



Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное  
учреждение высшего образования  
«Московский государственный технический университет имени  
Н. Э. Баумана  
(национальный исследовательский университет)»  
(МГТУ им. Н. Э. Баумана)

---

ФАКУЛЬТЕТ «Информатика и системы управления»

---

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

---

## РУБЕЖНЫЙ КОНТРОЛЬ № 1

### по дисциплине «Анализ алгоритмов»

Студент Фам Минь Хиеу

Группа ИУ7-52Б

Оценка (баллы) \_\_\_\_\_

Преподаватель Волкова Л. Л.

Москва — 2024 г.

# СОДЕРЖАНИЕ

<b>ВВЕДЕНИЕ</b>	<b>3</b>
<b>1 Аналитическая часть</b>	<b>4</b>
1.1 Трудоёмкость в худшем и лучшем случаях . . . . .	4
1.2 Трудоёмкость в среднем . . . . .	4
1.3 Алгоритм полного перебора . . . . .	4
<b>2 Конструкторская часть</b>	<b>5</b>
2.1 Разработка алгоритмов . . . . .	5
2.2 Модель вычислений для оценки трудоемкости алгоритмов . . .	6
2.3 Трудоемкость алгоритмов . . . . .	7
2.4 Требования к программному обеспечению . . . . .	8
<b>3 Технологическая часть</b>	<b>9</b>
3.1 Средства реализации . . . . .	9
3.2 Реализация алгоритмов . . . . .	9
3.3 Функциональные тесты . . . . .	10
<b>4 Исследовательская часть</b>	<b>11</b>
4.1 Технические характеристики . . . . .	11
4.2 Демонстрация работы программы . . . . .	11
4.3 Вывод . . . . .	12
<b>ЗАКЛЮЧЕНИЕ</b>	<b>13</b>
<b>СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ</b>	<b>14</b>

# ВВЕДЕНИЕ

Целью данной контрольной работы является выполнение следующих задания:

1. Что такое трудоёмкость в среднем?
2. Поиск элемента в массиве из  $N$  различных целых чисел. Найти трудоёмкость в среднем, трудоёмкость в лучшем случае и в худшем случае.

# 1 Аналитическая часть

## 1.1 Трудоемкость в худшем и лучшем случаях

Под худшим случаем будем понимать такой вход  $D$  длины  $n$ , на котором алгоритм  $A$  задает наибольшее количество элементарных операций, при этом максимум берется по всем  $D \in D_n$ . Трудоемкость алгоритма на этом входе будем называть трудоемкостью в худшем случае, и обозначать ее через  $f_A^\wedge(n)$ , тогда

$$f_A^\wedge(n) = \max_{D \in D_n} \{f_A(D)\} \quad (1.1)$$

по аналогии через  $f_A^\vee(n)$  будем обозначать трудоемкость в лучшем случае, как трудоемкость с наименьшим количеством операций на всех входах длины  $n$ :

$$f_A^\vee(n) = \min_{D \in D_n} \{f_A(D)\} \quad (1.2)$$

## 1.2 Трудоемкость в среднем

Трудоемкость алгоритма в среднем — это среднее количество операций, задаваемых алгоритмом  $A$  на входах длины  $n$ , где усреднение берется по всем  $D \in D_n$ . Введем для трудоемкости в среднем обозначение  $\overline{f_A}(n)$ , тогда

$$\overline{f_A}(n) = \sum_{D \in D_n} p(D) \cdot f_A(D), \quad (1.3)$$

где  $p(D)$  есть частотная встречаемость входа  $D$  для анализируемой области применения алгоритма.

## 1.3 Алгоритм полного перебора

Алгоритм полного перебора — метод решения задачи, при котором по очереди рассматриваются все возможные варианты решения. В случае реализации алгоритма в рамках данной работы будут последовательно перебираться каждый элемент до тех пор, пока не будет найден нужный.

## 2 Конструкторская часть

В данном разделе будут представлены схемы алгоритма поиска элемента в массиве, требования к программному обеспечению.

### 2.1 Разработка алгоритмов

На рисунке 2.1 приведена схема алгоритма поиска элемента в массиве методом полного перебора.

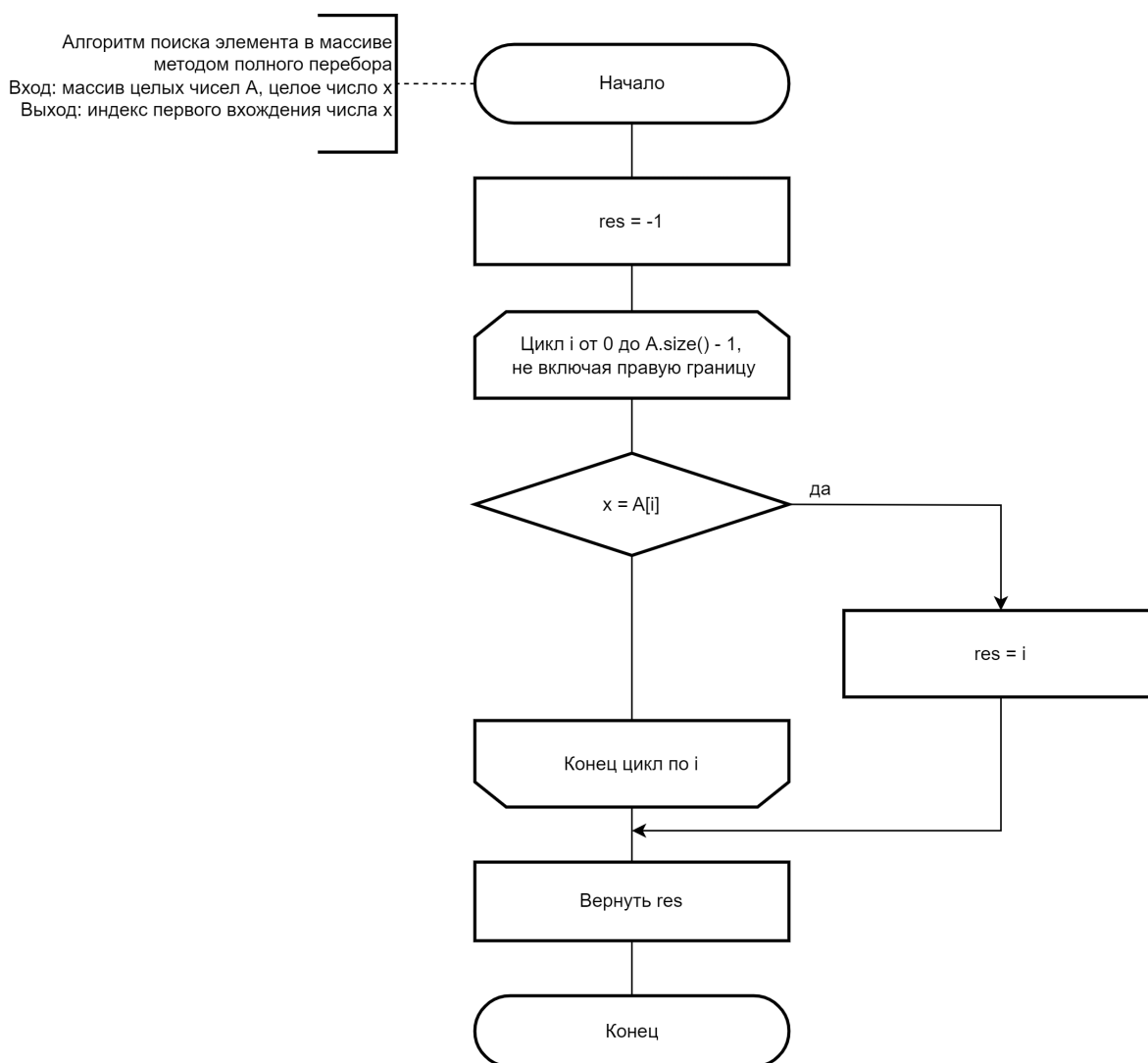


Рисунок 2.1 – Алгоритм поиска элемента в массиве методом полного перебора

## 2.2 Модель вычислений для оценки трудоемкости алгоритмов

Для определения трудоемкости алгоритмов необходимо ввести модель вычислений [1]:

1. операции из списка (2.1) имеют трудоемкость равную 2;

$$*, /, \%, * =, / =, \% = \quad (2.1)$$

2. операции из списка (2.2) имеют трудоемкость равную 1;

$$+, -, + =, - =, =, ==, ! =, <, >, < =, > =, [], ++, -- \quad (2.2)$$

3. трудоемкость оператора выбора `if условие then A else B` рассчитывается как (2.3);

$$f_{if} = f_{условия} + \begin{cases} f_A, & \text{если условие выполняется,} \\ f_B, & \text{иначе.} \end{cases} \quad (2.3)$$

4. трудоемкость цикла рассчитывается, как (2.4);

$$f_{for} = f_{инициализации} + f_{сравнения} + N(f_{тела} + f_{инкремент} + f_{сравнения}) \quad (2.4)$$

5. трудоемкость вызова функции равна 0.

## 2.3 Трудоемкость алгоритмов

Трудоемкость данного алгоритма будет складываться из следующих составляющих:

- трудоемкости создания переменной `res`, равной 1;
- трудоемкости вычисления длины массива, равной  $N$ ;
- трудоемкости тела, нахождения заданного элемента, равной  $2 + 5 \cdot N$ ;

Далее рассмотрены лучший и худший случаи трудоёмкости и условия их наступления.

В худшем случае количество итераций в цикле будет максимально, это ситуация, когда в массиве нет заданного элемента. В этом случае трудоемкость алгоритма равна

$$f_A^\wedge(n) = 1 + N + 2 + 5 \cdot N + 1 = 6 \cdot N + 4 = O(N) \quad (2.5)$$

В лучшем случае количество итераций в цикле будет минимально, это ситуация, когда заданный элемент находится в начале массива. В этом случае трудоемкость алгоритма равна:

$$f_A^\vee(n) = 1 + 2 + 1 = 4 = (1) \quad (2.6)$$

Далее рассмотрены трудоёмкость в среднем. Пусть алгоритм поиска без выполнения цикла затрачивает  $k_0 = 4$  операций, а при каждой итерации  $k_1 = 5$  операций. Алгоритм нашёл элементы на первом сравнении (лучший случай), тогда будет затрачено  $k_0 + k_1$  операций, на втором —  $k_0 + 2 \cdot k_1$ , на последнем (худший случай) —  $k_0 + (N - 1) \cdot k_1$ . Если такой элемент отсутствует в массиве, то это, только перебрав все элементы, таким образом трудоёмкость такого случая равно трудоёмкости худшем случае.

Трудоёмкость в среднем может быть рассчитана как математическое ожидание по формуле (2.7) ( $\Omega$  — множество всех возможных исходов и все исходы равновероятны).

$$\begin{aligned}
\sum_{i \in \Omega} p_i \cdot f_i &= (k_0 + k_1) \cdot \frac{1}{N+1} + (k_0 + 2 \cdot k_1) \cdot \frac{1}{N+1} + (k_0 + 3 \cdot k_1) \cdot \frac{1}{N+1} + \\
&\quad + (k_0 + (N-1) \cdot k_1) \cdot \frac{1}{N+1} + (k_0 + (N-1) \cdot k_1) \cdot \frac{1}{N+1} = \\
&= k_0 \frac{N+1}{N+1} + k_1 + \frac{1+2+\dots+N-1+N-1}{N+1} = k_0 + k_1 \cdot \left( \frac{N-3}{N+1} + \frac{N}{2} \right) = \\
&= k_0 + k_1 \cdot \left( 1 + \frac{N}{2} - \frac{4}{N+1} \right) = 4 + 5 \cdot \left( 1 + \frac{N}{2} - \frac{4}{N+1} \right) = O(N) \quad (2.7)
\end{aligned}$$

## 2.4 Требования к программному обеспечению

К программе предъявляются следующие требования:

- входные данные — массив целых чисел, целое число  $x$ ;
- выходные данные — индекс 1-го вхождения заданного элемента, если найти результат, иначе -1.

## Вывод

Были представлены схемы алгоритмов поиска элемента в массиве методом полным перебором и требования к программному обеспечению. Проведённая теоретическая оценка трудоёмкости алгоритмов показала, что в трудоёмкость выполнения данного алгоритма равна  $O(N)$  в худшем случае,  $O(1)$  в лучшем случае и трудоёмкость в среднем  $O(N)$ .



## 3 Технологическая часть

В данном разделе будут указаны средства реализации, будут представлены реализация алгоритма, а также функциональные тесты.

### 3.1 Средства реализации

Данная программа разработана на языке C++, поддерживаемом многими операционными системами [2].

### 3.2 Реализация алгоритмов

Алгоритм поиска элемент в массиве приведен в листинге 3.1.

Листинг 3.1 – Алгоритм поиска элемента в массиве

```
1 int solution(vector<int> arr, int n, int x)
2 {
3     int res = -1;
4     for (int i = 0; i < n; i++)
5         if (arr[i] == x)
6             {
7                 res = i;
8                 break;
9             }
10    return res;
11 }
```

### 3.3 Функциональные тесты

В таблице 3.1 приведены функциональные тесты для функций, реализующих алгоритмы. Все тесты пройдены успешно.

Таблица 3.1 – Функциональные тесты

№	Входные данные	Ожидаемый результат
1	1 2 3 4 5, 10	-1
2	1 2 3 4 5, 5	4
3	1 2 3 4 5, 1	0

### Вывод

Были реализованы функции поиска полным перебором. Было проведено функциональное тестирование данного алгоритма.

## 4 Исследовательская часть

В данном разделе будут приведены примеры работы программы и представлена информация о технических характеристиках устройства.

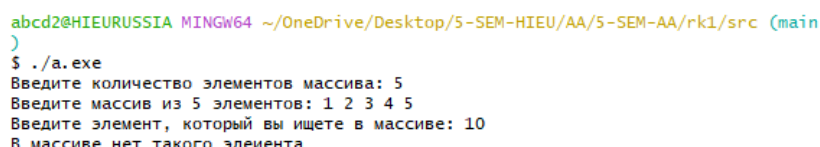
### 4.1 Технические характеристики

Тестирование проводилось на устройстве со следующими техническими характеристиками:

- операционная система Window 10 Home Single Language;
- память 8 Гб;
- процессор 11th Gen Intel(R) Core(TM) i7-1165G7 2.80 ГГц, 4 ядра.

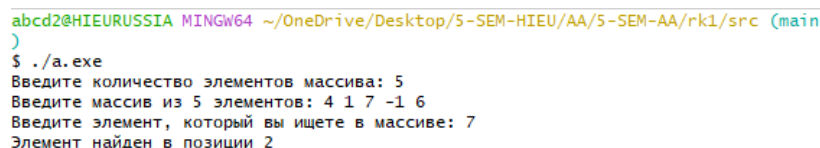
### 4.2 Демонстрация работы программы

На рисунках 4.1 – 4.2 приведены примеры работы программы.



```
abcd2@HIEURUSSIA MINGW64 ~/OneDrive/Desktop/5-SEM-HIEU/AA/5-SEM-AA/rk1/src (main)
$ ./a.exe
Введите количество элементов массива: 5
Введите массив из 5 элементов: 1 2 3 4 5
Введите элемент, который вы ищете в массиве: 10
В массиве нет такого элемента
```

Рисунок 4.1 – Пример работы данного алгоритма при успешно поиске



```
abcd2@HIEURUSSIA MINGW64 ~/OneDrive/Desktop/5-SEM-HIEU/AA/5-SEM-AA/rk1/src (main)
$ ./a.exe
Введите количество элементов массива: 5
Введите массив из 5 элементов: 4 1 7 -1 6
Введите элемент, который вы ищете в массиве: 7
Элемент найден в позиции 2
```

Рисунок 4.2 – Пример работы данного алгоритма при неудачном поиске

### 4.3 Вывод

В данном разделе были приведены примеры работы программы.

## ЗАКЛЮЧЕНИЕ

Проведённая теоретическая оценка трудоёмкости алгоритмов показала, что в трудоёмкость выполнения данного алгоритма равна  $O(N)$  в худшем случае,  $O(1)$  в лучшем случае и трудоёмкость в среднем  $O(N)$ .

Цель, поставленная перед началом работы, была достигнута.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Ульянов М. В. Ресурсно-эффективные компьютерные алгоритмы. Разработка и анализ. — М.: Физматлит, 2007. — 304 с.
2. C++ reference [Электронный ресурс]. Режим доступа: <https://en.cppreference.com/w/>