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

«Дослідження лінійного пошуку в
послідовностях»

Варіант 10

Виконав студент ІП-13 Дейнега Владислав Миколайович
(шифр, прізвище, ім'я, по батькові)

Перевірив Вечерковська Анастасія Сергіївна
(прізвище, ім'я, по батькові)

Київ 2021

Лабораторна робота 9
ДОСЛІДЖЕННЯ АЛГОРИТМІВ ОБХОДУ МАСИВІВ

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

Варіант 10

Розробити алгоритм та написати програму, яка складається з наступних дій:

1. Опису дійсного двовимірного массиву розміром m x n.
2.Знаходження максимального елемента у кожному стовпцю.
3.Обміняти знайдене значення X з елементом першого стовпця.

Постановка задачі

Результатом є виведення максимального елемента кожного стовпця та його місцезнаходження, та матриці із зміною місцями максимального елемента та елемента першого стовпця.

Побудова математичної моделі

Таблиця імен змінних

Змінна	**Тип**	**Ім’я**	**Призначення**
Кількість рядків	Цілий	m	Вхідне дане
Кількість стовпців	Цілий	n	Вхідне дане
Рядок максимального елемента	Цілий	max_rows	Результат
Стовпець максимального елемента	Цілий	max_cols	Результат
Значення максимального елемента	Дійсний	max	Результат
Матриця	Дійсний	matr	Проміжне дане

Фугкція matr_fill

Змінна	**Тип**	**Ім’я**	**Призначення**

Матриця	Дійсний	matr	Проміжне дане
Кількість рядків	Цілий	rows	Проміжне дане
Кількість стовпців	Цілий	cols	Проміжне дане
Лічильник i	Цілий	i	Проміжне дане
Лічильник j	Цілий	j	Проміжне дане

Функція matr_out

Змінна	**Тип**	**Ім’я**	**Призначення**
Матриця	Дійсний	matr	Проміжне дане
Кількість рядків	Цілий	rows	Проміжне дане
Кількість стовпців	Цілий	cols	Проміжне дане
Лічильник i	Цілий	i	Проміжне дане
Лічильник j	Цілий	j	Проміжне дане

Функція matr_max

Змінна	**Тип**	**Ім’я**	**Призначення**
Матриця	Дійсний	matr	Проміжне дане
Кількість рядків	Цілий	rows	Проміжне дане
Кількість стовпців	Цілий	cols	Проміжне дане
Лічильник i	Цілий	i	Проміжне дане
Лічильник j	Цілий	j	Проміжне дане
Рядок максимального елемента	Цілий	max_rows	Результат
Стовпець максимального елемента	Цілий	max_cols	Результат
Значення максимального елемента	Дійсний	max	Результат

Функія matr_switch

Змінна	**Тип**	**Ім’я**	**Призначення**
Матриця	Дійсний	matr	Проміжне дане
Кількість рядків	Цілий	rows	Проміжне дане
Кількість стовпців	Цілий	cols	Проміжне дане
Рядок максимального елемента	Цілий	max_rows	Результат
Стовпець максимального елемента	Цілий	max_cols	Результат
Значення максимального елемента	Дійсний	max	Результат
Проміжна змінна	Дійсний	matr_temp	Проміжне дане

Спочатку ініціюємо матрицю розміром mxn і заповнюємо її випадковими числами з допомогою функції rand(). Потім знаходимо максимальний елемент у кожному стовпці та його місцезнаходження. Міняємо місцями елемент першого стовпця та максимальний елемент матриці.

Псевдокод

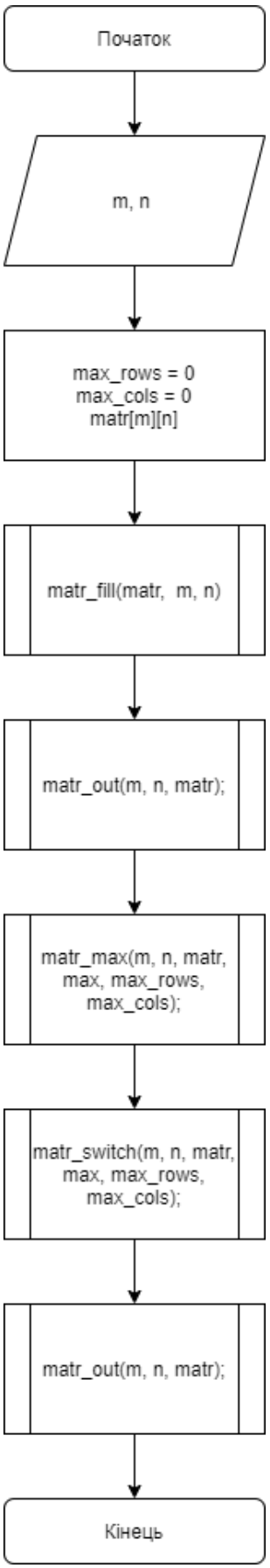
Початок
Введення m, n
max_rows = 0
max_cols = 0
max = 0
matr[i][j]
matr_fill(matr, m, n)
matr_out(m, n, matr)
matr_max(m, n, matr, max, max_rows, max_cols)
matr_switch(m, n, matr, max, max_rows, max_cols)
matr_out(m, n, matr)
Кінець

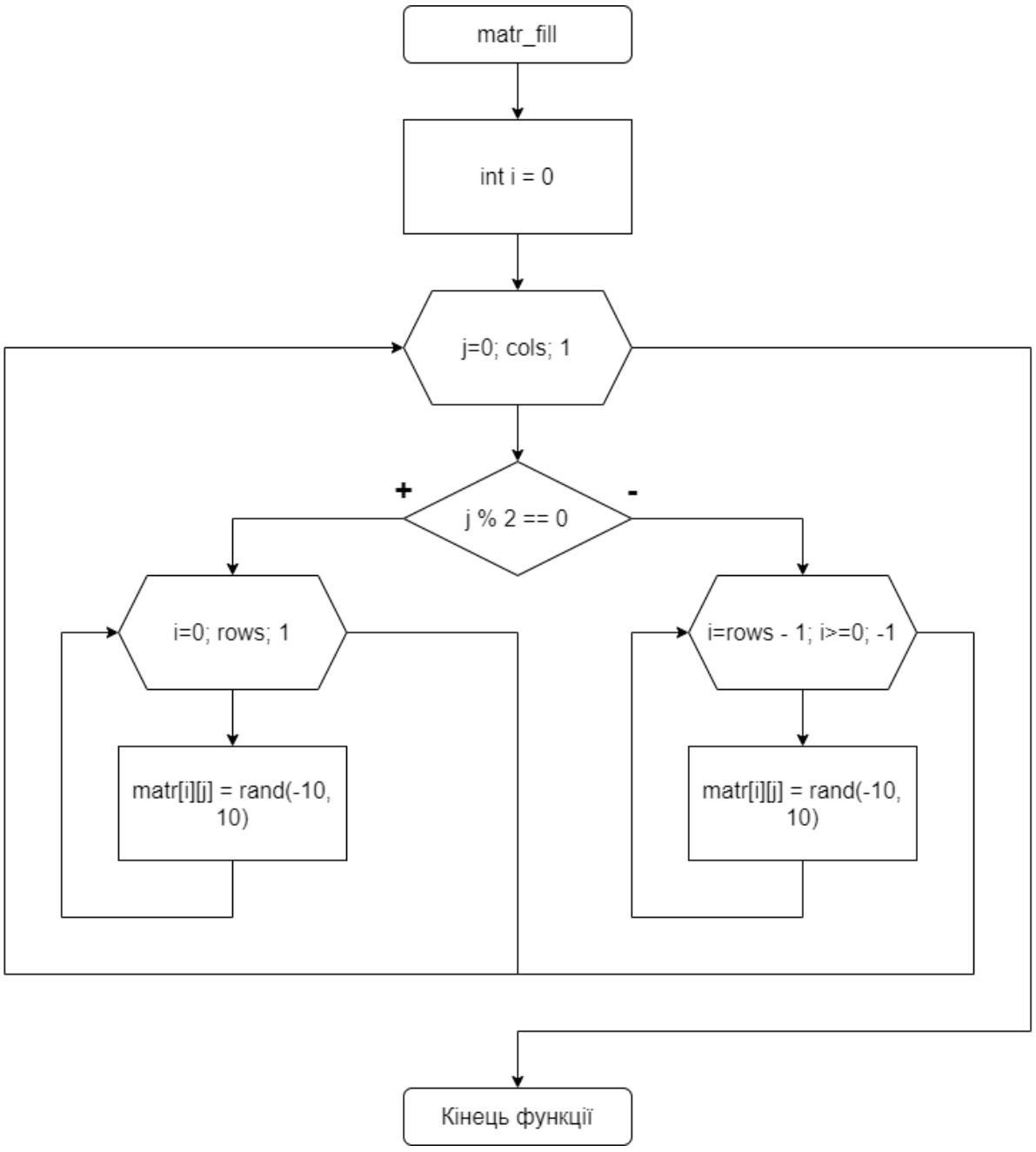
Функція matr_fill(matr, rows, cols)

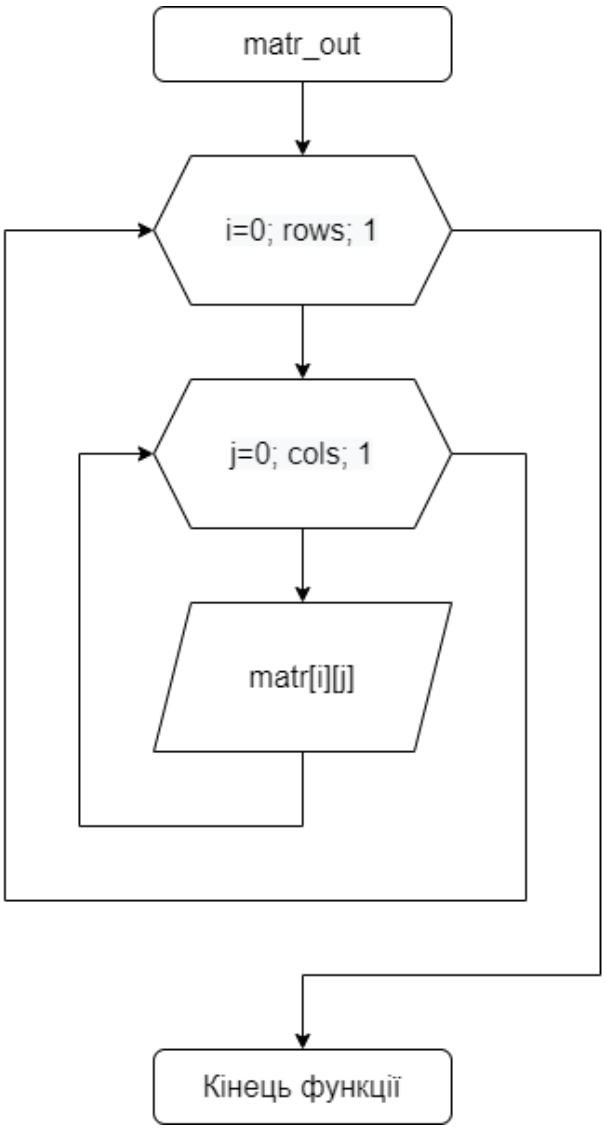
```
    i = 0
    для j від 0 до cols повторити
      якщо j % 2 == 0
          для i від 0 до raws повторити
              matr[i][j] = rand(-10, 10)
          все повторити
      інакше
          для i від rows - 1 до 0 повторити з кроком -1
              matr[i][j] = rand(-10, 10)
          все повторити
      все якщо
    все повторити
все функція

Функція matr_out(rows, cols, matr)
    для i від 0 до rows повторити
        для j від 0 до cols повторити
            Вивести matr[i][j]
        все повторити
    все повторити
все функція
```

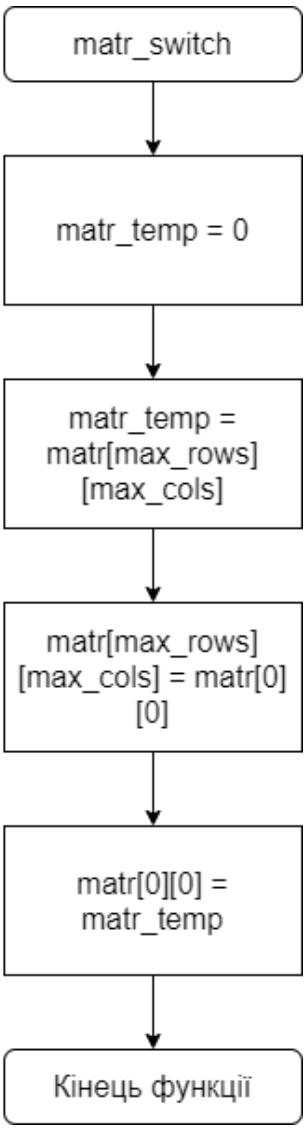
```
Функція matr_max(rows, cols, matr, max, max_rows, max_cols)
    для j від 0 до cols повторити
      max_local = matr[0][j]
      якщо j % 2 == 0
        для i від 0 до raws повторити
          якщо matr[i][j] >= max_local
            max_local = matr[i][j]
            rows_temp = i
            якщо max_local >= max
              max = max_local
              max_cols = j
              max_rows = i
            все якщо
          все якщо
        все повторити
      інакше
        для i від rows - 1 до 0 повторити з кроком -1
          якщо matr[i][j] >= max_local
            max_local = matr[i][j]
            rows_temp = i
            якщо max_local >= max
              max = max_local
              max_cols = j
              max_rows = i
            все якщо
          все якщо
        все повторити
      все якщо
    все повторити
все функція
```

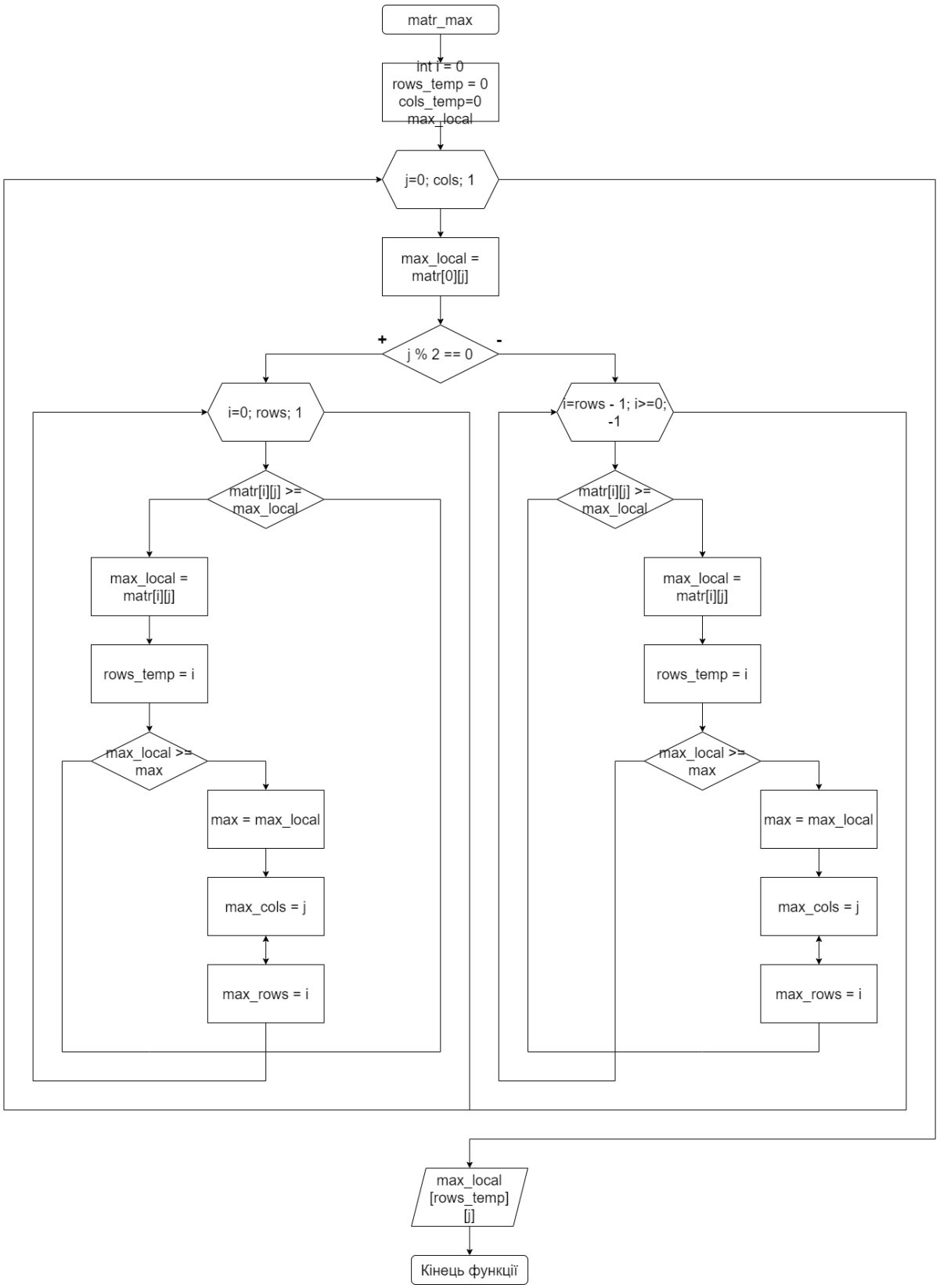
Блок-схема


Початок
m, n
max_rows = 0
max_cols = 0
matr[m][n]
matr_fill(matr, m, n)
matr_out(m, n, matr);
matr_max(m, n, matr,
max, max_rows,
max_cols);
matr_switch(m, n, matr,
max, max_rows,
max_cols);
matr_out(m, n, matr);
Кінець


matr_fill
int i = 0
j=0; cols; 1
+
-
j % 2 == 0
i=0; rows; 1
matr[i][j] = rand(-10, 10)
i=rows - 1; i>=0; -1
matr[i][j] = rand(-10, 10)
Кінець функції


matr_out
i=0; rows; 1
j=0; cols; 1
matr[i][j]
Кінець функції


matr_switch
matr_temp = 0
matr_temp = matr[max_rows] [max_cols]
matr[max_rows] [max_cols] = matr[0] [0]
matr[0][0] = matr_temp
Кінець функції


matr_max
int i = 0
rows_temp = 0
cols_temp=0
max_local
j=0; cols; 1
max_local = matr[0][j]
+
j % 2 == 0
-
i=0; rows; 1
matr[i][j] >= max_local
max_local = matr[i][j]
rows_temp = i
max_local >= max
max = max_local
max_cols = j
max_rows = i
i=rows - 1; i>=0; -1
matr[i][j] >= max_local
max_local = matr[i][j]
rows_temp = i
max_local >= max
max = max_local
max_cols = j
max_rows = i
max_local [rows_temp] [j]
Кінець функції

Код программы

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

using namespace std;

void matr_fill(float**, int, int);
void matr_out(int , int , float**);
void matr_max(int, int, float**, float&, int&, int&);
void matr_switch(int, int, float**, float, int, int);

int main()
{
    srand(time(NULL));
    int m, n, max_rows = 0, max_cols = 0;
    float max = 0;
    cin >> m;
    cin >> n;

    float** matr = new float* [m];
    for (int i = 0; i < m; i++)
    {
        matr[i] = new float[n];
    }

    matr_fill(matr,  m, n);
    matr_out(m, n, matr);
    matr_max(m, n, matr, max, max_rows, max_cols);
    matr_switch(m, n, matr, max, max_rows, max_cols);
    matr_out(m, n, matr);
}

void matr_fill(float** matr, int rows, int cols)
{
    int i = 0;
    for (int j = 0; j < cols; j++)
    {
        if (j % 2 == 0)
        {
            for (i = 0; i < rows; i++)
            {
                matr[i][j] = rand() % 21 - 10;
            }
        }
        else
```

```
45              {
46                  for (i = rows - 1; i >= 0; i--)
47                  {
48                      matr[i][j] = rand() % 21 - 10;
49                  }
50              }
51          }
52      }
53
54      void matr_out(int rows, int cols, float** matr)
55      {
56          for (int i = 0; i < rows; i++)
57          {
58              for (int j = 0; j < cols; j++)
59              {
60                  cout << setw(2) << matr[i][j] << "\t";
61              }
62              cout << endl;
63          }
64      }
65
66      void matr_max(int rows, int cols, float** matr, float& max, int& max_rows, int& max_cols)
67      {
68          float max_local;
69          int i, rows_temp = 0, cols_temp=0;
70          for (int j = 0; j < cols; j++)
71          {
72              max_local = matr[0][j];
73              if (j % 2 == 0)
74              {
75                  for (i = 0; i < rows; i++)
76                  {
77                      if (matr[i][j] >= max_local)
78                      {
79                          max_local = matr[i][j];
80                          rows_temp = i;
81                          if (max_local >= max)
82                          {
83                              max = max_local;
84                              max_cols = j;
85                              max_rows = i;
86                          }
87                      }
88                  }
```

```
89              }
90              else
91              {
92                  for (i = rows - 1; i >= 0; i--)
93                  {
94                      if (matr[i][j] >= max_local)
95                      {
96                          max_local = matr[i][j];
97                          rows_temp = i;
98                          if (max_local >= max)
99                          {
100                             max = max_local;
101                             max_cols = j;
102                             max_rows = i;
103                         }
104                     }
105                 }
106             }
107             cout << "Max element " << j + 1 << " colons: " << max_local << " [" << rows_temp << "]" "[" << j << "]" << endl;
108         }
109     }
110
111     void matr_switch(int rows, int cols, float** matr, float max, int max_rows, int max_cols)
112     {
113         float matr_temp = 0;
114         matr_temp = matr[max_rows][max_cols];
115         matr[max_rows][max_cols] = matr[0][0];
116         matr[0][0] = matr_temp;
117     }
118
119
```

Висновок

Під час виконання лабораторної роботи, я дослідив особливості алгоритмів обходу масивів та набув практичних навичок їх використання.