



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
"МИРЭА - Российский технологический университет"

РТУ МИРЭА

Институт информационных технологий (ИТ)
Кафедра математического обеспечения и стандартизации информационных
технологий (МОСИТ)

ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №1_1
по дисциплине

«Структуры и алгоритмы обработки данных»

Тема. Вычислительная сложность алгоритма

Выполнил студент группы ИКБО-63-23		Зуев Д.А.
Принял старший преподаватель		Скворцова Л.А.

Москва 2025

Оглавление

1 УСЛОВИЕ ЗАДАНИЯ.....	3
1.1 Задание.....	3
1.2 Требование к выполнению задания.....	3
1.3 Варианты работы.....	3
2 Разработка решения.....	4
2.1 Таблица, заполненная согласно требованиям задания.....	4
2.2 Функциональные зависимости, полученные в результате анализа алгоритма. Описание получения функций для различных случаев, если для алгоритма они должны быть рассмотрены.....	5
2.3 Код алгоритма на языке C++.....	5
2.4 Разработанные тесты.....	7
2.5 Скрины результатов тестирования алгоритма.....	7
2.6 Результаты исследования алгоритма (тестирования) на различных объемах данных и получение времени его выполнения.....	10
2.7 Анализ выполнения алгоритма.....	10

1. УСЛОВИЕ ЗАДАНИЯ

1.1 Задание

Разработать алгоритм задачи варианта. Определить функцию, показывающую зависимость количества выполняемых инструкций от размера задачи, функцию емкостной сложности алгоритма.

1.2 Требования к выполнению задания

1. Выполнить разработку алгоритма задачи варианта, представляя последовательность как массив из n значений, записать алгоритм на псевдокоде.
2. Определить, для полученного алгоритма, функциональную зависимость (функцию), указывающей зависимость количества выполняемых инструкций от размера задачи
3. Технологию подсчета количества инструкций алгоритма представить в таблице табл. 1. При этом: – псевдокод алгоритма разместить в столбце *Инструкции (оператор) алгоритма* таблицы табл. 1, каждая управляющая инструкция строго в отдельной строке таблицы;
– в строке столбца *Количество выполнений инструкции* разместить формулу (или значение), определяющую (определяющее) количество выполнений инструкции на заданном размере задачи (n).
4. Определить функцию (функции: *наилучший, наихудший и средний случаи*) зависимости количества инструкций алгоритма от размера задачи и от данных. Для этого выполнить суммарный подсчет всех значений столбца *Количество выполнений инструкции*, учитывая влияние данных на количество выполняемых инструкций.
5. Реализовать алгоритм.
6. Разработать тесты для доказательства корректной работы алгоритма при $n=20$.
7. Выполнить отладку и тестирование алгоритма.
8. Определить емкостную сложность алгоритма.

1.3 Вариант работы

Дано натуральное число n и последовательность действительных чисел x_1, \dots, x_n . Получить значение выражения $(1+r)/(1+s)$ где r – сумма тех чисел последовательности, которые больше 1, а s – сумма тех чисел последовательности, которые не превышают 1.

2. РАЗРАБОТКА РЕШЕНИЯ

2.1 Таблица, заполненная согласно требованиям задания

Таблица 1. Форма представления алгоритма при получении функции зависимости количества выполняемых инструкций от размера задачи

Номер Строки Инструкции алгоритма	Инструкция (оператор) алгоритма	Количество выполнений инструкции
1	double r = 0	1
2	double s = 0	1
3	for (double num : x)	n + 1
4	if (num > 1)	n
5	r += num	$\sum_{i=1}^n [x[i] > 1]$
6	else	$n - \sum_{i=1}^n [x[i] > 1]$
7	s += num	$\sum_{i=1}^n [x[i] \leq 1]$
8	double result = (1 + r) / (1 + s)	1
9	return result	1

calculateExpression		
1	int n;	1
2	cout << "Введите количество чисел (n): ";	1
3	cin >> n;	1
4	if (n <= 0)	1
5	cout << "Ошибка: количество чисел должно быть положительным.";	1
6	return 1;	1
7	vector<double> x(n);	1
8	cout << "Введите последовательность из " << n << " чисел:";	1
9	for (int i = 0; i < n; ++i)	n+1n+1

10	<code>cout << "x[" << i + 1 << "]:</code> <code>";</code>	<code>n</code>
11	<code>cin >> x[i];</code>	<code>n</code>
12	<code>double result =</code> <code>calculateExpression(x);</code>	<code>1</code>
13	<code>cout << "Результат: " <<</code> <code>result;</code>	<code>1</code>
14	<code>return 0;</code>	<code>1</code>

Получим сложность для функции `calculateExpression`: $T(n) = 2 + (n + 1) + n + n + 2 = 4n + 5$

Получим сложность для функции `main`: $T(n) = 3 + 1 + (n + 1) + 2n + 2 = 3n + 7$

Получим функцию роста, сложим `main` и `calculateExpression`: $T(n) = (4n + 5) + (3n + 7) = 7n + 12$

2.2 Функциональные зависимости, полученные в результате анализа алгоритма. Описание получения функций для различных случаев, если для алгоритма они должны быть рассмотрены

В лучшем случае сложность алгоритма будет равна: $T(n) = 7n + 12 - n = 6n + 12$, так как если все числа больше 1, то в функции `calculateExpression` операция `s += num` не выполняется.

В худшем случае сложность алгоритма будет равна: $T(n) = 7n + 12$ так как если все числа не превышают 1, то в функции `calculateExpression` операция `r += num` не выполняется.

В среднем случае сложность алгоритма будет равна: $T(n) = 7n + 12$ так как если примерно половина чисел больше 1, а половина не превышает 1.

2.3 Код алгоритма на языке C++

На рисунках 1 и 2 приведена реализация алгоритма на C++.

Примечание: в данной реализации сложность алгоритма будет постоянной $O(n)$, так как в коде основной цикл, который проходит по всем элементам массива `x`, выполняется `n` раз (где `n` — количество элементов в массиве). Внутри цикла выполняются операции, которые занимают константное время (например, проверка условия `if (num > 1)` и сложение `r += num` или `s += num`). Таким образом, общее количество операций в цикле пропорционально `n`.

```

#include <iostream>
#include <vector>

using namespace std;

double calculateExpression(const vector<double>& x) {
    double r = 0;
    double s = 0;
    for (double num : x) {
        if (num > 1) {
            r += num;
        } else {
            s += num;
        }
    }

    double result = (1 + r) / (1 + s);
    return result;
}

```

Рисунок 1 - реализация алгоритма на C++

```

int main() {
    int n;
    cout << "Введите количество чисел (n): ";
    cin >> n;
    vector<double> x(n);
    cout << "Введите последовательность из " << n << " чисел:" << endl;
    for (int i = 0; i < n; ++i) {
        cout << "x[" << i + 1 << "]: ";
        cin >> x[i];
    }
    double result = calculateExpression(x);
    cout << "Результат: " << result << endl;
    return 0;
}

```

Рисунок 2 - реализация алгоритма на C++

2.4 Разработанные тесты

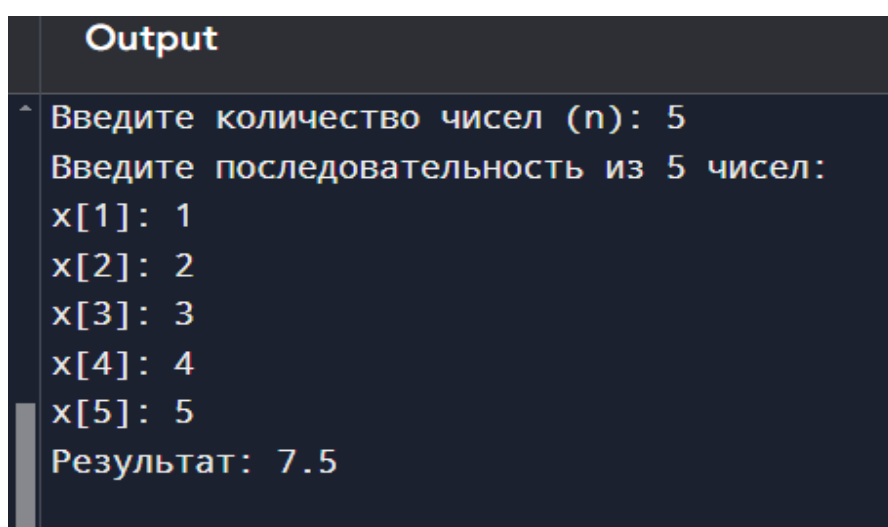
Таблица - 2. Таблица тестов алгоритма

calculateExpression

Номер теста	Входные данные	Эталон результата (ожидаемые результаты)
1	$n=5, x=\{1, 2, 3, 4, 5\}$	7.5
2	$n=20, x=\{1, 2, 3, \dots, 20\}$	105
3	$n=10, x=\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$	27.5
4	$n=25, x=\{1, 2, 3, \dots, 25\}$	162.5
5	$n=50, x=\{1, 2, 3, \dots, 50\}$	637.5

2.5 Скрины результатов тестирования алгоритма

На рисунках 3-7 приведены скрины результатов тестирования программы.

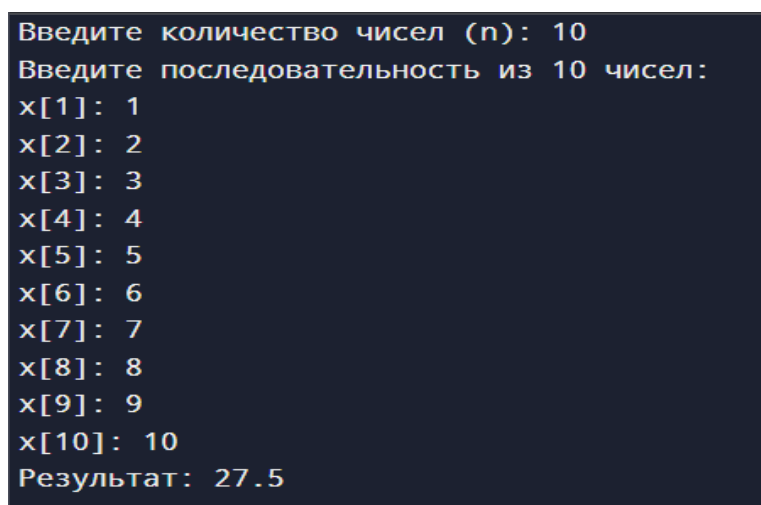


```

Output
^ Введите количество чисел (n): 5
  Введите последовательность из 5 чисел:
  x[1]: 1
  x[2]: 2
  x[3]: 3
  x[4]: 4
  x[5]: 5
  Результат: 7.5

```

Рисунок 3 - тест номер 1



```

Введите количество чисел (n): 10
Введите последовательность из 10 чисел:
x[1]: 1
x[2]: 2
x[3]: 3
x[4]: 4
x[5]: 5
x[6]: 6
x[7]: 7
x[8]: 8
x[9]: 9
x[10]: 10
Результат: 27.5

```

Рисунок 4 - тест номер 2

```
Output
Введите количество чисел (n): 20
Введите последовательность из 20 чисел:
x[1]: 1
x[2]: 2
x[3]: 3
x[4]: 4
x[5]: 5
x[6]: 6
x[7]: 7
x[8]: 8
x[9]: 9
x[10]: 10
x[11]: 11
x[12]: 12
x[13]: 13
x[14]: 14
x[15]: 15
x[16]: 16
x[17]: 17
x[18]: 18
x[19]: 19
x[20]: 20
Результат: 105
```

Рисунок 5 - тест номер 3

```
Введите количество чисел (n): 25
Введите последовательность из 25 чисел:
x[1]: 1
x[2]: 2
x[3]: 3
x[4]: 4
x[5]: 5
x[6]: 6
x[7]: 7
x[8]: 8
x[9]: 9
x[10]: 10
x[11]: 11
x[12]: 12
x[13]: 13
x[14]: 14
x[15]: 15
x[16]: 16
x[17]: 17
x[18]: 18
x[19]: 19
x[20]: 20
x[21]: 21
x[22]: 22
x[23]: 23
x[24]: 24
x[25]: 25
Результат: 162.5
```

Рисунок 6 - тест номер 4


```

x[40]: 40
x[41]: 41
x[42]: 42
x[43]: 43
x[44]: 44
x[45]: 45
x[46]: 46
x[47]: 47
x[48]: 48
x[49]: 49
x[50]: 50
Результат: 637.5

```

Рисунок 7 - тест номер 5

2.6 Результаты исследования алгоритма (тестирования) на различных объемах данных и получение времени его выполнения

Таблица 3. Параметры алгоритма при оценке сложности алгоритма

Размер задачи (n)	Время выполнения алгоритма (сек)	Количество инструкций по формуле функции (Т)	Время выполнения Т инструкций на компьютере (Т/быстродействие комп.) (сек)
1	0.002	19	$1.9 \cdot 10^{-8}$
100	0.003	712	$7.12 \cdot 10^{-7}$
1000	0.0074	7012	$7.012 \cdot 10^{-6}$
5000	0.0473	35012	$3.5012 \cdot 10^{-5}$

Примечание к расчетам: базовая частота процессора, на котором выполнялась работа, равняется 2.3 ГГц, или же 2300000000 Гц. Возьмем m за разрядность максимально допустимого `int` – 19

2.7 Анализ выполнения алгоритма

Основываясь на данных таблицы 3 можно заметить рост времени при увеличении размера задачи, согласно формуле функции.