ЗВІТ

про виконання лабораторної роботи № 6.4.2

«Опрацювання та впорядкування одновимірних динамічних масивів»

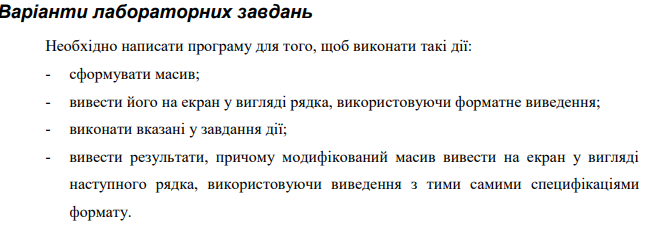
з дисципліни

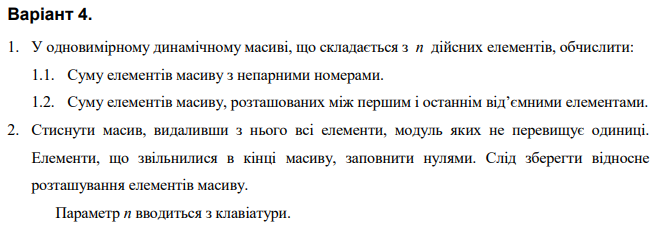
«Алгоритмізація та програмування»

студента групи РІ-11

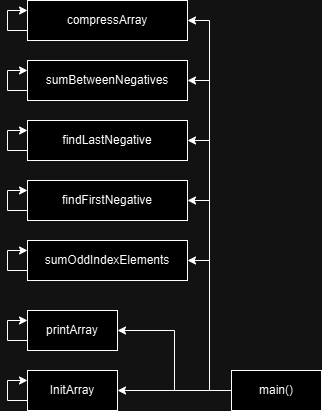
Ганина Дмитра Юрійовича

Умова завдання:





Структурна схема програми:



Текст програми:

#include <iostream>

#include <cstdlib> // для rand() і srand()

#include <ctime> // для time()

#include <cmath> // для abs()

#include<windows.h>

using namespace std;

void InitArray(int\* arr, const int size, int i) {

if (i < size) {

arr[i] = rand() % 21 - 10;

InitArray(arr, size, i + 1);

}

}

// Рекурсивна функція для виведення масиву

void printArray(const int\* arr, int size, int index = 0) {

if (index == size) {

cout << endl;

return;

}

cout << arr[index] << " ";

printArray(arr, size, index + 1);

}

// Рекурсивна функція для обчислення суми елементів з непарними індексами

int sumOddIndexElements(const int\* arr, int size, int index = 1) {

if (index >= size) return 0;

return arr[index] + sumOddIndexElements(arr, size, index + 2);

}

// Рекурсивна функція для пошуку першого від'ємного елемента

int findFirstNegative(const int\* arr, int size, int index = 0) {

if (index >= size) return -1;

if (arr[index] < 0) return index;

return findFirstNegative(arr, size, index + 1);

}

// Рекурсивна функція для пошуку останнього від'ємного елемента

int findLastNegative(const int\* arr, int index) {

if (index < 0) return -1;

if (arr[index] < 0) return index;

return findLastNegative(arr, index - 1);

}

// Рекурсивна функція для обчислення суми між першим та останнім від'ємними елементами

int sumBetweenNegatives(const int\* arr, int start, int end) {

if (start >= end) return 0;

return arr[start] + sumBetweenNegatives(arr, start + 1, end);

}

// Рекурсивна функція для стискання масиву (видалення елементів, модуль яких <= 1)

void compressArray(int\* arr, int size, int index = 0, int newIndex = 0) {

if (index == size) {

for (int i = newIndex; i < size; i++) {

arr[i] = 0; // Заповнюємо решту нулями

}

return;

}

if ((abs(arr[index]) > 1)) {

arr[newIndex] = arr[index];

compressArray(arr, size, index + 1, newIndex + 1);

}

else {

compressArray(arr, size, index + 1, newIndex);

}

}

int main() {

// Ініціалізуємо генератор випадкових чисел

srand(time(0));

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int n;

cout << "Введіть кількість елементів масиву: ";

cin >> n;

// Динамічне виділення пам'яті для масиву

int\* arr = new int[n];

InitArray(arr, n ,0);

cout << "Початковий масив: ";

printArray(arr, n);

// 1.1. Сума елементів з непарними індексами

int sumOdd = sumOddIndexElements(arr, n);

cout << "Сума елементів з непарними індексами: " << sumOdd << endl;

// 1.2. Сума між першим та останнім від'ємними елементами

int firstNeg = findFirstNegative(arr, n);

int lastNeg = findLastNegative(arr, n - 1);

if (firstNeg != -1 && lastNeg != -1 && firstNeg < lastNeg) {

int sumBetween = sumBetweenNegatives(arr, firstNeg + 1, lastNeg);

cout << "Сума елементів між першим і останнім від'ємними: " << sumBetween << endl;

}

else {

cout << "Не знайдено двох від'ємних елементів для обчислення суми." << endl;

}

// 2. Стиснення масиву

compressArray(arr, n);

cout << "Масив після стиснення: ";

printArray(arr, n);

// Звільнення динамічної пам'яті

delete[] arr;

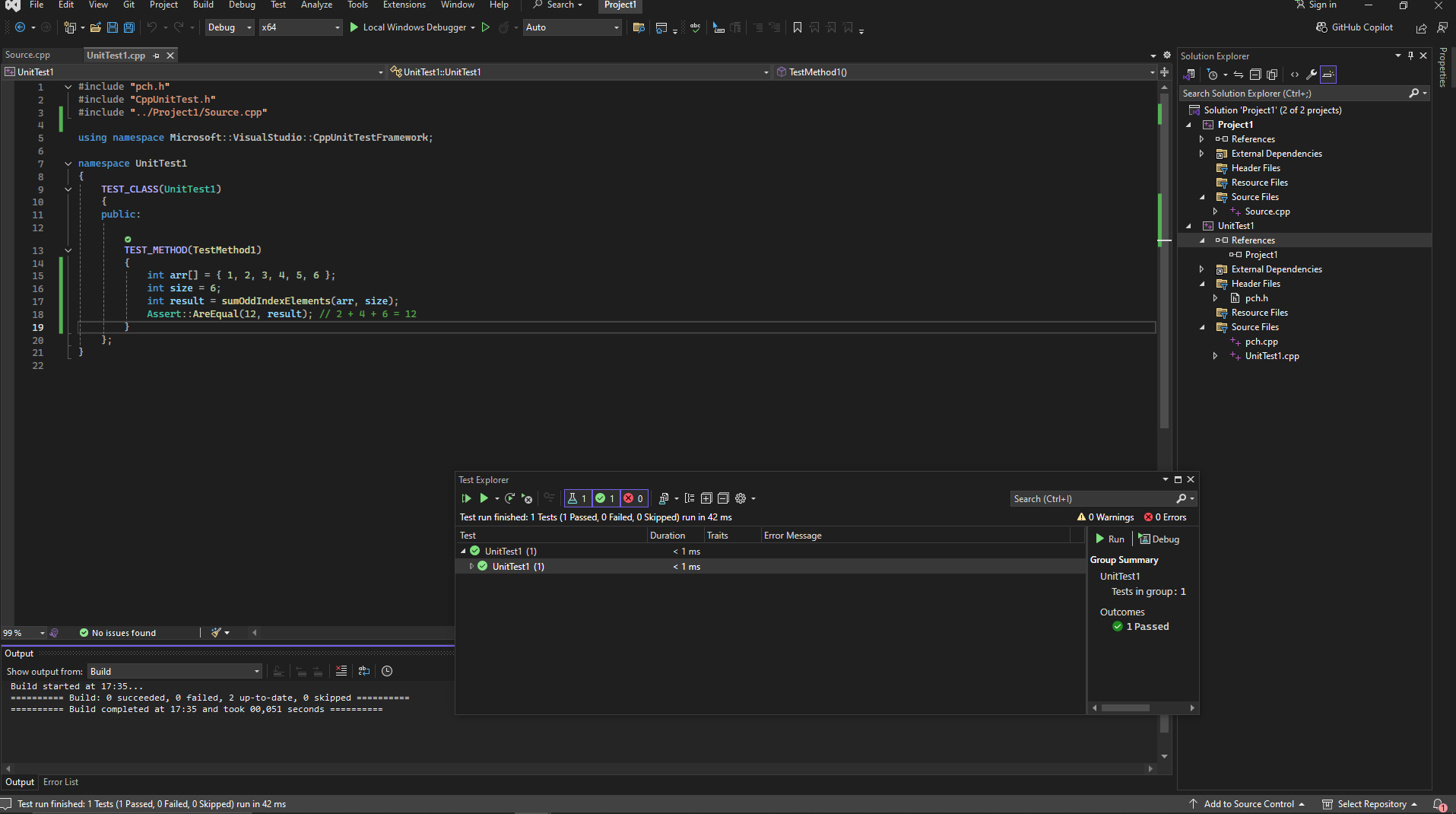
return 0;

}

Посилання на git-репозиторій з проектом:

https://github.com/HanynDmytro/Lab6.4-rek

Результати unit-тесту



Висновки:

На цій лабораторній роботі я навчився опрацьовувати одновимірні динамічні масиви.