# 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 19

Виконав студент: ІП-11 Лисенко Андрій Юрійович

Перевірила: Вітковська Ірина Іванівна

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.

**Індивідуальне завдання**

**Варіант 19**

Розробити алгоритм та написати програму, яка складається з наступних

дій:

1. Опису змінної індексованого типу (двовимірний масив) згідно з варіантом.

2. Ініціювання змінної, що описана в п.1 даного завдання.

3. Обчислення змінної, що описана в п.1, згідно з варіантом.

|  |  |
| --- | --- |
| № | Опис варіанту |
| 19 | Задано матрицю дійсних чисел A[m,n]. У кожному рядку матриці знайти перший нульовий елемент Х і його місцезнаходження. Обміняти знайдене значення Х з елементом останнього стовбця. |

**Постановка задачі**

Використаємо обхід матриці по рядкам. Для знаходження нульвого елементу використаємо функцію *swap\_null\_in\_mtx()* у якій лінійним пошуком у заданому рядку матриці знайдемо шуканий елемент. Обмінямо його з останнім елементом цього рядку функцією *swap().*

**Побудова математичної моделі**

Складемо таблицю змінних

|  |  |  |  |
| --- | --- | --- | --- |
| Змінна | Тип | Ім’я | Призначення |
| Матриця | *double\*\** | *matrix* | Зберігати значення елементів матриці |
| Розмірність x, y | *int* | *size\_x, size\_y* | Зберігати розмір матриці |
| Діапазон генерації початок, кінець | *int* | *rng\_start, rng\_end* | Зберігати діапазон генерації випадкових чисел |
| Точність | *int* | *precision* | Зберігати кількість цифр після коми |
| Рядок, стовпчик | *int* | *row, col* | Зберігати поточний рядок та стовпчик матриці |

Складемо таблицю підпрограм:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Підпрограма | Тип | Ім’я | Параметри | Призначення |
| Опис матриці | *double\*\** | *declr\_mtx* | *size\_x, size\_y* | Виділення пам'яті для зберіганя елементів матриці |
| Генерація випадкових дійсних | *double* | *rand\_double* | *rng\_start, rng\_end, precision* | Генерація випадкових дійсних чисел |
| Генерація матриці | *void* | *gen\_mtx* | *matrix, size\_x, size\_y, rng\_start, rng\_end, precision* | Генерація матриці випадкових дійсних чисел |
| Обмін нулів матриці | *void* | *swap\_null\_in\_mtx* | *matrix, size\_x, size\_y* | Обмін першого нульового елементу і його місцезнаходження та обмін його з останнім елементом стовпця. |
| Знайти нульове значення | *int* | *find\_null* | *matrix, size\_x, size\_y* | Знайти перший нульовий елемент рядку |
| Обмін | *void* | *swap* | *a, b* | Обмін двух значень двух змінних |
| Виведення матриці | *void* | *out\_mtx* | *matrix, size\_x, size\_y, rng\_start, rng\_end, precision* | Виведення матриці |

**Розв’язання**

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

***Крок 1:***Конкретизаціядії опису матриці

***Крок 2:*** Конкретизація дії генерації матриці

***Крок 3:*** Конкретизація дії обміну знайденого нульового значення з елементом першого стовбця

**Псевдокод**

**Основна програма:**

**Початок**

*double\*\* matrix = declr\_mtx(size\_x, size\_y);*

*gen\_mtx(matrix, size\_x, size\_y, rng\_start, rng\_end, precision);*

**Виведення** *matrix;*

*swap\_null\_in\_mtx(matrix, size\_x, size\_y);*

**Виведення** *matrix;*

**Кінець**

**Підпрограма** ***declr\_mtx:***

**Початок** ***declr\_mtx***

*matrix = new double\* [size\_x];*

**Повторити для** *i* **від** *0* **до** *size\_x:*

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

**Все повторити**

**Повернути** *matrix;*

**Кінець *declr\_mtx***

**Підпрограма *rand\_double:***

**Початок *rand\_double***

*y = -rng\_start;*

*x = rng\_end + y;*

*rand\_int = rand() % x - y;*

*rand\_frac = rand() % int(pow(10, precision));*

*random = rand\_int + rand\_frac \* pow(0.1, precision);*

**Повернути** *random;*

**Кінець *rand\_double***

**Підпрограма *gen\_mtx:***

**Початок *gen\_mtx***

**Повторити для** *i* **від** *0* **до** *size\_x***:**

**Повторити для** *j* **від** *0* **до** *size\_y***:**

*\*(\*(matrix + j) + i) = rand\_double(rng\_start, rng\_end, precision);*

**Все повторити**

**Все повторити**

**Кінець *gen\_mtx***

**Підпрограма *swap\_null\_in\_mtx:***

**Початок *swap\_null\_in\_mtx***

**Повторити для** *row* **від** *0* **до** *size\_x***:**

*col = find\_null(matrix, size\_x, row);*

**Якщо** *col != -1* **то:**

*swap(\*(\*(matrix + col) + row), \*(\*(matrix + size\_y - 1) + row));*

**Виведення** *row, col;*

**Інакше:**

**Виведення** *row, col;*

**Все якщо**

**Все повторити**

**Кінець *swap\_null\_in\_mtx***

**Підпрограма *find\_null:***

**Початок *find\_null***

**Повторити для** *col* **від** *0* **до** *size\_x***:**

**Якщо** *\*(\*(matrix + col) + row) == 0* **то:**

**Повернути** *col***;**

**Все якщо**

**Все повторити**

**Повернути** *-1;*

**Кінець *find\_null***

**Підпрограма *swap:***

**Початок *swap***

*c = a;*

*a = b;*

*b = c;*

**Кінець *swap***

**Підпрограма *out\_mtx:***

**Початок *out\_mtx***

**Повторитидля***j***від***0***до***size\_y****:***

**Повторити для***i***від***0***до** *size\_x****:***

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

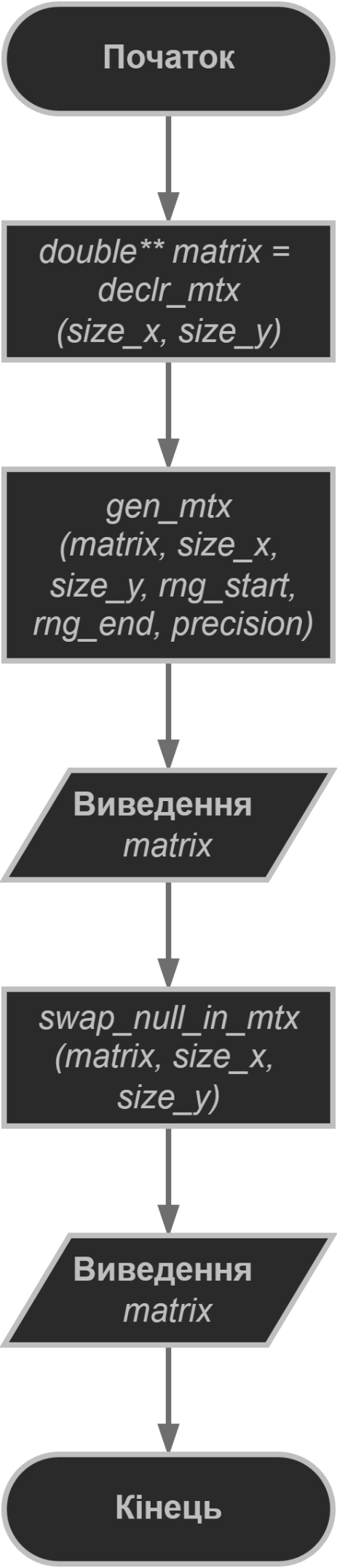
**Все повторити**

**Все повторити**

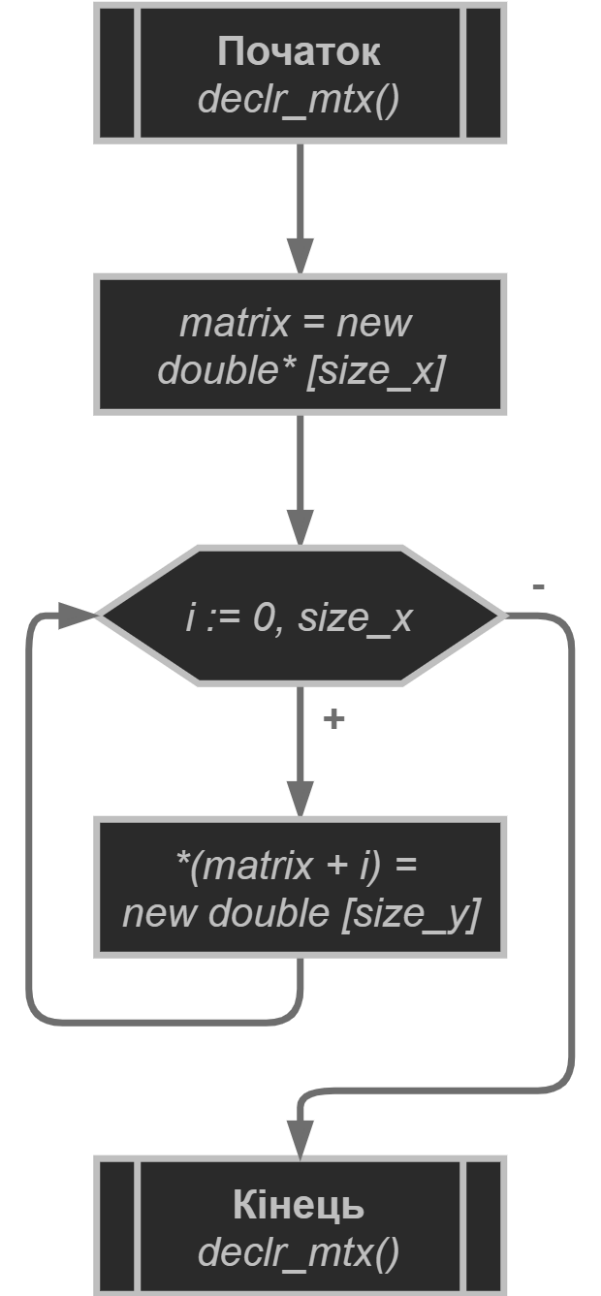
**Кінець *out\_mtx***

**Блок-схема**

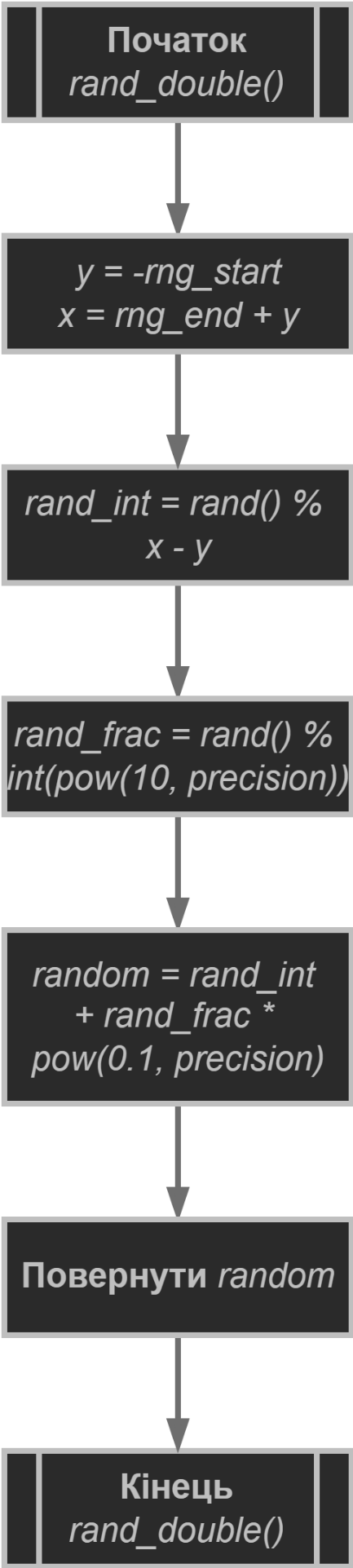
**Основна програма**

****

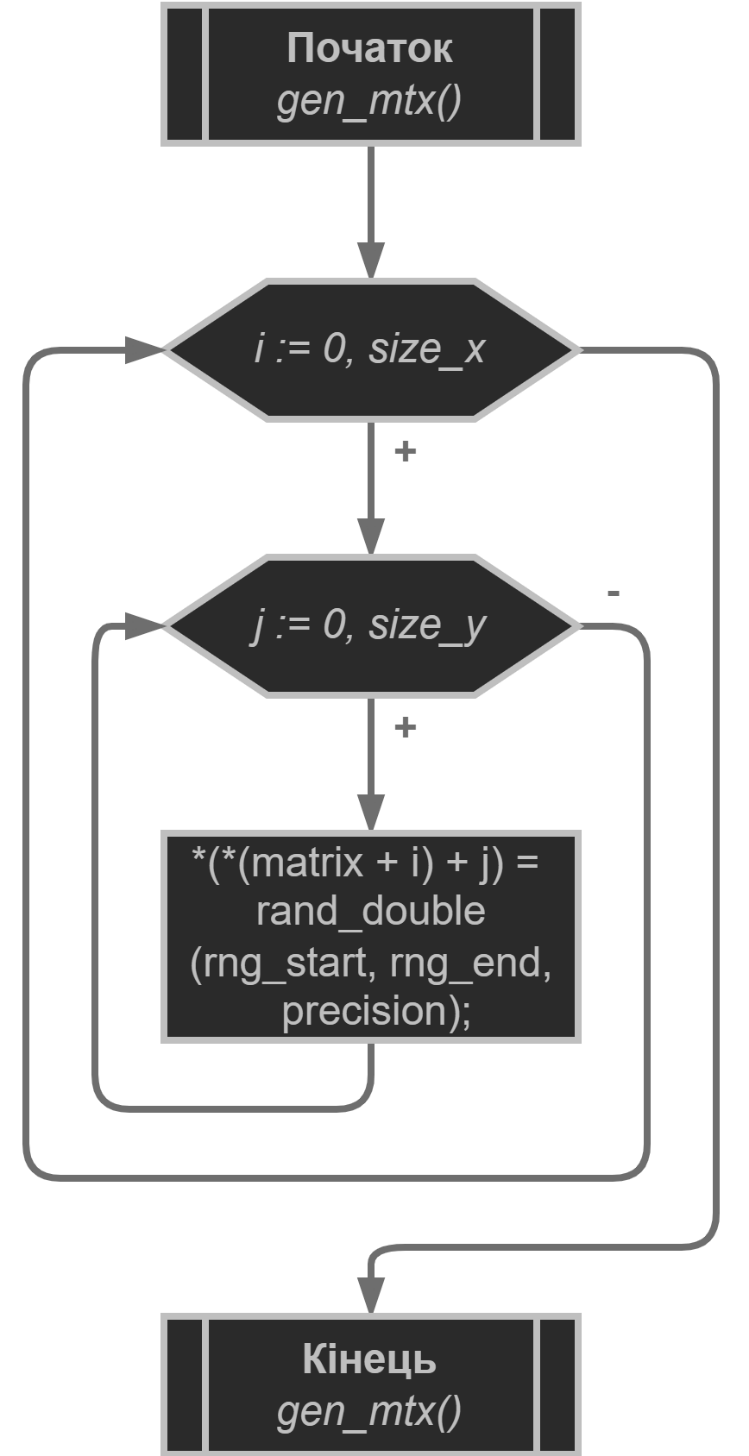
**Підпрограма declr\_mtx**

****

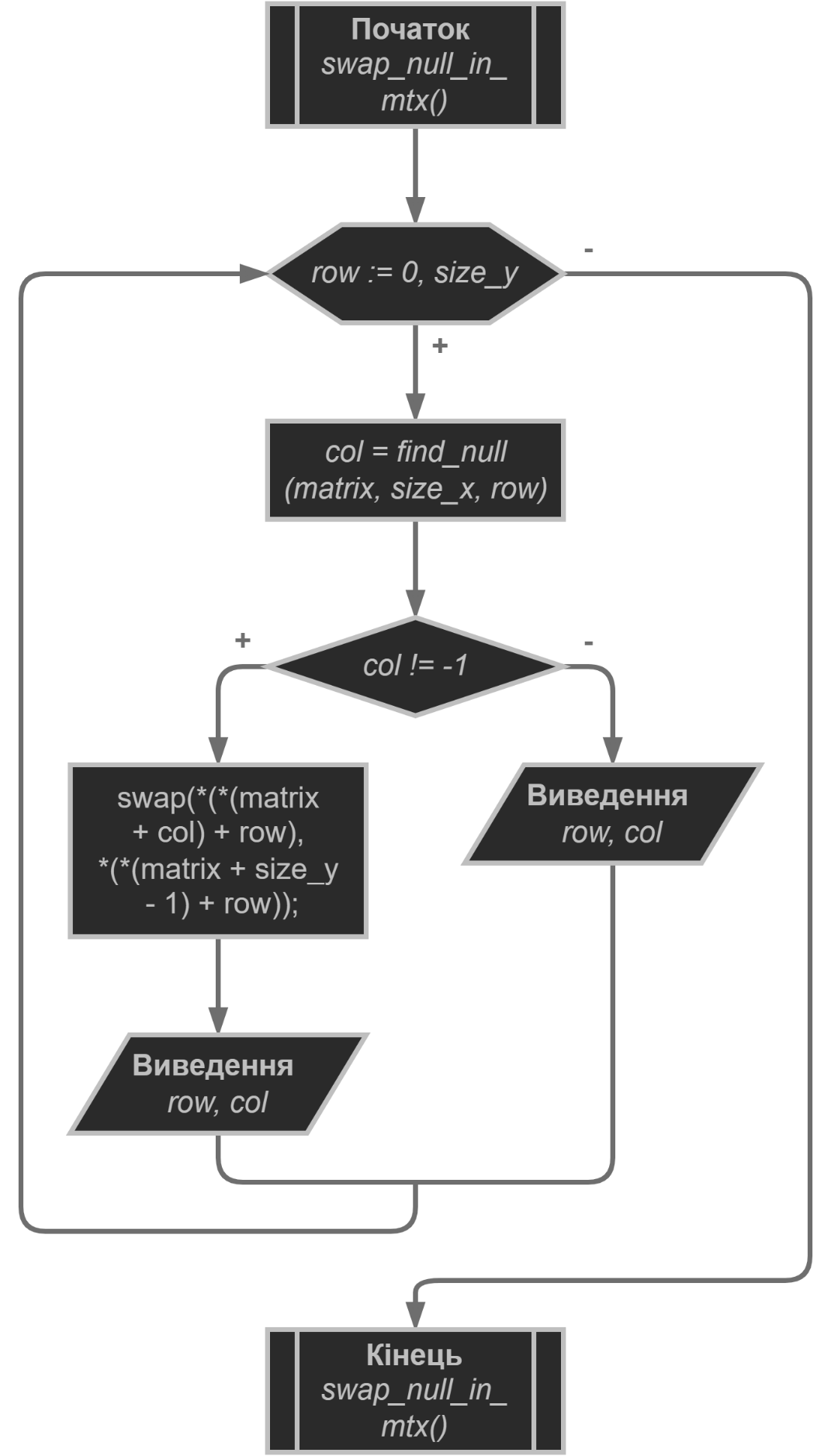
**Підпрограма rand\_double**

****

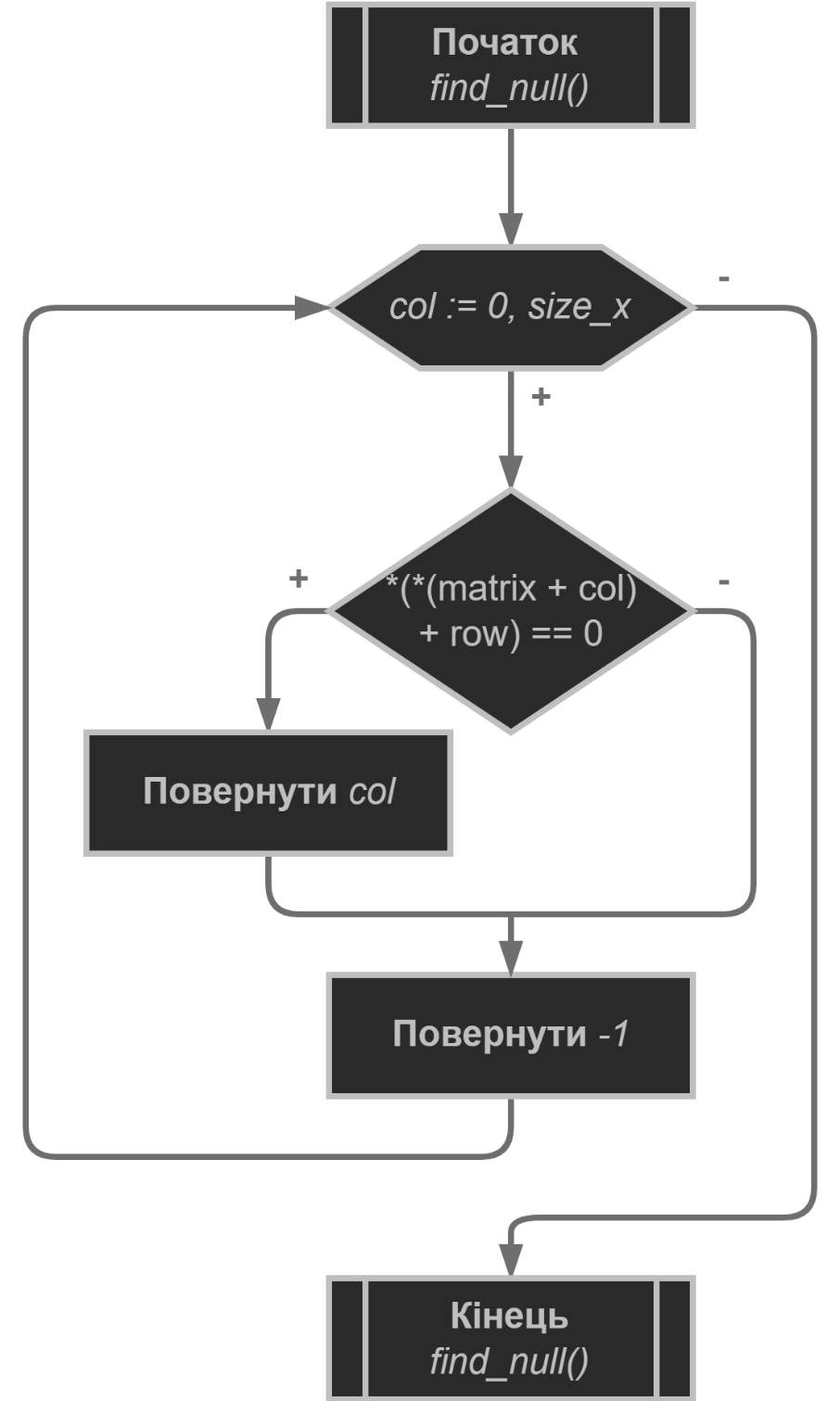
**Підпрограма gen\_mtx**

****

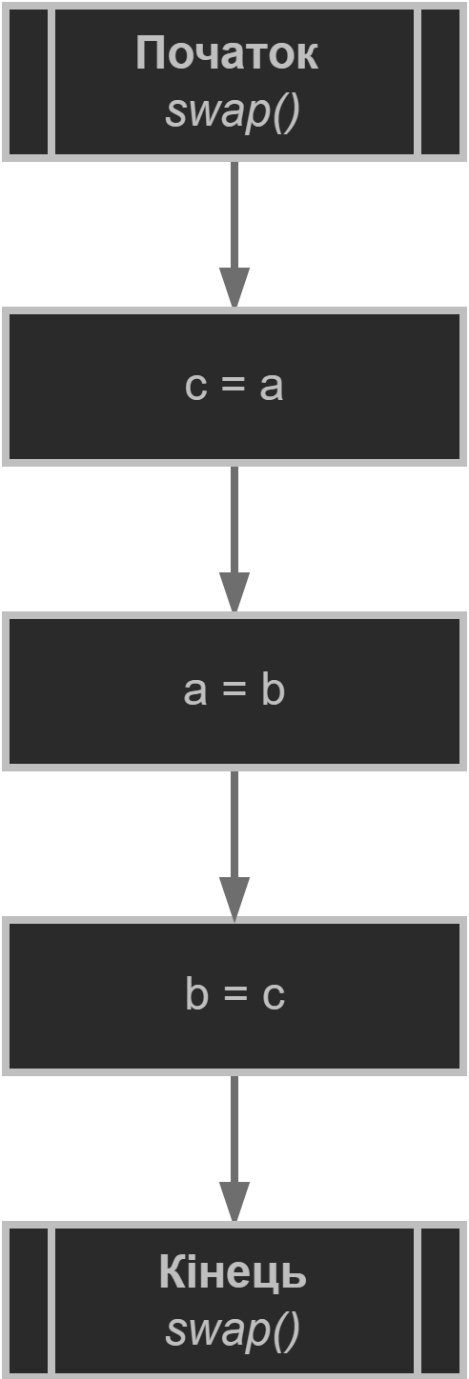
**Підпрограма swap\_null\_in\_mtx**

****

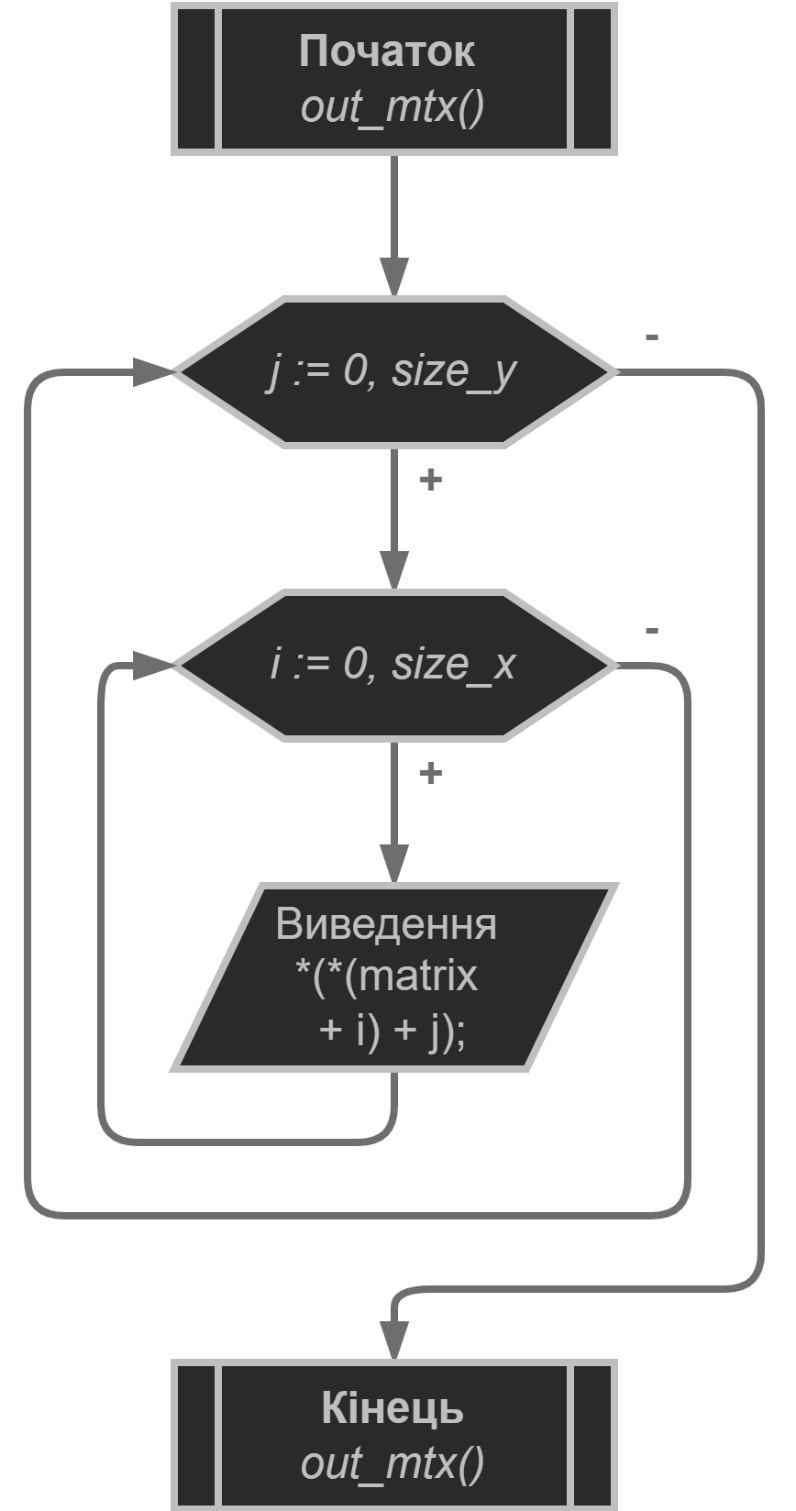
**Підпрограма find\_null**

****

**Підпрограма swap**

****

**Підпрограма out\_mtx**

****

**Код на С++**

#include <iostream>

#include <cmath>

#include <iomanip>

using namespace std;

double\*\* declr\_mtx(int, int);

double rand\_double(int, int, int);

void gen\_mtx(double\*\*, int, int, int, int, int);

void swap\_null\_in\_mtx(double\*\*, int, int);

int find\_null(double\*\*, int, int);

void swap(double&, double&);

void out\_mtx(double\*\*, int, int, int, int, int);

int main() {

int size\_x, size\_y, rng\_start, rng\_end, precision;

cout << "Enter matrix size (m by n): "; cin >> size\_y >> size\_x;

cout << "Enter range (from - to) [|range|<=16383]: "; cin >> rng\_start >> rng\_end;

cout << "Enter precision: [<=4]: "; cin >> precision;

srand((unsigned int)time(NULL));

double\*\* matrix = declr\_mtx(size\_x, size\_y);

gen\_mtx(matrix, size\_x, size\_y, rng\_start, rng\_end, precision);

cout << "\nGenerated matrix:\n";

out\_mtx(matrix, size\_x, size\_y, rng\_start, rng\_end, precision);

swap\_null\_in\_mtx(matrix, size\_x, size\_y);

cout << "\nMatrix after swapping:\n";

out\_mtx(matrix, size\_x, size\_y, rng\_start, rng\_end, precision);

system("pause");

return 0;

}

double\*\* declr\_mtx(int size\_x, int size\_y) {

double\*\* matrix = new double\* [size\_x];

for (int i = 0; i < size\_x; i++)

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

return matrix;

}

void gen\_mtx(double\*\* matrix, int size\_x, int size\_y, int rng\_start, int rng\_end, int precision) {

int y = -rng\_start;

int x = rng\_end + y + 1;

for (int j = 0; j < size\_y; j++)

for (int i = 0; i < size\_x; i++)

\*(\*(matrix + i) + j) = rand\_double(rng\_start, rng\_end, precision);

}

double rand\_double(int rng\_start, int rng\_end, int precision) {

double random;

int rand\_int, rand\_frac;

int y = -rng\_start;

int x = rng\_end + y;

rand\_int = rand() % x - y;

rand\_frac = rand() % int(pow(10, precision));

random = rand\_int + rand\_frac \* pow(0.1, precision);

return random;

}

void swap\_null\_in\_mtx(double\*\* matrix, int size\_x, int size\_y) {

int col;

for (int row = 0; row < size\_x; row++) {

col = find\_null(matrix, size\_x, row);

if (col != -1) {

swap(\*(\*(matrix + col) + row), \*(\*(matrix + size\_y - 1) + row));

cout << "Row " << row + 1 << " contains a zero at column " << col + 1 << "\n";

} else

cout << "Row " << row + 1 << " does not contain a zero\n";

}

}

int find\_null(double\*\* matrix, int size\_x, int row) {

for (int i = 0; i < size\_x; i++)

if (\*(\*(matrix + i) + row) == 0)

return i;

return -1;

}

void swap(double &a, double &b) {

double c = a;

a = b;

b = c;

}

void out\_mtx(double\*\* matrix, int size\_x, int size\_y, int rng\_start, int rng\_end, int precision) {

int range;

if (fabs(rng\_start) < fabs(rng\_end))

range = fabs(rng\_end);

else

range = fabs(rng\_start);

int width = (int)log10(range) + precision + 4;

int width\_full = width \* size\_x + 3;

cout << "\n";

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

cout << "|";

for (int i = 0; i < size\_x; i++)

cout << setw(width) << setprecision(precision) << fixed

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

cout << " |\n";

if (j != size\_y - 1)

cout << "|"

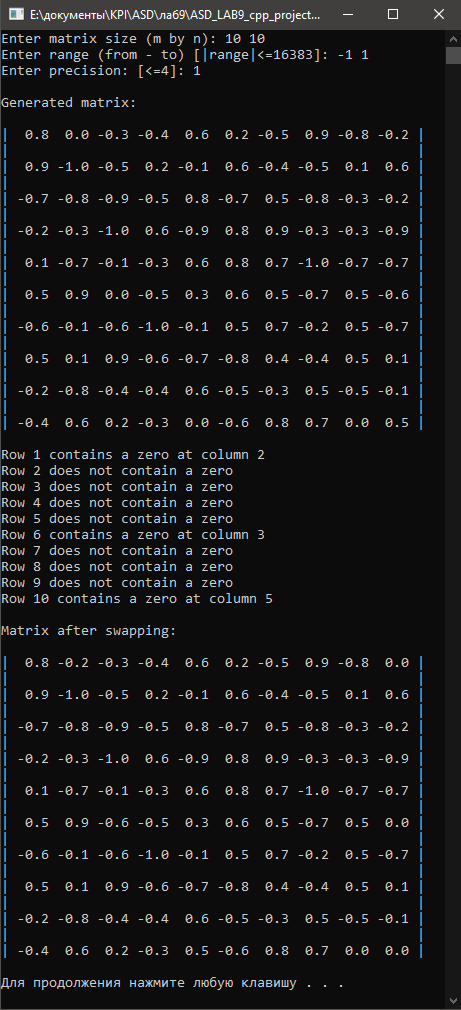
<< setw(width\_full)

<< "|\n";

}

cout << "\n";

}

****

**Випробування**

|  |  |
| --- | --- |
| Блок | Дія |
| 0 | Початок |
| 1 | Сгенерована матриця:  | -0.3 -0.4 -1.0 |  | |  | -0.8 -0.0 -0.7 |  | |  | 0.2 0.5 0.4 | |
| 2 | *Row 1 doesn`t contain a zero*  *Row 2 contans a zero at column 2* (Нуль наявний в другому рядку на другій позиції)  *Row 3 doesn`t contain a zero* |
| 3 | Матриця після обміну:  | -0.3 -0.4 -1.0 |  | |  | -0.8 -0.7 -0.0 |  | |  | 0.2 0.5 0.4 |  Обмін проведено правильно |
| 4 | Кінець |

**Висновок**

Ми використали обхід по рядкам та лінійний пошук для знаходження першого нульового елементу та обміняли його з останнім елементом поточного рядка. Отже ми дослідили алгоритми обходу масивів, набули практичних навичок використання цих алгоритмів під час складання програмних специфікацій.