Додаток 1

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_11__

Виконав студент __ІП-13, Дем’янчук Олександр Петрович_
(шифр, прізвище, ім'я, по батькові)

Перевірив ____Вєчерковська Анастасія Сергіївна____
(прізвище, ім'я, по батькові)

Лабораторна 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 11

Завдання

11	Задано матрицю дійсних чисел A[n,n], ініціалізувати матрицю обходом по рядках. На побічній діагоналі матриці знайти перший максимальний елемент і останній мінімальний елементи, та поміняти їх місцями з елементами головної діагоналі.

1. Постановка задачі

Ініціалізуємо двовимірний масив дійсного типу та розмірності, що обирається користувачем. Квадратна матриця ініціалізується обходом по рядках і заповнюється випадковими значеннями. Завдяки вкладеним арифметичним циклам шукаємо мінімальне та максимальне значення на побічній діагоналі та міняємо його місцями з елементом на головній діагоналі матриці, що розташований на одному рядку з даним елементом.

Математична модель

Побудуємо таблицю імен змінних:

Змінна	***Тип***	***Ім'я***	***Призначення***
Двовимірний масив	Дійсний	matrix	Проміжні дані
Порядок квадратної матриці	Цілий	size	Вхідні дані
Значення мінімального елемента побічної діагоналі	Дійсний	minimal	Проміжні дані
Значення максимального елемента побічної діагоналі	Дійсний	minimal	Проміжні дані
Значення для обходу матриці рядками	Цілий	direction	Проміжні дані

Таблиця функцій:

Змінна	***Ім'я***
Генератор випадкових чисел від -15 до 15	random[-15.0; 15.0]
Заповнення матриці	create_matrix
Виведення елементів матриці	output_matrix
Пошук індексів першого максимального елемента	max_element
Пошук індексів останнього мінімального елемента	min_element
Виведення елементів матриці	output_matrix

Розмірність матриці вводиться користувачем і зберігається у **size**.

Створення та заповнення двовимірного масиву відбувається через підпрограму **create_matrix**, ініціалізація відбувається обходом по рядках, а елементи заповнюються випадковими дійсними числами через **random[-15.0, 15.0]**.

Виведення елементів матриці у підпрограмі **output_matrix** реалізоване для перегляду проміжних результатів та оцінки роботи алгоритму.

Підпрограма **min_element** реалізує пошук останнього мінімального елемента на побічній діагоналі, і присвоює це значення змінній **minimal.**

Підпрограма **max_element** реалізує пошук першого максимального елемента на побічній діагоналі, і присвоює це значення змінній **maximal.**

Розв'язання:

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

Крок 1. Визначимо основні дії;

Крок 2. Деталізуємо дію визначення розмірності матриці через введення значення у змінну **size;**

Крок 3. Деталізуємо підпрограму ініціалізації та заповнення двовимірного масиву **create_matrix**;

Крок 4. Деталізуємо підпрограму виведення двовимірного масиву **output_matrix**;

Крок 5. Деталізуємо підпрограму пошуку мінімального значення **minimal**;

Крок 6.Деталізуємо підпрограму пошуку максимального значення **maximal**;

Крок 7. Деталізуємо підпрограму заміни мінімального й максимального значень на побічній діагоналі та відповідних їм на головній через **switch_elements**;

Псевдокод:

Крок 1

початок

введення **size**

виклик підпрограми **create_matrix** для ініціалізації та заповнення двовимірного масиву;

виклик підпрограми **output_matrix** для виведення елементів двовимірного масиву

виклик підпрограми **min_element** для пошуку мінімального значення **minimal;**

виклик підпрограми **max_element** для пошуку мінімального значення **maximal;**

виклик підпрограми **switch_elements** для заміни місцями мінімального значення **minimal** й відповідного йому на головній

виклик підпрограми **switch_elements** для заміни місцями максимального значення **maximal** й відповідного йому на головній

виклик підпрограми **output_matrix** для виведення елементів нового двовимірного масиву

кінець

Крок 7

початок

введення **size**

create_matrix(size)

output_matrix(matrix, size)

minimal := min_element(matrix, size)

maximal := max_element(matrix, size)

switch_elements(matrix, size, minimal)

switch_elements(matrix, size, maximal)

output_matrix(matrix, size)

кінець

підпрограма create_matrix(size)

direction := 1

```
        повторити для i від 0 до size з кроком 1
            якщо direction > 0 то
                повторити для j від 0 до size з кроком 1
                    matrix[i][j] = випадкове[-10.0; 10.0]
                все повторити
            інакше
                повторити для j від size до 0 з кроком -1
                    matrix[i][j] = випадкове[-10.0; 10.0]
                все повторити
            все якщо
            direction := direction * -1
return matrix
все підпрограма

підпрограма output_matrix(matrix, size)
        повторити для i від 0 до size з кроком 1
            повторити для j від 0 до size з кроком 1
                вивести matrix[i][j]
            все повторити
        все повторити
все підпрограма

підпрограма min_element(matrix, size)
```

```
        min := matrix[0][0]
        повторити для i від 0 до size з кроком 1
            j := size - i - 1
            якщо matrix[i][j] <= min то
                        min = matrix[i][j]
            все якщо
        все повторити
return min
все підпрограма

підпрограма max_element(matrix, size)
        max := matrix[0][0]
        повторити для i від 0 до size з кроком 1
            j : size - i - 1
            якщо matrix[i][j] >= max то
                        max = matrix[i][j]
            все якщо
        все повторити
return max
все підпрограма




підпрограма switch_elements(matrix, size, num)
        повторити для i від 0 до size з кроком 1
```

повторити для i **від** 0 **до** size **з кроком 1**

якщо matrix[i][j] == num то

tmp := matrix[i][j]

matrix[i][j] := matrix[i][i]

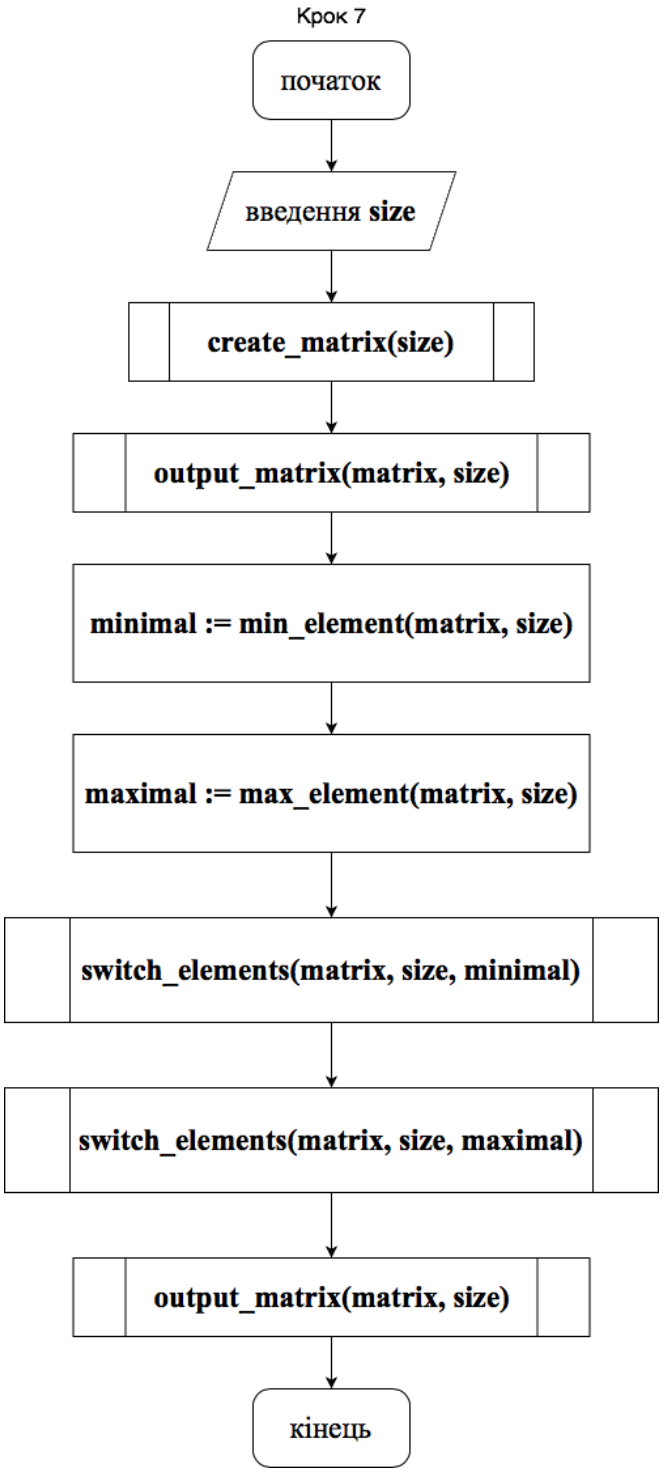
matrix[i][i] := tmp

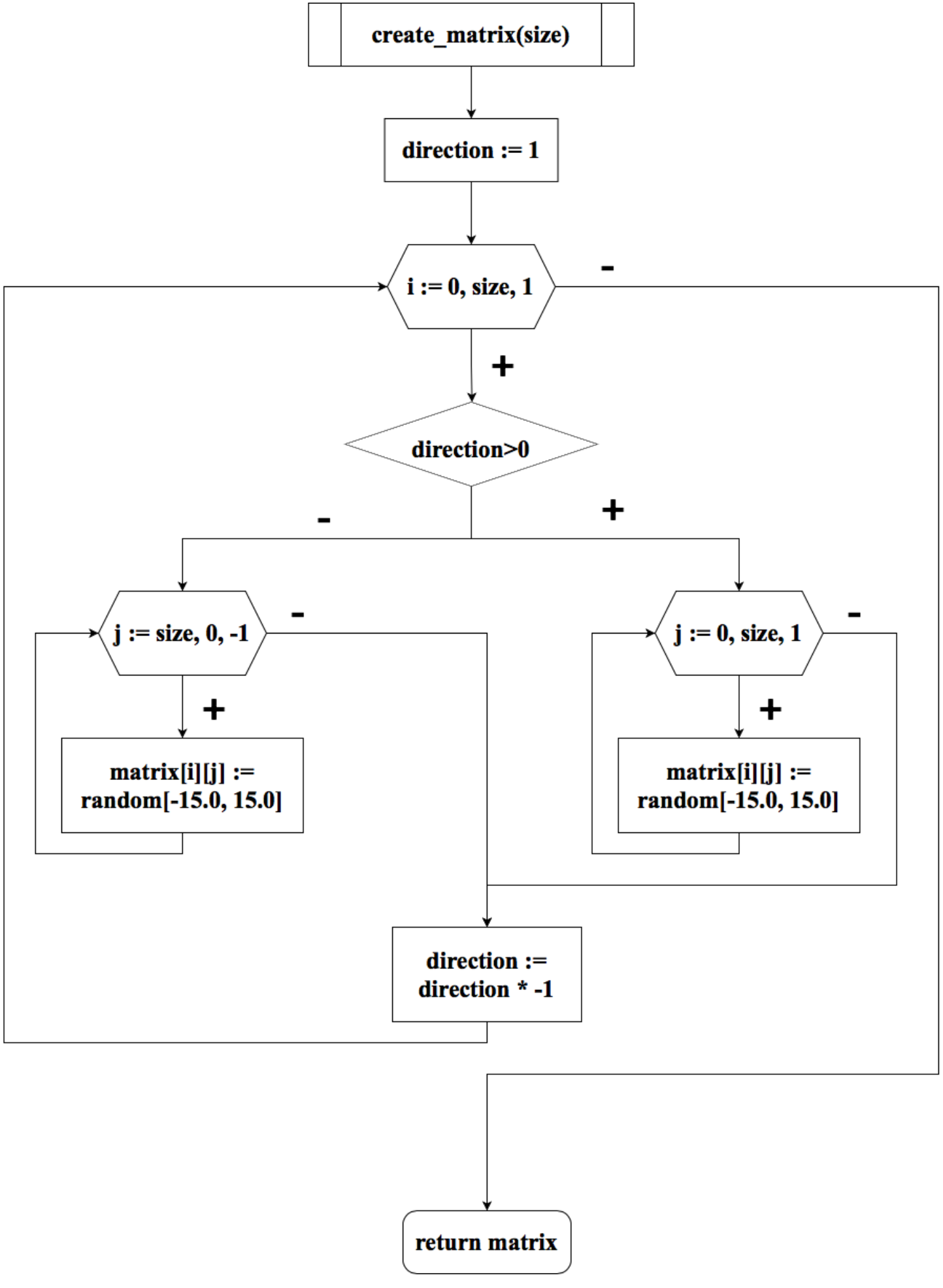
все якщо

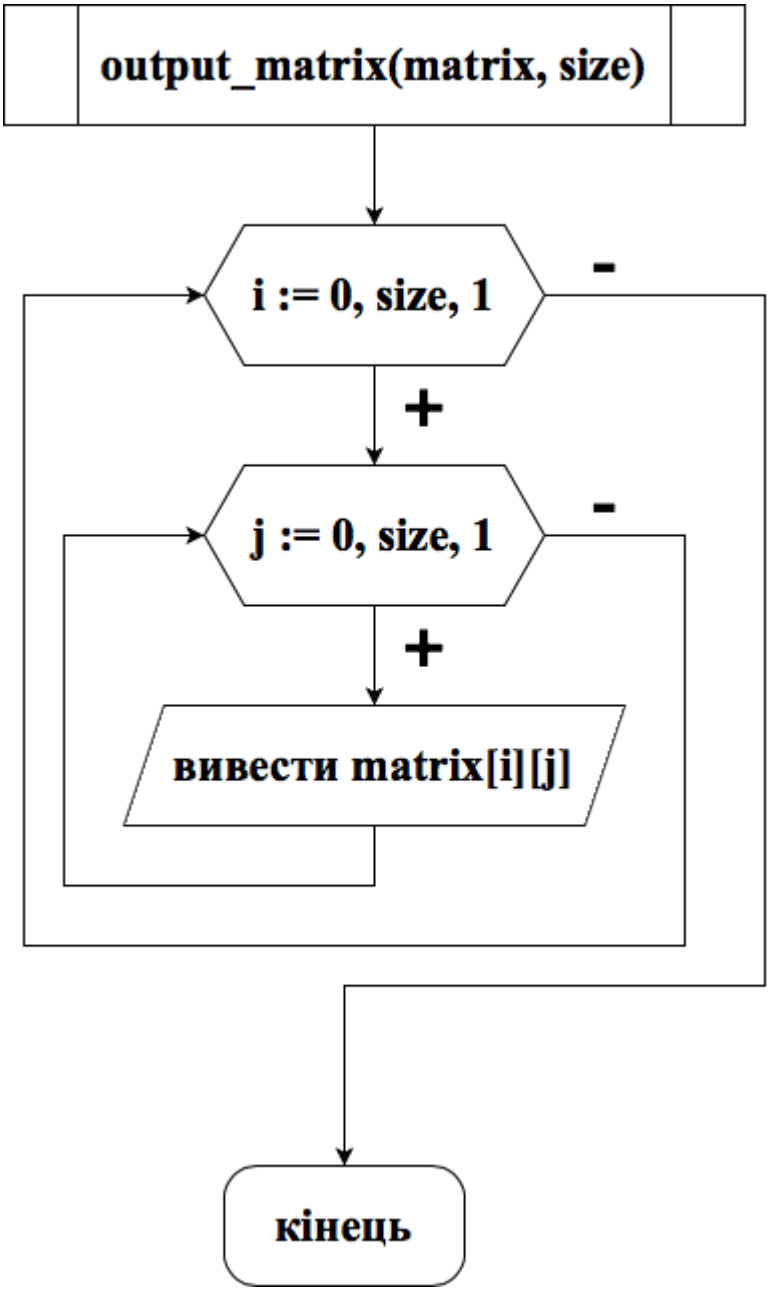
все повторити

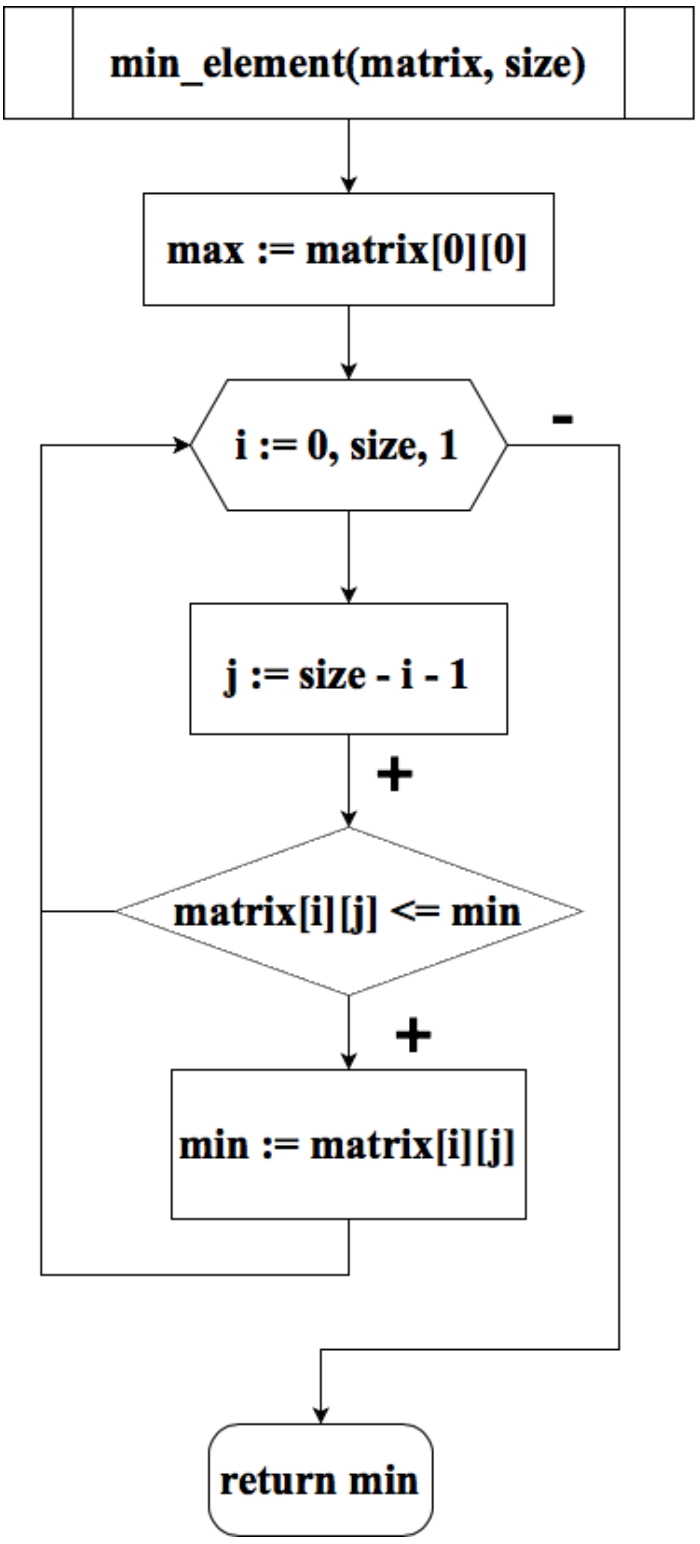
все повторити

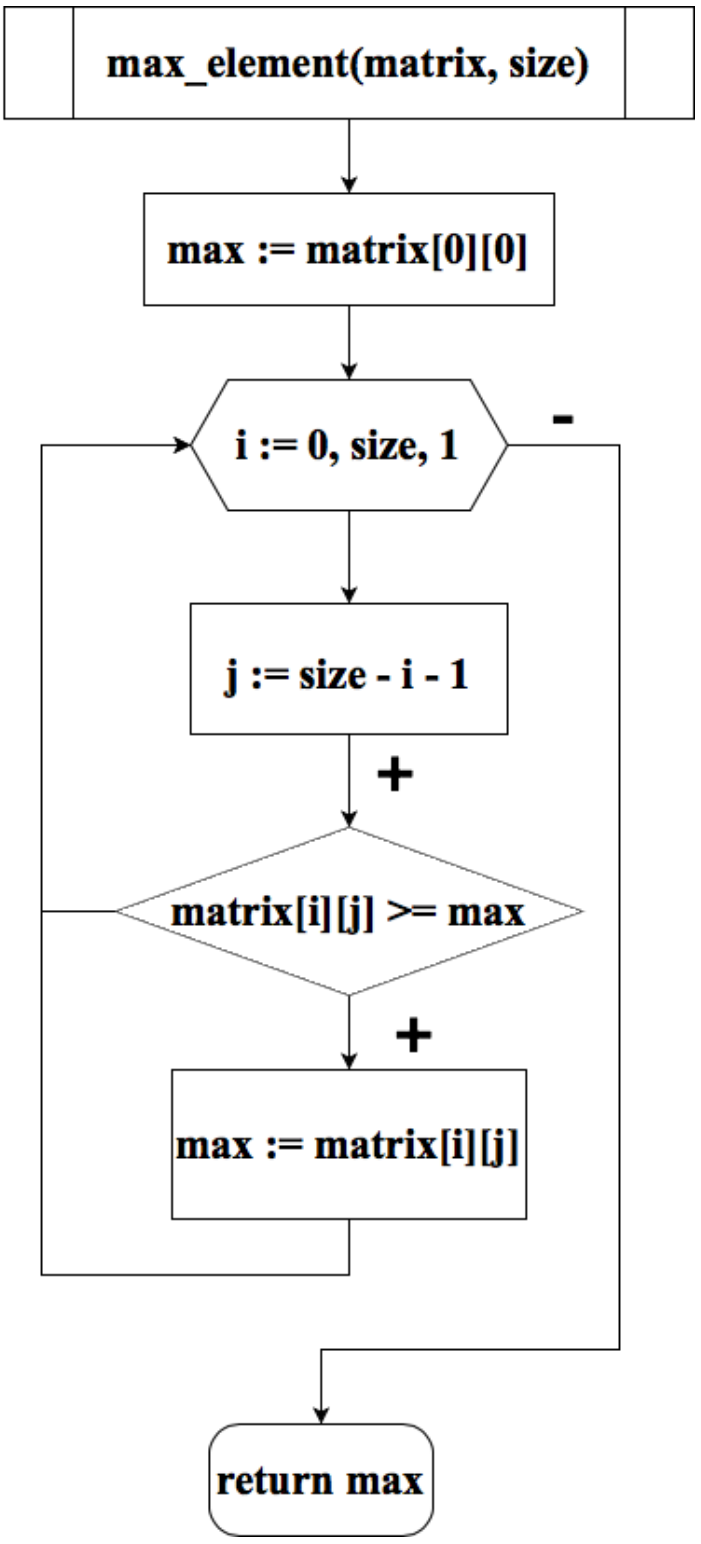
Блок-схема


Крок 7
початок
введення size
create_matrix(size)
output_matrix(matrix, size)
minimal := min_element(matrix, size)
maximal := max_element(matrix, size)
switch_elements(matrix, size, minimal)
switch_elements(matrix, size, maximal)
output_matrix(matrix, size)
кінець


create_matrix(size)
direction := 1
i := 0, size, 1
-
+
direction>0
-
+
j := size, 0, -1
-
+
matrix[i][j] :=
random[-15.0, 15.0]
j := 0, size, 1
-
+
matrix[i][j] :=
random[-15.0, 15.0]
direction :=
direction * -1
return matrix


output_matrix(matrix, size)
i := 0, size, 1
-
+
j := 0, size, 1
-
+
вивести matrix[i][j]
кінець


min_element(matrix, size)
max := matrix[0][0]
i := 0, size, 1
-
j := size - i - 1
+
matrix[i][j] <= min
+
min := matrix[i][j]
return min


max_element(matrix, size)
max := matrix[0][0]
i := 0, size, 1
-
j := size - i - 1
+
matrix[i][j] >= max
+
max := matrix[i][j]
return max

Програмний код

```cpp
#include <iostream>
#include <iomanip>

using namespace std;

double** create_matrix(int);
void output_matrix(double**, int);
double min_element(double**, int);
double max_element(double**, int);
void switch_elements(double**, int, double);
void delete_matrix(double**, int);

int main()
{
    srand(time(NULL));
    int size;
    cout << "Enter the size of matrix: \n";
    cin >> size;
    double** matrix = create_matrix(size);
    cout << "Matrix A[" << size << " x " << size << "]: \n";
    output_matrix(matrix, size);

    double minimal = min_element(matrix, size);
    double maximal = max_element(matrix, size);

    switch_elements(matrix, size, minimal);
    switch_elements(matrix, size, maximal);

    cout << "\nNew matrix A(" << size << " x " << size << "): \n";
    output_matrix(matrix, size);

    delete_matrix(matrix, size);
}
```

```
34
35  double** create_matrix(int size)
36  {
37      double upper, lower;
38       double lower
39      lower = -15.0;
40      double** matrix = new double* [size];
41      for (int i = 0; i < size; i++)
42      {
43          matrix[i] = new double[size];
44      }
45      int direction = 1;
46      for (int i = 0; i < size; i++)
47      {
48          if(direction > 0)
49          {
50              for (int j = 0; j < size; j++)
51              {
52                  matrix[i][j] = (double)(rand()) / RAND_MAX * (upper - lower + 1) + lower;
53              }
54          }
55          else
56          {
57              for (int j = size - 1; j >= 0; j--)
58              {
59                  matrix[i][j] = (double)(rand()) / RAND_MAX * (upper - lower + 1) + lower;
60              }
61          }
62          direction *= -1;
63      }
64      return matrix;
65  }
```

```
66
67  void output_matrix(double** matrix, int size)
68  {
69      for (int i = 0; i < size; i++)
70      {
71          for (int j = 0; j < size; j++)
72          {
73              cout << setw(10) << matrix[i][j];
74          }
75          cout << "\n";
76      }
77      cout << endl;
78  }
79
80  double min_element(double** matrix, int size)
81  {
82      double min = matrix[0][0];
83      int row = 0;
84      int col = 0;
85      for(int i = 0; i < size; i++)
86      {
87          int j = size - i - 1;
88          if(matrix[i][j] <= min)
89          {
90              min = matrix[i][j];
91              row = i;
92              col = j;
93          }
94
95      }
96      cout << "the minimal element is [" << row << "][" << col << "]\n";
97      return min;
98  }
99
```

```
100  double max_element(double** matrix, int size)
101  {
102      double max = matrix[0][0];
103      int row = 0;
104      int col = 0;
105      for(int i = 0; i < size; i++)
106      {
107          int j = size - i - 1;
108          if(matrix[i][j] >= max)
109          {
110              max = matrix[i][j];
111              row = i;
112              col = j;
113          }
114      }
115      cout << "the maximal element is [" << row << "][" << col << "]\n";
116      return max;
117  }
118
119  void switch_elements(double** matrix, int size, double num)
120  {
121      double tmp;
122      for (int i = 0; i < size; i++)
123      {
124          for(int j = 0; j < size; j++)
125          {
126              if(matrix[i][j] == num)
127              {
128                  tmp = matrix[i][j];
129                  matrix[i][j] = matrix[i][i];
130                  matrix[i][i] = tmp;
131              }
132          }
133      }
134  }
```

```
136    void delete_matrix(double** matrix, int size)
137    {
138        for (int i = 0; i < size; i++)
139        {
140            delete[] matrix[i];
141        }
142        delete[] matrix;
143    }
```

```
MacBook-Pro-user:Documents user$ cd "/Users/user/Documents/" && g++ lab9
ents/"lab9_asd
Enter the size of matrix:
5
Matrix A[5 x 5]:
   2.55165    10.555   13.9521   6.85728   -9.7448
  -10.8605    8.2295   11.0785   9.51824   -9.8414
  -6.18465  -4.46225  -10.0693  -7.51259  -3.14472
   13.5909   10.1916   15.2397  -2.89295 -0.306235
   12.1898  -7.30718   8.23677  -12.5609   -3.2583

the minimal element is [2][2]
the maximal element is [4][0]

New matrix A(5 x 5):
   2.55165    10.555   13.9521   6.85728   -9.7448
  -10.8605    8.2295   11.0785   9.51824   -9.8414
  -6.18465  -4.46225  -10.0693  -7.51259  -3.14472
   13.5909   10.1916   15.2397  -2.89295 -0.306235
   -3.2583  -7.30718   8.23677  -12.5609   12.1898

MacBook-Pro-user:Documents user$
```

Висновок

На лабораторній роботі дослідив алгоритми обходу масивів, набути практичних навичок використання цих алгоритмів під час складання програмних специфікацій.