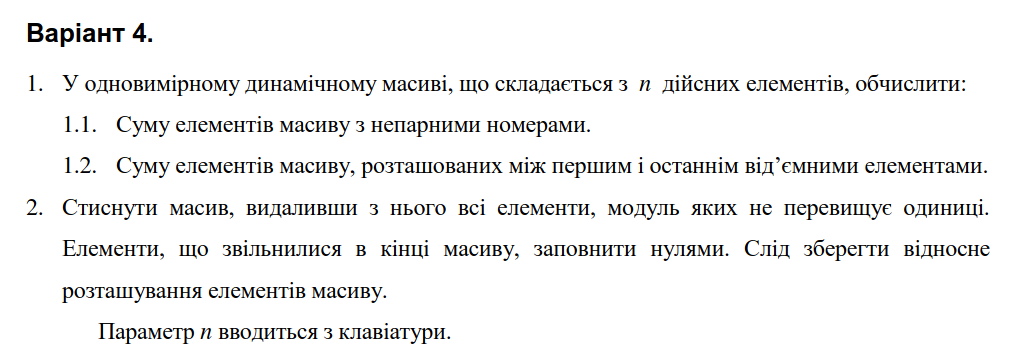
**ЗВІТ  
про виконання лабораторної роботи № 06.4.іт  
Опрацювання та впорядкування одновимірних динамічних масивів  
з дисципліни  
"Алгоритмізація та програмування"  
студента групи РІ-12  
Грушевського Івана Олександровича**

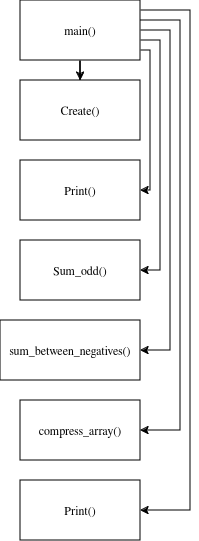
# Мета: навчитися опрацьовувати одновимірні динамічні масиви

# 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 результати, причому модифікований масив вивести на екран 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.  
Інформацію у функції слід передавати 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));  
  
 int n = 0;  
 cout << "n = "; cin >> n;  
  
 double \*a = new double[n];  
  
 double Low = -20.0;  
 double High = 20.0;  
  
 lib::Create(a, n, Low, High);  
 lib::Print(a, n);  
  
 cout << "Sum of elements with odd indexes: " << lib::Sum\_odd(a, n) << endl;  
 cout << "Sum of elements between first and last negative: " << lib::sum\_between\_negatives(a, n) << endl;  
  
 lib::compress\_array(a, n);  
 lib::Print(a, n);  
   
 delete[] a;  
 return 0;  
}

## function.cpp

// include header  
#include "function.h"  
  
// include libraries  
#include <iostream>  
  
using namespace std;  
  
namespace lib {  
 // functions script  
  
 void Create(double \*a, const int size, const double Low, const double High)  
 {  
 for (int i=0; i<size; i++)  
 {  
 double random = Low + static\_cast<double>(rand()) / (static\_cast<double>(RAND\_MAX / (High - Low)));  
 a[i] = random;  
 }  
 }  
  
 void Print(double \*a, const int size)  
 {  
 cout << "a = [";  
 for (int i=0; i<size; i++)  
 {  
 cout << " " << a[i];  
 }  
  
 cout << " ]" << endl;  
 }  
  
 double Sum\_odd(double \*a, const int size)  
 {  
 double sum = 0;  
 for (int i=0; i<size; i++)  
 {  
 if (i%2 != 0)  
 sum += a[i];  
 else  
 sum += 0;  
 }  
   
 return sum;  
 }  
  
 int first\_neg\_el(double \*a, const int size)  
 {  
   
 for (int i=0; i<size; i++)  
 {  
 if (a[i] < 0)  
 {  
 return i;  
   
 }  
 }  
  
 return -1;  
 }  
  
 int last\_neg\_el(double \*a, const int size)  
 {  
   
 for (int i=size-1; i>=0; i--)  
 {  
 if (a[i] < 0)  
 {  
 return i;  
   
 }  
 }  
  
 return -1;  
 }  
  
 double sum\_between\_negatives(double \*a, const int size) {  
 int firstIndex = first\_neg\_el(a, size);  
 int lastIndex = last\_neg\_el(a, size);  
   
   
 if (firstIndex == -1 || lastIndex == -1 || firstIndex >= lastIndex)   
 {  
 return 0.0;  
 }  
  
 double sum = 0.0;  
 for (int i = firstIndex + 1; i < lastIndex; i++)   
 {   
 sum += a[i];  
 }  
 return sum;   
 }  
  
 void compress\_array(double \*a, int &size) {  
 int j = 0;  
  
   
 for (int i = 0; i < size; i++) {  
   
 if (abs(a[i]) > 1) {  
 a[j++] = a[i];  
 }  
 }  
  
 for (; j < size; j++) {  
 a[j] = 0;  
 }  
  
 size = j;   
 }  
   
}

## 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 for the Sum\_odd function  
TEST(ArrayFunctionsTest, SumOdd) {  
 double arr1[] = {1.0, 2.0, 3.0, 4.0, 5.0};  
 EXPECT\_DOUBLE\_EQ(lib::Sum\_odd(arr1, 5), 6.0); // 2.0 + 4.0 = 6.0  
  
 double arr2[] = {0.5, -1.5, 2.5, -2.5, 1.5, 3.0};  
 EXPECT\_DOUBLE\_EQ(lib::Sum\_odd(arr2, 6), -1.0); // -1.5 + 3.0 = 1.5  
  
 double arr3[] = {0.0, 0.0, 0.0, 0.0};  
 EXPECT\_DOUBLE\_EQ(lib::Sum\_odd(arr3, 4), 0.0); // all zeros  
}  
  
// Test for the first\_neg\_el function  
TEST(ArrayFunctionsTest, FirstNegativeElement) {  
 double arr1[] = {1.0, 2.0, -3.0, 4.0};  
 EXPECT\_EQ(lib::first\_neg\_el(arr1, 4), 2); // first negative at index 2  
  
 double arr2[] = {1.0, 2.0, 3.0, 4.0};  
 EXPECT\_EQ(lib::first\_neg\_el(arr2, 4), -1); // no negative elements  
  
 double arr3[] = {-1.0, -2.0, 3.0};  
 EXPECT\_EQ(lib::first\_neg\_el(arr3, 3), 0); // first negative at index 0  
}  
  
// Test for the last\_neg\_el function  
TEST(ArrayFunctionsTest, LastNegativeElement) {  
 double arr1[] = {1.0, -2.0, 3.0, -4.0};  
 EXPECT\_EQ(lib::last\_neg\_el(arr1, 4), 3); // last negative at index 3  
  
 double arr2[] = {1.0, 2.0, 3.0};  
 EXPECT\_EQ(lib::last\_neg\_el(arr2, 3), -1); // no negative elements  
  
 double arr3[] = {-1.0, -2.0, -3.0};  
 EXPECT\_EQ(lib::last\_neg\_el(arr3, 3), 2); // last negative at index 2  
}  
  
// Test for the sum\_between\_negatives function  
TEST(ArrayFunctionsTest, SumBetweenNegatives) {  
 double arr1[] = {1.0, -2.0, 3.0, -4.0, 5.0};  
 EXPECT\_DOUBLE\_EQ(lib::sum\_between\_negatives(arr1, 5), 3.0); // 3.0 is between -2.0 and -4.0  
  
 double arr2[] = {1.0, 2.0, 3.0};  
 EXPECT\_DOUBLE\_EQ(lib::sum\_between\_negatives(arr2, 3), 0.0); // no negative elements  
  
 double arr3[] = {-1.0, 2.0, -3.0, 4.0};  
 EXPECT\_DOUBLE\_EQ(lib::sum\_between\_negatives(arr3, 4), 2.0); // 2.0 is between -1.0 and -3.0  
}

## Вивід unit-тесту:

Internal ctest changing into directory: /home/pirum/University-labs/algorithmization\_and\_programming/lab\_06/lab\_06\_4\_it/build  
Test project /home/pirum/University-labs/algorithmization\_and\_programming/lab\_06/lab\_06\_4\_it/build  
 Start 1: ArrayFunctionsTest.SumOdd  
 Start 2: ArrayFunctionsTest.FirstNegativeElement  
 Start 4: ArrayFunctionsTest.SumBetweenNegatives  
 Start 3: ArrayFunctionsTest.LastNegativeElement  
1/4 Test #1: ArrayFunctionsTest.SumOdd ................. Passed 0.01 sec  
2/4 Test #2: ArrayFunctionsTest.FirstNegativeElement ... Passed 0.00 sec  
3/4 Test #4: ArrayFunctionsTest.SumBetweenNegatives .... Passed 0.00 sec  
4/4 Test #3: ArrayFunctionsTest.LastNegativeElement .... 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 опрацьовувати одновимірні динамічні масиви