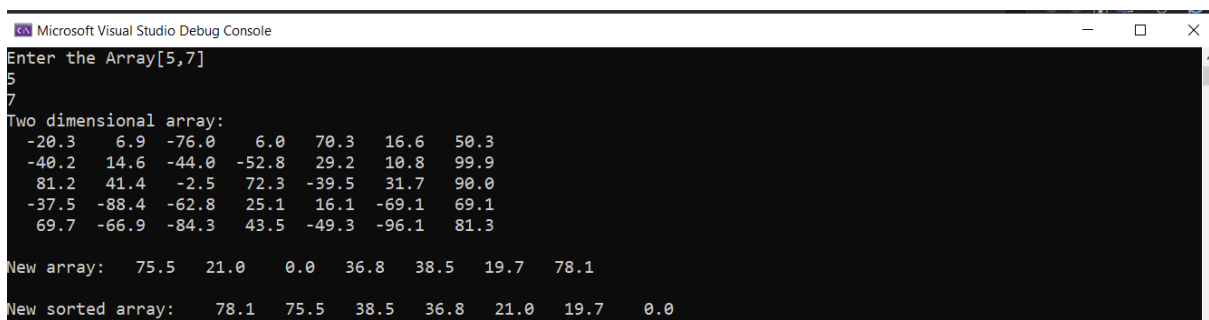
```

```

81     for (int i = 0; i < m; i++) {
82         for (int j = i + 1; j < m; j++) {
83             if (NewArr[i] < NewArr[j]) {
84                 add = NewArr[i];
85                 NewArr[i] = NewArr[j];
86                 NewArr[j] = add;
87             }
88         }
89         cout << setw(7) << NewArr[i];
90     }
91     delete[] NewArr;
92 }
93 //Видалення матриці
94 void remove(float** arr, int n) {
95     for (int i = 0; i < n; i++) {
96         delete[] arr[i];
97     }
98     delete[] arr;
99 }

```

Тестування програми:



Microsoft Visual Studio Debug Console

```

Enter the Array[5,7]
5
7
Two dimensional array:
-20.3   6.9  -76.0   6.0   70.3   16.6   50.3
-40.2  14.6  -44.0  -52.8   29.2   10.8   99.9
 81.2  41.4   -2.5   72.3  -39.5   31.7   90.0
-37.5 -88.4  -62.8   25.1   16.1  -69.1   69.1
 69.7 -66.9  -84.3   43.5  -49.3  -96.1   81.3

New array:   75.5   21.0    0.0   36.8   38.5   19.7   78.1
New sorted array:  78.1   75.5   38.5   36.8   21.0   19.7    0.0

```

Висновок: На цій лабораторній роботі, ми дослідили алгоритми пошуку та сортування, набули практичних навичок використання цих алгоритмів під час складання програмних специфікацій. Ми створили програму, яка з стовпчиків двовимірного масиву створює новий одновимірний масив, який потім ми сортуємо за спаданням.