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)

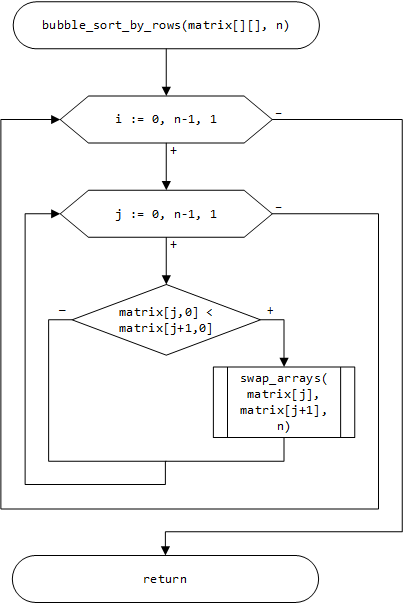
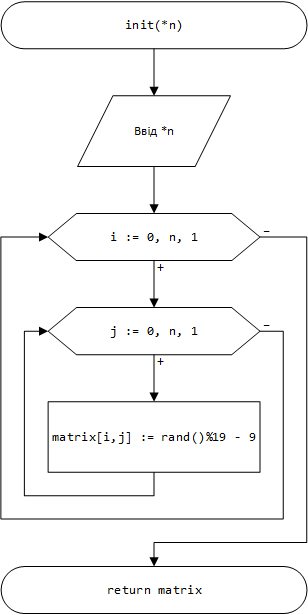
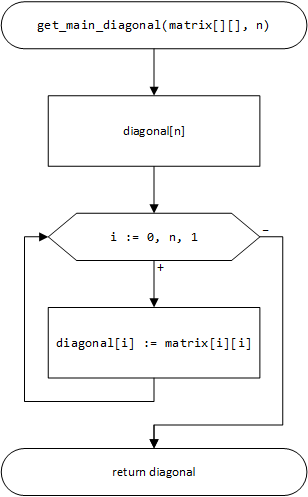
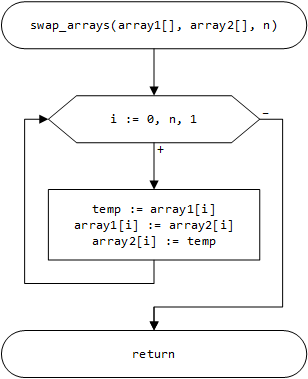
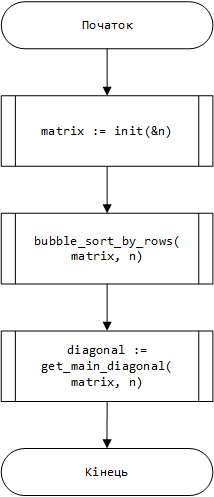
Київ 2020

# 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;

/\*\*

\* Повертає матрицю заповнену випадковими числами від -9 до 9.

\* @param n[out] Вказіник розмірність матриці.

\* @return Вказівник на динамічний двовимірний масив (матрицю).

\*/

**int**\*\* **init\_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**(**int**\*\*, **int**, **int** (\*)(**int**[], **int**));

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

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

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

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

/\*\*

\* Повертає головну діагональ матриці.

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

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

\* @return Вказіник на динамічний масив з головною діагоналлю.

\*/

**int**\* **get\_main\_diagonal**(**int**\*\*, **int**);

**inline** **int** **get\_first\_element**(**int**[], **int**); // Повертає перший елемент масиву

**void** **print\_array**(**int**\*, **int**); // Виводить одновимірний масив

**void** **print\_matrix**(**int**\*\*, **int**); // Виводить двовимірний масив

**void** **delete\_matrix**(**int** \*\*, **int**); // Видаляє динамічний двовимірний масив

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

**int** n, \*\*matrix = init\_matrix(&n);

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

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

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

**int** \*diagonal = get\_main\_diagonal(matrix, n);

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

**delete**[] diagonal;

delete\_matrix(matrix, n);

**return** **0**;

}

// Повертає матрицю заповнену випадковими числами від -9 до 9.

**int**\*\* **init\_matrix**(**int** \*n) {

srand(time(NULL));

cout << "Введіть розмірність квадратної матриці n: ";

cin >> \*n;

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

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

matrix[i] = **new** **int** [\*n];

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

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

}

**return** matrix;

}

// Повертає перший елемент масиву

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

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

}

// Повертає головну діагональ матриці.

**int**\* **get\_main\_diagonal**(**int** \*\*matrix, **int** n) {

**int**\* diagonal = **new** **int**[n];

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

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

**return** diagonal;

}

// Виводить одновимірний масив

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

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

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

cout << endl;

}

// Виводить двовимірний масив

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

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

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

cout << endl;

}

// Видаляє динамічний двовимірний масив

**void** **delete\_matrix**(**int** \*\*matrix, **int** n) {

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

**delete**[] matrix[i];

**delete**[] matrix;

}

// Сортує рядки матриці за спаданням функції-ключа (сортування бульбашкою)

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

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

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

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

**if** ((\*key)(matrix[j], n) < (\*key)(matrix[j+**1**], n))

swap\_arrays(matrix[j], matrix[j+**1**], n);

}

// Сортує рядки матриці за спаданням функції-ключа (сортування вставками)

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

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

**int** j, \*temp = **new** **int**[n];

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

assign\_array(temp, matrix[i], n); // temp = matrix[i]

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

assign\_array(matrix[j+**1**], matrix[j], n); // matrix[j] = matrix[j-1]

assign\_array(matrix[j], temp, n); // matrix[j] = temp

}

**delete**[] temp;

}

// Сортує рядки матриці за спаданням функції-ключа (сортування Шела)

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

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

**int** j, \*temp = **new** **int**[n];

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

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

assign\_array(temp, matrix[i], n); // temp = matrix[i]

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

assign\_array(matrix[j], matrix[j-gap], n); // matrix[j] = matrix[j-gap]

assign\_array(matrix[j], temp, n); // matrix[j] = temp

}

**delete**[] temp;

}

// Сортує рядки матриці за спаданням функції-ключа (сортування вибором)

**void** **selection\_sort\_by\_rows**(**int**\*\* 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 (швидке сортування)

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

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

}

}

// Міняє місцями два масива

**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;

}

}

// Присвоює першому масиву другий (array1 = array2)

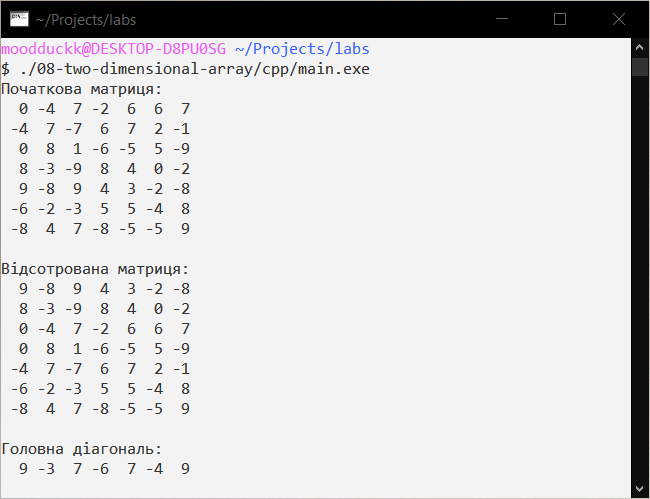
**void** **assign\_array**(**int** array1[], **int** array2[], **int** n) {

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

array1[i] = array2[i];

}

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 та присвоєння значень одно масиву іншому.