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»

Отчет по лабораторной работе №7 по курсу «Анализ алгоритмов»

Тема Алгоритмы поиска

Студент Пискунов П.

Группа ИУ7-56Б

Преподаватель Волкова Л.Л., Строганов Д.В.

Москва — 2024 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В мире программирования и алгоритмов поиск элемента в массиве – одна из фундаментальных задач, которая выступает в роли ключа к разгадке многих аспектов эффективной обработки данных. Эта задача, хоть и проста на первый взгляд, замаскирована внутри себя множеством сложных решений и стратегий. Каждый подход несет в себе не только методику нахождения элемента, но и философию взаимодействия с множеством данных.

Целью данной лабораторной работы является изучение алгоритмов поиска в массиве – стандартный алгоритм и алгоритм бинарного поиска.

Для достижения цели, требуется выполнить следующие задачи:

1) проанализировать стандартный алгоритм и алгоритм бинарного поиска с единственным сравнением для каждого медианного элемента;
2) определить средства программной реализации выбранного алгоритма;
3) провести сравнительный анализ по времени реализованного алгоритма;
4) подготовить отчет о лабораторной работе.

1 Аналитическая часть

1.1 Массив

Массив – тип данных, в котором хранится упорядоченный набор однотипных элементов.

Поиск – обработка некоторого множества данных с целью выявления подмножества данных, соответствующего критериям поиска [1].

Все алгоритмы поиска делятся на:

- поиск в неупорядоченном множестве данных;
- поиск в упорядоченном множестве данных.

Упорядоченность – наличие отсортированного ключевого поля.

1.2 Алгоритм бинарного поиска

Бинарный поиск – тип поискового алгоритма, который последовательно делит пополам заранее отсортированный массив данных, чтобы обнаружить нужный элемент [2].

1.3 Стандартный алгоритм поиска

Стандартный алгоритм поиска – метод решения, при котором поочередно перебираются все элементы массива, пока не будет найден нужный. Сложность такого алгоритма зависит от количества всех возможных решений, а время решения может стремиться к экспоненциальному времени работы. Чем дальше искомый ключ от начала массива, тем выше трудоемкость алгоритма.

Вывод

В данном разделе была представлена информация о алгоритмах поиска элемента в массиве.

2 Конструкторская часть

В этом разделе будут представлены требования к программному обеспечению (ПО), трудоемкость и схема алгоритмов.

2.1 Требования к программному обеспечению

Программе передаются массив целыми числами и элемент, который необходимо найти, а на выход получается индекс элемента массива. Кроме того, необходимо сообщить пользователю затраченное каждым алгоритмом процессорное время.

В создаваемом приложении пользователю должен быть доступен выбор желаемого алгоритма.

2.2 Разработка стандартного алгоритма поиска

На рисунке 2.1 представлен стандартный алгоритм поиска.

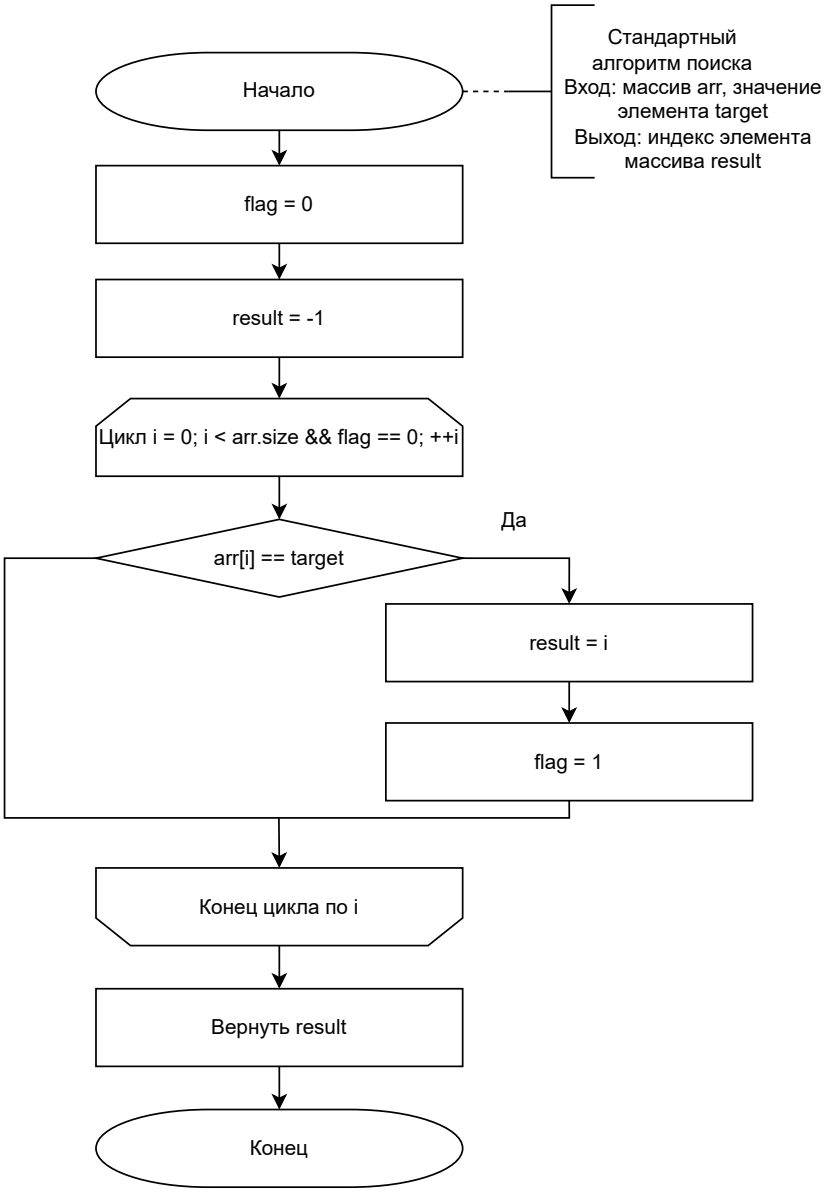

Рисунок 2.1 – Стандартный алгоритм поиска

2.3 Разработка алгоритма бинарного поиска

На рисунке 2.2 представлен алгоритм бинарного поиска.

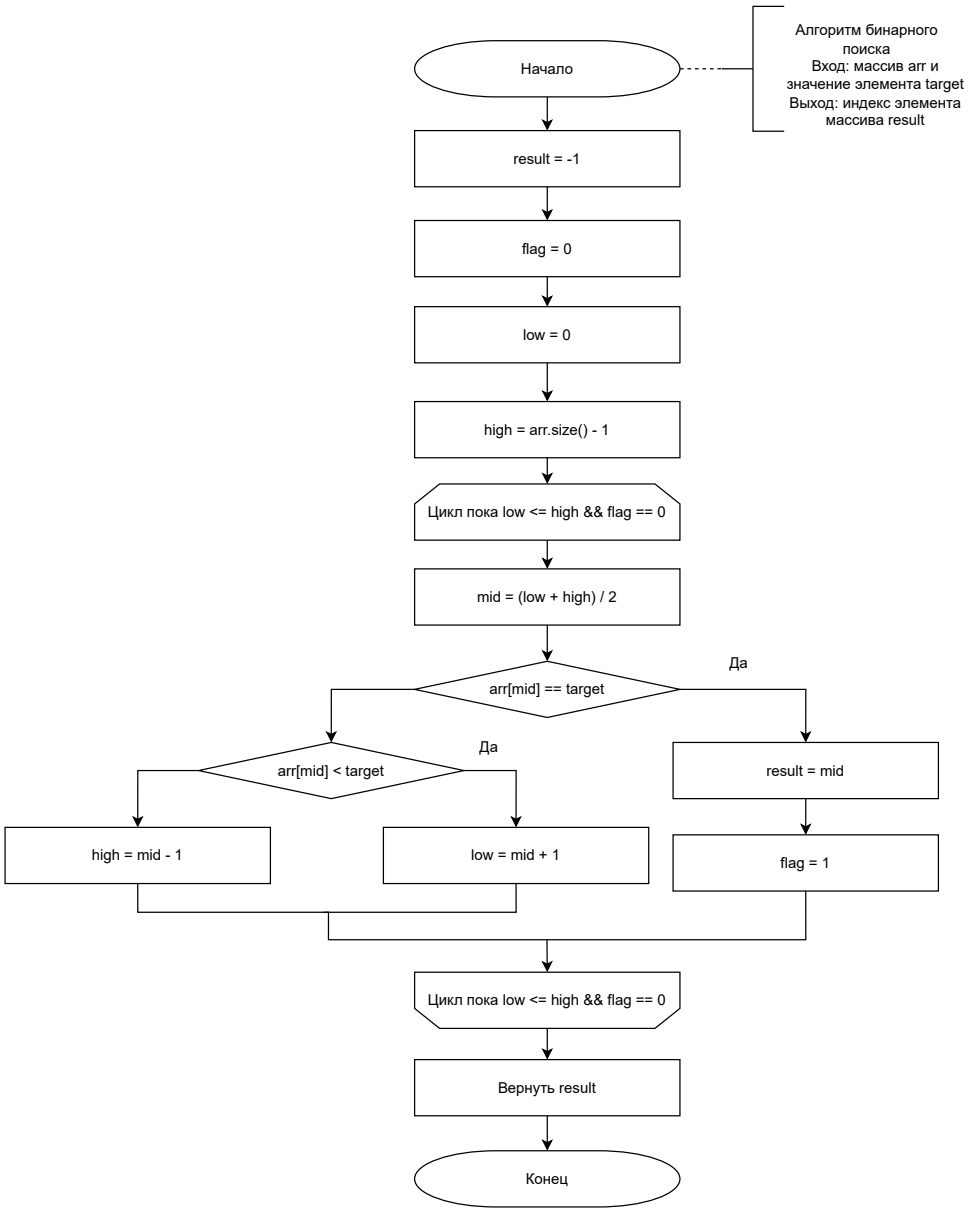

Рисунок 2.2 – Алгоритм бинарного поиска

2.4 Модель вычислений

Для дальнейшего анализа трудоемкости необходимо ввести модель вычислений.

Операции из списка (2.1) имеют трудоемкость 1.

$$=, +=, -=, +, -, ==, !=, <, >, <=, >=, [], ++, -- \tag{2.1}$$

Операции из списка (2.2) имеют трудоемкость 2.

$$*, /, \% \tag{2.2}$$

Трудоемкость оператора выбора `if условие then A else B` рассчитывается как:

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

Трудоемкость цикла рассчитывается как:

$$f_{for} = f_{\text{инициализации}} + f_{\text{сравнения}} + N(f_{\text{тела}} + f_{\text{инкремента}} + f_{\text{сравнения}}) \tag{2.4}$$

Трудоемкость вызова функции равна 0.

2.5 Трудоемкость алгоритмов

Далее размер массива обозначается как N.

2.5.1 Стандартный алгоритм поиска

Пусть на старте алгоритм затрагивает k_0 операций, а при сравнении k_1 операций, тогда:

- в лучшем случае элемент будет найден на первом сравнении за $k_0 + k_1$ операций;
- в худшем случае элемент будет найден на последнем сравнении за $k_0 + Nk_1$ операций;

- элемент будет найден на i-ом сравнении за $k_0 + ik_1$ операций.

Тогда средняя трудоемкость может быть рассчитана по следующей формуле:

$$f = k_0 + k_1 \cdot \left(1 + \frac{N}{2} - \frac{1}{N+1}\right) \tag{2.5}$$

2.5.2 Алгоритм бинарного поиска

Пусть на старте алгоритм затрагивает k_0 операций, тогда:

- в лучшем случае элемент будет найден на первом сравнении с средним элементом с трудоемкостью $k_0 + log_2 1$;
- в худшем случае элемент будет найден на последнем сравнении с трудоёмкостью $b + log_2 N$;
- элемент будет найден на i-ом сравнении с трудоемкостью $b + log_2 i$;

Вывод

На основе теоретических знаний, полученных в аналитическом разделе, были разработаны схемы алгоритмов, благодаря которым может быть найден элемент массива разными способами. Также для каждого из них были рассчитаны и оценены лучшие и худшие случаи.

3 Технологическая часть

В этом разделе предоставляются листинги реализованных алгоритмов и осуществляется выбор средств реализации.

3.1 Выбор средств реализации

Для выполнения данной лабораторной работы был выбран язык программирования C++. Время измерялось с помощью функции clock() из библиотеки time.h [3].

3.2 Реализация алгоритмов

В листинге 3.1 представлена реализация стандартного алгоритма поиска.

Листинг 3.1 – Стандартный алгоритм поиска

```
1    int standard_search(const vector<int> &arr, const int target)
2    {
3        for (size_t i = 0; i < arr.size(); ++i)
4            if (arr[i] == target)
5                return i;
6
7        return -1;
8    }
```

В листинге 3.2 представлена реализация алгоритма бинарного поиска.

Листинг 3.2 – Алгоритм бинарного поиска

```
int binary_search(const vector<int>& arr, const int target)
{
    int low = 0, high = arr.size() - 1;

    while (low <= high)
    {
        int mid = (low + high) / 2;

        if (arr[mid] == target)
            return mid;
        else if (arr[mid] < target)
            low = mid + 1;
        else
            high = mid - 1;
    }

    return -1;
}
```

Вывод

Были выбраны инструменты для реализации и разработаны стандартный алгоритм и алгоритм бинарного поиска. Также предоставлены листинги кода на выбранном языке программирования.

4 Исследовательская часть

4.1 Интерфейс приложения

На рисунке 4.1 представлен интерфейс приложения.

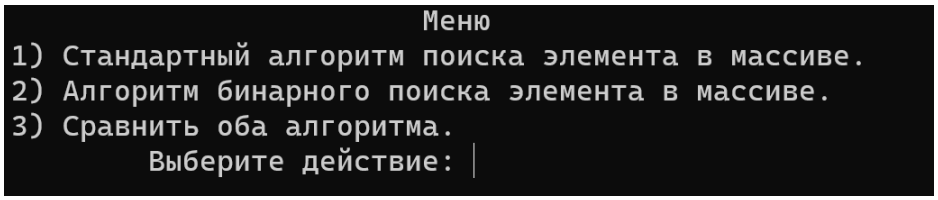

Рисунок 4.1 – Интерфейс приложения

4.2 Технические характеристики

Технические характеристики устройства:

- операционная система — Windows 11 Pro 64 – разрядная система [4];
- оперативная память — 16 Гбайт;
- процессор — 11th Gen Intel(R) Core(TM) i7-1165G7 с тактовой частотой 2.8 ГГц.

4.3 Время выполнения реализаций алгоритмов

На рисунке 4.2 представлено сравнение времени выполнения стандартного алгоритма и алгоритма бинарного поиска элемента в массиве в лучшем случае, когда искомый элемент расположен на первом месте. Время выполнения измерялось как среднее значение из десяти измерений.

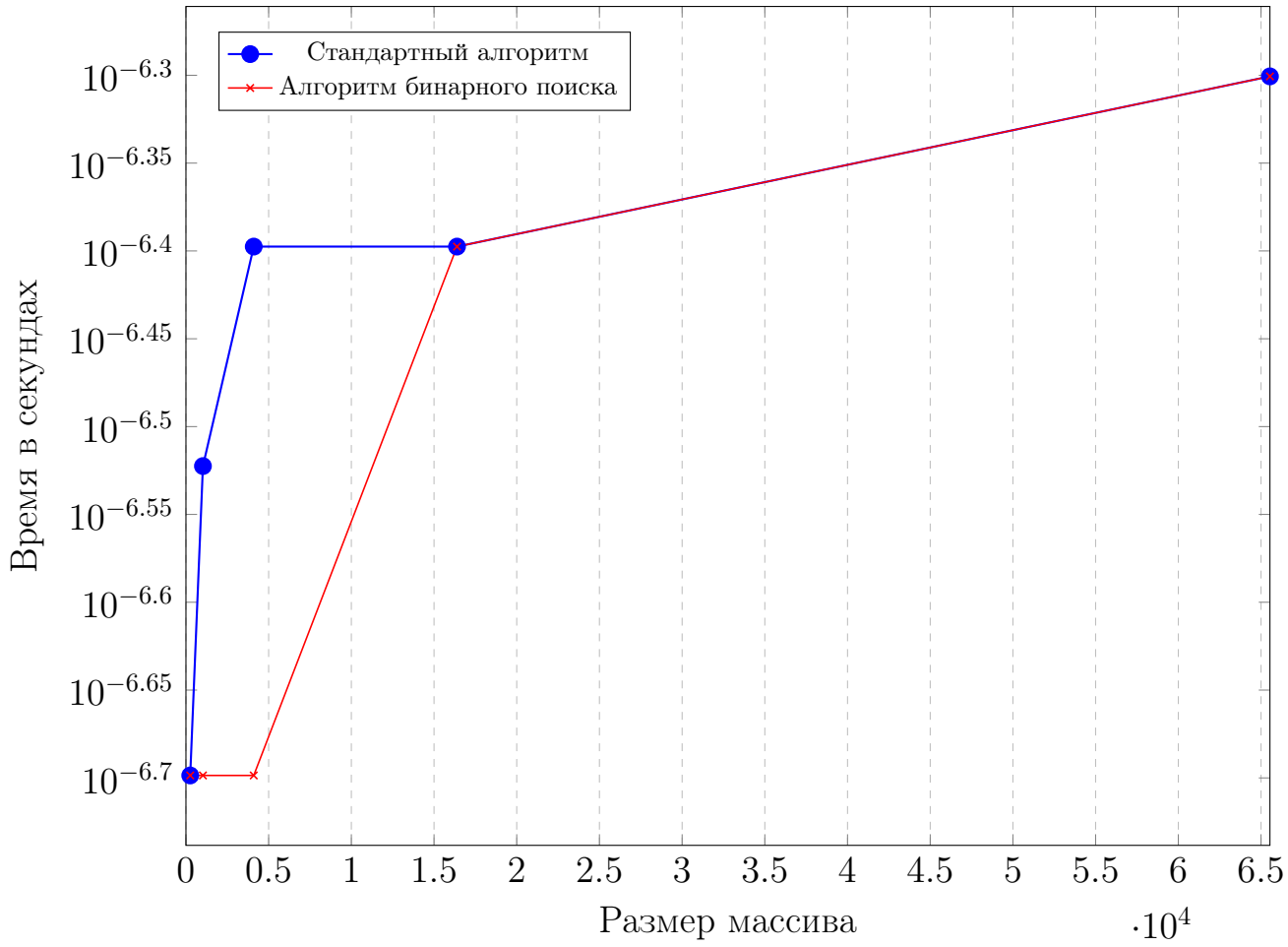

Рисунок 4.2 – Сравнение алгоритмов по времени в лучшем случае

На рисунке 4.3 представлено сравнение времени выполнения стандартного алгоритма и алгоритма бинарного поиска элемента в массиве в худшем случае, когда искомый элемент отсутствует в массиве. Время выполнения было измерено как среднее значение по десяти измерениям.

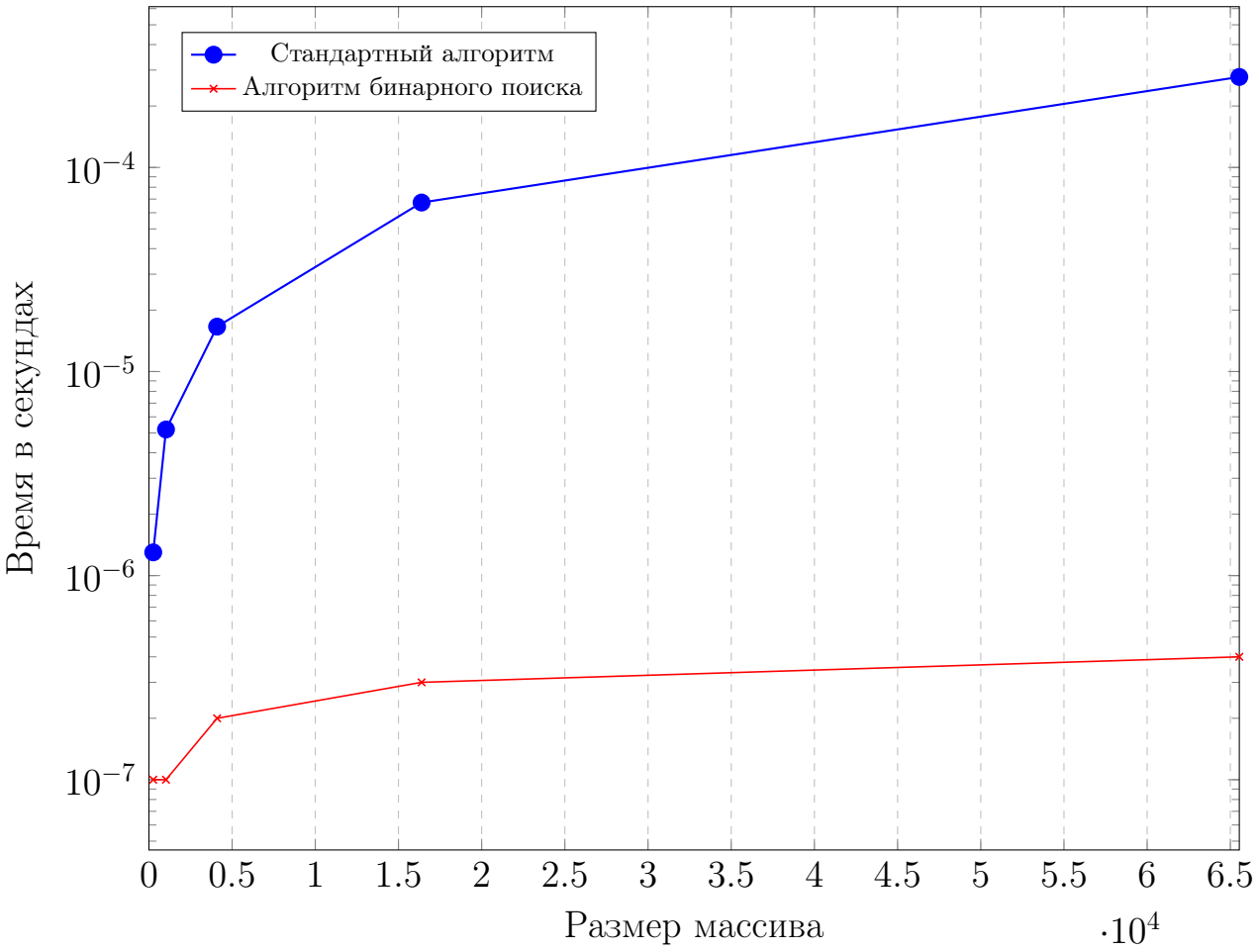

Рисунок 4.3 – Сравнение алгоритмов по времени в худшем случае

Вывод

В результате исследования стандартного алгоритма поиска и алгоритма бинарного поиска с единственным сравнением для каждого медианного элемента выявлено, что в худшем случае бинарный поиск оказывается более эффективным из - за лучшей трудоемкости по сравнению со стандартным алгоритмом. Это связано с тем, что бинарный поиск в упорядоченном массиве делает гораздо меньше сравнений, что уменьшает общее время выполнения.

В лучшем случае бинарный поиск также демонстрирует небольшое превосходство по сравнению с стандартным алгоритмом. Хотя разница не столь значительна, бинарный поиск все же немного опережает стандартный алгоритм благодаря более оптимальной стратегии поиска в упорядоченном массиве.

ЗАКЛЮЧЕНИЕ

В результате исследования стандартного алгоритма поиска и алгоритма бинарного поиска с единственным сравнением для каждого медианного элемента выявлено, что в худшем случае бинарный поиск оказывается более эффективным из - за лучшей трудоемкости по сравнению со стандартным алгоритмом. Это связано с тем, что бинарный поиск в упорядоченном массиве делает гораздо меньше сравнений, что уменьшает общее время выполнения.

В лучшем случае бинарный поиск также демонстрирует небольшое превосходство по сравнению с стандартным алгоритмом. Хотя разница не столь значительна, бинарный поиск все же немного опережает стандартный алгоритм благодаря более оптимальной стратегии поиска в упорядоченном массиве.

В результате выполнения лабораторной работы были выполнены следующие задачи:

1) проанализированы стандартный алгоритм и алгоритм бинарного поиска с единственным сравнением для каждого медианного элемента;

2) определены средства программной реализации выбранного алгоритма;

3) проведен сравнительный анализ по времени реализованного алгоритма;

4) подготовлен отчет о лабораторной работе.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Search [Электронный ресурс]. — Режим доступа: https://prog-cpp.ru/algorithm-sort/ (дата обращения: 04.02.2024).

2 Бинарный поиск [Электронный ресурс]. — Режим доступа: https://blog.skillfactory.ru/glossary/binarnyj-poisk/ (дата обращения: 04.02.2024).

3 C library function clock() [Электронный ресурс]. — Режим доступа: https://en.cppreference.com/w/c/chrono/clock_t (дата обращения: 16.01.2024).

4 Windows 11 Pro [Электронный ресурс]. — Режим доступа: https://www.microsoft.com/en-us/windows/business/windows-11-pro#windows11security (дата обращения: 16.01.2024).