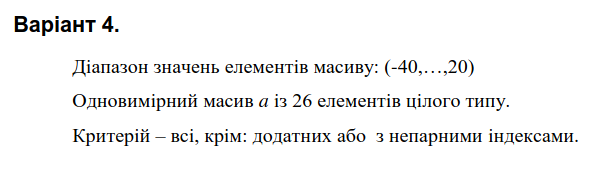
**ЗВІТ  
про виконання лабораторної роботи № 06.1.іт  
Пошук елементів одноимірного масиву ітераційним та рекурсивним способом  
з дисципліни  
"Алгоритмізація та програмування"  
студента групи РІ-12  
Грушевського Івана Олександровича**

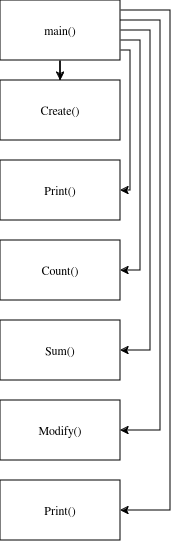
# Мета: Навчитися програмувати пошук послідовним переглядом, обчислення кількості та суми заданих елементів одновимірного масиву.

# Умова завдання:

Написати програму, яка за допомогою генератора випадкових чисел формує вказаний  
масив – такий, що значення його елементів належать заданому діапазону. Обчислити  
кількість та суму тих елементів, які задовольняють вказаному критерію; а також замінити  
нулями ці елементи.  
Необхідно:  
- сформувати масив;  
- вивести його на екран у вигляді рядка, використовуючи форматне виведення;  
- виконати вказані у завдання дії (обчислити кількість та суму тих елементів, які  
задовольняють вказаному критерію; а також замінити нулями ці елементи);  
- вивести результат – обчислену кількість та суму, та вивести на екран  
модифікований масив у вигляді наступного рядка, використовуючи виведення з  
тими самими специфікаціями формату.  
Всі вказані дії необхідно реалізувати за допомогою окремих функцій.  
Спосіб 1. Всі функції, у т.ч. ті, що реалізують обчислення кількості, суми, та заміну  
нулями вказаних елементів, – використовують ітераційні алгоритми.  
Спосіб 2. Всі функції, у т.ч. ті, що реалізують обчислення кількості, суми, та заміну  
нулями вказаних елементів, – мають бути рекурсивними.  
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 <time.h>  
  
  
using namespace std;  
  
int main() {  
 // script  
   
 srand((unsigned)time(NULL));  
 const int size = 26;  
 int a[size];  
  
 int Low = -40;  
 int High = 20;  
  
 lib::Create(a, size, Low, High);  
 cout << "a = " << lib::Print(a, size) << endl;  
  
 cout << "Number of elements: " << lib::Count(a, size) << endl;  
 cout << "Sum of elements: " << lib::Sum(a, size) << endl;  
  
 lib::Modify(a, size);  
 cout << "Modified a = " << lib::Print(a, size) << endl;  
  
 return 0;  
}

## function.cpp

// include header  
#include "function.h"  
  
// include libraries  
#include <string>  
  
  
using namespace std;  
  
namespace lib {  
 // functions script  
  
 void Create(int \*a, const int size, const int Low, const int High)  
 {  
 for (int i=0; i < size; i++) {  
 a[i] = Low + rand() % (High-Low+1);  
 }  
 }  
  
 string Print(int \*a, const int size)  
 {  
 string arr = "[";  
  
 for (int i=0; i < size; i++)  
 {  
 arr += (" " + to\_string(a[i]));   
 }  
 arr += " ]";  
  
 return arr;  
 }  
  
 int Count(int \*a, const int size)  
 {  
 int number = 0;  
  
 for (int i=0; i < size; i++)  
 {  
 if (i%2 != 0 || a[i] > 0)  
 number += 0;  
 else  
 number += 1;  
   
 }  
  
 return number;  
 }  
  
 int Sum(int \*a, const int size)  
 {  
 int sum = 0;  
  
 for (int i=0; i < size; i++)  
 {  
 if (i%2 != 0 || a[i] > 0)  
 sum += 0;  
 else  
 sum += a[i];  
   
 }  
  
 return sum;  
 }  
  
 void Modify(int \*a, const int size)  
 {  
 for (int i=0; i < size; i++)  
 {  
 if (i%2 != 0 || a[i] > 0)  
 a[i] = 0;  
 }  
 }  
  
}

## 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 the Count function  
TEST(CountTest, HandlesEvenIndexesAndNegativeValues) {  
 int arr[] = {0, -1, 2, -3, -4, 5};  
 int size = sizeof(arr) / sizeof(arr[0]);  
  
 EXPECT\_EQ(lib::Count(arr, size), 2);  
}  
  
TEST(CountTest, HandlesAllPositiveValues) {  
 int arr[] = {1, 2, 3, 4, 5, 6};  
 int size = sizeof(arr) / sizeof(arr[0]);  
  
 EXPECT\_EQ(lib::Count(arr, size), 0);  
}  
  
// Test suite for the Sum function  
TEST(SumTest, HandlesNegativeAndEvenIndexes) {  
 int arr[] = {0, -1, 2, -3, -4, 5};  
 int size = sizeof(arr) / sizeof(arr[0]);  
  
 EXPECT\_EQ(lib::Sum(arr, size), -4);  
}  
  
TEST(SumTest, HandlesAllPositiveValues) {  
 int arr[] = {1, 2, 3, 4, 5, 6};   
 int size = sizeof(arr) / sizeof(arr[0]);  
  
 EXPECT\_EQ(lib::Sum(arr, size), 0);  
}

## Вивід unit-тесту:

Internal ctest changing into directory: /home/pirum/University-labs/algorithmization\_and\_programming/lab\_06/lab\_06\_1\_it/build  
Test project /home/pirum/University-labs/algorithmization\_and\_programming/lab\_06/lab\_06\_1\_it/build  
 Start 1: CountTest.HandlesEvenIndexesAndNegativeValues  
 Start 2: CountTest.HandlesAllPositiveValues  
 Start 3: SumTest.HandlesNegativeAndEvenIndexes  
 Start 4: SumTest.HandlesAllPositiveValues  
1/4 Test #1: CountTest.HandlesEvenIndexesAndNegativeValues ... Passed 0.01 sec  
2/4 Test #2: CountTest.HandlesAllPositiveValues .............. Passed 0.01 sec  
3/4 Test #3: SumTest.HandlesNegativeAndEvenIndexes ........... Passed 0.00 sec  
4/4 Test #4: SumTest.HandlesAllPositiveValues ................ Passed 0.00 sec  
  
100% tests passed, 0 tests failed out of 4  
  
Total Test time (real) = 0.01 sec

# Висновки:

У результаті виконання лабораторної роботи я зміг Навчитися програмувати пошук послідовним переглядом, обчислення кількості та суми заданих елементів одновимірного масиву.