

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ  
федеральное государственное автономное образовательное учреждение высшего образования  
"Омский государственный технический университет"

Кафедра «Автоматизированные системы обработки информации и управления»

## **ПРАКТИЧЕСКАЯ РАБОТА №2**

**по дисциплине**

**«Измерительные средства аналитики программных систем и технологий»**

**по теме**

**«Метрики Мак-Кейба.»**

Выполнил:	Шмидт А.В. гр. ИВТ - 244
Проверил:	Зубарев А.А. к.т.н., доцент

Омск 2025

## **ЗАДАНИЕ**

1. Выбрать программу, в которой **ОБЯЗАТЕЛЬНО** должны быть **ВЕТВЛЕНИЯ**, или **ЦИКЛЫ** (или то и другое). Представить графически схему алгоритма программы.
2. Представить поток управления в виде графа, сделав оценку свойств передачи управления в программе («дерево обязательного предшествования»).
3. Определить цикломатическую сложность графа потока управления.
4. Сделать выводы по работе.

## ПРИМЕР ОПРЕДЕЛЕНИЯ МЕТРИК ПО МАК-КЕЙБУ

Программа на C++ считывает количество элементов массива, запрашивает у пользователя значения и сохраняет их во внутренний динамический массив. Для каждого числа программа определяет знак, чётность и вычисляет сумму его цифр с помощью отдельной подпрограммы. В процессе обработки формируются статистические показатели: количество положительных, отрицательных, чётных и нечётных элементов. После завершения цикла программа выводит сводную статистику и сравнивает, каких чисел больше — положительных или отрицательных. В финале консоль ожидает нажатия клавиши, что предотвращает её мгновенное закрытие. Ниже представлен листинг программы.

```
#include <iostream>
#include <vector>
using namespace std;

int sumOfDigits(int x) {
    x = abs(x);
    int s = 0;
    while (x > 0) {
        s += x % 10;
        x /= 10;
    }
    return s;
}

int main() {
    int n;
    cout << "Enter amount: ";
    cin >> n;

    vector<int> arr(n);

    for (int i = 0; i < n; i++) {
        cout << "Enter element " << i + 1 << ": ";
        cin >> arr[i];
    }

    int positiveCount = 0;
    int negativeCount = 0;
    int evenCount = 0;
    int oddCount = 0;

    for (int i = 0; i < n; i++) {
        int val = arr[i];

        if (val > 0) {
            positiveCount++;
        } else if (val < 0) {
```

```

        negativeCount++;
    }

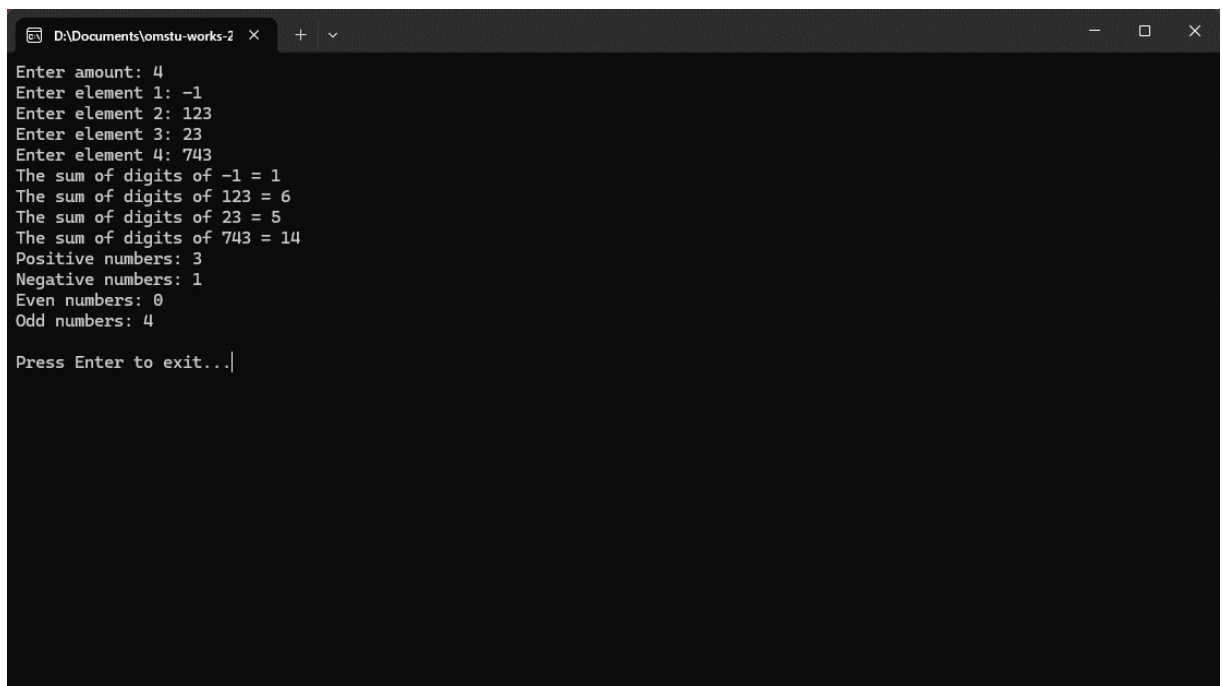
    if (val % 2 == 0) {
        evenCount++;
    } else {
        oddCount++;
    }

    int s = sumOfDigits(val);
    cout << "The sum of digits of " << val << " = " << s << "\n";
}

cout << "Positive numbers: " << positiveCount << "\n";
cout << "Negative numbers: " << negativeCount << "\n";
cout << "Even numbers: " << evenCount << "\n";
cout << "Odd numbers: " << oddCount << "\n";

cout << "\nPress Enter to exit...";
cin.ignore();
cin.get();
return 0;
}

```



```

D:\Documents\omstu-works-2
Enter amount: 4
Enter element 1: -1
Enter element 2: 123
Enter element 3: 23
Enter element 4: 743
The sum of digits of -1 = 1
The sum of digits of 123 = 6
The sum of digits of 23 = 5
The sum of digits of 743 = 14
Positive numbers: 3
Negative numbers: 1
Even numbers: 0
Odd numbers: 4
Press Enter to exit...

```

Рисунок 1 – Результат работы программы.

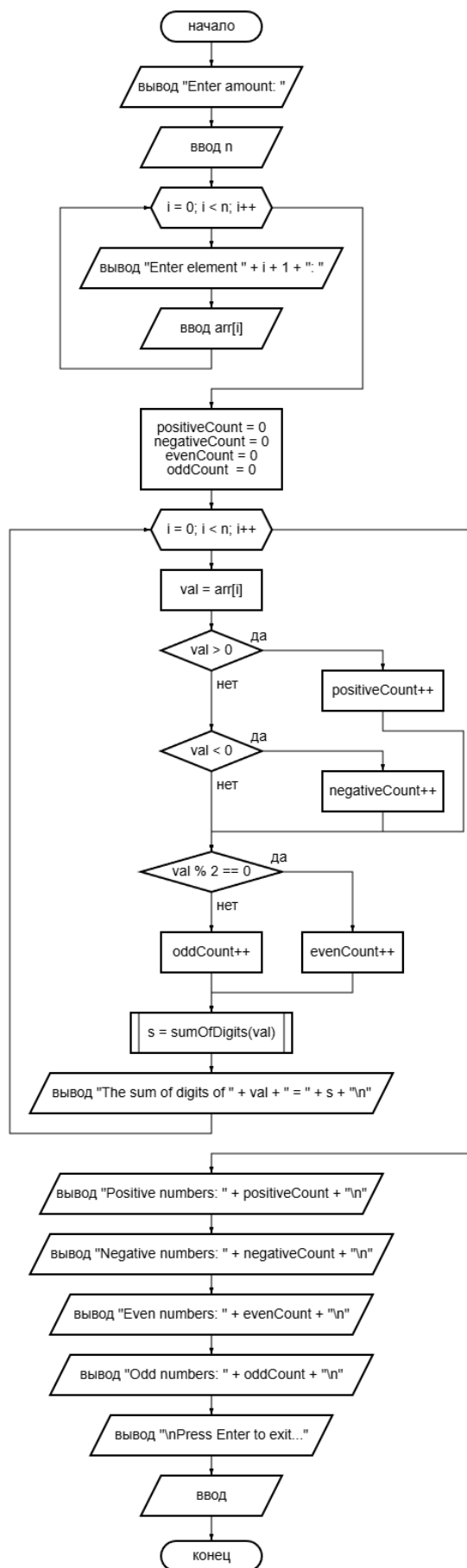


Рисунок 2 – Схема алгоритма главной функции.

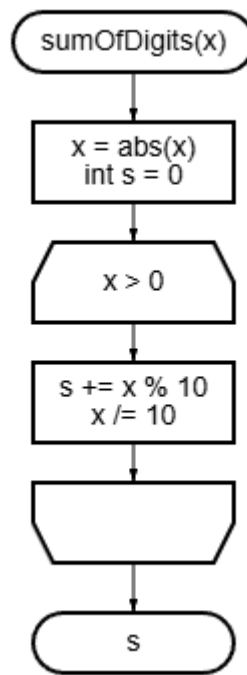


Рисунок 3 – Схема алгоритма подпрограммы для подсчёта суммы цифр.

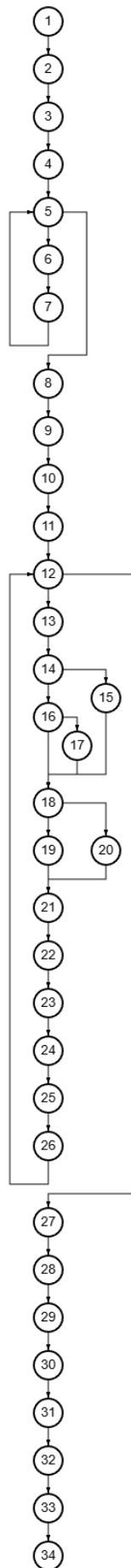


Рисунок 4 – Граф потока управления.

Таблица 1 – Пояснение к шагам графа.

Номер шага графа	Действие шага
1	<code>int n;</code>
2	<code>cout &lt;&lt; "Enter amount: ";</code>
3	<code>cin &gt;&gt; n;</code>
4	<code>vector&lt;int&gt; arr(n);</code>
5	<code>for (int i = 0; i &lt; n; i++)</code>
6	<code>cout &lt;&lt; "Enter element " &lt;&lt; i + 1 &lt;&lt; ": ";</code>
7	<code>cin &gt;&gt; arr[i];</code>
8	<code>int positiveCount = 0;</code>
9	<code>int negativeCount = 0;</code>
10	<code>int evenCount = 0;</code>
11	<code>int oddCount = 0;</code>
12	<code>for (int i = 0; i &lt; n; i++)</code>
13	<code>int val = arr[i];</code>
14	<code>if (val &gt; 0)</code>
15	<code>positiveCount++;</code>
16	<code>else if (val &lt; 0)</code>
17	<code>negativeCount++;</code>
18	<code>if (val % 2 == 0)</code>
19	<code>evenCount++;</code>
20	<code>oddCount++;</code>
21	<code>x = abs(x);</code>
22	<code>int s = 0;</code>
23	<code>while (x &gt; 0)</code>
24	<code>s += x % 10;</code>
25	<code>x /= 10;</code>
26	<code>cout &lt;&lt; "The sum of digits of " &lt;&lt; val &lt;&lt; " = " &lt;&lt; s &lt;&lt; "\n";</code>
27	<code>cout &lt;&lt; "Positive numbers: " &lt;&lt; positiveCount &lt;&lt; "\n";</code>
28	<code>cout &lt;&lt; "Negative numbers: " &lt;&lt; negativeCount &lt;&lt; "\n";</code>
29	<code>cout &lt;&lt; "Even numbers: " &lt;&lt; evenCount &lt;&lt; "\n";</code>
30	<code>cout &lt;&lt; "Odd numbers: " &lt;&lt; oddCount &lt;&lt; "\n";</code>
31	<code>cout &lt;&lt; "\nPress Enter to exit...";</code>
32	<code>cin.ignore();</code>
33	<code>cin.get();</code>



Продолжение таблицы 2 – Пояснение к шагам графа.

34	return 0;
----	-----------

Формула для вычисления цикломатического числа Мак-Кейба  
(цикломатической сложности)  $Z(G)$ :

$$Z(G) = e - v + 2p, \text{ где:}$$

$e$  – количество рёбер в графе потока управления;

$v$  – количество узлов (вершин);

$p$  – количество связных компонент (здесь  $p = 1$  — одна программа).

В нашем случае имеем:

$$Z(G) = 38 - 34 + 2 \cdot 1 = 6$$

## ВЫВОД

В ходе работы проанализирована программа на C++, которая принимает набор чисел, определяет для каждого знак, чётность и сумму цифр, подсчитывает статистику по введённым значениям и выводит итоговое сравнение. Построен граф потока управления. По формуле Мак-Кейба  $Z(G) = e - v + 2p$  цикломатическая сложность программы составила 6 ( $p = 1$ ).

Значение не превышает порог 10, что указывает на нормальную сложность и программа не требует рефакторинга.

Метрика Мак-Кейба подтвердила свою актуальность: она наглядно показала минимальное количество тестовых прогонов (6), необходимых для полного покрытия всех ветвей и контуров программы, и позволила объективно оценить структурную сложность кода.