# 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 № 1 з дисципліни

«Основи програмування-1.

Базові конструкції»

«Багатовимірні масиви»

Варіант 19

Виконав студент: ІП-11 Лисенко Андрій Юрійович

Перевірила: Вітковська Ірина Іванівна

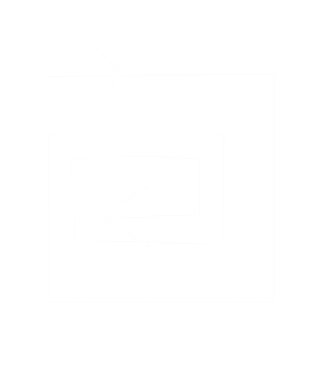
Київ 2021

**Лабораторна робота №8**

**Багатовимірні масиви**

**Мета –** опанувати технологію використання двовимірних масивів даних (матриць), навчитись розробляти алгоритми та програми із застосуванням матриць.

**Індивідуальне завдання**

**Варіант 19**

Побудувати квадратну матрицю з елементами 1*,* 2*, ..., n,* розміщеними по спіралі, починаючи з лівого верхнього кута. Знайти мінімальне значення кутових елементів отриманої матриці.

**Постановка задачі**

Для вирішення цієї задачі використаємо функцію *incr()*. За домогою цієї функцію ми будемо визначати напрям обходу по матриці та інкрементувати/декрементувати відповідні значення лічильників та заповнемо матрицю по спіралі. Функція *find\_min()* знайде мінімальне кутове значення.

**Побудова математичної моделі**

Складемо таблицю змінних

|  |  |  |  |
| --- | --- | --- | --- |
| Змінна | Тип | Ім’я | Призначення |
| Шукана матриця | *int\*\** | *matrix* | Зберігати дані елементів матриці |
| Розмір | *int* | *size* | Зберігати розмір квадратної матриці |
| Лічильники *i, j* | *int* | i, j | Зберігати дані про розташування відповідного елемента матриці |
| Маркери розташування | *int* | *UL, UR, DR, DL* | Зберігати дані про те чи знаходиться даний елемент у кутах матриці |
| Напрямок | *int* | *dir* | Зберігати дані про напрямок заповнення таблиці |
| Значення | *int* | *val* | Зберігати дані про значення елемента матриці |
| Масштабування | *int* | *scale* | Зберігати дані про масштабування елемента спіралі обходження матриці |

Складемо таблицю підпрограм:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Підпрограма | Тип | Ім’я | Параметри | Призначення |
| Ініціалізація | *int\*\** | *init()* | size | Опис квадратної матриці |
| Генерація | *void* | *generate()* | *\*\*matrix, size* | Генерація спіральної матриці |
| Інкремент | *void* | *incr()* | \*i, \*j, pos, val, \*dir, \*scale | Інкремент крока обходу матриці |
| Виведення | *void* | *output()* | \*\**matrix, size* | Виведення матриці |
| Знаходження мінімуму | *int* | *find\_min()* | \*\**matrix, size* | Знахолдження мінімума кутових елементів матриці |
| Видалення | *void* | *del()* | \*\**matrix, size* | Видалення матриці |

**Псевдокод**

**Основна програма:**

**Початок**

**Введення** *size*;

*matrix = init(size)*;

*generate(matrix, size)*;

*output(matrix, size)*;

**Виведення** *find\_min(matrix, size)*;

*del(matrix, size)*;

**Кінець**

**Підпрограма init():**

**Початок init()**

*int\*\* matrix = new int\* [size];*

**Для** *i* **від** 0 **до** *size* **повторити**:

*matrix[i] = new int[size];*

**Все повторити**

**Повернути** *matrix;*

**Кінець init()**

**Підпрограма incr():**

**Початок *incr()***

*UL = (\*i == \*scale && \*j == \*scale + 1);*

*UR = (\*i == pos - \*scale && \*j == \*scale);*

*DR = (\*i == pos - \*scale && \*j == pos - \*scale);*

*DL = (\*i == \*scale && \*j == pos - \*scale);*

**Якщо** *UR || DR || DL || UL* **то:**

*\*dir = (\*dir + 1) % 4;*

**Якщо** *UL* **то:**

*(\*scale)++;*

**Все якщо**

**Все якщо**

**Якщо** *\*dir == 0* **то** *(\*i)++;*

**Інакше якщо** *\*dir == 1* **то** *(\*j)++;*

**Інакше якщо** *\*dir == 2* **то** *(\*i)--;*

**Інакше** *(\*j)--;*

**Кінець *incr()***

**Підпрограма** ***generate():***

**Початок *generate()***

*i = 0, j = 0, dir = 0, pos = size - 1, scale = 0;*

**Для** *val* **від** 1 **до** *size \* size* **повторити**:

*\*(\*(matrix + i) + j) = val;*

*incr(&i, &j, pos, val, &dir, &scale);*

**Все повторити**

**Кінець *generate()***

**Підпрограма** ***output():***

**Початок *output()***

**Для** *i* **від** 0 **до** *size* **повторити:**

**Для** *j* **від** 0 **до** *size* **повторити:**

**Виведення** *\*(\*(matrix + j) + i);*

**Все повторити**

**Все повторити**

**Підпрограма *find\_min():***

**Початок *find\_min()***

*UL = \*\*matrix;*

*UR = \*\*(matrix + size - 1);*

*DR = \*(\*(matrix + size - 1) + size - 1);*

*DL = \*(\*matrix + size - 1);*

**Якщо** *UL < UR && UL < DR && UL < DL* **то повернути** *UL;*

**Інакше якщо** *UR < UL && UR < DR && UR < DL* **то повернути**

*UR;*

**Інакше якщо** *DR < UL && DR < UR && DR < DL* **то повернути** *DR;*

**Інакше повернути** *DL;*

**Підпрограма *del():***

**Початок *del()***

**Для** *i* **від** 0 **до** *size* **повторити:**

*delete[] \*(matrix + i);*

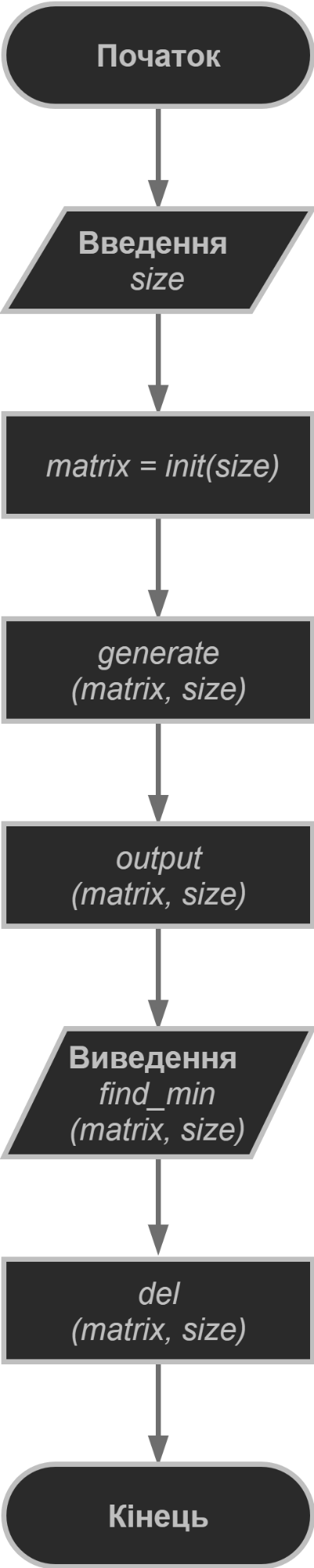
**Все повторити**

*delete[] matrix;*

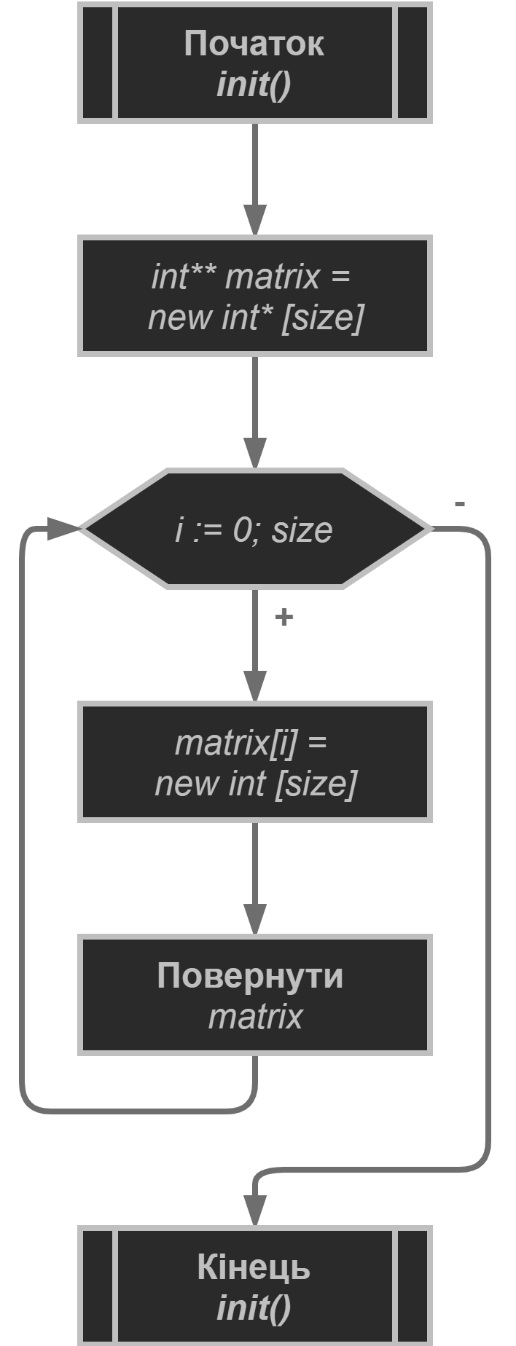
**Кінець *del()***

**Блок-схема**

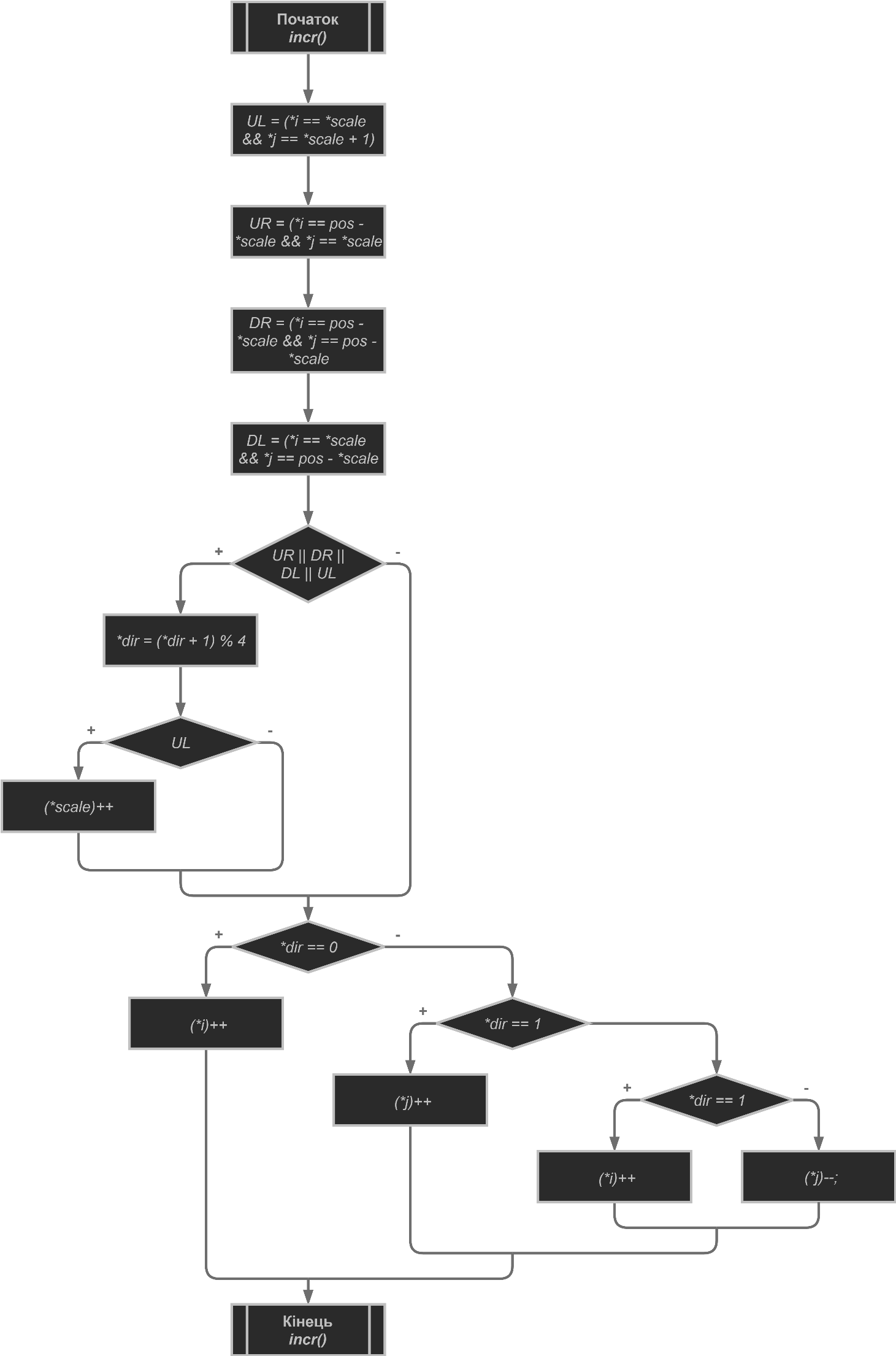
**Основна програма:**

****

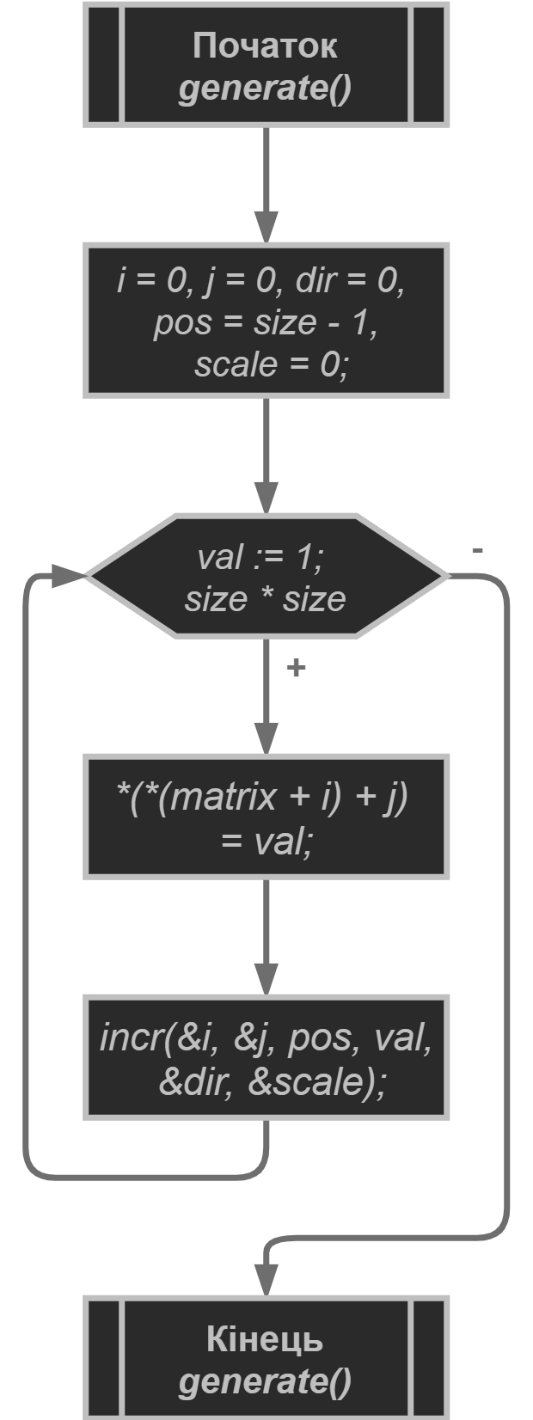
**Підпрограма *init():***

******

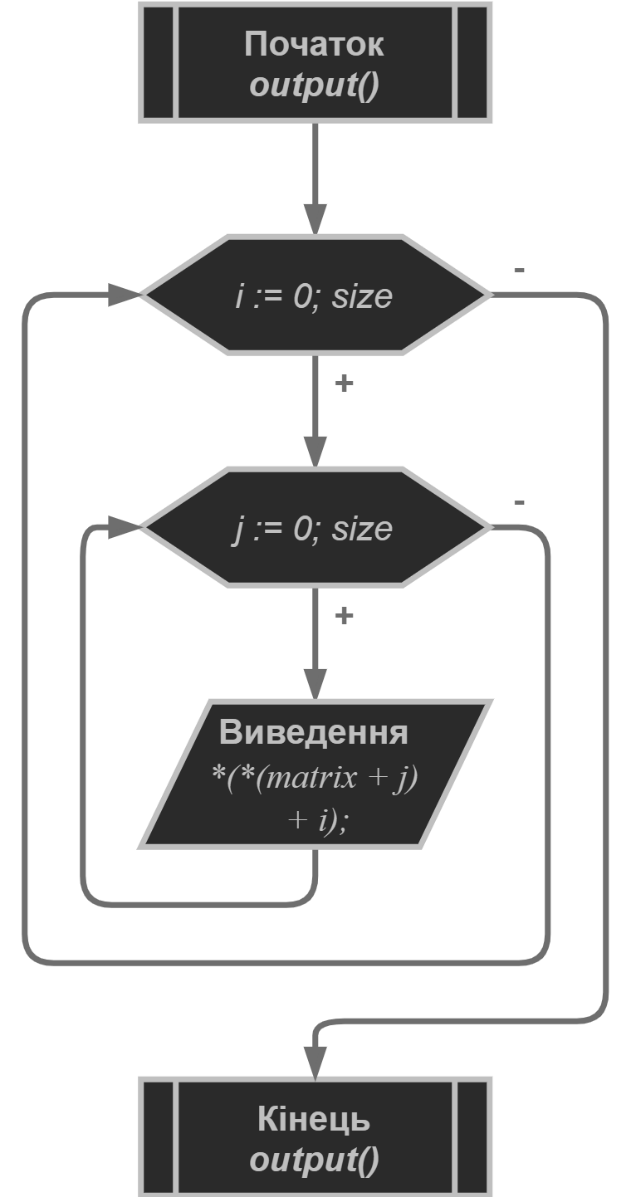
**Підпрограма *incr():***

******

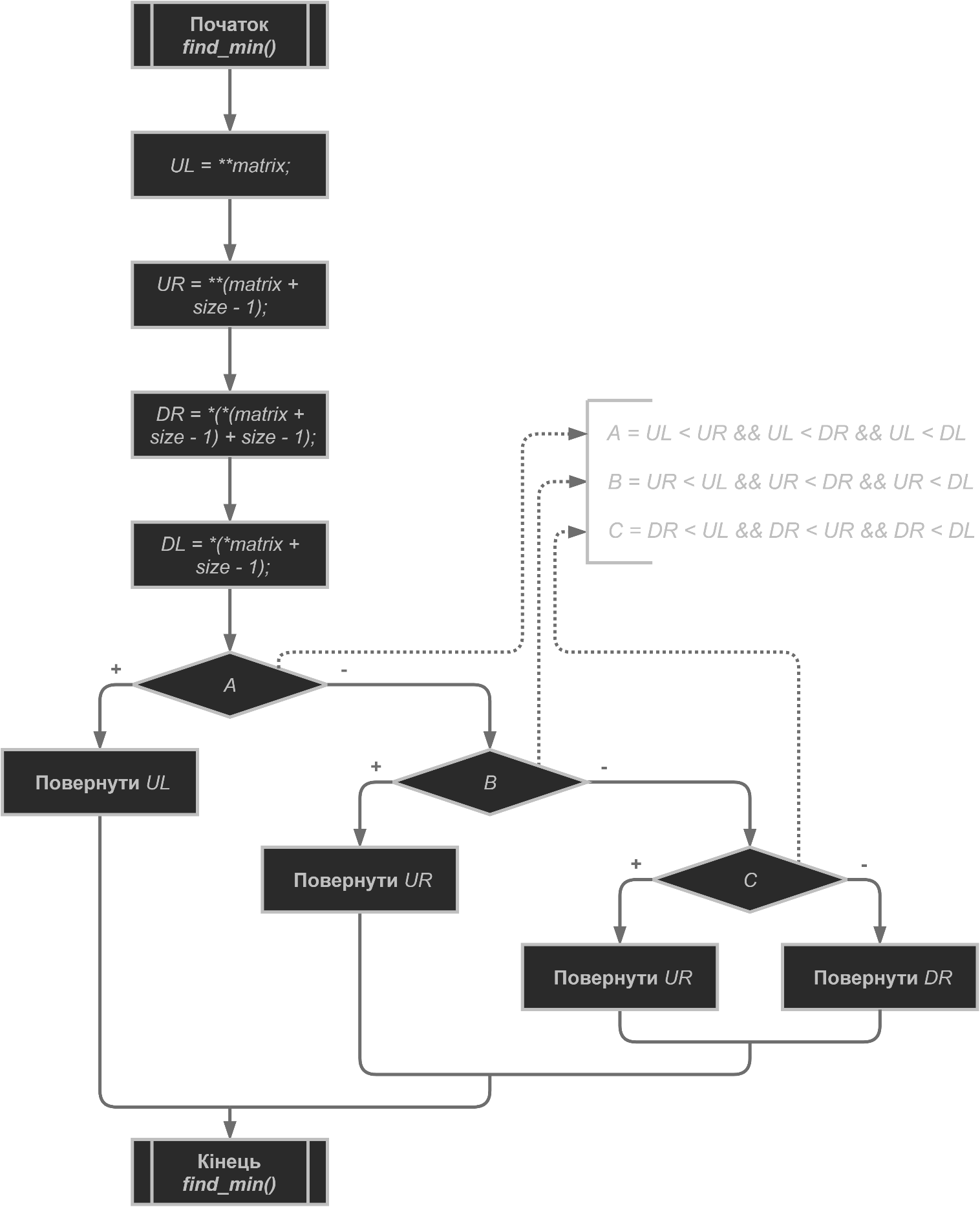
**Підпрограма *generate():***

******

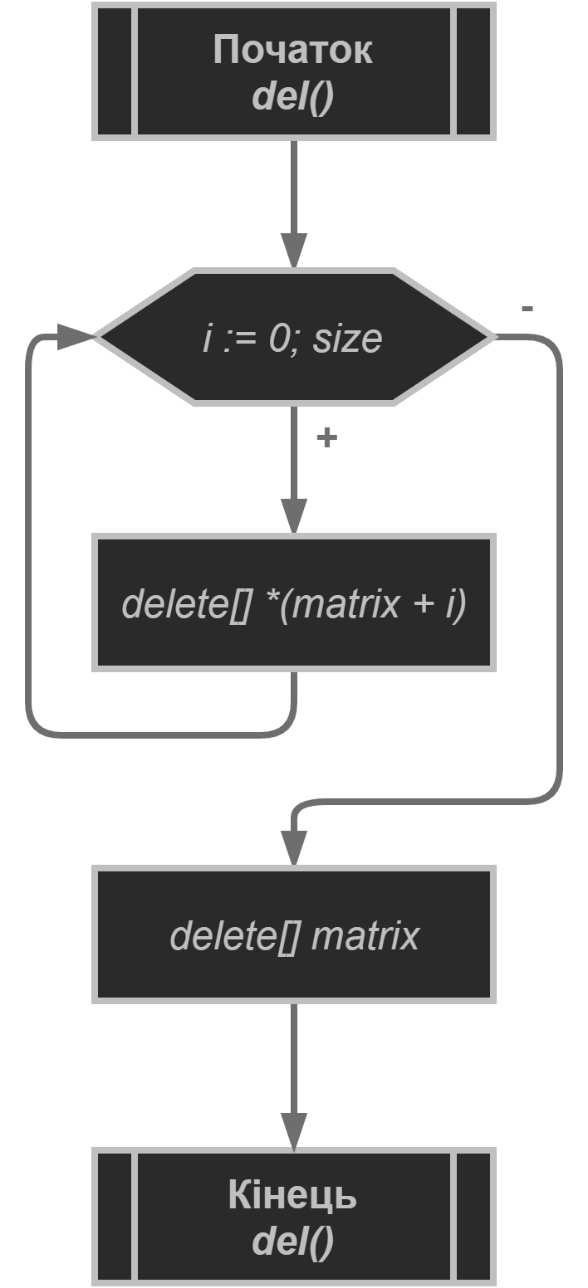
**Підпрограма *output():***

******

**Підпрограма *find\_min():***

******

**Підпрограма *del():***

******

**Код на С++**

#include <iostream>

#include <iomanip>

#include <stdlib.h>

#include <math.h>

using namespace std;

int\*\* init(int);

void generate(int\*\*, int);

void incr(int\*, int\*, int, int, int\*, int\*);

void output(int\*\*, int);

int find\_min(int\*\*, int);

void del(int\*\*, int);

int main() {

int size;

int\*\* matrix;

cout << "Enter the size of the spiral matrix: "; cin >> size;

matrix = init(size);

generate(matrix, size);

cout << "\nThe spiral matrix with size " << size << ":" << endl;

output(matrix, size);

cout << "The minimal angular elements of the matrix is " << find\_min(matrix, size) << "\n\n";

del(matrix, size);

system("pause");

return 0;

}

int\*\* init(int size) {

int\*\* matrix = new int\* [size];

for (int i = 0; i < size; i++) {

\*(matrix + i) = new int[size];

}

return matrix;

}

void incr(int\* i, int\* j, int pos, int val, int\* dir, int\* scale) {

bool UL = (\*i == \*scale && \*j == \*scale + 1);

bool UR = (\*i == pos - \*scale && \*j == \*scale);

bool DR = (\*i == pos - \*scale && \*j == pos - \*scale);

bool DL = (\*i == \*scale && \*j == pos - \*scale);

if (UR || DR || DL || UL) {

\*dir = (\*dir + 1) % 4;

if (UL) (\*scale)++;

}

if (\*dir == 0) (\*i)++;

else if (\*dir == 1) (\*j)++;

else if (\*dir == 2) (\*i)--;

else (\*j)--;

}

void generate(int \*\*matrix, int size) {

int i = 0, j = 0, dir = 0, pos = size - 1, scale = 0;

for (int val = 1; val <= size \* size; val++) {

\*(\*(matrix + i) + j) = val;

incr(&i, &j, pos, val, &dir, &scale);

}

}

void output(int \*\*matrix, int size) {

int width0 = floor(log(size \* size) / log(10) + 1);

int width1 = 1 + width0;

cout << endl;

for (int i = 0; i < size; i++) {

cout << "|";

for (int j = 0; j < size; j++) {

cout << setw(width1) << \*(\*(matrix + j) + i);

}

cout << setw(width0) << "|" << endl;

if (i != size - 1) {

int width\_full = width1 \* (size + 1) - 1;

cout << "|" << setw(width\_full) << "|" << endl;

}

}

cout << endl;

}

int find\_min(int\*\* matrix, int size) {

int UL = \*\*matrix;

int UR = \*\*(matrix + size - 1);

int DR = \*(\*(matrix + size - 1) + size - 1);

int DL = \*(\*matrix + size - 1);

if (UL < UR && UL < DR && UL < DL) return UL;

else if (UR < UL && UR < DR && UR < DL) return UR;

else if (DR < UL && DR < UR && DR < DL) return DR;

else return DL;

}

void del(int \*\*matrix, int size) {

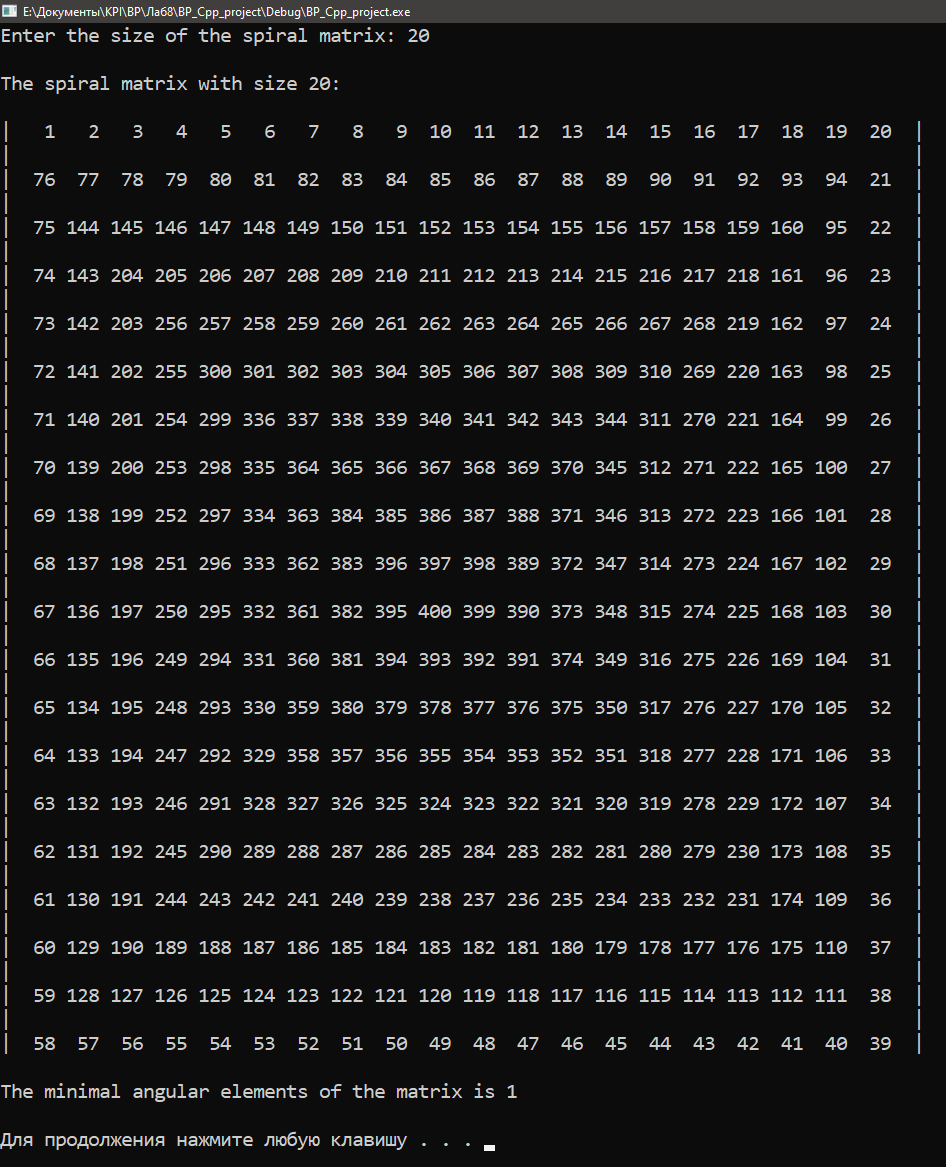
for (int i = 0; i < size; i++) {

delete[] \*(matrix + i);

}

delete[] matrix;

}

****

**Висновок**

Ми використали масив покажчиків, що у свою чергу є адресами перших елементів окремих масивів для опису матриці. Для того, щоб ініціалізувати двомірний масив(матрицю) спіральним чином ми використали підпрограму incr(), що за допомогою даних про розположення елемента інкрементувала/декрементували лічильник стовпця або рядка. Також підпрограма *find\_min()* була використана для порівняння можливих значень кутових елементів для знайдення мінімального. Отже, ми опанували технологію використання двовимірних масивів даних (матриць) та навчились розробляти алгоритми та програми із застосуванням матриць.