ЗВІТ

про виконання лабораторної роботи № 7.01

«Пошук заданих елементів та впорядкування рядків / стовпчиків матриці» з дисципліни

«Алгоритмізація та програмування» студента групи РІ-12

Мілянця Мар`яна

# Мета роботи:

Навчитися обчислювати кількість, суму заданих елементів матриці; а також впорядковувати матрицю, переставляючи місцями її рядки /

стовпчики.

# Умова завдання:

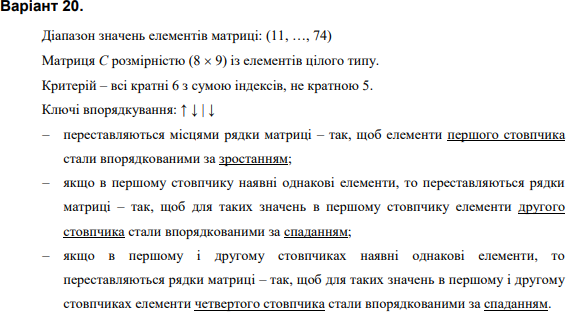
Написати програму, яка: 1. за допомогою генератора випадкових чисел формує вказану матрицю – так, що значення її елементів належать заданому діапазону. Обчислити кількість та суму тих елементів, які задовольняють вказаному критерію; а також замінити нулями ці елементи; 2. впорядковує матрицю, переставляючи місцями її рядки / стовпчики відповідно до вказаних ключів впорядкування. Пояснити призначення змінних. Необхідно: - (завдання 1 і 2) сформувати матрицю; - (завдання 1 і 2) вивести її на екран у вигляді таблиці, використовуючи форматне виведення; - (завдання 2) впорядкувати матрицю, переставляючи місцями її рядки / стовпчики відповідно до вказаних ключів впорядкування; - (завдання 2) вивести на екран модифіковану матрицю у вигляді окремої таблиці, використовуючи виведення з тими самими специфікаціями формату; - (завдання 1) виконати вказані у завданні дії (обчислити кількість та суму тих елементів, які задовольняють вказаному критерію; а також замінити нулями ці елементи); - (завдання 1) вивести результат – обчислену кількість і суму, та вивести на екран модифіковану матрицю у вигляді окремої таблиці, використовуючи вивення з тими самими специфікаціями формату. Усі вказані дії необхідно реалізувати за допомогою окремих функцій. Інформація у функції має передаватися лише за допомогою параметрів. Використання глобальних змінних – не допускається. Кожна функція має виконувати лише одну роль, і ця роль має бути відображена у назві функції. Рекурсивний та ітераційний способи – це 2 різні проекти, для яких потрібно 2 різних unit-тести і 2 різних звіти. «Функція, яка повертає / обчислює / шукає ...» – має не виводити ці значення, а повернути їх у місце виклику як результат функції або як відповідний вихідний параметр. Діапазон значень елементів матриці: (7, …, 65) Матриця A розмірністю (5  9) із елементів цілого типу. Критерій – всі парні і крім кратних 7. Ключі впорядкування: ↓

↓ | ↑ – переставляються місцями рядки матриці – так, щоб елементи першого стовпчика стали впорядкованими за спаданням; – якщо в першому стовпчику наявні однакові елементи, то переставляються рядки матриці – так, щоб для таких значень в першому стовпчику елементи другого стовпчика стали впорядкованими за спаданням; – якщо в першому і другому

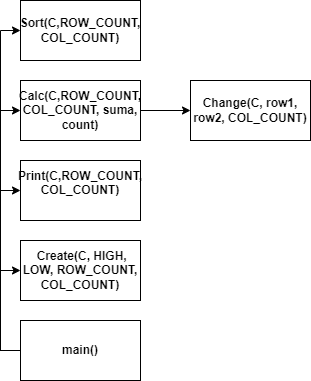
стовпчиках наявні однакові елементи, то переставляються рядки матриці – так, щоб для таких значень в першому і другому стовпчиках елементи четвертого стовпчика стали впорядкованими за зростанням. 89 Алгоритми Алгоритми формування масиву та його виведення на екран описані в розділі

«Приклади». Алгоритм обчислення кількості, суми тих елементів, які задовольняють певній умові та заміни вказаних елементів нулями (функція Calc()): Значення суми S та кількості k ініціюються нулями. В циклі з параметром i, що змінюється від 0 до rowCount-1, переглядаються всі рядки матриці. У вкладеному циклі з параметром j, що змінюється від 0 до colCount-1, переглядаються всі елементи поточного рядка. Якщо поточний елемент задовольняє умові, то сума збільшується на значення поточного елемента, кількість збільшується на 1, а поточний елемент отримує значення

0. Алгоритм сортування рядків (функція Sort()): Використовується метод обміну («бульбашки»): якщо елементи двох сусідніх рядків не в тому порядку, як слід,– то ці рядки міняються місцями. Алгоритм обміну рядків місцями (функція Change()): В циклі з параметром j, що змінюється від 0 до colCount-1, переглядаються всі елементи двох рядків. Здійснюється обмін значеннями поточних елементів двох рядків: Змінна tmp отримує значення поточного елемента першого рядка. Поточний елемент першого рядка отримує значення поточного елемента другого рядка. Поточний елемент другого рядка отримує значення змінної tmp.



# Структурна схема програми:



**Ітераційний спосіб:**

# Текст програми:

#include <iostream> #include <Windows.h>

using namespace std; const int ROW\_COUNT = 8; const int COL\_COUNT = 9;

void Create(int\*\* C, const int HIGH, const int LOW, const int ROW\_COUNT, const int COL\_COUNT);

void Print(int\*\* C, const int ROW\_COUNT, const int COL\_COUNT);

void Calc(int\*\* C, const int ROW\_COUNT, const int COL\_COUNT, int& suma, int& count);

void Sort(int\*\* C, const int ROW\_COUNT, const int COL\_COUNT);

void Change(int\*\* C, const int row1, const int row2, const int COL\_COUNT);

int main()

{

srand(time(NULL));

const int HIGH = 74; const int LOW = 11; const int ROW\_COUNT = 8; const int COL\_COUNT = 9;

int\*\* C = new int\* [ROW\_COUNT];

for (int i = 0; i < ROW\_COUNT; i++) C[i] = new int[COL\_COUNT];

cout << "F: " << "\n";

Create(C, HIGH, LOW, ROW\_COUNT, COL\_COUNT); Print(C, ROW\_COUNT, COL\_COUNT);

cout << "D: " << "\n"; Sort(C, ROW\_COUNT, COL\_COUNT);

Print(C, ROW\_COUNT, COL\_COUNT);

int suma = 0; int count = 0;

Calc(C, ROW\_COUNT, COL\_COUNT, suma, count); cout << "S: " << suma << "\n" << "\n"; cout << "N: " << count << "\n" << "\n"; cout << "H: " << "\n";

Print(C, ROW\_COUNT, COL\_COUNT);

for (int i = 0; i < ROW\_COUNT; i++) delete[] C[i];

delete[] C;

system("pause"); return 0;

}

void Create(int\*\* C, const int HIGH, const int LOW, const int ROW\_COUNT, const int COL\_COUNT)

{

for (int i = 0; i < ROW\_COUNT; i++)

{

for (int j = 0; j < COL\_COUNT; j++)

C[i][j] = LOW + rand() % (HIGH - LOW + 1);

}

}

void Print(int\*\* C, const int ROW\_COUNT, const int COL\_COUNT)

{

for (int i = 0; i < ROW\_COUNT; i++)

{

for (int j = 0; j < COL\_COUNT; j++)

{

cout << C[i][j] << "\t";

}

cout << "\n";

}

cout << "\n";

}

void Calc(int\*\* C, const int ROW\_COUNT, const int COL\_COUNT, int& suma, int& count)

{

for (int i = 0; i < ROW\_COUNT; i++)

{

for (int j = 0; j < COL\_COUNT; j++)

{

if (C[i][j] % 6 == 0 && (i + j) % 5 != 0)

{

suma += C[i][j]; count++; C[i][j] = 0;

}

}

}

}

void Change(int\*\* C, const int row1, const int row2, const int COL\_COUNT)

{

int tmp;

for (int j = 0; j < COL\_COUNT; j++)

{

tmp = C[row1][j]; C[row1][j] = C[row2][j]; C[row2][j] = tmp;

}

}

void Sort(int\*\* C, const int ROW\_COUNT, const int COL\_COUNT)

{

for (int i0 = 0; i0 < ROW\_COUNT - 1; i0++)

for (int i1 = 0; i1 < ROW\_COUNT - i0 - 1; i1++) if ((C[i1][0] < C[i1 + 1][0])

||

(C[i1][0] == C[i1 + 1][0] &&

C[i1][1] < C[i1 + 1][1])

}

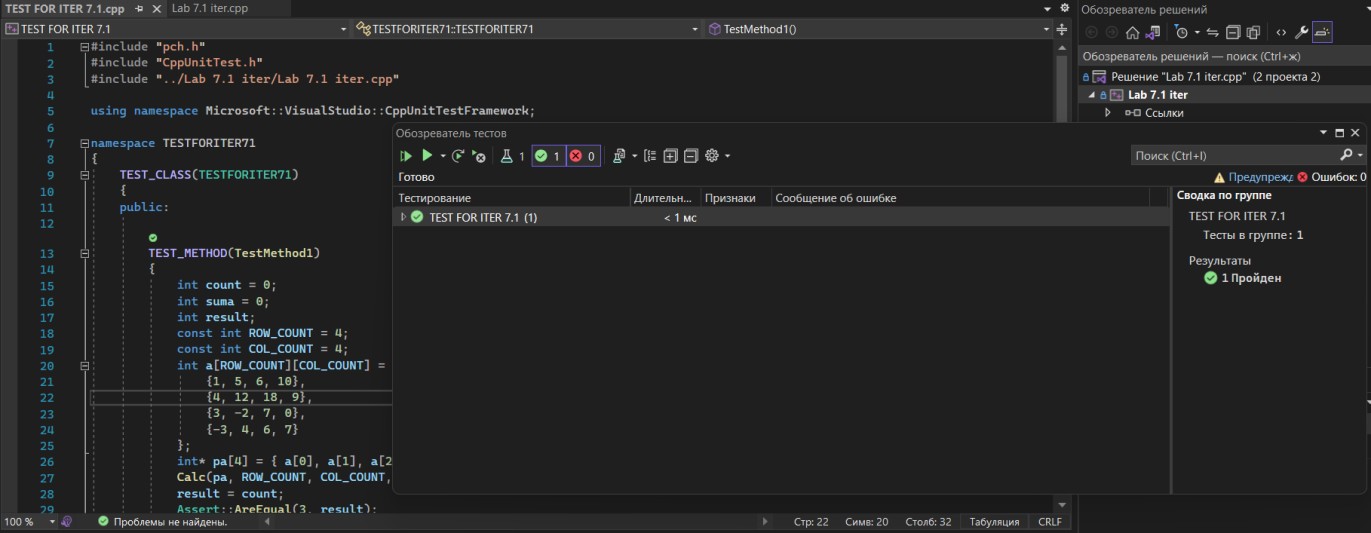
# Результати unit-тесту:

||

(C[i1][0] == C[i1 + 1][0] &&

C[i1][1] == C[i1 + 1][1] &&

C[i1][3] < C[i1 + 1][3])) Change(C, i1, i1 + 1, COL\_COUNT);



# Текст unit-тесту:

#include "pch.h" #include "CppUnitTest.h"

#include "../Lab 7.1 iter/Lab 7.1 iter.cpp"

using namespace Microsoft::VisualStudio::CppUnitTestFramework; namespace TESTFORITER71

{

TEST\_CLASS(TESTFORITER71)

{

public:

TEST\_METHOD(TestMethod1)

{

int count = 0; int suma = 0; int result;

const int ROW\_COUNT = 4; const int COL\_COUNT = 4;

int a[ROW\_COUNT][COL\_COUNT] = {

{1, 5, 6, 10},

{4, 12, 18, 9},

{3, -2, 7, 0},

{-3, 4, 6, 7}

};

int\* pa[4] = { a[0], a[1], a[2], a[3] }; Calc(pa, ROW\_COUNT, COL\_COUNT, suma, count); result = count;

Assert::AreEqual(3, result);

}

};

}

**Посилання**: <https://github.com/milianets20/Lab-7.1-iter>

**Висновок**: виконуючи лабораторну роботу я навчився обчислювати кількість, суму заданих елементів матриці; а також впорядковувати матрицю, переставляючи місцями її рядки / стовпчики.