



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 по дисциплине «Структуры и алгоритмы обработки данных»

Тема «Оценка вычислительной сложности алгоритма»

Выполнил студент группы ИКБО-50-23

Русаков М.Ю.

Принял старший преподаватель

Скворцова Л.А.

Оглавление

1. Условие задачи	3
2. Разработка решения	4
2.1. Таблица с количеством выполняемых инструкций, заполненная согласно требованиям задания	4
2.2. Функциональные зависимости, полученные в результате анализа алгоритма	5
2.3. Код алгоритма на языке C++	6
2.4. Разработанные тесты	8
2.5. Скриншоты результатов тестирования	9
2.6. Результаты исследования алгоритма на различных объемах данных и получение времени его выполнения	10
2.7. Анализ выполнения алгоритма	11

1. Условие задачи

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

Требования:

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), указывающую зависимость количества выполняемых инструкций от размера задачи.
3. Технологию подсчета количества инструкций алгоритма представить в виде таблицы.
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. Подготовку тестов выполнить в таблице.
7. Выполнить отладку и тестирование алгоритма. Определить емкостную сложность алгоритма.

Вариант 22. Дано натуральное число n и последовательность натуральных чисел. Определить количество чисел последовательности, являющихся палиндромами.

2. Разработка решения

2.1. Таблица с количеством выполняемых инструкций, заполненная согласно требованиям задания

Пусть $\rho(n) = \log_{10} n + 1$ – функция, вычисляющая количество выполнений некоторой инструкции.

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	quantity $\leftarrow 0$	1
2	if array[i] = 0 then	2
3	continue	1
4	endIf	
5	pow $\leftarrow 1$	1
6	while array[i] mod pow \neq array[i] do	$4 \times \rho(\text{array}[i])$
7	pow \leftarrow pow $\times 10$: (
8	od	
9	reversed $\leftarrow 0$	1
10	temp \leftarrow array[i]	1
11	while pow > 1 do	1
12	digit \leftarrow temp mod 10	: (2
13	reversed \leftarrow reversed $\times 10 +$ digit	: (3
14	temp \leftarrow temp div 10	: (2
15	pow \leftarrow pow div 10	: (2
16	od	
17	if array[i] = reversed then	1
18	quantity \leftarrow quantity + 1	2
19	endIf	
20	return quantity	1

2.2. Функциональные зависимости, полученные в результате анализа алгоритма

Здесь просто переписаны результаты для неэффективного алгоритма, так что нахождение истинных функциональных зависимостей оставляется в качестве самостоятельного упражнения для заинтересованных читателей .

1. Заметим, что в лучшем случае все числа будут отрицательными, а значит не будут являться палиндромами. В таком случае выполняется $F_{\text{лучш}} = 3 \times n + 2$ инструкций.
2. В худшем случае все числа не будут являться палиндромами, а их длина будет наибольшей. Для типа `int` в языке C++ максимальная длина составляет 10 символов (для чисел, не меньших 2×10^9). В таком случае будет выполняться $F_{\text{худш}} = 2 + n(8 + 11 \times l_i) = 118 \times n + 2$ инструкций.

2.3. Код алгоритма на языке C++

```
#include <iostream>
#include <chrono>

int palindromes(int size, int array[]) {
    int quantity = 0;

    for (int i = 0; i < size; i++) {
        // Отрицательные числа не являются палиндромами по определению
        if (array[i] < 0) {
            continue;
        }

        int pow = 1;
        while (array[i] % (pow) != array[i]) {
            pow *= 10;
        }

        int reversed = 0, temp = array[i];
        while (pow > 1) {
            int digit = temp % 10;
            reversed = reversed * 10 + digit;
            temp /= 10;
            pow /= 10;
        }

        if (array[i] == reversed) {
            quantity++;
        }
    }

    return quantity;
}

int main() {
    const int ARRAY_SIZE = 100;
    int array[ARRAY_SIZE];

    std::cout << "Введите длину массива: ";
    int size;
    std::cin >> size;

    // Длина массива ограничена диапазоном [1;100]
    if (!(size >= 1 && size <= 100)) {
        std::cout << "Неверное значение длины массива!";
        return 1;
    }
}
```

```
}

for (int i{}; i < size; i++) {
    std::cin >> array[i];
}

auto start = std::chrono::high_resolution_clock::now();
int pals = palindromes(size, array);
auto end = std::chrono::high_resolution_clock::now();

std::cout << "Количество чисел-палиндромов в массиве: " << pals <<
'\n';
std::cout << "Время выполнения алгоритма: " <<
std::chrono::duration_cast<std::chrono::milliseconds>(end -
start).count() << " ms";
}
```

2.4. Разработанные тесты

Таблица 2 – Таблица тестов

Номер теста	Входные данные	Эталон результата (ожидаемый результат)
1	$n = 1$ arr = {3}	1
2	$n = 3$ arr = {-1, -2, -3}	0
3	$n = 6$ arr = {121, 16, 85, 13, 22, 5}	3
4	$n = 20$ arr = {376, 451, 438, 146, 717, 565, 216, 498, 391, 111, 213, 617, 204, 495, 484, 604, 540, 411, 46, 651}	4

2.5. Скриншоты результатов тестирования

```
1
3
Количество палиндромов = 1
Время выполнения программы: 0 ms
```

Рисунок 1 – Тест 1

```
3
-1 -2 -3
Количество палиндромов = 0
Время выполнения программы: 0 ms
```

Рисунок 2 – Тест 2

```
6
121 16 85 13 22 5
Количество палиндромов = 3
Время выполнения программы: 0 ms
```

Рисунок 3 – Тест 3

```
● 20
376 451 438 146 717 565 216 498 391 111 213 617 204 495 484 604 540 411 46 651
Количество палиндромов = 4
Время выполнения программы: 0 ms
```

Рисунок 4 – Тест 4

2.6. Результаты исследования алгоритма 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 (T)	Время выполнения T инструкций на компьютере (T / быстродействие процессора) (сек)
1	$< 10^{-9}$	120	0,00000003
2	0,000061	11802	0,000003
3	0,000553	118020	0,00003
4	0,002	590002	0,00016

2.7. Анализ выполнения алгоритма

Анализируя данные из таблицы 3, можно прийти к выводу, что время выполнения алгоритма линейно растет при увеличении размера задачи.