ЗВІТ

про виконання лабораторної роботи № 6.04

«Опрацювання та впорядкування одновимірних динамічних масивів» з дисципліни

«Алгоритмізація та програмування» студента групи РІ-12

Мілянця Мар`яна

# Мета роботи:

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

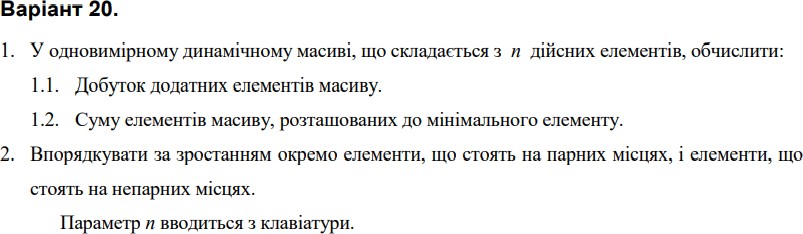
специфікаціями формату. Спосіб 1. Всі вказані дії необхідно реалізувати за допомогою окремих функцій, що використовують ітераційні алгоритми.

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

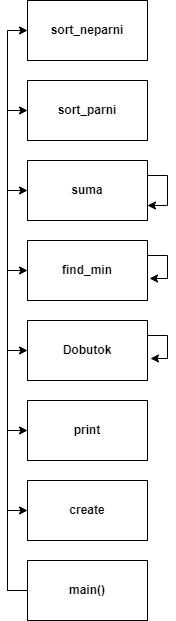
допускається. Кожна функція має виконувати лише одну роль, і ця роль має бути відображена у назві функції. Рекурсивний та ітераційний способи

– 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:



**Рекурсивний спосіб:**

# Текст програми:

#include <iostream> #include <ctime> #include <Windows.h>

using namespace std;

void create(double\* a, int n, const int i, const double HIGH, const double LOW); void print(double\* a, int n, const int i);

double Dobutok(double\* a, int n, const int i, double& dobutok); double find\_min(double\* a, int n, const int i, double& min);

double Suma(double\* a, int n, const int i, double& suma, double& min); void sort\_parni(double\* a, const int n, const int i);

void sort\_neparni(double\* a, const int n, const int i);

int main()

{

srand(time(NULL));

cout << "F: " << endl; int n;

cout << "n = "; cin >> n; double\* a = new double[n]; const double HIGH = 53; const double LOW = -17; const int i = 0;

cout << "a[] = { ";

create(a, n, i, HIGH, LOW);

print(a, n, i); cout << "}" << endl; double dobutok = 1;

cout << "D = " << Dobutok(a, n, i, dobutok) << endl; double suma = 0;

double min = a[0];

cout << "Product of positive elements = " << find\_min(a, n, i, min) << endl;

cout << " Summary of elements before minimal = " << Suma(a, n, i, suma, min) <<

endl;

cout << "G: " << endl; cout << "a[] = { "; sort\_parni(a, n, i + 1); print(a, n, i);

cout << "}" << endl; cout << "J: " << endl; sort\_neparni(a, n, i); cout << "a[] = { ";

print(a, n, i); cout << "}" << endl;

}

void create(double\* a, int n, const int i, const double HIGH, const double LOW)

{

a[i] = LOW + rand() \* (HIGH - LOW) / RAND\_MAX;

if (i < n - 1)

create(a, n, i + 1, HIGH, LOW);

}

void print(double\* a, int n, const int i)

{

cout << a[i] << " "; if (i < n - 1)

print(a, n, i + 1);

}

double Dobutok(double\* a, int n, const int i, double& dobutok)

{

bool umova = false; if (a[i] > 0)

{

umova = true;

}

if (umova)

{

dobutok \*= a[i];

}

if (i < n - 1)

Dobutok(a, n, i + 1, dobutok);

else

return dobutok;

}

double find\_min(double\* a, int n, const int i, double& min)

{

if (i < n)

{

if (a[i] < min)

min = a[i]; find\_min(a, n, i + 1, min);

}

return min;

}

double Suma(double\* a, int n, const int i, double& suma, double& min)

{

double minimum\_element = find\_min(a, n, i, min); suma += a[i];

if (a[i] == minimum\_element) return suma;

if (i < n - 1)

Suma(a, n, i + 1, suma, min);

}

void sort\_parni(double\* a, const int n, const int i)

{

for (int j = i + 2; j < n; j++) if (a[j] < a[i])

{

double temp = a[j]; a[j] = a[i];

a[i] = temp;

}

if (i < n - 1)

sort\_parni(a, n, i + 1);

}

void sort\_neparni(double\* a, const int n, const int i)

{

for (int j = i + 1; j < n; j++) if (a[j] < a[i])

{

double temp = a[j]; a[j] = a[i];

a[i] = temp;

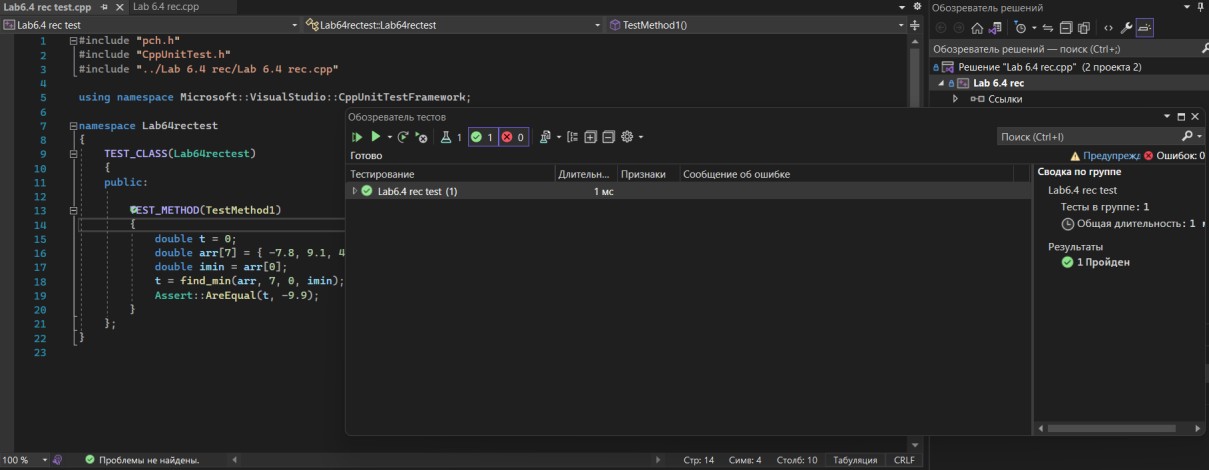
}

if (i < n - 1)

sort\_neparni(a, n, i + 1);

}

# Результати unit-тесту:



**Текст unit-тесту:**

#include "pch.h" #include "CppUnitTest.h"

#include "../Lab 6.4 rec/Lab 6.4 rec.cpp"

using namespace Microsoft::VisualStudio::CppUnitTestFramework; namespace Lab64rectest

{

TEST\_CLASS(Lab64rectest)

{

public:

TEST\_METHOD(TestMethod1)

{

double t = 0;

double arr[7] = { -7.8, 9.1, 4.2, 0.7, 9.4, -9.9, 1.2 };

double imin = arr[0];

t = find\_min(arr, 7, 0, imin);

Assert::AreEqual(t, -9.9);

}

};

}

**Посилання**: <https://github.com/milianets20/Lab-6.4-rec>

**Висновок**: виконуючи лабораторну роботу я навчився опрацьовувати одновимірні динамічні масиви