ЗВІТ  
про виконання лабораторної роботи № 6.1.2  
«Пошук елементів одновимірного масиву ітераційним та рекурсивним способом»  
з дисципліни  
«Алгоритмізація та програмування»  
студента групи ІН-105 Б  
Горанов Анастас

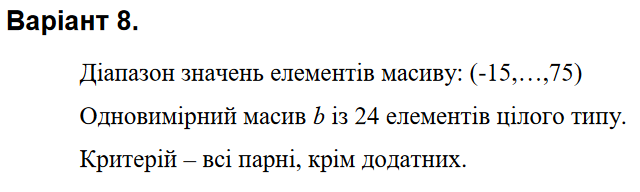
**Мета роботи:** навчитися програмувати пошук послідовним переглядом, обчислення кількості та суми заданих елементів одновимірного масиву.

**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

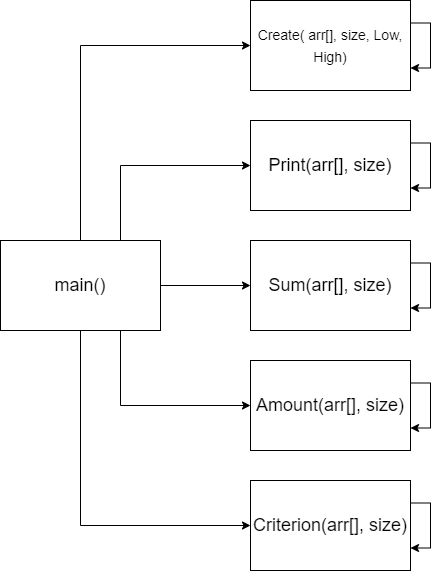
вказаному критерію; а також замінити нулями ці елементи);

- 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:**

// PR6/1/2.cpp

// Горанов Анастас

// Лабораторна робота №6.1.2

// Пошук елементів одновимірного масиву ітераційним та рекурсивним способом

// Варіант 8

#include <iostream>

#include <iomanip>

#include <time.h>

using namespace std;

void Create(int arr[], const int size, const int Low, const int High, int i)

{

arr[i] = Low + rand() % (High - Low + 1);

if (i < size - 1)

{

Create(arr, size, Low, High, i + 1);

}

}

void Print(int arr[], const int size, int i)

{

cout << arr[i] << " ";

if (i < size - 1)

{

Print(arr, size, i + 1);

}

}

int Sum(int arr[], const int size, int i)

{

if ((arr[i] < 0) && (arr[i] % 2 == 0))

{

if (i < size)

return arr[i] + Sum(arr, size, i + 1);

else

return 0;

}

else

return Sum(arr, size, i + 1);

}

int Amount(int arr[], const int size, int i)

{

if ((arr[i] < 0) && (arr[i] % 2 == 0))

{

if (i < size)

return 1 + Amount(arr, size, i + 1);

else

return 0;

}

else

return Amount(arr, size, i + 1);

}

int Criterion(int arr[], const int size, int i)

{

if (i == size)

return 0;

if ((arr[i] < 0) && (arr[i] % 2 == 0))

arr[i] = 0;

return Criterion(arr, size, i + 1);

}

int main()

{

srand((unsigned)time(NULL));

const int n = 24;

int b[n];

int Low = -15;

int High = 75;

Create(b, n, Low, High, 0);

cout << "p[ ";

Print(b, n, 0);

cout << "]" << endl;

cout << "Sum = " << Sum(b, n, 0) << endl;

cout << "Amount = " << Amount(b, n, 0) << endl;

Criterion(b, n, 0);

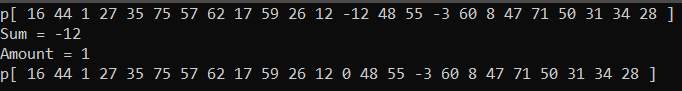
cout << "p[ ";

Print(b, n, 0);

cout << "]" << endl;

return 0;

}

****

**Посилання на Git-репозиторій:** [*https://github.com/StassNG/PR6.1.2*](https://github.com/StassNG/PR6.1.2)

**Результати unit-тесту:**

Зображення, що містить текст

Автоматично згенерований опис

**Текст Unit-тесту:**

#include "pch.h"

#include "CppUnitTest.h"

#include "../PR-6.1-rec/PR-6.1-rec.cpp"

using namespace Microsoft::VisualStudio::CppUnitTestFramework;

namespace UnitTest61

{

TEST\_CLASS(UnitTest61)

{

public:

TEST\_METHOD(TestMethod1)

{

int t;

const int k = 3;

int b[k] = { -2, -4, -6 };

t = Sum(b, k, 0);

Assert::AreEqual(t, -12);

}

};

}

**Висновок:** навчився програмувати пошук послідовним переглядом, обчислення кількості та суми заданих елементів одновимірного масиву.