**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра «АПУ»**

**ОТЧЁТ**

**по лабораторной работе № 3**

**по дисциплине «Программирование»**

**«СТАТИЧЕСКИЕ МАССИВЫ И ФУНКЦИИ»**

|  |  |  |
| --- | --- | --- |
| Студент гр. 4391 | **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | Мухачев Д. О. |
| Преподаватель | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Писарев А.С. |

Санкт-Петербург

2024

**Цель работы:**

Задачей была представлены 5 действительных чисел и функция

Необходимо заполнить 2 статических массива значениями и вывести на экран некоторые данные об этих массивах.

**Ход работы:**

Первым шагом была реализация функции :

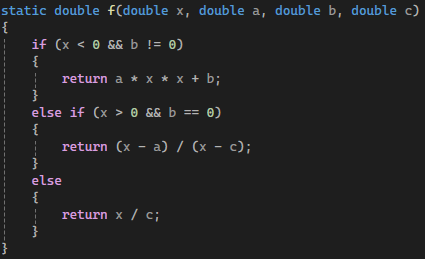
****

Рисунок 1

Также была написана функция проверки, является ли число степенью двойки:

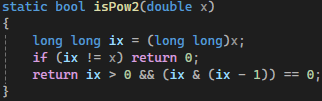
****

Рисунок 2

Была написана функция сортировки выбором для последующей сортировки массива согласно условиям задачи:

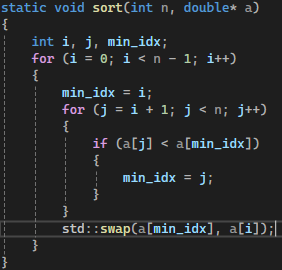


Рисунок 3

Далее был написан основной цикл программы, включающий в себя интерфейс, позволяющий вводить последовательность в соответствии с условиями задачи и использующий Unicode для вывода кириллицы:

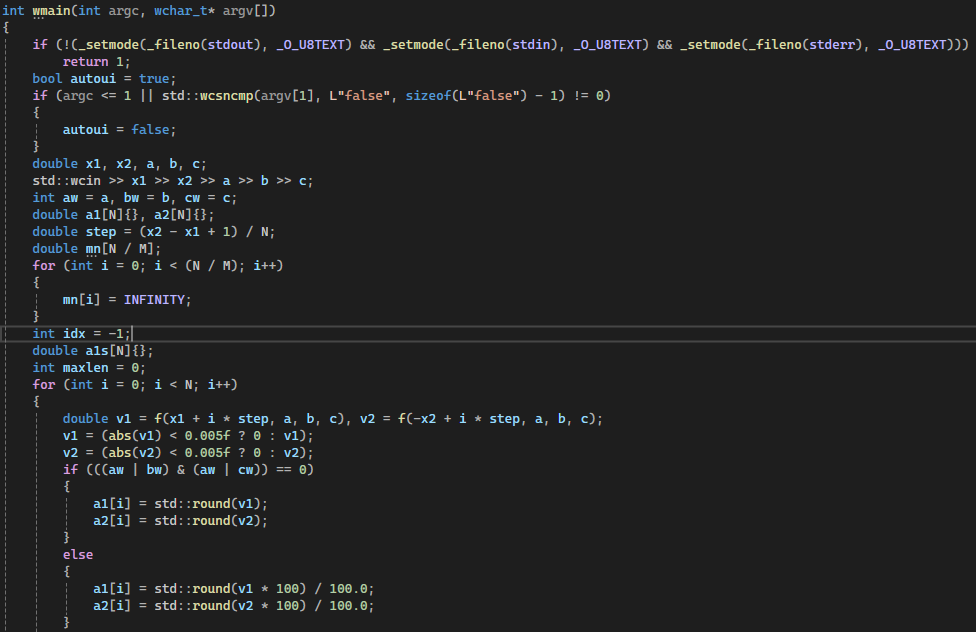


Рисунок 4

Каждый подсчитанное значение округляется в соответствии с условием задачи, вычисляется минимум для каждой из подгрупп, ведется поиск подстроки массива, состоящей из возрастающих степеней двойки, с помощью ранее написанной функции:

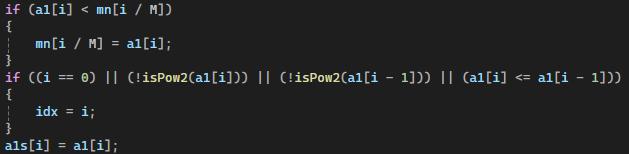


Рисунок 5

После завершения цикла идет проверка существования последовательности из степеней числа 2, первый массив сортируется, выполняется подсчет количество чисел, встречающихся в массиве более одного раза:

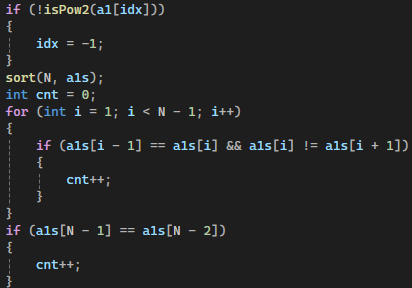


Рисунок 6

Для реализации вывода потребовалось написать функции вывода массива в 2 вариантах:

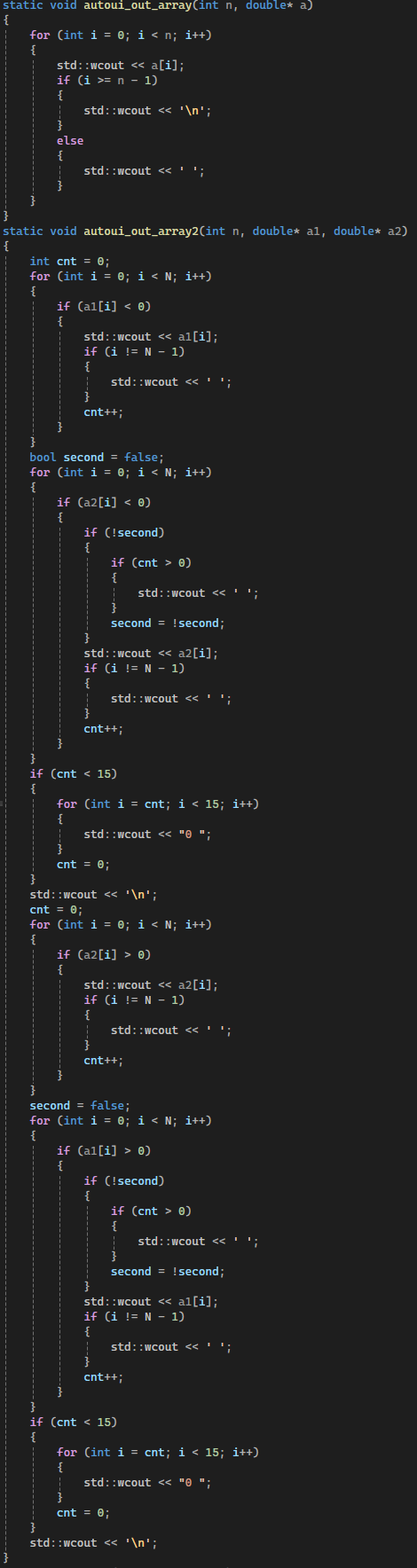


Рисунок 7

После обработки всей последовательности, программа выдает искомые величины в нужном формате:

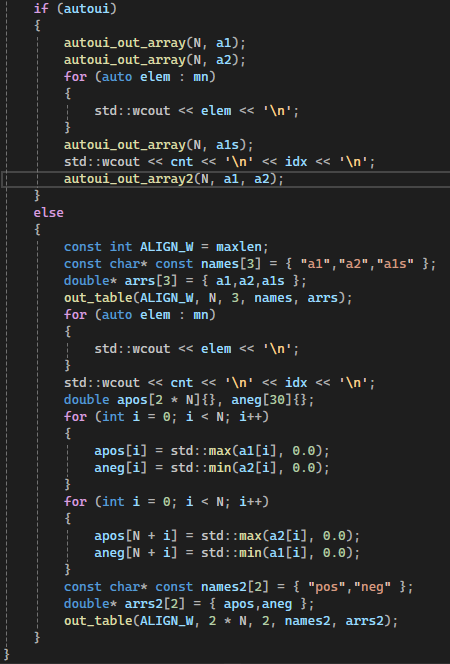


Рисунок 8

В завершении, были подключены все необходимые для работы программы библиотеки:

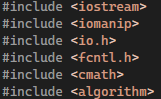


Рисунок 9

**Проверка работоспособности программы:**

Для теста были использованы входные данные из методических материалов:

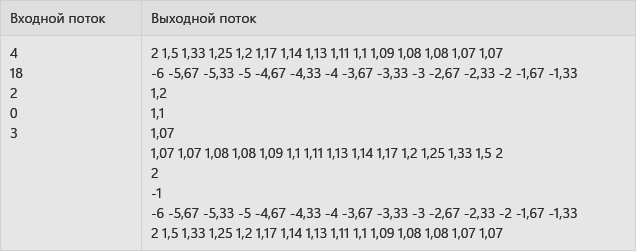


Рисунок 10

Полученные результаты:

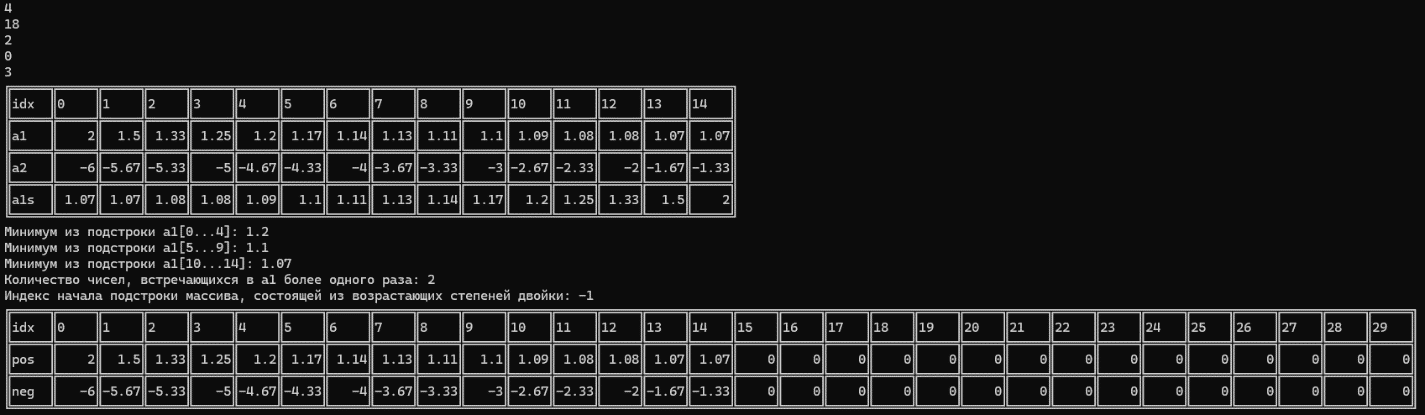


Рисунок 11

**Вывод:** В ходе выполнения лабораторной работы, целью которой была разработка программы для определения некоторых свойств функции , был получен практический опыт в нескольких ключевых областях. Было освоено создание различных типов интерфейсов, использование циклов, условных операторов и разработка функций для решения математических задач. Теоретические знания, полученные на лекциях, помогли в написании алгоритма сортировки массива.

Приложение 1

Код программы

#include <iostream>

#include <iomanip>

#include <io.h>

#include <fcntl.h>

#include <cmath>

#include <algorithm>

constexpr int N = 15, M = 5;

constexpr wchar\_t LINE\_LR = L'\u2550';

constexpr wchar\_t LINE\_TB = L'\u2551';

constexpr wchar\_t LINE\_TR = L'\u2554';

constexpr wchar\_t LINE\_TL = L'\u2557';

constexpr wchar\_t LINE\_BR = L'\u255A';

constexpr wchar\_t LINE\_BL = L'\u255D';

constexpr wchar\_t LINE\_TBR = L'\u2560';

constexpr wchar\_t LINE\_TBL = L'\u2563';

constexpr wchar\_t LINE\_BRL = L'\u2566';

constexpr wchar\_t LINE\_TRL = L'\u2569';

constexpr wchar\_t LINE\_TLBR = L'\u256C';

static void out\_sep(int ALIGN\_W, int LEN)

{

std::wcout << '\n' << LINE\_TBR;

for (int i = 0; i < LEN + 1; i++)

{

for (int j = 0; j < ALIGN\_W; j++)

{

std::wcout << LINE\_LR;

}

if (i < LEN)

{

std::wcout << LINE\_TLBR;

}

}

}

static void out\_array(int ALIGN\_W, int LEN, const char\* const name, double\* a)

{

std::wcout << LINE\_TBL << '\n' << LINE\_TB;

std::wcout << std::left << std::setw(ALIGN\_W) << std::setfill(L' ') << name;

std::wcout << LINE\_TB;

for (int i = 0; i < LEN; i++)

{

std::wcout << std::right << std::setw(ALIGN\_W) << std::setfill(L' ') << a[i];

std::wcout << LINE\_TB;

}

}

static void out\_table(int ALIGN\_W, int LEN, int n, const char\* const\* const names, double\*\* arrs)

{

std::wcout << LINE\_TR;

for (int i = 0; i < LEN + 1; i++)

{

for (int j = 0; j < ALIGN\_W; j++)

{

std::wcout << LINE\_LR;

}

if (i < LEN)

{

std::wcout << LINE\_BRL;

}

}

std::wcout << LINE\_TL << '\n' << LINE\_TB;

std::wcout << std::left << std::setw(ALIGN\_W) << std::setfill(L' ') << "idx";

std::wcout << LINE\_TB;

for (int i = 0; i < LEN; i++)

{

std::wcout << std::left << std::setw(ALIGN\_W) << std::setfill(L' ') << i;

std::wcout << LINE\_TB;

}

out\_sep(ALIGN\_W, LEN);

for (int i = 0; i < n - 1; i++)

{

out\_array(ALIGN\_W, LEN, names[i], arrs[i]);

out\_sep(ALIGN\_W, LEN);

}

out\_array(ALIGN\_W, LEN, names[n - 1], arrs[n - 1]);

std::wcout << '\n' << LINE\_BR;

for (int i = 0; i < LEN + 1; i++)

{

for (int j = 0; j < ALIGN\_W; j++)

{

std::wcout << LINE\_LR;

}

if (i < LEN)

{

std::wcout << LINE\_TRL;

}

}

std::wcout << LINE\_BL << '\n';

}

static void autoui\_out\_array(int n, double\* a)

{

for (int i = 0; i < n; i++)

{

std::wcout << a[i];

if (i >= n - 1)

{

std::wcout << '\n';

}

else

{

std::wcout << ' ';

}

}

}

static void autoui\_out\_array2(int n, double\* a1, double\* a2)

{

int cnt = 0;

for (int i = 0; i < N; i++)

{

if (a1[i] < 0)

{

std::wcout << a1[i];

if (i != N - 1)

{

std::wcout << ' ';

}

cnt++;

}

}

bool second = false;

for (int i = 0; i < N; i++)

{

if (a2[i] < 0)

{

if (!second)

{

if (cnt > 0)

{

std::wcout << ' ';

}

second = !second;

}

std::wcout << a2[i];

if (i != N - 1)

{

std::wcout << ' ';

}

cnt++;

}

}

if (cnt < 15)

{

for (int i = cnt; i < 15; i++)

{

std::wcout << "0 ";

}

cnt = 0;

}

std::wcout << '\n';

cnt = 0;

for (int i = 0; i < N; i++)

{

if (a2[i] > 0)

{

std::wcout << a2[i];

if (i != N - 1)

{

std::wcout << ' ';

}

cnt++;

}

}

second = false;

for (int i = 0; i < N; i++)

{

if (a1[i] > 0)

{

if (!second)

{

if (cnt > 0)

{

std::wcout << ' ';

}

second = !second;

}

std::wcout << a1[i];

if (i != N - 1)

{

std::wcout << ' ';

}

cnt++;

}

}

if (cnt < 15)

{

for (int i = cnt; i < 15; i++)

{

std::wcout << "0 ";

}

cnt = 0;

}

std::wcout << '\n';

}

static void sort(int n, double\* a)

{

int i, j, min\_idx;

for (i = 0; i < n - 1; i++)

{

min\_idx = i;

for (j = i + 1; j < n; j++)

{

if (a[j] < a[min\_idx])

{

min\_idx = j;

}

std::swap(a[min\_idx], a[i]);

}

}

}

static bool isPow2(double x)

{

long long ix = (long long)x;

if (ix != x) return 0;

return ix > 0 && (ix & (ix - 1)) == 0;

}

static double f(double x, double a, double b, double c)

{

if (x < 0 && b != 0)

{

return a \* x \* x + b;

}

else if (x > 0 && b == 0)

{

return (x - a) / (x - c);

}

else

{

return x / c;

}

}

int wmain(int argc, wchar\_t\* argv[])

{

if (!(\_setmode(\_fileno(stdout), \_O\_U8TEXT) && \_setmode(\_fileno(stdin), \_O\_U8TEXT) && \_setmode(\_fileno(stderr), \_O\_U8TEXT)))

return 1;

bool autoui = true;

if (argc <= 1 || std::wcsncmp(argv[1], L"false", sizeof(L"false") - 1) != 0)

{

autoui = false;

}

double x1, x2, a, b, c;

std::wcin >> x1 >> x2 >> a >> b >> c;

int aw = a, bw = b, cw = c;

double a1[N]{}, a2[N]{};

double step = (x2 - x1 + 1) / N;

double mn[N / M];

for (int i = 0; i < (N / M); i++)

{

mn[i] = INFINITY;

}

int idx = -1;

double a1s[N]{};

int maxlen = 0;

for (int i = 0; i < N; i++)

{

double v1 = f(x1 + i \* step, a, b, c), v2 = f(-x2 + i \* step, a, b, c);

v1 = (abs(v1) < 0.005f ? 0 : v1);

v2 = (abs(v2) < 0.005f ? 0 : v2);

if (((aw | bw) & (aw | cw)) == 0)

{

a1[i] = std::round(v1);

a2[i] = std::round(v2);

}

else

{

a1[i] = std::round(v1 \* 100) / 100.0;

a2[i] = std::round(v2 \* 100) / 100.0;

}

if (a1[i] < mn[i / M])

{

mn[i / M] = a1[i];

}

if ((i == 0) || (!isPow2(a1[i])) || (!isPow2(a1[i - 1])) || (!(a1[i] > a1[i - 1])))

{

idx = -1;

}

else

{

idx = i;

}

a1s[i] = a1[i];

if (!autoui)

{

int len = std::snprintf(nullptr, 0, "%.2f", a1[i]);

if (len > maxlen)

{

maxlen = len;

}

len = std::snprintf(nullptr, 0, "%.2f", a2[i]);

if (len > maxlen)

{

maxlen = len;

}

}

}

sort(N, a1s);

int cnt = 0;

for (int i = 1; i < N - 1; i++)

{

if (a1s[i - 1] == a1s[i] && a1s[i] != a1s[i + 1])

{

cnt++;

}

}

if (a1s[N - 1] == a1s[N - 2])

{

cnt++;

}

if (autoui)

{

autoui\_out\_array(N, a1);

autoui\_out\_array(N, a2);

for (auto elem : mn)

{

std::wcout << elem << '\n';

}

autoui\_out\_array(N, a1s);

std::wcout << cnt << '\n' << idx << '\n';

autoui\_out\_array2(N, a1, a2);

}

else

{

const int ALIGN\_W = maxlen;

const char\* const names[3] = { "a1","a2","a1s" };

double\* arrs[3] = { a1,a2,a1s };

out\_table(ALIGN\_W, N, 3, names, arrs);

for (int i = 0; i < (N / M); i++)

{

std::wcout << L"Минимум из подстроки a1[" << i \* M << "..." << i \* M + M - 1 << "]: " << mn[i] << '\n';

}

std::wcout << L"Количество чисел, встречающихся в a1 более одного раза: " << cnt << '\n';

std::wcout << L"Индекс начала подстроки массива, состоящей из возрастающих степеней двойки: " << idx << '\n';

double apos[2 \* N]{}, aneg[2 \* N]{};

for (int i = 0; i < N; i++)

{

apos[i] = std::max(a1[i], 0.0);

aneg[i] = std::min(a2[i], 0.0);

}

for (int i = 0; i < N; i++)

{

apos[N + i] = std::max(a2[i], 0.0);

aneg[N + i] = std::min(a1[i], 0.0);

}

const char\* const names2[2] = { "pos","neg" };

double\* arrs2[2] = { apos,aneg };

out\_table(ALIGN\_W, 2 \* N, 2, names2, arrs2);

}

}