

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 28

Виконав студент ІП-11 Сідак Кирил Ігорович
(шифр, прізвище, ім'я, по батькові)

Перевірив _____
(прізвище, ім'я, по батькові)

Київ 2021

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

Варіант 28

У двох заданих дійсних квадратних матрицях розмірності n поміняти місцями їх головні діагоналі, попередньо упряdkувавши за збільшенням елементи цих діагоналей.

Постановка задачі:

Спочатку треба створити два динамічні двовимірні масиви розмірності n на n та заповнити їх випадковими дійсними числами з певного проміжку, потім в даних масивах за допомогою алгоритму сортування упорядкувати за збільшенням елементи головних діагоналей та замінити кожний елемент головної діагоналі одного масиву відповідним елементом головної діагоналі іншого масиву. Після цього потрібно видалити з пам'яті обидва динамічні масиви.

Програма на C++:

```
/* Варіант 28
У двох заданих дійсних квадратних матрицях розмірності n поміняти
місцями їх
    головні діагоналі, попередньо упряdkувавши за збільшенням
елементи цих діагоналей.*/
#include <iostream>
#include <ctime>
#include <iomanip>
using namespace std;
double random_double(int, int); // Функція для генерація
випадкового дісного числа з заданого проміжка
double** generate_matrix(int, int); // Функція для створення
матриці заданої розмірності
void display_matrix(double**, int); // Функція для вивдення
матриці
void sort_diagonal_asc(double**, int); // Функція для сортування
елементів головної діагоналі за збільшенням
void swap_diagonals(double**, double**, int); /* Функція для
заміни головної діагоналі однієї матриці відповідною
    діагоналлю іншої матриці*/
void delete_matrix(double**, int); // Функція для видалення
двомірного масиву з пам'яті

int main() {
    int n;
    double** matrix_1;
    double** matrix_2;
    srand(time(NULL));
    cout << "Enter the size of the square matrices: ";
    cin >> n;
    matrix_1 = generate_matrix(n, n);
    matrix_2 = generate_matrix(n, n);
    cout << "Matrix 1:\n";
```

```

display_matrix(matrix_1, n);
cout << "Matrix 2: \n";
display_matrix(matrix_2, n);
sort_diagonal_asc(matrix_1, n);
sort_diagonal_asc(matrix_2, n);
cout << "Matrix 1 with sorted main diagonal:\n";
display_matrix(matrix_1, n);
cout << "Matrix 2 with sorted main diagonal:\n";
display_matrix(matrix_2, n);
swap_diagonals(matrix_1, matrix_2, n);
cout << "Matrix 1 with main diagonal from matrix 2:\n";
display_matrix(matrix_1, n);
cout << "Matrix 2 with main diagonal from matrix 1:\n";
display_matrix(matrix_2, n);
delete_matrix(matrix_1, n);
delete_matrix(matrix_2, n);
return 0;
}

double random_double(int min, int max) {
    double num;
    int rnd_of_max = rand()%(max + 1);
    double fraction = (double)(rnd_of_max) / max;
    num = min + rand()%(max - min) + fraction;
    return num;
}

double** generate_matrix(int rows, int columns) {
    double** matrix = new double*[rows];
    for (int i=0; i < rows; i++) {
        matrix[i] = new double[columns];
    }
    for (int i=0; i < rows; i++) {
        for (int j = 0; j < columns; j++) {
            matrix[i][j] = random_double(1, 10);
        }
    }
    return matrix;
}

void display_matrix(double** matrix, int size) {
    for (int i = 0; i < size; i++) {
        for (int j = 0; j < size; j++) {
            cout << setw(8) << matrix[i][j];
        }
        cout << '\n';
    }
}

void sort_diagonal_asc(double** matrix, int size) {
    int i;
    double temp_num;
    bool sorted, swapped;
    sorted = false;
    i = 0;
    while (i < size - 1 && !sorted){
        swapped = false;

```

```

        for (int j = 0; j < size - 1 - i; ++j) {
            if (matrix[j][j] > matrix[j+1][j+1]) {
                temp_num = matrix[j][j];
                matrix[j][j] = matrix[j+1][j+1];
                matrix[j+1][j+1] = temp_num;
                swapped = true;
            }
        }
        if (!swapped) {
            sorted = true;
        }
        i++;
    }
}

void swap_diagonals(double** matrix_1, double** matrix_2, int
size) {
    double temp_num;
    for (int i = 0; i < size; i++) {
        temp_num = matrix_1[i][i];
        matrix_1[i][i] = matrix_2[i][i];
        matrix_2[i][i] = temp_num;
    }
}

void delete_matrix(double** matrix, int size) {
    for (int i = 0; i < size; i++) {
        delete[] matrix[i];
    }
    delete[] matrix;
}

```

Результат на C++:

The screenshot shows a C++ IDE with a terminal window displaying the output of a program. The program prompts the user to enter the size of the square matrices (4). It then displays two 4x4 matrices, Matrix 1 and Matrix 2. Matrix 1 is sorted by its main diagonal, and Matrix 2 is sorted by its main diagonal. The sorted matrices are then swapped, and the final state of both matrices is displayed.

```

Lab8_OP main.cpp
CMakeLists.txt main.cpp
Run: Lab8_OP
Enter the size of the square matrices: 4
Matrix 1:
3.7 3 7.9 7.8
9 1.3 1.2 1.3
2.3 5 1.7 4.3
1.3 4.8 10 1.6
Matrix 2:
8.7 3.4 6.9 8.8
9.7 2.4 9.6 6
8.9 4.2 3.8 8.8
3.5 8.6 5.7 7
Matrix 1 with sorted main diagonal:
1.3 3 7.9 7.8
9 1.6 1.2 1.3
2.3 5 1.7 4.3
1.3 4.8 10 3.7
Matrix 2 with sorted main diagonal:
2.4 3.4 6.9 8.8
9.7 3.8 9.6 6
8.9 4.2 7 8.8
3.5 8.6 5.7 8.7
Matrix 1 with main diagonal from matrix 2:
2.4 3 7.9 7.8
9 3.8 1.2 1.3
2.3 5 7 4.3
1.3 4.8 10 8.7
Matrix 2 with main diagonal from matrix 1:
1.3 3.4 6.9 8.8
9.7 1.6 9.6 6
8.9 4.2 1.7 8.8
3.5 8.6 5.7 3.7

```

```
Lab8_OP main.cpp
CMakeLists.txt main.cpp
Run: Lab8_OP
/Users/kyryl/Desktop/Lab8_OP/cmake-build-debug/Lab8_OP
Enter the size of the square matrices: 3
Matrix 1:
7.3 2.1 2.4
6.1 6 2.1
1.6 6 6.4
Matrix 2:
6.9 5.2 7.9
6.4 4.5 9.1
6.7 3 7.9
Matrix 1 with sorted main diagonal:
6 2.1 2.4
6.1 6.4 2.1
1.6 6 7.3
Matrix 2 with sorted main diagonal:
4.5 5.2 7.9
6.4 6.9 9.1
6.7 3 7.9
Matrix 1 with main diagonal from matrix 2:
4.5 2.1 2.4
6.1 6.9 2.1
1.6 6 7.9
Matrix 2 with main diagonal from matrix 1:
6 5.2 7.9
6.4 6.4 9.1
6.7 3 7.3
Process finished with exit code 0
```

Висновок

Отже, я дослідив двовимірні масиви, опанував технологію їх використання та написав алгоритм для заміни головних діагоналей двох двовимірних масивів, попередньо створивши алгоритм для сортування їх елементів за збільшенням, та отримав коректний результат.