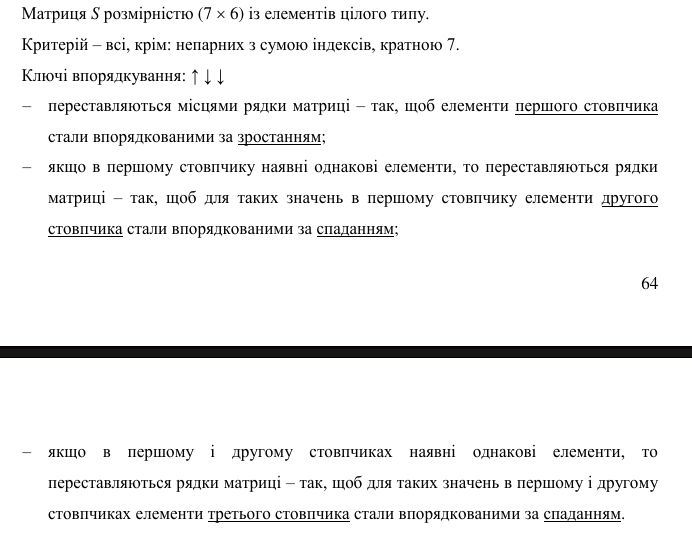
**ЗВІТ  
про виконання лабораторної роботи № 07.1.рек  
Пошук заданих елементів та впорядкування рядків / стовпчиків матриці  
з дисципліни  
"Алгоритмізація та програмування"  
студента групи РІ-12  
Грушевського Івана Олександровича**

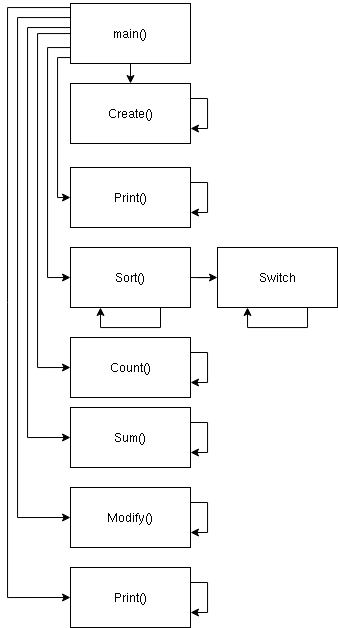
# Мета: навчитися обчислювати кількість, суму заданих елементів матриці; а також впорядковувати матрицю, переставляючи місцями її рядки / стовпчики.

# 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.



# Структурна схема програми:



# Текст програми:

## main.cpp

#include "../lib/function.h"  
  
// include libraries  
#include <iostream>  
#include <cstdlib>  
#include <ctime>  
  
using namespace std;  
  
int main() {  
 // script  
   
 srand((unsigned)time(NULL));  
  
 // variable declaration  
 const int rowCount = 7;  
 const int colCount = 6;  
 const int Low = 9;  
 const int High = 61;  
  
 // creation of 2D array  
 int \*\*a = new int\*[rowCount];  
  
 for (int i=0; i<rowCount; i++)  
 a[i] = new int [colCount];  
  
 // main work  
 lib::Create(a, rowCount, colCount, Low, High, 0, 0);  
  
 cout << endl;  
 lib::Print(a, rowCount, colCount, 0, 0);  
  
 lib::Sort(a, rowCount, colCount, 0, 0);  
  
 cout << "Count: " << lib::Count(a, rowCount, colCount,   
 0, 0, 0) << endl;  
  
  
 cout << "Sum: " << lib::Sum(a, rowCount,colCount,   
 0, 0, 0) << endl;  
 lib::Modify(a, rowCount, colCount, 0, 0);  
  
 cout << endl;  
 lib::Print(a, rowCount, colCount, 0, 0);  
  
 // cleaning  
 lib::Clean(a, rowCount, 0);  
  
 delete [] a;   
  
 return 0;  
}

## function.cpp

// include header  
#include "function.h"  
  
// include libraries  
#include <iostream>  
#include <iomanip>  
#include <cstdlib>  
  
using namespace std;  
  
namespace lib {  
 // functions script  
  
 void Create(int\*\* a, const int rowCount, const int colCount,  
 const int Low, const int High, int i, int j)  
 {  
 a[i][j] = Low + rand() % (High-Low+1);  
  
 if (j<colCount-1) {  
 Create(a, rowCount, colCount, Low, High, i, j+1);  
 }  
 else {  
 if (i<rowCount-1) {  
 Create(a, rowCount, colCount, Low, High, i+1, 0);  
 }  
 }  
 }  
  
 void Modify(int\*\* a, const int rowCount, const int colCount,   
 int i, int j)  
 {  
 if(!(((a[i][j]%2 != 0) && ((i+j)%7 == 0))))  
 a[i][j] = 0;  
  
 if (j<colCount-1)  
 Modify(a,rowCount, colCount, i, j+1);  
 else   
 if (i<rowCount-1)  
 Modify(a, rowCount, colCount, i+1, 0);  
 }  
  
 int Count(int\*\* a, const int rowCount, const int colCount,  
 int result, int i, int j)  
 {  
 if(!(((a[i][j]%2 != 0) && ((i+j)%7 == 0))))  
 result += 1;  
  
 if (j<colCount-1)  
 return Count(a,rowCount, colCount, result, i, j+1);  
 else {   
 if (i<rowCount-1)  
 return Count(a, rowCount, colCount, result, i+1, 0);  
 }  
  
 return result;  
 }  
  
 int Sum(int\*\* a, const int rowCount, const int colCount,   
 int sum, int i, int j)  
 {  
 if(!(((a[i][j]%2 != 0) && ((i+j)%7 == 0))))  
 sum += a[i][j];  
  
 if (j<colCount-1)  
 return Sum(a, rowCount, colCount, sum, i, j+1);  
 else {   
 if (i<rowCount-1)   
 return Sum(a, rowCount, colCount, sum, i+1, 0);  
 }   
 return sum;  
   
 }  
  
 void Print(int\*\* a, const int rowCount, const int colCount,   
 int i, int j)  
 {  
 cout << setw(4) << a[i][j];  
 if (j<colCount-1) {  
 Print(a, rowCount, colCount, i, j+1);  
 }  
 else {  
 if (i<rowCount-1) {  
 cout << endl;  
 Print(a, rowCount, colCount, i+1, 0);  
 }  
 else {  
 cout << endl << endl;  
 }  
 }  
 }  
  
 void Switch(int\*\* a, const int row1, const int row2, const int colCount, int j)  
 {  
 int tmp;  
   
 tmp = a[row1][j];  
 a[row1][j] = a[row2][j];  
 a[row2][j] = tmp;  
  
 if (j<colCount-1)  
 Switch(a, row1, row2, colCount, j+1);  
 }  
  
 void Sort(int\*\* a, const int rowCount, const int colCount,  
 int count, int i)  
 {  
 if (a[i][0] > a[i+1][0]) {  
 Switch(a, i, i+1, colCount, 0);  
 }  
 else if (a[i][0] == a[i+1][0]) {  
 if (a[i][1] < a[i+1][1]) {  
 Switch(a, i, i+1, colCount, 0);  
 cout << "SECOND SORT:\n";  
 Print(a, rowCount, colCount, 0, 0);  
 }  
 else if (a[i][1] == a[i+1][1]) {  
 if (a[i][2] < a[i+1][2]) {  
 Switch(a, i, i+1, colCount, 0);  
 cout << "THIRD SORT:\n";  
 Print(a, rowCount, colCount, 0, 0);  
 }  
 }  
 }  
  
 if (i<rowCount-2)  
 Sort(a, rowCount,colCount, count, i+1);  
 else {  
 count += 1;  
 if (count != rowCount)  
 Sort(a, rowCount, colCount, count, 0);  
 else {   
 cout << "RESULT OF SORT:\n";  
 Print(a, rowCount, colCount, 0, 0);  
 }  
 }  
 }  
  
 void Clean(int\*\* a, const int rowCount, int i)  
 {  
 delete [] a[i];  
 if (i<rowCount-2)  
 Clean(a, rowCount, i+1);  
 }  
}

## Посилання на git-репозиторій з проктом

https://github.com/p-i-r-u-m/University-labs/tree/master/AP

# Результати unit-тесту:

## unit\_tests.cpp

// Include the gtest header  
# include <gtest/gtest.h>  
  
// Include lib headers  
#include "../lib/function.h"  
  
// Helper function to create a 2D dynamic array  
int\*\* createArray(const int rowCount, const int colCount) {  
 int\*\* arr = new int\*[rowCount];  
 for (int i = 0; i < rowCount; i++) {  
 arr[i] = new int[colCount];  
 }  
 return arr;  
}  
  
// Helper function to delete the 2D dynamic array  
void deleteArray(int\*\* arr, const int rowCount) {  
 for (int i = 0; i < rowCount; i++) {  
 delete[] arr[i];  
 }  
 delete[] arr;  
}  
  
// Test case for Count function  
TEST(ArrayFunctionsTest, CountTest) {  
 int rowCount = 3, colCount = 3;  
 int\*\* a = createArray(rowCount, colCount);  
  
 // Fill array with test values  
 int values[3][3] = {  
 {1, 2, 3},  
 {4, 5, 6},  
 {7, 8, 9}  
 };  
 for (int i = 0; i < rowCount; i++) {  
 for (int j = 0; j < colCount; j++) {  
 a[i][j] = values[i][j];  
 }  
 }  
  
 EXPECT\_EQ(lib::Count(a, rowCount, colCount, 0, 0, 0), 8);  
  
 deleteArray(a, rowCount);  
}  
  
// Test case for Sum function  
TEST(ArrayFunctionsTest, SumTest) {  
 int rowCount = 3, colCount = 3;  
 int\*\* a = createArray(rowCount, colCount);  
  
 // Fill array with test values  
 int values[3][3] = {  
 {1, 2, 3},  
 {4, 5, 6},  
 {7, 8, 9}  
 };  
 for (int i = 0; i < rowCount; i++) {  
 for (int j = 0; j < colCount; j++) {  
 a[i][j] = values[i][j];  
 }  
 }  
  
 EXPECT\_EQ(lib::Sum(a, rowCount, colCount, 0, 0, 0), 44);  
  
 deleteArray(a, rowCount);  
}

## Вивід unit-тесту:

Internal ctest changing into directory: /home/pirum/University-labs/algorithmization\_and\_programming/7\_lab/1\_lab\_rec/build  
Test project /home/pirum/University-labs/algorithmization\_and\_programming/7\_lab/1\_lab\_rec/build  
 Start 1: ArrayFunctionsTest.CountTest  
 Start 2: ArrayFunctionsTest.SumTest  
1/2 Test #1: ArrayFunctionsTest.CountTest ..... Passed 0.00 sec  
2/2 Test #2: ArrayFunctionsTest.SumTest ....... Passed 0.00 sec  
  
100% tests passed, 0 tests failed out of 2  
  
Total Test time (real) = 0.01 sec

# Висновки:

У результаті виконання лабораторної роботи я зміг навчитися обчислювати кількість, суму заданих елементів матриці; а також впорядковувати матрицю, переставляючи місцями її рядки / стовпчики.