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 26

Виконав студент ІП-11 Рябов Юрій Ігорович
(шифр, прізвище, ім'я, по батькові)

Перевірив Мартинова Оксана Петрівна
(прізвище, ім'я, по батькові)

Київ 2021

Лабараторна робота№8
Дослідження алгоритмів пошуку та сортування

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

Індивідуальне завдання:

Варіант 26

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).

26	6 x 5	Дійсний	Із середнього арифметичного від'ємних значень елементів рядків двовимірного масиву. Відсортувати методом бульбашки за спаданням.

Постановка задачі

Необхідно створити та заповнити випадковими дійсними числами матрицю 6*5, в кожному рядку якої обрахувати середнє значення від'ємних елементів, скласти з цих середніх значень послідовність та відсортувати її за спаданням. Вхідних даних достатньо, результатом виконання програми є відсортована послідовність.

Побудова математичної моделі

Складемо таблицю змінних

Змінна	Тип	Ім'я	Призначення
Матриця	дійсний	matrix	Вхідне дане
Послідовність	дійсний	array	Проміжне дане, результат
Кількість від'ємних елементів	цілий	numberOfNegative	Змінна підпрограми
Сума від'ємних елементів	дійсний	sumOfNegative	Змінна підпрограми
Середнє від'ємних	дійсний	averageNegative	Змінна підпрограми
Буфер	дійсний	buffer	Змінна підпрограми

Згенеруємо матрицю 5*6, використавши в псевдокоді та блоксхемі позначення random(), за допомогою підпрограми FillAverageNegative заповнимо масив середніми від'ємних членім кожного рядка матриці, які заходитемо за допомогою підпрограми FindAverageNegative, після чого відсортуємо його сортуванням бульбашкою за допомогою підпрограми SortBubbleDescent.

Розв'язання

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

Крок 1. Визначимо основні дії

Крок 2. Деталізуємо заповнення масиву середніми від'ємних членів рядків

Крок 3. Деталізуємо знаходження середнього від'ємних членів рядка

Крок 4. Деталізуємо сортування масиву бульбашкою

Крок 5. Деталізуємо виведення матриці

Крок 6. Деталізуємо виведення масиву

Псевдокод

Основна програма

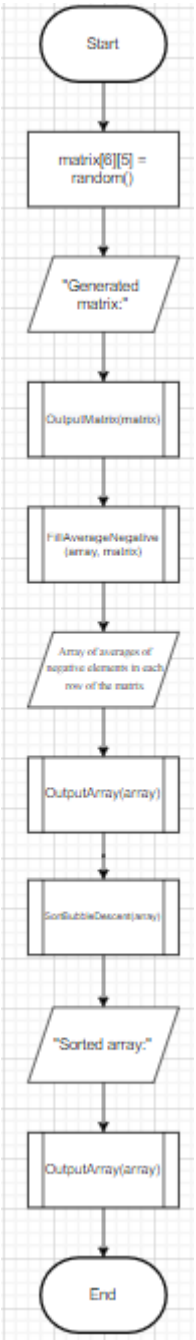
```
Start
        matrix[6][5] = random()
        output "Generated matrix"
        OutputMatrix(matrix)
        FillAverageNegative(array, matrix)
        output "Array of averages of negative elements in each row of the
matrix"
        OutputArray(array)
        SortBubbleDescent(array)
        output "Sorted array:"
        OutputArray(array)
End
```

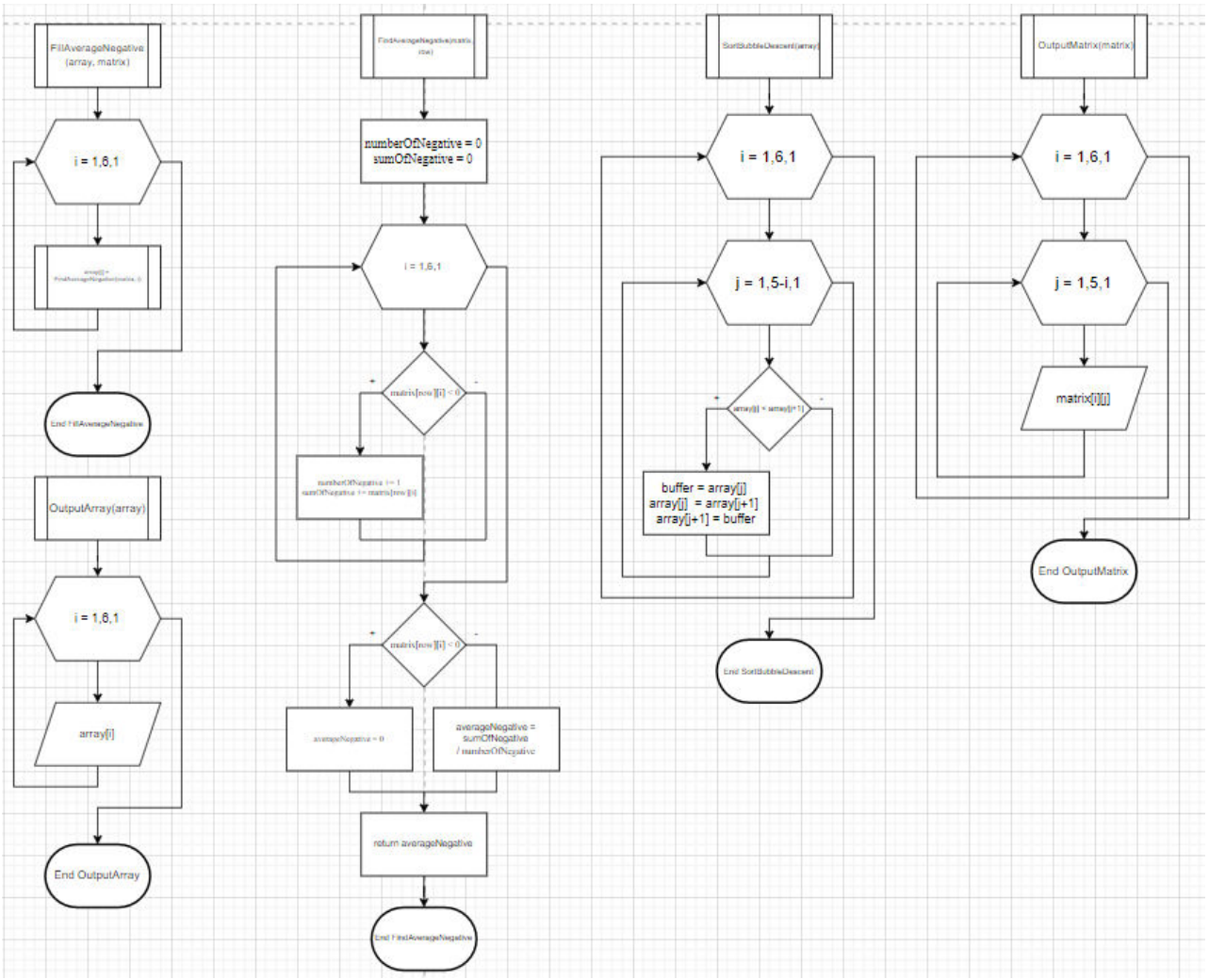
Підпрограми

```
FillAverageNegative(array, matrix)
        repeat for i from 1 to 6
            array[i] =  FindAverageNegative(matrix, i)
        end repeat
End FillAverageNegative


FindAverageNegative(matrix, row)
            numberOfNegative = 0
            sumOfNegative = 0
            repeat for i from 1 to 5
                if matrix[row][i] < 0
                        sumOfNegative += matrix[row][i]
                        numberOfNegative += 1
                end if
            end repeat
            if  numberOfNegative == 0
                averageNegative = 0
                else
                        averageNegative =  sumOfNegative /  numberOfNegative
            end if
            return  averageNegative
End FindAverageNegative


SortBubbleDescent(array)
            repeat for i from 1 to 6
                repeat for j from 1 to 5-i
                        if array[j] < array[j+1]
                                buffer = array[j]
                                array[j] = array[j+1]
                                array[j+1] = buffer
                        end if
                end repeat
```

```
        end repeat
End SortBubbleDescent

OutputMatrix(matrix)
        repeat for i from 1 to 6
            repeat for j from 1 to 5
                output matrix[i][j]
            end repeat
        end repeat
End OutputMatrix

OutputArray(array)
        repeat for i from 1 to 6
            output array[i]
        end repeat
End OutputArray
```

Блок-схема


Start
matrix[6][5] = random()
"Generated matrix:"
OutputMatrix(matrix)
FillAverageNegative (array, matrix)
Array of averages of negative elements in each row of the matrix
OutputArray(array)
SortBubbleDescent(array)
"Sorted array:"
OutputArray(array)
End


FillAverageNegative (array, matrix)
i = 1,6,1
End FillAverageNegative
OutputArray(array)
i = 1,6,1
array[i]
End OutputArray
numberOfNegative = 0
sumOfNegative = 0
i = 1,6,1
matrix[row][i] < 0
averageNegative = 0
averageNegative = sumOfNegative / numberOfNegative
return averageNegative
SortBubbleDescent(array)
i = 1,6,1
j = 1,5-i,1
buffer = array[j]
array[j] = array[j+1]
array[j+1] = buffer
End SortBubbleDescent
OutputMatrix(matrix)
i = 1,6,1
j = 1,5,1
matrix[i][j]
End OutputMatrix

Код програми

```
#include <iostream>

#include <ctime>
#include <iomanip>
#include <cmath>

double** GenerateMatrix();
void FillAverageNegative(double* array, double** matrix);
double FindAvergeNegative(double** matrix, int row);
void SortBubbleDescent(double* array);
void OutputMatrix(double** matrix);
void OutputArray(double* array);
void DeleteMatrix(double** matrix);

int main()
{
	double** matrix;
	double* array = new double[6];

	matrix = GenerateMatrix();
	std::cout << "Generated matrix:\n";
	OutputMatrix(matrix);

	FillAverageNegative(array, matrix);
	std::cout << "Array of averages of negative elements in each row of the matrix:\n";
	OutputArray(array);

	SortBubbleDescent(array);
	std::cout << "Sorted array:\n";
	OutputArray(array);

	DeleteMatrix(matrix);
	delete[] array;

	system("pause");
}

double** GenerateMatrix()
{
	double** matrix = new double* [6];
	for (int i = 0; i < 6; i++)
	{
		matrix[i] = new double [5];
	}

	srand(time(NULL));
	for (int i = 0; i < 6; i++)
	{
		for (int j = 0; j < 5; j++)
		{
			matrix[i][j] = round(100 * ((double)(rand() % 2001 - 1000) / (double)(rand() % 1000 + 1))) /
100.0;
		}
	}

	return matrix;
}

void FillAverageNegative(double* array, double** matrix)
{
	for (int i = 0; i < 6; i++)
	{
		array[i] = FindAvergeNegative(matrix, i);
	}
}

double FindAvergeNegative(double** matrix, int row)
{
```

```
	double sumOfNegative = 0;
	int numberOfNegative = 0;
	double averageNegative;

	for (int i = 0; i < 5; i++)
	{
		if (matrix[row][i] < 0)
		{
			sumOfNegative += matrix[row][i];
			numberOfNegative += 1;
		}
	}

	if (numberOfNegative == 0)
	{
		averageNegative = 0;
	}
	else
	{
		averageNegative = round(100 * sumOfNegative / numberOfNegative) / 100.0;
	}

	return averageNegative;
}

void SortBubbleDescent(double* array)
{
	double buffer;

	for (int i = 0; i < 6; i++)
	{
		for (int j = 0; j < 5 - i; j++)
		{
			if (array[j] < array[j + 1])
			{
				buffer = array[j];
				array[j] = array[j + 1];
				array[j + 1] = buffer;
			}
		}
	}
}

void OutputMatrix(double** matrix)
{
	for (int i = 0; i < 6; i++)
	{
		for (int j = 0; j < 5; j++)
		{
			std::cout << std::setw(7) << matrix[i][j];
		}
		std::cout << "\n";
	}
}

void OutputArray(double* array)
{
	for (int i = 0; i < 6; i++)
	{
		std::cout << array[i] << " ";
	}
	std::cout << "\n";
}

void DeleteMatrix(double** matrix)
{
	for (int i = 0; i < 6; i++)
	{
		delete[] matrix[i];
```

```
        }
        delete[] matrix;
}
```

```
Generated matrix:
   1.22  -1.62  -1.54  -4.33   3.98
   0.33  -0.62   1.26  -1.01   0.12
   3.09  13.47   1.08   0.44   1.01
   0.74   6.48  -0.28  -0.91   1.51
  11.93  -0.89  -0.05   0.96   1.19
   0.81   5.11   -1.6   0.31   0.64
Array of averages of negative elements in each row of the matrix:
-2.5 -0.82 0 -0.6 -0.47 -1.6
Sorted array:
0 -0.47 -0.6 -0.82 -1.6 -2.5
Press any key to continue . . .
```

Перевірка

Блок	Дія
	Початок
1	Згенерована матриця: -0.38 -1.37 -26 -0 -0.27 3.32 0.92 -0.34 -0.59 -1.14 1.86 8.26 -0.02 0.59 -0.57 -2.38 -1.39 1.85 -0.6 -0.68 2 -0.35 1.15 12.41 -4.55 -0.03 -1.52 1.43 0.14 1.42
2	Виведення матриці
3	Середнє значення від’ємних елементів 1 рядка: 2.5, 2 рядка: -0.825, 3 рядка: 0(всі елементи додатні), 4 рядка: -0.6, 5 рядка: -0.47, 6 рядка: -1.6
4	Виведення послідовності
5	Відсортована послідовність: 0 -0.47 -0.6 -0.82 -1.6 -2.5
6	Виведення відсортованої послідовності
	Кінець

Висновок

Отже, ми дослідили алгоритми сортування, створивши алгоритм з використанням сортування бульбашкою