МИНИСТЕРСТВО науки и высшего ОБРАЗОВАНИЯ РОссИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(национальный исследовательский университет)»

Институт №3 «Системы управления, информатика и электроэнергетика»

Кафедра № 304 «Вычислительные машины, системы и сети»

Информатика

Отчет по лабораторной работе № 3

Вариант № 9

Выполнила бригада группы М3О-111Бк-21

Багиров Э. Р.

Нуриев Н. Н.

Проверил Секретарев Виталий Евгеньевич

Москва 2021 г.

Оглавление

[**Задание** 3](#_Toc86736331)

[**Диаграмма алгоритма** 4](#_Toc86736332)

[**Алгоритмический вид программы** 5](#_Toc86736333)

[**Исходный код программы** 6](#_Toc86736334)

[**Некорректные тесты** 7](#_Toc86736335)

[**Корректные тесты** 8](#_Toc86736336)

[**Протокол расчета ожидаемого результата** 11](#_Toc86736337)

[**Вывод** 12](#_Toc86736338)

# **Задание**

Кафедра: 304 Курс: ИНФОРМАТИКА

Задание 3: Одномерные массивы

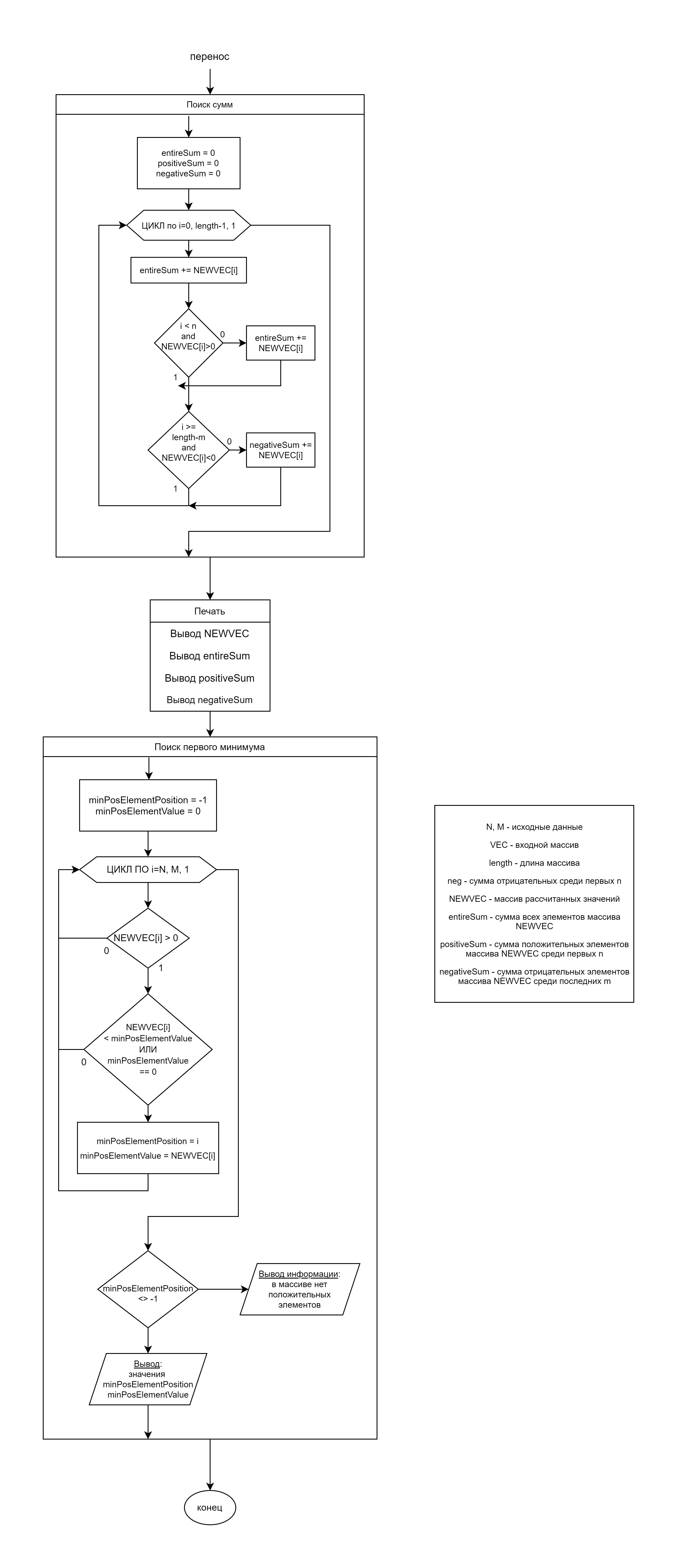
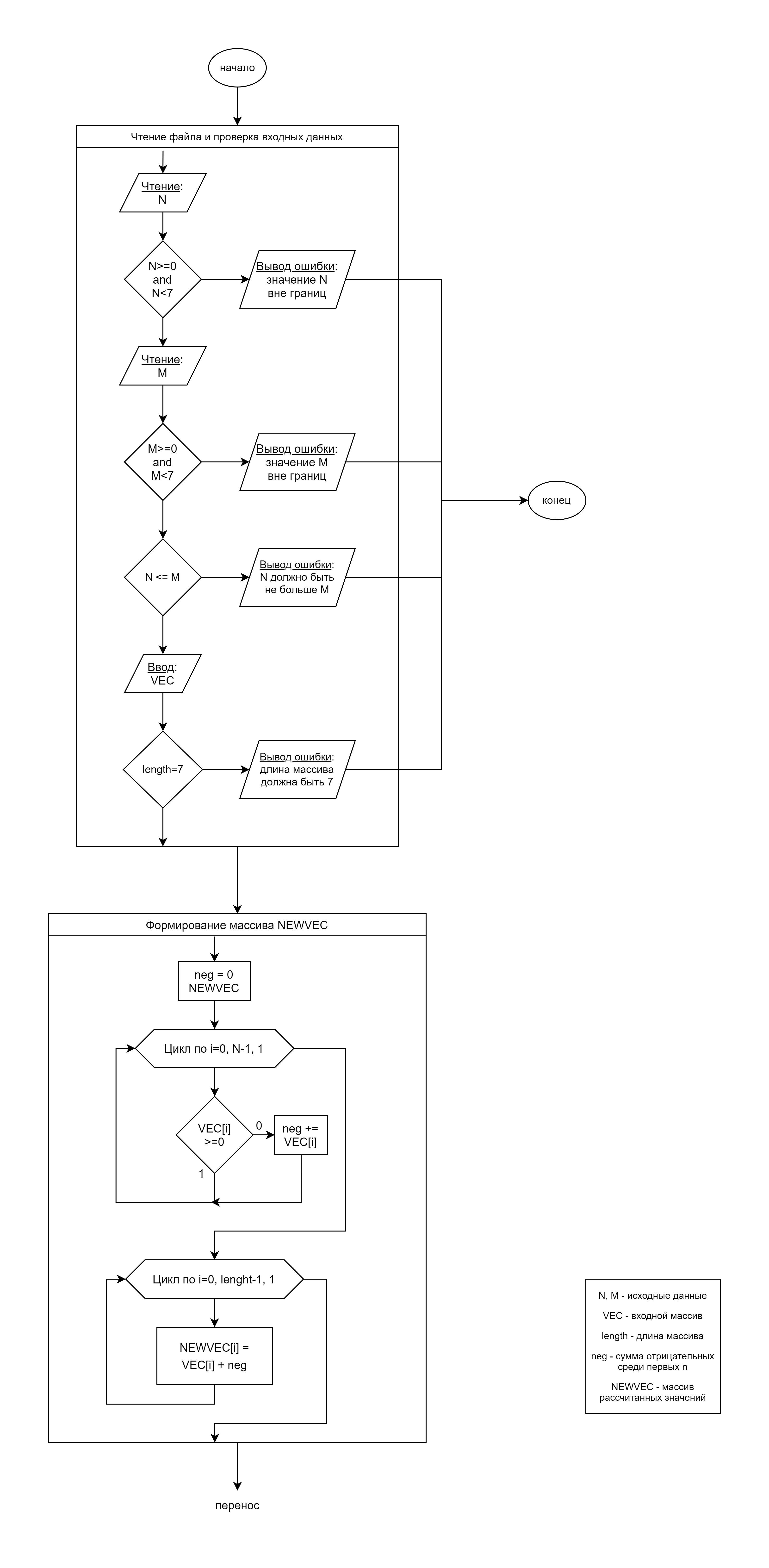
ВАРИАНТ № 9

Файл исходных данных содержит значения величин N, M и элементов массива VEC[0],..VEC[6]. Написать, отладить и протестировать программу, выполняющую:

1. Чтение данных из файла.
2. Формирование нового массива NEWVEC, каждый элемент которого есть сумма соответствующего элемента массива VEC со значением, полученным в результате суммирования отрицательных элементов массива VEC среди первых N.
3. Суммирование всех элементов сформированного массива, положительных элементов среди первых N, отрицательных элементов среди последних M.
4. Печать сформированного массива и значений сумм.
5. Печать индекса наименьшего положительного элемента массива NEWVEC (только в диапазоне индексов от N до M), печать индекса и значения указанного элемента.

Программа должна быть параметризована и обеспечивать входной контроль.

# **Диаграмма алгоритма**



# **Алгоритмический вид программы**

Лабораторная работа 3 по информатике

9 вариант

АЛГ Одномерные массивы

ПЕР normal\_length, length, N, M, VEC, NEWVEC, neg, entireSum, positiveSum, negativeSum, minPosElementPosition, minPosElementValue

НАЧАЛО

/\* ОБЪЯВЛЕНИЕ ПЕРЕМЕННЫХ \*/

normal\_length = 7 // нормальная длина массива

N // первое входное число

M // второе входное число

VEC // входной массив

NEWVEC // создаваемый массив

neg = 0 // сумма отрицательных элементов VEC среди первых N

entireSum = 0 // полная сумма элементов NEWVEC

positiveSum = 0 // сумма отрицательных элементов NEWVEC среди первых N

negativeSum = 0 // сумма положительных элементов NEWVEC среди последних M

minPosElementPosition = -1 // расположение наименьшего положительного элемента

minPosElementValue = 0 // значение наименьшего положительного элемента

/\* КОНЕЦ ОБЪЯВЛЕНИЯ ПЕРЕМЕННЫХ \*/

/\* ЧТЕНИЕ ФАЙЛА И ПРОВЕРКА ДАННЫХ \*/

СОЗДАНИЕ ОБЪЕКТА КЛАССА ifstream И СВЯЗЫВАНИЕ ЕГО С ФАЙЛОМ

ЧТЕНИЕ N ИЗ ФАЙЛА

ЕСЛИ КОНЕЦ ФАЙЛА // ПРОВЕРКА НА НАЛИЧИЕ ДАННЫХ В ФАЙЛЕ

ТО ПЕЧАТЬ (сообщение об ошибке)

КОНЕЦ

КОНЕСЛИ

ЕСЛИ N<0 ИЛИ N>7 // ПРОВЕРКА N

ТО ПЕЧАТЬ (сообщение об ошибке)

КОНЕЦ

КОНЕСЛИ

ЧТЕНИЕ M ИЗ ФАЙЛА

ЕСЛИ M<0 ИЛИ M>7 // ПРОВЕРКА M

ТО ПЕЧАТЬ (сообщение об ошибке)

КОНЕЦ

КОНЕСЛИ

ЕСЛИ N>M

ТО ПЕЧАТЬ (сообщение об ошибке)

КОНЕЦ

КОНЕСЛИ

ОБЪЯВЛЕНИЕ ПЕРЕМЕННОЙ length = 0 ДЛЯ ПОДСЧЕТА КОЛИЧЕСТВА ЭЛЕМЕНТОВ В МАССИВЕ

ПОКА НЕ КОНЕЦ ФАЙЛА // СЧИТЫВАНИЕ ДАННЫХ В МАССИВ VEC И СОХРАНЕНИЕ ЕГО ДЛИНЫ В length

НЦ

ЧТЕНИЕ VEC[length] ИЗ ФАЙЛА

УВЕЛИЧЕНИЕ length на 1

КЦ

КОНПОКА

ЕСЛИ length != normal\_length // ПРОВЕРКА ДЛИНЫ МАССИВА VEC

ТО ПЕЧАТЬ (сообщение об ошибке)

КОНЕЦ

КОНЕСЛИ

ЗАКРЫТИЕ ФАЙЛА

/\* КОНЕЦ ЧТЕНИЯ ФАЙЛА И ПРОВЕРКИ ДАННЫХ \*/

/\* ФОРМИРОВАНИЕ МАССИВА NEWVEC \*/

ДЛЯ i ОТ 0 ДО N-1 // НАХОЖДЕНИЕ СУММЫ ОТРИЦАТЕЛЬНЫХ ЭЛЕМЕНТОВ СРЕДИ ПЕРВЫХ N

НЦ

ЕСЛИ VEC[i] < 0

ТО neg += VEC[i]

КОНЕСЛИ

КЦ

ДЛЯ i ОТ 0 до normal\_length - 1 // НАХОЖДЕНИЕ ЭЛЕМЕНТОВ МАССИВА NEWVEC

НЦ

NEWVEC[i] = VEC[i] + neg

КЦ

/\* КОНЕЦ ФОРМИРОВАНИЯ МАССИВА NEWVEC \*/

/\* ПОИСК СУММ \*/

ДЛЯ i ОТ 0 ДО normal\_length - 1 // ПОСИК СУММ

НЦ

entireSum += NEWVEC[i]

ЕСЛИ i < N И NEWVEC[i] > 0

ТО positiveSum += NEWVEC[i]

КОНЕСЛИ

ЕСЛИ i>=normal\_length - M И NEWVEC[i] < 0

ТО negativeSum += NEWVEC[i]

КОНЕСЛИ

КЦ

/\* КОНЕЦ ПОИСКА СУММ \*/

/\* ПЕЧАТЬ NEWVEC И СУММ \*/

ПЕЧАТЬ МАССИВА NEWVEC

ПЕЧАТЬ entireSum

ПЕЧАТЬ positiveSum

ПЕЧАТЬ negativeSum

/\* КОНЕЦ ПЕЧАТИ NEWVEC И СУММ \*/

/\* ПОИСК НАИМЕНЬШЕГО ПОЛОЖИТЕЛЬНОГО \*/

ДЛЯ i ОТ N ДО M // ПОИСК НАИМЕШЬШЕГО ПОЛОЖИТЕЛЬНОГО ЭЛЕМЕНТА

НЦ

ЕСЛИ NEWVEC[i]>0

ТО ЕСЛИ NEWVEC[i] < minPosElementValue ИЛИ minPosElementValue = 0

ТО minPosElementValue = NEWVEC[i]

minPosElementPosition = i

КОНЕСЛИ

КОНЕСЛИ

КЦ

ЕСЛИ minPosElementPosition != -1 // ПРОВЕРКА НА НАЛИЧИЕ НАИМЕНЬШЕГО ПОЛОЖИТЕЛЬНОГО

ТО ПЕЧАТЬ minPosElementPosition И minPosElementValue

ИНАЧЕ ПЕЧАТЬ (В заданном интервале нет положительных элементов)

/\* КОНЕЦ ПОИСКА НАИМЕНЬШЕГО ПОЛОЖИТЕЛЬНОГО \*/

КОНЕЦ

# **Исходный код программы**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\

\* КАФЕДРА № 304 1 КУРС CPP ЛАБОРАТОРНАЯ РАБОТА \*

\*----------------------------------------------------------------\*

\* Project Type : Win32 Console Application \*

\* Project Name : laba2 \*

\* File Name : laba2.cpp \*

\* Language : C/C++ \*

\* Programmer(s) : Багиров Э., Нуриев Н. \*

\* Modifyed By : \*

\* Created : 11/10/21 \*

\* Last Revision : 11/24/21 \*

\* Comment(s) : Код к ЛР№2 - "Табулирование функций" \*

\\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream>

#include <iomanip>

using namespace std;

// псевдографика - заголовок таблицы

void HeadTable() {

cout << "\t"

<< char(218) << setfill(char(196)) << setw(20)

<< char(194) << setfill(char(196)) << setw(20)

<< char(194) << setfill(char(196)) << setw(20)

<< char(194) << setfill(char(196)) << setw(20)

<< char(191) << endl;

cout << "\t"

<< char(179) << setfill(' ') << setw(19)<< "#"

<< char(179) << setfill(' ') << setw(19)<< "x"

<< char(179) << setfill(' ') << setw(19)<< "F(x)"

<< char(179) << setfill(' ') << setw(19)<< "G(x)"

<< char(179) << endl;

cout << "\t"

<< char(195) << setfill(char(196)) << setw(20)

<< char(197) << setfill(char(196)) << setw(20)

<< char(197) << setw(20)

<< char(197) << setw(20) << char(180) << endl;

cout.setf(ios::showpoint | ios::fixed);

}

// псевдографика - завершение таблицы

void BottomTable() {

cout << "\t"

<< char(192) << setfill(char(196)) << setw(20)

<< char(193) << setfill(char(196)) << setw(20)

<< char(193) << setfill(char(196)) << setw(20)

<< char(193) << setfill(char(196)) << setw(20)

<< char(217) << endl;

}

// псевдографика - тело таблицы

void BetweenTheRaws() {

cout << "\t"

<< char(195) << setfill(char(196)) << setw(20)

<< char(197) << setfill(char(196)) << setw(20)

<< char(197) << setfill(char(196)) << setw(20)

<< char(197) << setfill(char(196)) << setw(20)

<< char(180) << endl;

}

// имплемантация возведения числа в степень

double power(double a, int n) {

double b = a;

for (int i = 1; i < n; i++) {

b \*= a;

}

return b;

}

// расчет значения функции F(x) в точке x

double F(double x) {

return power(x - 1, 3);

}

// расчет значения функции G(x) в точке x

double G(double x) {

return power(x + 5, 3) / (1 + power(sin(x), 2));

}

// вывод строк таблицы с автоматическим закрытием таблицы

void PrintTable(int n, double x, double f, double g, bool last) {

cout << "\t" << char(179) << setw(19) << setfill(char(255)) << n

<< char(179) << setw(19) << setprecision(4) << x

<< setfill(char(255))

<< char(179) << setw(19) << setprecision(4) << f

<< setfill(char(255))

<< char(179) << setw(19) << setprecision(4) << g

<< setfill(char(255)) << char(179) << endl;

if (last) {

BottomTable();

}

else {

BetweenTheRaws();

}

}

int main()

{

setlocale(LC\_ALL, "Russian"); // подключение русского языка

system("color F0");

//Инициализация переменных

double A; // начало отрезка

double B; // конец отрезка

int N; // количество интервалов, на которое делится отрезок [A;B]

double h; // шаг

double x; // аргумент приращения

int i; // индекс приращения

// Ввод и проверка данных

cout << "Введите А - координату начала отрезка [A;B]: \t";

cin >> A;

cout << "Введите B - координату конца отрезка [A;B]: \t";

cin >> B;

if (A > B) {

cout << "\nОШИБКА! А должен быть меньше или равен В!";

return 1; // выход из программы возвращением кода 1

}

if (A != B) {

cout << "Введите N - количество разделений отрезка [A;B]: \t";

cin >> N;

if (N <= 0) {

cout << "\nОШИБКА! N должно быть больше 0! ";

return 2; // выход из программы возвращением кода 2

}

h = (B - A) / N; // Расчет величины шага

}

else {

// Если входной отрезок есть точка, изменить количество разделений

N = 0;

h = 0;

}

cout << "Величина шага h = " << h << "\n";

// Подготовка таблицы

setlocale(LC\_ALL, "C"); // отключение русского языка

HeadTable(); // вывод заголовка таблицы

for (i = 0; i <= N; i++) { // посторная печать таблицы в цикле

x = A + i \* h; // вычисление инкремируемой переменной

double Fx = F(x); // расчет значения функции F(x) в точке x

double Gx = G(x); // расчет значения функции G(x) в точке x

bool last = i == N; // флаг последней итерации цикла

// функции-принтеру передаются значения:

// i - индекс итерации

// x - аргумент приращения

// Fx - значение функции F(x) в точке x

// Gx - значение функции G(x) в точке x

// last - флаг последней итерации цикла

PrintTable(i, x, Fx, Gx, last);

}

return 0; // выход из программы возвращением кода 0

}

# **Некорректные тесты**

**Тест №1**

Цель теста: проверить работу программы при пустом файле

Исходные данные: (пустой файл)

Ожидаемый результат: вывод сообщения об ошибке:“ Ошибка: файл пуст.”

Полученный результат:



Вывод: полученный результат совпал с ожидаемым. Тест ошибки не обнаружил.

**Тест №2**

Цель теста: проверить работу программы при пустом файле

Исходные данные: “ф а 1 2 3 4 5 6 7”

Ожидаемый результат: вывод сообщения об ошибке:“ Ошибка: сбой при чтении файла.”

Полученный результат:



Вывод: полученный результат совпал с ожидаемым. Тест ошибки не обнаружил.

**Тест №3**

Цель теста: проверить работу программы при пустом файле

Исходные данные: 10 5 1 2 3 4 5 6 7

Ожидаемый результат: вывод сообщения об ошибке:“ Ошибка: значение N вне границ.”

Полученный результат:



Вывод: полученный результат совпал с ожидаемым. Тест ошибки не обнаружил.

**Тест №4**

Цель теста: проверить работу программы при пустом файле

Исходные данные: 5 10 1 2 3 4 5 6 7

Ожидаемый результат: вывод сообщения об ошибке:“ Ошибка: значение M вне границ.”

Полученный результат:



Вывод: полученный результат совпал с ожидаемым. Тест ошибки не обнаружил.

**Тест №5**

Цель теста: проверить работу программы при пустом файле

Исходные данные: 6 5 1 2 3 4 5 6 7

Ожидаемый результат: вывод сообщения об ошибке:“ Ошибка: N должно быть не больше M.”

Полученный результат:



Вывод: полученный результат совпал с ожидаемым. Тест ошибки не обнаружил.

**Тест №6**

Цель теста: проверить работу программы при пустом файле

Исходные данные: 5 6 1 2 3 4 5 6

Ожидаемый результат: вывод сообщения об ошибке:“ Ошибка: длина массива должна быть равна 7. “

Полученный результат:



Вывод: полученный результат совпал с ожидаемым. Тест ошибки не обнаружил.

**Тест №7**

Цель теста: проверить работу программы при пустом файле

Исходные данные: (пустой файл)

Ожидаемый результат: вывод сообщения об ошибке:“ Ошибка: файл пуст.”

Полученный результат:



Вывод: полученный результат совпал с ожидаемым. Тест ошибки не обнаружил.

# **Корректные тесты**

**Тест №1**

Цель теста: проверить работу программы на границе корректной области исходных данных

Исходные данные: A = 4, B = 4

Ожидаемый результат: вывод значения функций F и G в точке x=4

Ожидаемые значения функций:

F(4) = (4-1)^3 = 3^3 = 27

G(4) = (4+5)^3 / (1+sin^2(4))= 9^3 / (1+0.0698^2) = 729/1.57275 = 463.5193

Полученный результат:

Изображение выглядит как текст, стол

Автоматически созданное описание

Вывод: полученный результат совпал с ожидаемым. Тест ошибки не обнаружил.

**Тест №2**

# **Вывод**

Проанализирован алгоритм по разбитию отрезка на интервалы, вычислению значения функций в найденных концах интервалов, форматированию выходных данных в виде таблицы. В ходе выполнения лабораторной работы были составлены блок-схема и псевдокод для дальнейшего написания программы. Проведены тесты для проверки соответствия исходных данных и корректные тесты. В результате проведения тестов ошибок в программе не обнаружено. Разработка программы завершена на основании того, что полученные результаты совпали с ожидаемыми, набор тестов считаем полным.