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 № 9 з дисципліни
«Алгоритми та структури даних-1.
Основи алгоритмізації»

«Дослідження алгоритмів обходу

масивів»

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№9
Дослідження алгоритмів

обходу масивів

Мета – дослідити алгоритми обходу масивів, набути практичних навичок використання цих алгоритмів під час складання програмних специфікацій.

Індивідуальне завдання:

Варіант 26

1. Опису змінної індексованого типу (двовимірний масив) згідно з варіантом (табл. 1).
2. Ініціювання змінної, що описана в п.1 даного завдання.
3. Обчислення змінної, що описана в п.1, згідно з варіантом (табл. 1).

26	Задано матрицю дійсних чисел A[m,n]. При обході матриці по стовпчиках знайти в ній останній додатний елемент X і його місцезнаходження. Підрахувати кількість елементів над побічною діагоналлю, більших за X.

Постановка задачі

Необхідно за допомогою обходу по стовпцях знайти в матриці останній додатний елемент, після чого порівняти його з усіма елементами вище побічної діагоналі і підрахувати кількість тих з них, що більше за останній додатний. Вхідних даних достатньо, результатом виконання алгоритму є кількість елементів матриці над побічною діагоналлю, що більше за останній додатний.

Побудова математичної моделі

Складемо таблицю змінних

Змінна	Тип	Ім’я	Призначення
Матриця	дійсний	matrix	Вхідне дане
Розмір матриці	цілий	size	Вхідне дане
Останній додатний елемент	дійсний	lastPositive	Результат
Ряд останнього додатного елемента	цілий	lastPositiveRow	Результат
Колонка останнього додатного елемента	цілий	lastPositiveColumn	Результат
Кількість елементів над побічною діагоналлю більше за останній додатний	цілий	numberAboveMore	Результат

Згенеруємо матрицю, заповнивши її випадковими цілими числами, що позначимо в псевдокоді та блоксхемі як random(), за допомогою підпрограми FindLastPositive, а також MoveUp і MoveDown обійдемо матрицю по стовпцям та знайдемо в ній останній додатний елемент, після чого за допомогою підпрограми FindNumberAboveMore знайдемо кількість елементів над побічною діагоналлю, що більше за знайдений останній додатний елемент.

Розв'язання

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

Крок 1. Визначимо основні дії

Крок 2. Деталізуємо знаходження останнього додатного елемента

Крок 3. Деталізуємо рух вниз по матриці

Крок 4. Деталізуємо рух вгору по матриці

Крок 5. Деталізуємо знаходження кількості елементів над побічною діагоналлю більше за останній додатний

Крок 6. Деталізуємо виведення матриці

Псевдокод

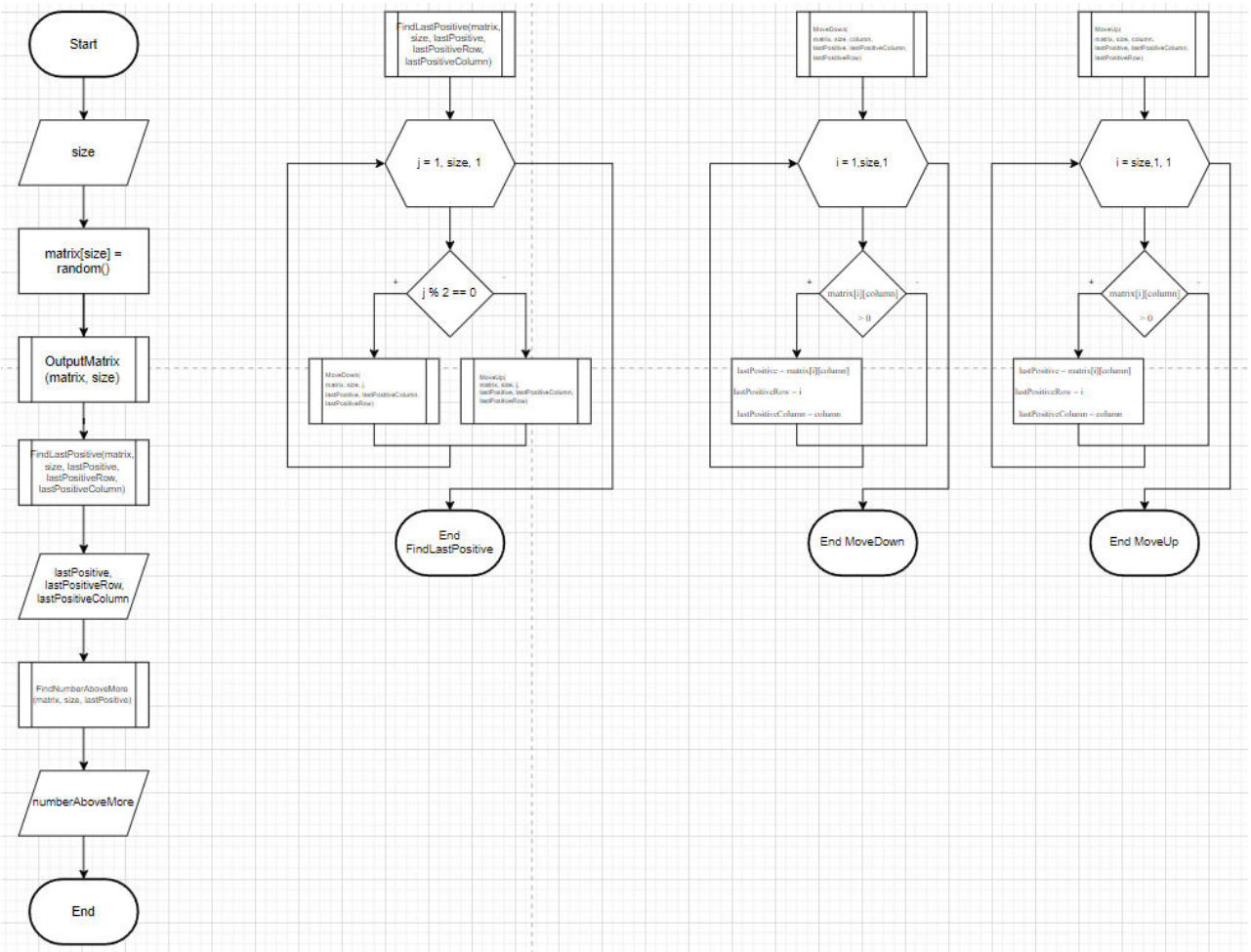
Основна програма

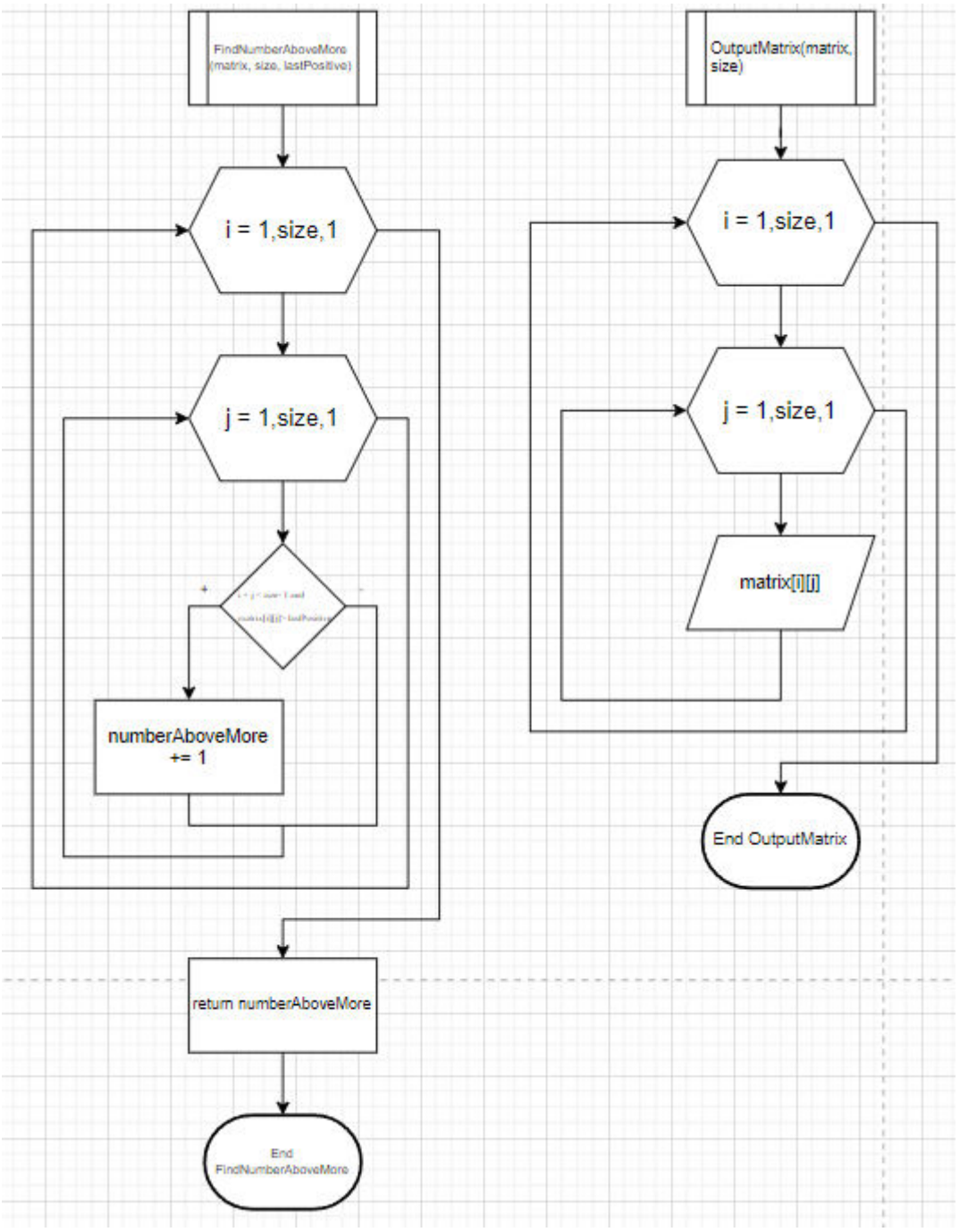
```
Start
        input size
        matrix[size] = random()
        OutputMatrix(matrix, size)
        FindLastPositive(matrix, size,  lastPositive, lastPositiveColumn,
lastPositiveRow)
        output  lastPositive, lastPositiveColumn, lastPositiveRow
        numberAboveMore = FindNumberAboveMore(matrix, size, lastPositive)
        output numberAboveMore
End
```

```
FindLastPositive(matrix, size,  lastPositive, lastPositiveColumn,
lastPositiveRow)
        repeat for j from 1 to size
            if j % 2 == 0
                MoveDown( matrix, size, j, lastPositive, lastPositiveColumn,
lastPositiveRow)
                else
                        MoveUp( matrix, size, j, lastPositive, lastPositiveColumn,
lastPositiveRow)
            end if
        end repeat
End FindLastPositive

MoveDown( matrix, size, column, lastPositive, lastPositiveColumn,
lastPositiveRow)
        repeat for i from size – 1 to 1
            if matrix[i][column] > 0
                lastPositive = matrix[i][column]
                lastPositiveRow = i
                lastPositiveColumn = column
            end if
        end repeat
end MoveDown

MoveUp( matrix, size, column, lastPositive, lastPositiveColumn,
lastPositiveRow)
        repeat for i from 1 to size
            if matrix[i][column] > 0
                lastPositive = matrix[i][column]
                lastPositiveRow = i
                lastPositiveColumn = column
            end if
```

```
            end repeat
end MoveUp

FindNumberAboveMore(matrix, size, lastPositive)
             repeat for i from 1 to size
                  repeat for j from 1 to size
                     if i + j < size - 1 and matrix[i][j] > lastPositive
                           numberAboveMore += 1
                     end if
                 end repeat
            end repeat
            return numberAboveMore
End FindNumberAboveMore

OutputMatrix(matrix, size)
            repeat for i from 1 to size
                  repeat for j from 1 to size
                     output matrix[i][j]
                 end repeat
            end repeat
End OutputMatrix
```

Блок-схема


Start
size
matrix[size] = random()
OutputMatrix (matrix, size)
FindLastPositive(matrix, size, lastPositive, lastPositiveRow, lastPositiveColumn)
lastPositive, lastPositiveRow, lastPositiveColumn
FindNumberAboveMore (matrix, size, lastPositive)
numberAboveMore
End
FindLastPositive(matrix, size, lastPositive, lastPositiveRow, lastPositiveColumn)
j = 1, size, 1
j % 2 == 0
End FindLastPositive
i = 1,size,1
matrix[i][column] > 0
lastPositive = matrix[i][column]
lastPositiveRow = i
lastPositiveColumn = column
End MoveDown
i = size,1, 1
matrix[i][column] > 0
lastPositive = matrix[i][column]
lastPositiveRow = i
lastPositiveColumn = column
End MoveUp


FindNumberAboveMore (matrix, size, lastPositive)
i = 1,size,1
j = 1,size,1
numberAboveMore += 1
return numberAboveMore
End FindNumberAboveMore
OutputMatrix(matrix, size)
i = 1,size,1
j = 1,size,1
matrix[i][j]
End OutputMatrix

Код програми

```
#include <iostream>

#include <ctime>
#include <iomanip>

double** GenerateMatrix(int size);
void FindLastPositive(double** matrix, int size, double& lastPositive, int& lastPositiveRow, int& lastPositiveColumn);
void MoveUp(double** matrix, int size, int column, double& lastPositive, int& lastPositiveRow, int& lastPositiveColumn);
void MoveDown(double** matrix, int size, int column, double& lastPositive, int& lastPositiveRow, int&
lastPositiveColumn);
int FindNumberAboveMore(double** matrix, int size, double lastPositive);
void OutputMatrix(double** matrix, int size);
void DeleteMatrix(double** matrix, int size);


int main()
{
	double** matrix;
	int size;
	double lastPositive;
	int lastPositiveRow;
	int lastPositiveColumn;
	int numberAboveMore;

	std::cout << "Enter matrix size: ";
	std::cin >> size;

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

	FindLastPositive(matrix, size, lastPositive, lastPositiveRow, lastPositiveColumn);
	std::cout << "\nLast positive element: " << lastPositive << "\nIt's row and column: " << lastPositiveRow << ", " <<
lastPositiveColumn;

	numberAboveMore = FindNumberAboveMore(matrix, size, lastPositive);
	std::cout << "\nNumber of elements above not main diagonal more than last positive element: " <<
numberAboveMore << "\n";

	DeleteMatrix(matrix, size);

	system("pause");
}

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

	srand(time(NULL));
	for (int i = 0; i < size; i++)
	{
		for (int j = 0; j < size; j++)
		{
			matrix[i][j] = (double)(rand() % 20000 - 10000) / 100.0;
		}
	}

	return matrix;
}

void FindLastPositive(double** matrix, int size, double& lastPositive, int& lastPositiveRow, int& lastPositiveColumn)
{
	for (int j = 0; j < size; j++)
	{
```

```
                if (j % 2 == 0)
                {
                        MoveDown(matrix, size, j, lastPositive, lastPositiveRow, lastPositiveColumn);
                }
                else
                {
                        MoveUp(matrix, size, j, lastPositive, lastPositiveRow, lastPositiveColumn);
                }
        }
}

void MoveUp(double** matrix, int size, int column, double& lastPositive, int& lastPositiveRow, int& lastPositiveColumn)
{
        for (int i = size - 1; i >= 0; i--)
        {
                if (matrix[i][column] > 0)
                {
                        lastPositive = matrix[i][column];
                        lastPositiveRow = i;
                        lastPositiveColumn = column;
                }
        }
}

void MoveDown(double** matrix, int size, int column, double& lastPositive, int& lastPositiveRow, int&
lastPositiveColumn)
{
        for (int i = 0; i < size; i++)
        {
                if (matrix[i][column] > 0)
                {
                        lastPositive = matrix[i][column];
                        lastPositiveRow = i;
                        lastPositiveColumn = column;
                }
        }
}

int FindNumberAboveMore(double** matrix, int size, double lastPositive)
{
        int numberAboveMore = 0;

        for (int i = 0; i < size; i++)
        {
                for (int j = 0; j < size; j++)
                {
                        if (i + j < size - 1 && matrix[i][j] > lastPositive)
                        {
                                numberAboveMore++;
                        }
                }
        }

        return numberAboveMore;
}

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

void DeleteMatrix(double** matrix, int size)
```

```
{
        for (int i = 0; i < size; i++)
        {
                delete[] matrix[i];
        }
        delete[] matrix;
}
```

```
Enter matrix size: 3
Generated matrix:
  -8.55 -77.33 -85.93
  73.53 -91.42 -65.38
  12.21 -48.48 -42.64

Last positive element: 12.21
It's row and column: 2, 0
Number of elements above not main diagonal more than last positive element: 1
Press any key to continue . . .
```

Перевірка

Блок	Дія
	Початок
1	Введення розміру матриці, генерація і виведення матриці
2	Останній додатний елемент — 12.21. рядок 2, колонка 0
3	Єдиний елемент над побічною діагоналлю більший за 12.21 — 73.53
	Кінець

Висновок

Отже, ми дослідили алгоритми обходи матриць, застосувавши алгоритм обходу по стовпцях для знаходження останнього додатного елемента матриці.