АБОРАТОРНАЯ РАБОТА №3

ИССЛЕДОВАНИЕ АЛГОРИТМОВ ОБРАБОТКИ ОДНОМЕРНЫХ СТАТИЧЕСКИХ МАССИВОВ

1. Цель работы:

Исследование способов представления массивов в памяти ЭВМ, получение практических навыков реализации алгоритмов обработки одномерных массивов.

2. Постановка задачи и вариант задания:

Вариант 5: В одномерном массиве, состоящем из N вещественных элементов, вычислить:

1) максимальный элемент массива;

2) сумму элементов массива, расположенных до последнего положительного элемента.

Заменить все элементы, модуль которых находится внутри отрезка [a, b] на число a и упорядочить элементы массива по возрастанию, используя алгоритм сортировки методом вставки. Значение a и b вводит пользователь.

3. Краткие теоретические сведения

Для выполнения лабораторной работы необходимо изучить особенности представления и обработки одномерных массивов в языках C/C++, а также алгоритмы сортировки массивов методами прямого обмена, вставки и прямого выбора.

ХОД РАБОТЫ

4. Структурная схема алгоритма

Структурная схема алгоритма представлена на рисунке 4.1.

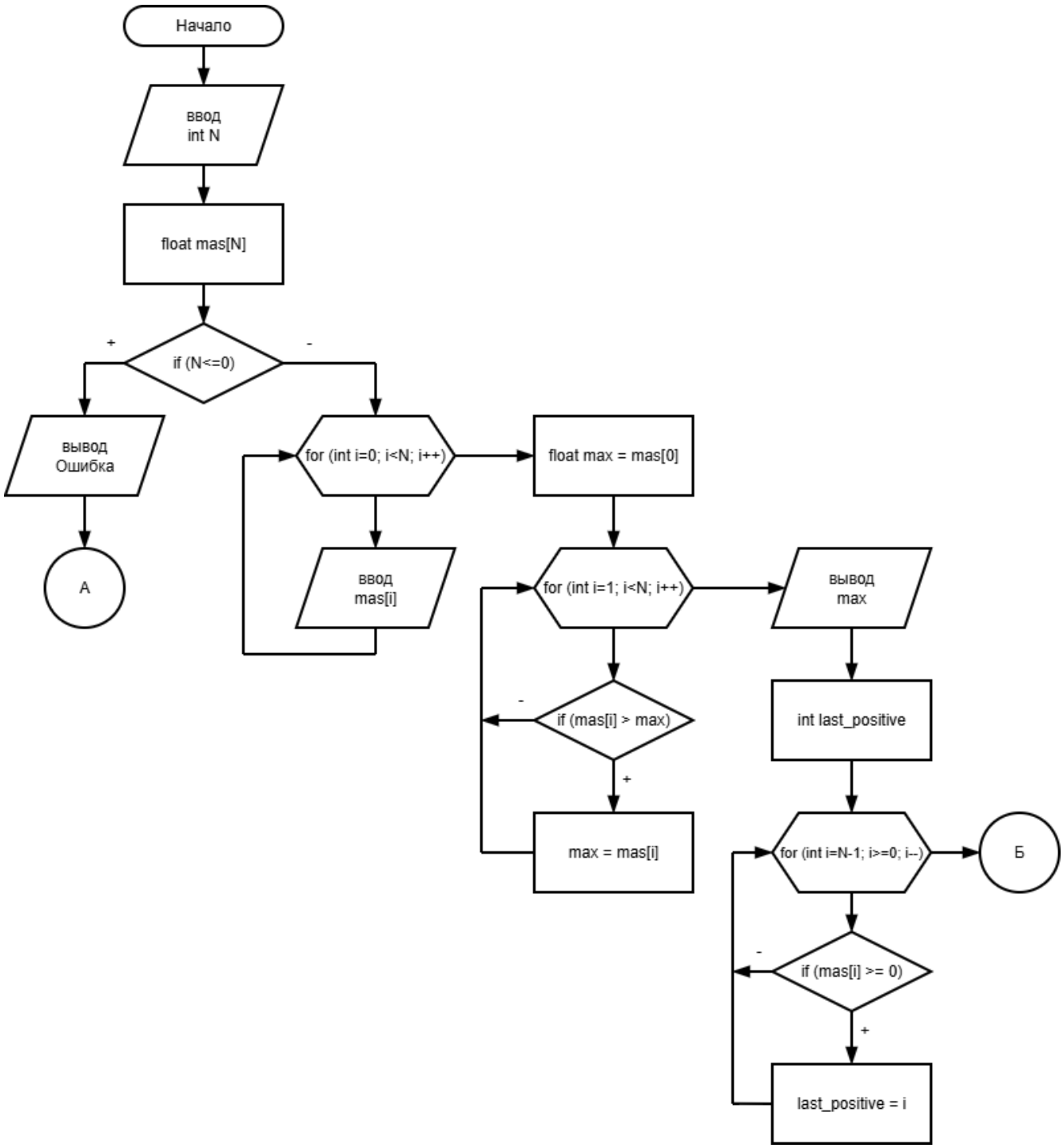

Рисунок 3.1 – Блок схема

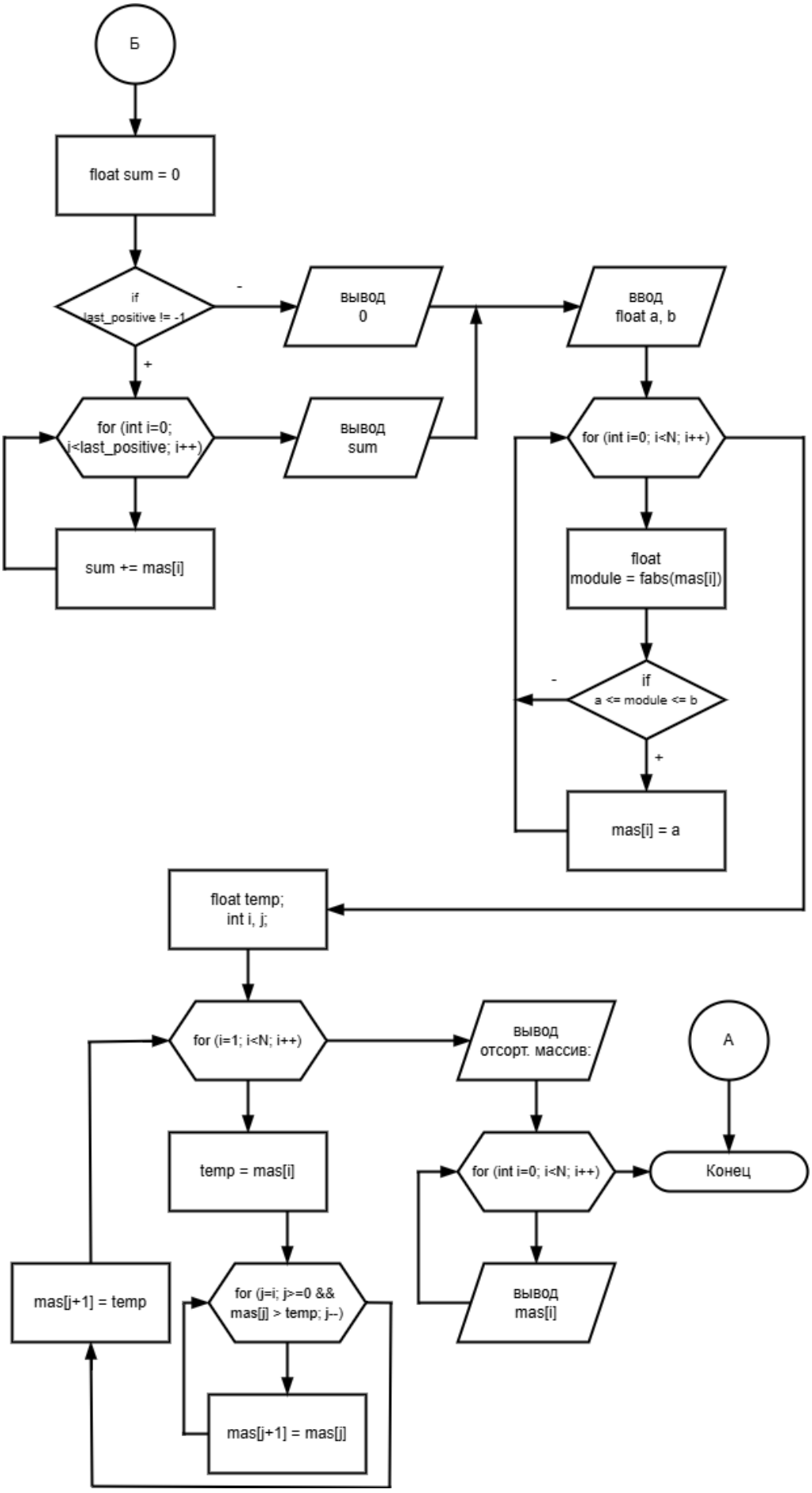

Рисунок 3.2 – Блок-схема (Лист 2)

5. Текст кода на языке С. Тестирование и отладка программы

```
#include <stdio.h>
#include <math.h>

int main() {
    int N;
    printf("Введите размер массива N: ");
    scanf("%d \n", &N);
    float mas[N], c;

    if (N <= 0)
    {
        printf("Число элементов (N) не может быть <= 0");
        return 1;
    }

    for (int i = 0; i < N; i++) {
        scanf("%f", &mas[i]);
    }

    float max = mas[0];
    for (int i = 1; i < N; i++) {
        if (mas[i] > max) {
            max = mas[i];
        }
    }
    printf("Максимальный элемент: %.2f\n", max);

    int last_positive = -1;
    for (int i = N - 1; i >= 0; i--) {
        if (mas[i] >= 0) {
            last_positive = i;
            break;
        }
    }
    float sum = 0;
    if (last_positive != -1) {
        for (int i = 0; i < last_positive; i++) {
            sum += mas[i];
        }
        printf("Сумма элементов до последнего положительного:
%.2f\n", sum);
    } else {
        printf("Положительных элементов нет. Сумма: 0\n");
    }

    float a, b;
    printf("Введите значение A и B: ");
    scanf("%f %f", &a, &b);
    if (a > b)
    {
        printf("Значение A не может быть больше значения B");
        return 1;
```

```
        }
        for (int i = 0; i < N; i++)
        {
            float module = fabs(mas[i]);
            if (a <= module && module <= b)
            {
                mas[i] = a;
            }
        }
        printf("Сумма элементов массива, модуль которых входит в
отрезок [%.2f; %.2f]: %.2f \n", a, b, sum);

        float temp;
        int i, j;
        for(i=1; i<N; i++) {
            temp=mas[i];
            for(j=i-1;(j>=0)&&(mas[j]>temp);j--) {
                mas[j+1]=mas[j];
            }
            mas[j+1]=temp;
        }
        printf("Отсортированный массив после замены:\n");
        for (int i = 0; i < N; i++) {
            printf("%.2f; ", mas[i]);
        }
        return 0;
    }
```

На рисунке 3.3 отображены результаты работы кода согласно всем условиям по заданию для языка С.

```
Введите размер массива N: 10
1
2
3
4
66
-66
13
-11
-2
-3
Максимальный элемент: 66.00
Сумма элементов до последнего положительного: 10.00
Введите значение A и B: 2 7
Сумма элементов массива, модуль которых входит в отрезок [2.00; 7.00]: 10.00
Отсортированный массив после замены:
-66.00; -11.00; 1.00; 2.00; 2.00; 2.00; 2.00; 2.00; 13.00; 66.00;
```

Рисунок 3.3 – Результат работы программы на языке С

6. Текст кода на языке C++. Тестирование и отладка программы

```
#include <iostream>
#include <math.h>

using namespace std;

int main()
{
    int N;
    cout << "Введите размер массива N: ";
    cin >> N;
    float mas[N], c;

    if (N <= 0)
    {
        cout << "Число элементов (N) не может быть <= 0" << endl;
        return 1;
    }

    for (int i = 0; i < N; i++)
    {
        cin >> mas[i];
    }

    float max = mas[0];
    for (int i = 1; i < N; i++)
    {
        if (mas[i] > max)
        {
            max = mas[i];
        }
    }
    cout << "Максимальный элемент: " << max << endl;

    int last_positive = -1;
    for (int i = N - 1; i >= 0; i--)
    {
        if (mas[i] >= 0)
        {
            last_positive = i;
        }
    }
    float sum = 0;
    if (last_positive != -1)
    {
        for (int i = 0; i < last_positive; i++)
        {
            sum += mas[i];
        }
        cout << "Сумма элементов до последнего положительного: "
<< sum << endl;
    }
    else
```

```
        {
            cout << "Положительных элементов нет. Сумма: 0" << endl;
        }

        float a, b;
        cout << "Введите значение A: " << endl;
        cin >> a;
        cout << "Введите значение B: " << endl;
        cin >> b;
        if (a > b)
        {
            cout << "Значение A не может быть больше значения B" <<
endl;
            return 1;
        }
        for (int i = 0; i < N; i++)
        {
            float module = fabs(mas[i]);
            if (a <= module && module <= b)
            {
                mas[i] = a;
            }
        }

        float temp;
        int i, j;
        for (i = 1; i < N; i++)
        {
            temp = mas[i];
            for (j = i - 1; (j >= 0) && (mas[j] > temp); j--)
            {
                mas[j + 1] = mas[j];
            }
            mas[j + 1] = temp;
        }
        cout << "Отсортированный массив после замены:" << endl;
        for (int i = 0; i < N; i++)
        {
            cout << mas[i] << ";  ";
        }
        return 0;
    }
```

На рисунке 3.4 отображены результаты работы кода согласно трем условиям по заданию для языка C++.

```
Введите размер массива N: 10
1
2
3
4
66
-66
13
-11
-2
-3
Максимальный элемент: 66
Сумма элементов до последнего положительного: 0
Введите значение A:
2
Введите значение B:
7
Отсортированный массив после замены:
-66;  -11;  1;  2;  2;  2;  2;  2;  13;  66;
```

Рисунок 3.4 – Результат работы программы на языке C++

ВЫВОД

В ходе лабораторной работы были исследованы способы представления и обработки одномерных статических массивов. На практике были успешно реализованы базовые алгоритмы, включая поиск, сортировку и фильтрацию элементов. Экспериментально подтверждено, что работа с массивами основана на использовании индексов для прямого доступа к элементам. Полученные навыки закрепили понимание принципов эффективного управления данными в памяти.