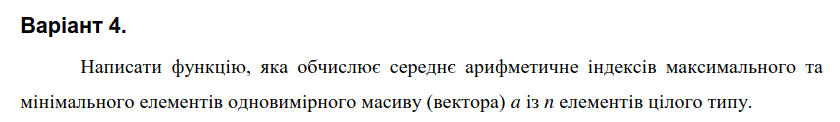
**ЗВІТ  
про виконання лабораторної роботи № 06.2.рек  
Опрацювання одновимірних масивів ітераційними та рекурсивними способами  
з дисципліни  
"Алгоритмізація та програмування"  
студента групи РІ-12  
Грушевського Івана Олександровича**

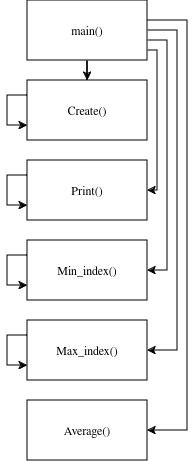
# Мета: навчитися опрацьовувати одновимірні масиви.

# Умова завдання:

Необхідно написати програму для того, щоб виконати такі дії:  
- сформувати масив;  
- вивести його на екран у вигляді рядка, використовуючи форматне виведення;  
- виконати вказані у завдання дії;  
- вивести результат, причому, якщо масив був змінений – то вивести на екран  
модифікований масив у вигляді наступного рядка, використовуючи виведення з  
тими самими специфікаціями формату.  
Спосіб 1. Всі вказані дії необхідно реалізувати за допомогою окремих функцій, що  
використовують ітераційні алгоритми.  
Спосіб 2. Всі 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 <time.h>  
#include <iostream>  
  
using namespace std;  
  
int main() {  
 // script  
  
 srand((unsigned)time(NULL));  
  
 int size = 0;  
 cout << "size = "; cin >> size;  
  
 int\* a = new int[size];  
  
 int Low = -10;  
 int High = 10;  
  
   
 lib::Create(a, size, Low, High, 0);  
 lib::Print(a, size, "", 0);  
   
  
 int min\_ind = lib::Min\_index(a, size, a[0], 0, 0);  
 cout << "index of minimum: " << min\_ind << endl;  
  
 int max\_ind = lib::Max\_index(a, size, a[0], 0, 0);  
 cout << "index of max: " << max\_ind << endl;  
  
 double avr = lib::Average(min\_ind, max\_ind);  
 cout << "Result: " << avr << endl;  
   
 delete[] a;  
  
 return 0;  
}

## function.cpp

// include header  
#include "function.h"  
  
// include libraries  
#include <iostream>  
#include <string>  
  
using namespace std;  
  
namespace lib {  
 // functions script  
   
 void Create(int \*a, const int size, const int Low, const int High, int i)  
 {  
 a[i] = Low + rand() % (High-Low+1);  
 if (i<size-1)  
 Create(a, size, Low, High, i+1);  
 }  
  
 void Print(int \*a, const int size, string arr, int i)  
 {  
 arr += (" " + to\_string(a[i]));  
 if (i<size-1)  
 Print(a, size, arr, i+1);  
 else  
 cout << "a = [" << arr << " ]" << endl;  
 }  
  
 int Max\_index(int \*a, const int size, int max, int max\_index, int i)  
 {  
 if (a[i]>max)  
 {  
 max = a[i];  
 max\_index = i;  
 }  
 if (i<size-1)  
 return Max\_index(a, size, max, max\_index, i+1);  
 else  
 return max\_index;  
 }  
  
 int Min\_index(int \*a, const int size, int min, int min\_index, int i)  
 {  
 if (a[i]<min)  
 {  
 min = a[i];  
 min\_index = i;  
 }  
 if (i<size-1)  
 return Min\_index(a, size, min, min\_index, i+1);  
 else  
 return min\_index;  
  
 }  
  
 double Average(const int min\_index, const int max\_index)  
 {  
 return (static\_cast<double>(min\_index) + static\_cast<double>(max\_index))/2;  
 }  
}

## 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 suite for Max\_index function  
TEST(MaxIndexTest, HandlesNormalValues) {  
 int arr[] = {1, 2, 3, 4, 5};  
 int size = sizeof(arr) / sizeof(arr[0]);  
 EXPECT\_EQ(lib::Max\_index(arr, size, arr[0], 0, 0), 4); // Index of max (5) is 4  
}  
  
TEST(MaxIndexTest, HandlesAllNegativeValues) {  
 int arr[] = {-1, -2, -3, -4, -5};  
 int size = sizeof(arr) / sizeof(arr[0]);  
 EXPECT\_EQ(lib::Max\_index(arr, size, arr[0], 0, 0), 0); // Index of max (-1) is 0  
}  
  
TEST(MaxIndexTest, HandlesSingleElement) {  
 int arr[] = {10};  
 int size = sizeof(arr) / sizeof(arr[0]);  
 EXPECT\_EQ(lib::Max\_index(arr, size, arr[0], 0, 0), 0); // Index of max (10) is 0  
}  
  
// Test suite for Min\_index function  
TEST(MinIndexTest, HandlesNormalValues) {  
 int arr[] = {5, 4, 3, 2, 1};  
 int size = sizeof(arr) / sizeof(arr[0]);  
 EXPECT\_EQ(lib::Min\_index(arr, size, arr[0], 0, 0), 4); // Index of min (1) is 4  
}  
  
TEST(MinIndexTest, HandlesAllNegativeValues) {  
 int arr[] = {-1, -2, -3, -4, -5};  
 int size = sizeof(arr) / sizeof(arr[0]);  
 EXPECT\_EQ(lib::Min\_index(arr, size, arr[0], 0, 0), 4); // Index of min (-5) is 4  
}  
  
TEST(MinIndexTest, HandlesSingleElement) {  
 int arr[] = {10};  
 int size = sizeof(arr) / sizeof(arr[0]);  
 EXPECT\_EQ(lib::Min\_index(arr, size, arr[0], 0, 0), 0); // Index of min (10) is 0  
}  
  
// Test suite for Avarage function  
TEST(AvarageTest, HandlesNormalIndices) {  
 int minIndex = 0;  
 int maxIndex = 4;  
 EXPECT\_EQ(lib::Average(minIndex, maxIndex), 2); // Average of 0 and 4 is 2  
}  
  
TEST(AvarageTest, HandlesSameIndices) {  
 int minIndex = 3;  
 int maxIndex = 3;  
 EXPECT\_EQ(lib::Average(minIndex, maxIndex), 3); // Average of 3 and 3 is 3  
}  
  
TEST(AvarageTest, HandlesNegativeIndices) {  
 int minIndex = -2;  
 int maxIndex = -1;  
 EXPECT\_EQ(lib::Average(minIndex, maxIndex), -1.5); // Average of -2 and -1 is -1  
}

## Вивід unit-тесту:

Internal ctest changing into directory: /home/pirum/University-labs/algorithmization\_and\_programming/lab\_06/lab\_06\_2\_rec/build  
Test project /home/pirum/University-labs/algorithmization\_and\_programming/lab\_06/lab\_06\_2\_rec/build  
 Start 1: MaxIndexTest.HandlesNormalValues  
 Start 2: MaxIndexTest.HandlesAllNegativeValues  
 Start 3: MaxIndexTest.HandlesSingleElement  
 Start 4: MinIndexTest.HandlesNormalValues  
 Start 5: MinIndexTest.HandlesAllNegativeValues  
 Start 6: MinIndexTest.HandlesSingleElement  
 Start 7: AvarageTest.HandlesNormalIndices  
 Start 8: AvarageTest.HandlesSameIndices  
 Start 9: AvarageTest.HandlesNegativeIndices  
1/9 Test #1: MaxIndexTest.HandlesNormalValues ........ Passed 0.01 sec  
2/9 Test #2: MaxIndexTest.HandlesAllNegativeValues ... Passed 0.01 sec  
3/9 Test #3: MaxIndexTest.HandlesSingleElement ....... Passed 0.01 sec  
4/9 Test #4: MinIndexTest.HandlesNormalValues ........ Passed 0.01 sec  
5/9 Test #5: MinIndexTest.HandlesAllNegativeValues ... Passed 0.01 sec  
6/9 Test #6: MinIndexTest.HandlesSingleElement ....... Passed 0.01 sec  
7/9 Test #7: AvarageTest.HandlesNormalIndices ........ Passed 0.01 sec  
8/9 Test #8: AvarageTest.HandlesSameIndices .......... Passed 0.01 sec  
9/9 Test #9: AvarageTest.HandlesNegativeIndices ...... Passed 0.00 sec  
  
100% tests passed, 0 tests failed out of 9  
  
Total Test time (real) = 0.02 sec

# Висновки:

У результаті виконання лабораторної роботи я зміг навчитися опрацьовувати одновимірні масиви.