Міністерство освіти і науки України

Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації

і управління

Звіт

з лабораторної роботи № 8 з дисципліни

«Основи програмування-1

Базові конструкції»

«Багатовимірні масиви»

Варіант     5

Виконав студент       ІП-02 Геращенко Дмитро Ігорович

(шифр, прізвище, ім'я, по батькові)

Перевірив              Вітковська Ірина Іванівна

( прізвище, ім'я, по батькові)

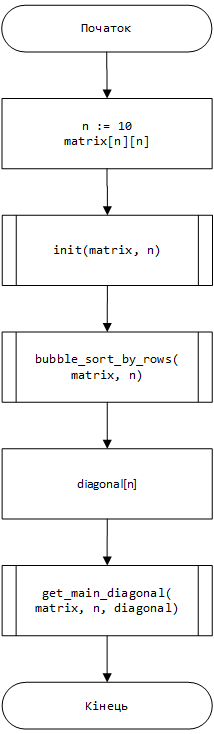
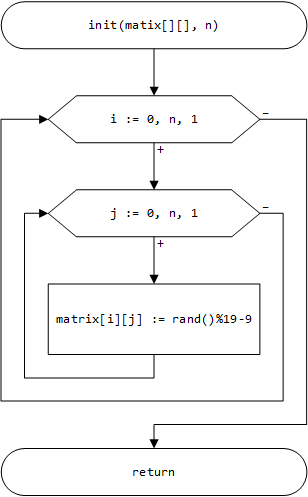
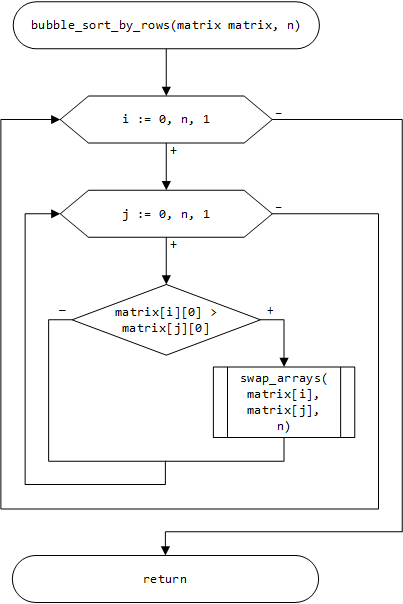
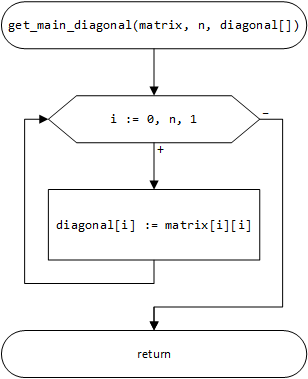
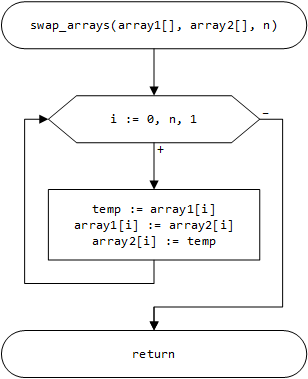
Київ 2020

# Лабораторна робота 8 Багатовимірні масиви Варіант 5

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

1. Постановка задачі

На основі заданої матриці *A*(*n* x *n*) побудувати вектор *X*(*n*) наступним чином: рядки матриці впродякувати по зменшенню елементів її першого стовптця і як *X* прийняти головну діагональ перетвореної матриці.

1. Блок схема
2. Текст програми

#include <iostream> // cout

#include <iomanip> // setw

#include <ctime> // time

#include <cmath> // rand

**using** **namespace** std;

**const** **int** n = **7**;

**typedef** **int** square\_matrix[n][n];

/\*\*

\* Заповнює матрицю випадковими числами від -9 до 9.

\* @param matrix[in, out] Матриця, що буде заповнена.

\* @param n[in] К-сть рядків і ствопців в матриці.

\*/

**void** **init**(square\_matrix, **int**);

/\*\*

\* Сортує матрицю за спаданням резльтату виклика ключа на рядках.

\* @param matrix[in, out] Матриця, що буде відсортована.

\* @param n[in] К-сть рядків і ствопців в матриці.

\* @param key[in] Функція ключ, за значенням якої сорується матриця.

\*/

**void** **bubble\_sort\_by\_rows**(square\_matrix, **int**, **int** (\*)(**int**[], **int**));

// void insertion\_sort\_by\_rows(square\_matrix, int, int (\*)(int[], int));

// void shell\_sort\_by\_rows(square\_matrix, int, int (\*)(int[], int));

// void selection\_sort\_by\_rows(square\_matrix, int, int (\*)(int[], int));

// void quick\_sort\_by\_rows(square\_matrix, int, int, int (\*)(int[], int));

**inline** **int** **get\_first\_element**(**int**[], **int**);

/\*\*

\* Повертає головну діагональ матриці.

\* @param matrix[in] Матриця, з якої буде взята головна діагональ.

\* @param n[in] К-сть рядків і ствопців в матриці.

\* @param diagonal[out] Одновимірний масив з елементами головної діагоналі

\*/

**void** **get\_main\_diagonal**(square\_matrix, **int**, **int**[]);

// Виводить одновимірний масив

**void** **print\_array**(**int**[], **int**);

// Виводить двовимірний масив

**void** **print\_matrix**(square\_matrix, **int**);

**int** **main**() {

square\_matrix matrix;

init(matrix, n);

cout << "Початкова матриця:" << endl; print\_matrix(matrix, n);

bubble\_sort\_by\_rows(matrix, n, get\_first\_element);

// Варіант з лябмда функцією

// bubble\_sort\_by\_rows(matrix, n,

// [](int row[], int n) {

// return row[0];

// });

cout << "Відсотрована матриця:" << endl; print\_matrix(matrix, n);

**int** diagonal[n];

get\_main\_diagonal(matrix, n, diagonal);

cout << "Головна діагональ:" << endl; print\_array(diagonal, n);

**return** **0**;

}

// Ініціалізує матрицю випадковими числами від -9 до 9

**void** **init**(square\_matrix matrix, **int** n) {

srand(time(NULL));

**for** (**int** i = **0**; i < n; i++)

**for** (**int** j = **0**; j < n; j++)

matrix[i][j] = rand()%**19** - **9**;

}

// Повертає перший елемент масиву

**inline** **int** **get\_first\_element**(**int** array[], **int** n) {

**return** array[**0**];

}

// Сортує рядки матриці за спаданням функції-ключа

**void** **bubble\_sort\_by\_rows**(square\_matrix matrix, **int** n, **int** (\*key)(**int**[], **int**)) {

**void** swap\_arrays(**int**[], **int**[], **int**);

**for** (**int** i = **0**; i < n; i++)

**for** (**int** j = **0**; j < n; j++)

**if** ((\*key)(matrix[i], n) > (\*key)(matrix[j], n))

swap\_arrays(matrix[i], matrix[j], n);

}

// Міняє місцями елементи двох одномірних масивів

**void** **swap\_arrays**(**int** array1[], **int** array2[], **int** n) {

**int** temp;

**for** (**int** i = **0**; i < n; i++) {

temp = array1[i];

array1[i] = array2[i];

array2[i] = temp;

}

}

// Повертає масив з елементів головної діагоналі квадратної матриці

**void** **get\_main\_diagonal**(square\_matrix matrix, **int** n, **int** diagonal[]) {

**for** (**int** i = **0**; i < n; i++)

diagonal[i] = matrix[i][i];

}

// Виводить одновимірний масив

**void** **print\_array**(**int** array[], **int** n) {

**for** (**int** j = **0**; j < n; j++)

cout << setw(**3**) << array[j];

cout << endl;

}

// Виводить двовимірний масив

**void** **print\_matrix**(square\_matrix matrix, **int** n) {

**for** (**int** i = **0**; i < n; i++)

print\_array(matrix[i], n);

cout << endl;

}

// Сортує рядки матриці за спаданням функції-ключа

**void** **insertion\_sort\_by\_rows**(square\_matrix matrix, **int** n, **int** (\*key)(**int**[], **int**)) {

**void** insert\_array(**int**[], **int**[], **int**);

**int** i, j, temp[n];

**for** (i = **1**; i < n; i++) {

insert\_array(temp, matrix[i], n);

**for** (j = i-**1**; j >= **0** && ((\*key)(matrix[j], n) < (\*key)(temp, n)); j--)

insert\_array(matrix[j+**1**], matrix[j], n);

insert\_array(matrix[j+**1**], temp, n);

}

}

// Сортує рядки матриці за спаданням функції-ключа

**void** **shell\_sort\_by\_rows**(square\_matrix matrix, **int** n, **int** (\*key)(**int**[], **int**)) {

**void** insert\_array(**int**[], **int**[], **int**);

**int** gap, i, j, temp[n];

**for** (gap = n/**2**; gap > **0**; gap /= **2**)

**for** (i = gap; i < n; i++) {

insert\_array(temp, matrix[i], n);

**for** (j = i; j >= gap && ((\*key)(matrix[j-gap], n) < (\*key)(temp, n)); j -= gap)

insert\_array(matrix[j], matrix[j-gap], n);

insert\_array(matrix[j], temp, n);

}

}

// Вставляє елементи другого масиву в перший

**void** **insert\_array**(**int** array1[], **int** array2[], **int** n) {

**for** (**int** i = **0**; i < n; i++)

array1[i] = array2[i];

}

// Сортує рядки матриці за спаданням функції-ключа

**void** **selection\_sort\_by\_rows**(square\_matrix matrix, **int** n, **int** (\*key)(**int**[], **int**)) {

**void** swap\_arrays(**int**[], **int**[], **int**);

**int** max\_idx;

**for** (**int** i = **0**; i < n-**1**; i++) {

max\_idx = i;

**for** (**int** j = i+**1**; j < n; j++)

**if** ((\*key)(matrix[j], n) > (\*key)(matrix[max\_idx], n))

max\_idx = j;

swap\_arrays(matrix[i], matrix[max\_idx], n);

}

}

// Сортує рядки матриці за спаданням функції-ключа

**void** **quick\_sort\_by\_rows**(square\_matrix matrix, **int** low, **int** high, **int** (\*key)(**int**[], **int**)) {

**void** swap\_arrays(**int**[], **int**[], **int**);

**if** (low < high) {

**int** \*pivot = matrix[high]; // опорний елемент

**int** i = low; // в резальтаті стане індексом опорного елементу

**for** (**int** j = low; j <= high-**1**; j++)

**if** ((\*key)(matrix[j], n) > (\*key)(pivot, n))

swap\_arrays(matrix[i++], matrix[j], n);

swap\_arrays(matrix[i], pivot, n);

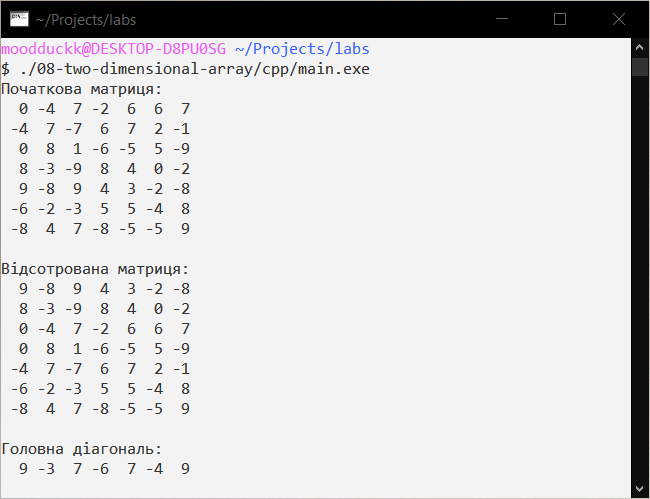
quick\_sort\_by\_rows(matrix, low, i - **1**, key);

quick\_sort\_by\_rows(matrix, i + **1**, high, key);

}

}

1. Копії екранних форм результатів роботи



1. Висновок

На цій лабораторній роботі використано та оброблено двовимірні вимірні масиви. Наприклад, використано алгоритм ініціалізації матриці випадковими числами, алгоритми сортування (бульбашкою, вставками, Шелла, вибіркою та швидкого сортування) та алгоритм знаходження елементів головної діагоналі матриці. Також, як допоміжна підпрограма для сортування, застосовано алгоритм обміну елементів двох одновимірних масивів.