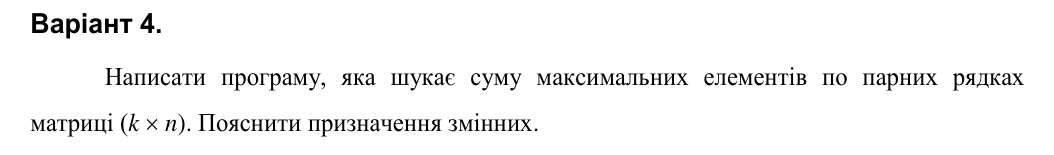
**ЗВІТ  
про виконання лабораторної роботи № 07.2.іт  
Опрацювання багатовимірних масивів ітераційним і рекурсивним способами  
з дисципліни  
"Алгоритмізація та програмування"  
студента групи РІ-12  
Грушевського Івана Олександровича**

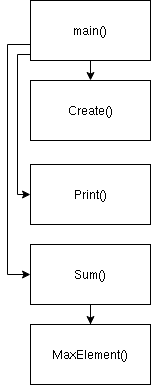
# Мета: Навчитися опрацьовувати двовимірні та багатовимірні масиви ітераційним і рекурсивним способами.

# Умова завдання:

Необхідно:  
-  
 сформувати матрицю із елементів цілого типу (величину кожної розмірності та  
діапазон значень елементів – вибрати самостійно);  
-  
 вивести її на екран у вигляді таблиці, використовуючи форматне виведення;  
-  
 виконати вказані у завданні дії;  
-  
 вивести результат, причому, якщо матриця була змінена – то вивести на екран  
модифіковану матрицю у вигляді окремої таблиці, використовуючи виведення з  
тими самими специфікаціями формату.  
  
1. Ітераційний спосіб. Усі вказані дії необхідно реалізувати за допомогою окремих  
функцій. Інформація у функції має передаватися лише за допомогою параметрів.  
Використання глобальних змінних – не допускається.  
2. Рекурсивний спосіб. Усі вказані дії необхідно реалізувати за допомогою окремих  
рекурсивних 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 header  
#include "../lib/function.h"  
  
// include libraries  
#include <iostream>  
#include <cstdlib>  
#include <ctime>  
  
using namespace std;  
  
int main() {  
 // script  
   
 srand((unsigned)time(NULL));  
  
 // variable declaration  
 int rowCount, colCount, Low, High;  
  
 cout << "Rows: "; cin >> rowCount;  
 cout << "Columbs: "; cin >> colCount;  
 cout << "Low: "; cin >> Low;  
 cout << "High: "; cin >> High;  
  
 if (rowCount <= 0 || colCount <= 0) {  
 cout << "Either rowCount or colCount is invalid" << endl;  
 return 0;  
 } else if (Low >= High) {  
 cout << "Low is not less than High" << endl;  
 return 0;  
 } else if (Low <= 0 || High <= 0) {  
 cout << "Low or High is not positive" << endl;  
 return 0;  
 }  
  
 // 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);  
 lib::Print(a, rowCount, colCount);  
  
 cout << "Sum: " << lib::Sum(a, rowCount, colCount) << endl;  
  
 // cleaning  
 for (int i=0; i<rowCount; i++)  
 delete [] a[i];  
  
 delete [] a;  
  
 return 0;  
}

## function.cpp

// include header  
#include "function.h"  
  
// include libraries  
#include <iostream>  
#include <iomanip>  
  
using namespace std;  
  
namespace lib {  
 // functions script  
  
 void Create(int\*\* a, const int rowCount, const int colCount,  
 const int Low, const int High)  
 {  
 for (int i=0; i<rowCount; i++)  
 for (int j=0; j<colCount; j++)  
 a[i][j] = Low + rand() % (High-Low+1);  
 }  
  
 void Print(int\*\* a, const int rowCount, const int colCount)  
 {  
 cout << endl;  
 for (int i=0; i<rowCount; i++)  
 {  
 for (int j=0; j<colCount; j++)  
 cout << setw(4) << a[i][j];  
 cout << endl;  
 }  
 cout << endl;  
 }  
  
 int MaxElement(int\*\* a, const int row, const int colCount)  
 {  
 int max = a[0][0];  
 for (int j=0; j<colCount; j++)  
 if (a[row][j] > max)  
 max = a[row][j];  
  
 return max;  
 }  
  
 int Sum(int\*\* a, const int rowCount, const int colCount)  
 {  
 int sum = 0;  
 for (int i=1; i<rowCount; i+=2)  
 sum += MaxElement(a, i, colCount);  
  
 return sum;  
 }  
}

## 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"  
  
// Test case for Sum function  
TEST(SumFunctionTest, BasicTests) {  
 // Create a 2D array for testing  
 const int rowCount = 4;  
 const int colCount = 4;  
  
 int\*\* a = new int\*[rowCount];  
 for (int i = 0; i < rowCount; ++i) {  
 a[i] = new int[colCount];  
 }  
  
 // Fill the array with some test data  
 int testData[4][4] = {  
 {1, 2, 3, 4}, // Row 0  
 {5, 6, 7, 8}, // Row 1  
 {9, 10, 11, 12}, // Row 2  
 {13, 14, 15, 16} // Row 3  
 };  
  
 for (int i = 0; i < rowCount; ++i) {  
 for (int j = 0; j < colCount; ++j) {  
 a[i][j] = testData[i][j];  
 }  
 }  
  
 // Test cases for Sum function  
 EXPECT\_EQ(lib::Sum(a, rowCount, colCount), 24); // Sum of max elements from rows 1 (8) and 3 (16)  
  
 // Clean up the allocated memory  
 for (int i = 0; i < rowCount; ++i) {  
 delete[] a[i];  
 }  
 delete[] a;  
}

## Вивід unit-тесту:

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

# Висновки:

У результаті виконання лабораторної роботи я зміг Навчитися опрацьовувати двовимірні та багатовимірні масиви ітераційним і рекурсивним способами.