



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»

Отчёт

по лабораторной работе № 7

Название: Поиск по словарю

Дисциплина: Анализ алгоритмов

Студент

ИУ7-55Б

(Группа)

Д.В. Сусликов

(Подпись, дата)

(И.О. Фамилия)

Преподаватель

Л.Л. Волкова

(Подпись, дата)

(И.О. Фамилия)

Москва, 2020

Содержание

Введение	3
1 Аналитический раздел	4
1.1 Словарь	4
1.2 Алгоритм полного перебора	4
1.3 Алгоритм бинарного поиска	4
1.4 Алгоритм поиска по сегментам	5
Вывод	5
2 Конструкторский раздел	6
2.1 Схемы алгоритмов	6
Вывод	9
3 Технологический раздел	10
3.1 Общие требования	10
3.2 Средства реализации	10
3.3 Листинг кода программы	11
Вывод	13
4 Экспериментальный раздел	13
4.1 Примеры работы программы	13
4.2 Описание экспериментов	13
Вывод	14
5 Заключение	15
Заключение	15
Литература	16

Введение

Цель работы: изучение алгоритмов поиска слов в словаре

В ходе лабораторной работы требуется:

- 1) описать алгоритм полного перебора;
- 2) описать алгоритм двоичного поиска;
- 3) описать алгоритм поиска слов по сегментам;
- 4) реализовать 3 алгоритма поиска по словарю;
- 5) провести замеры времени работы алгоритмов.

1 Аналитический раздел

В данном разделе представлено описание трех выбранных алгоритмов.

1.1 Словарь

Словари играют большую роль в современной культуре, в них отражаются знания, накопленные обществом на протяжении веков. Они служат целям описания и нормализации языка, содействуют повышению правильности и выразительности речи его носителей.

Словарь – справочная книга, содержащая собрание слов (или морфем, словосочетаний, идиом и т. д.), расположенных по определенному принципу, и дающая сведения об их значениях, употреблении, происхождении, переводе на др. язык и т. п. (лингвистические словари) или информацию о понятиях и предметах, ими обозначаемых, о деятелях в каких-либо областях науки, культуры и др[6].

1.2 Алгоритм полного перебора

Алгоритм полного перебора — это алгоритм разрешения математических задач, который можно отнести к классу способов нахождения решения рассмотрением всех возможных вариантов.

Под полным перебором понимается методика разрешения задач математики путем рассмотрения всех возможных вариантов. Уровень сложности при полном переборе напрямую связан с количеством допустимых решений задачи. В случае, когда область решений огромна, время полного перебора может исчисляться десятками и даже сотнями лет, и при этом итоговый результат возможно ещё не будет найден[4].

1.3 Алгоритм бинарного поиска

Бинарный поиск производится в упорядоченном массиве.

При бинарном поиске искомый ключ сравнивается с ключом среднего элемента в массиве. Если они равны, то поиск успешен. В противном случае поиск осуществляется аналогично в левой или правой частях массива.

Алгоритм может быть определен в рекурсивной и нерекурсивной формах.

Бинарный поиск также называют поиском методом деления отрезка пополам или дихотомии.

Двоичный поиск заключается в том, что на каждом шаге множество объектов делится на две части и в работе остаётся та часть множества, где находится искомый объект. Или же, в зависимости от постановки задачи, мы можем остановить процесс, когда мы получим первый или же последний индекс вхождения элемента. Последнее условие — это левосторонний/правосторонний двоичный поиск[5].

1.4 Алгоритм поиска по сегментам

Суть заключается в том, что требуется разбить словарь по сегментам. Каждый сегмент подразумевает под собой первую букву слов, что в нём находятся.

Поиск слова осуществляется сперва по сегменту, а затем по его содержанию.

Вывод

В данном разделе были рассмотрены алгоритмы для поиска слова в словаре.

2 Конструкторский раздел

В данном разделе показаны схемы разработанных алгоритмов.

2.1 Схемы алгоритмов

На Рисунке 1 представлена схема алгоритма полного перебора.

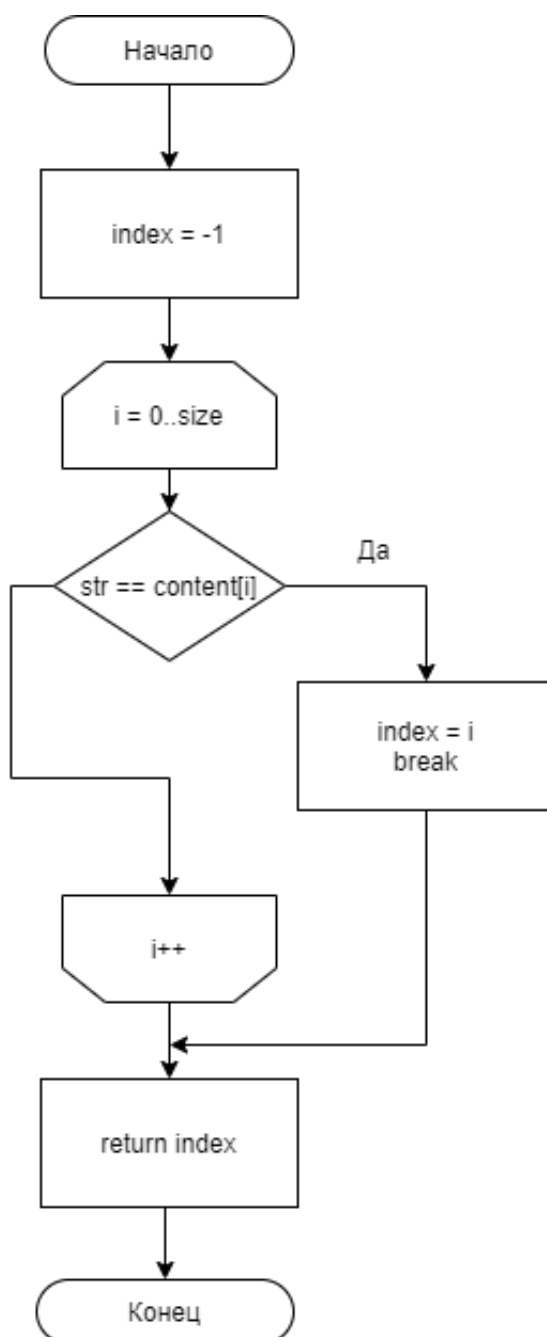


Рисунок 1 – Схема алгоритма полного перебора

На Рисунке 2 показана схема алгоритма бинарного поиска.

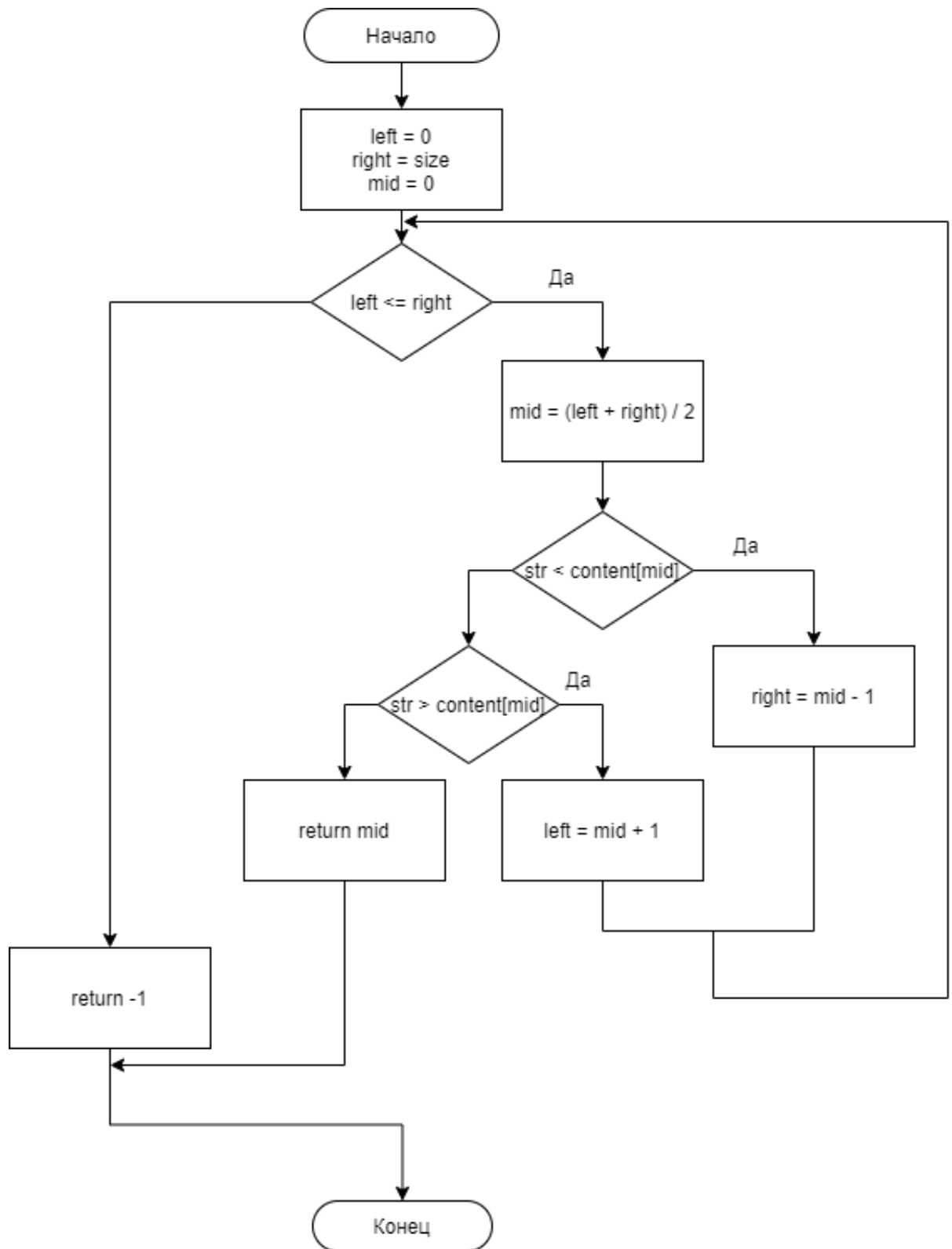


Рисунок 2 – Схема алгоритма бинарного поиска

На Рисунке 3 показана схема алгоритма поиска по сегментам.

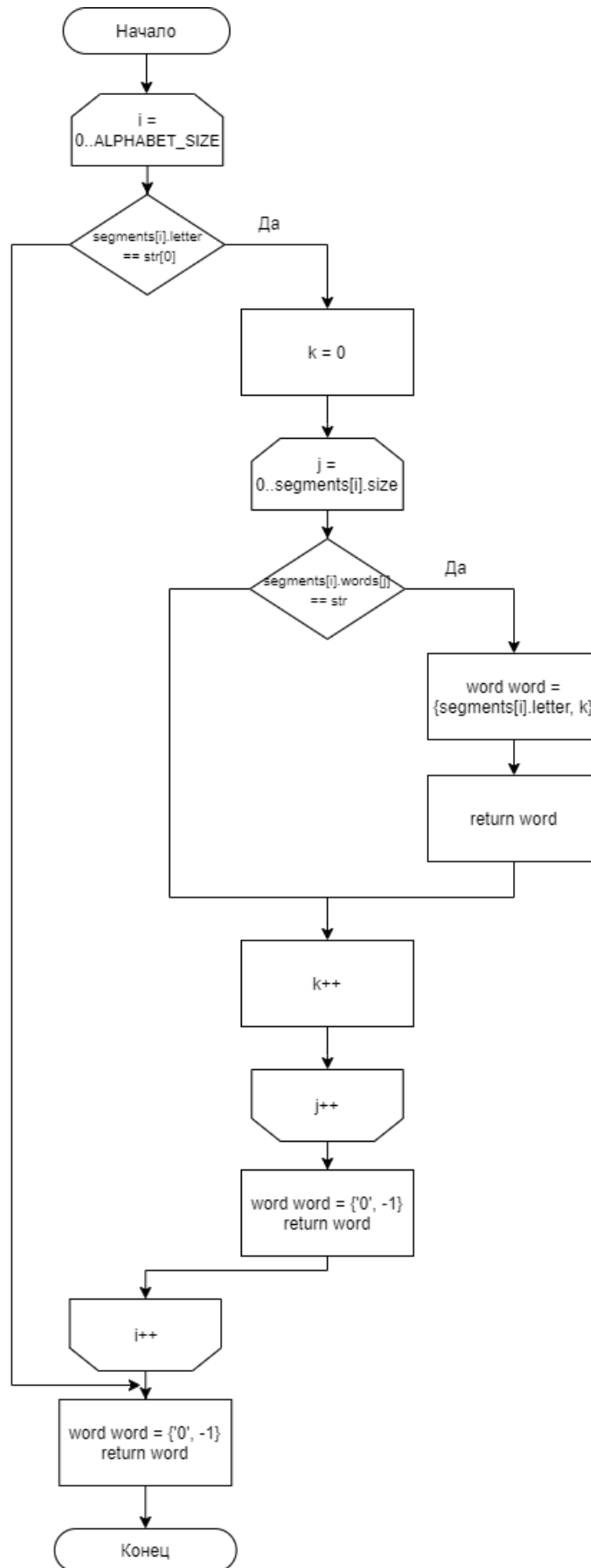


Рисунок 3 – Схема алгоритма поиска по сегментам

Вывод

В данном разделе были рассмотрены схемы алгоритмов.

3 Технологический раздел

В данном разделе даны общие требования к программе, средства реализации и сама реализация алгоритмов.

3.1 Общие требования

Требования к вводу:

- 1) вводится слово;
- 2) в первых 2 алгоритмах результатом будет являться ключ (в первом по обычному словарю, а во втором по отсортированному словарю), в третьем результатом будет являться ключ (первая буква или же сегмент) и индекс слова в сегменте.

Требования к программе

- 1) при вводе слова, которого нет в словаре, программа не должна завершиться аварийно;
- 2) ключ или же ключ-индекс должны быть корректными (соответствовать нужному слову).

3.2 Средства реализации

В лабораторной работе был использован язык *C++*[1], так как он известен, и на нём было написано множество предыдущих работ.

Среда разработки - *Qt*[2].

Для замеров процессорного времени была использована функция *clock()*[3].

3.3 Листинг кода программы

В Листинге 1 реализован алгоритм полного перебора слов.

Листинг 1 – Алгоритм полного перебора слов

```
1      int Dictionary::brute_force(std::string str)
2      {
3          for (size_t i = 0; i < size; i++)
4          {
5              if (str == content[i])
6                  return i;
7          }
8
9          return -1;
10     }
```

В Листинге 2 реализован алгоритм двоичного поиска

Листинг 2 – Алгоритм двоичного поиска

```
1      int Dictionary::binary_find(std::string str)
2      {
3          int left = 0;
4          int right = size;
5          int mid = 0;
6
7          while(left <= right)
8          {
9              mid = (left + right) / 2;
10
11              if (str < content[mid])
12                  right = mid - 1;
13              else if (str > content[mid])
14                  left = mid + 1;
15              else return mid;
16          }
17          return -1;
```

В Листинге 3 реализован алгоритм поиска по сегментам

Листинг 3 – Алгоритм поиска по сегментам

```
1 SegDictionary::SegDictionary(Dictionary& dic)
2 {
3     char alphabet[] = {'e', 't', 'a', 'o', 'i', 'n', 's', 'h',
4         'r', 'd', 'l', 'c', \
5         'u', 'm', 'w', 'f', 'g', 'y', 'p', 'b', 'v', 'k', 'x',
6         'j', \
7         'q', 'z'};
8
9     for (size_t i = 0; i < ALPHABET_SIZE; i++)
10     {
11         char cur_letter = alphabet[i];
12         size_t amount = 0;
13         segment seg;
14         seg.letter = cur_letter;
15
16         for (size_t j = 0; j < dic.size; j++)
17         {
18             if (dic.content[j][0] == cur_letter)
19             {
20                 seg.words[amount] = dic.content[j];
21                 amount++;
22             }
23
24         seg.size = amount;
25         segments[i] = seg;
26     }
```

Вывод

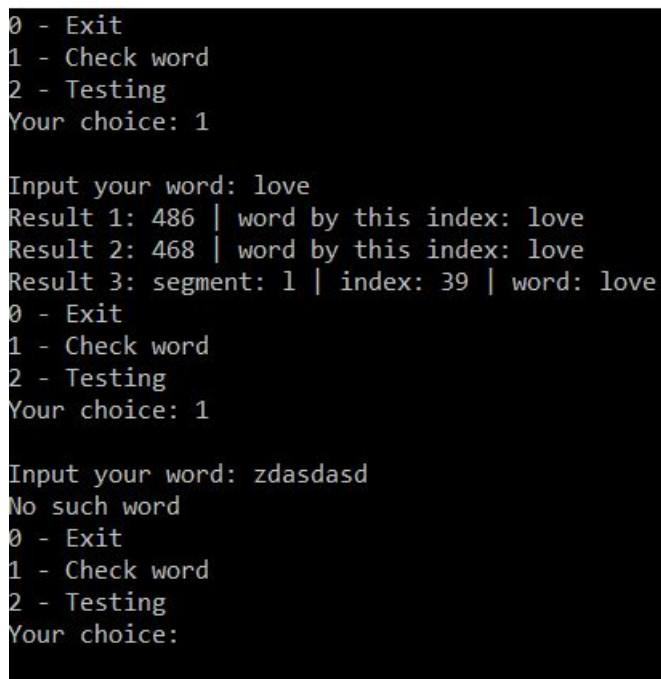
В данном разделе были даны общие требования к программе, описаны средства реализации, а также реализованы алгоритмы поиска по словарю.

4 Экспериментальный раздел

В данном разделе представлены результаты работы программы и приведен анализ времени работы алгоритмов.

4.1 Примеры работы программы

На Рисунке 4 представлен пример работы программы.



```
0 - Exit
1 - Check word
2 - Testing
Your choice: 1

Input your word: love
Result 1: 486 | word by this index: love
Result 2: 468 | word by this index: love
Result 3: segment: 1 | index: 39 | word: love
0 - Exit
1 - Check word
2 - Testing
Your choice: 1

Input your word: zdasdasd
No such word
0 - Exit
1 - Check word
2 - Testing
Your choice:
```

Рисунок 4 – Примеры работы программы

4.2 Описание экспериментов

Производится замер времени для $n + 1$ возможных случаев, где n - длина словаря.

На Рисунке 5 представлены результаты сравнения трех алгоритмов поиска.

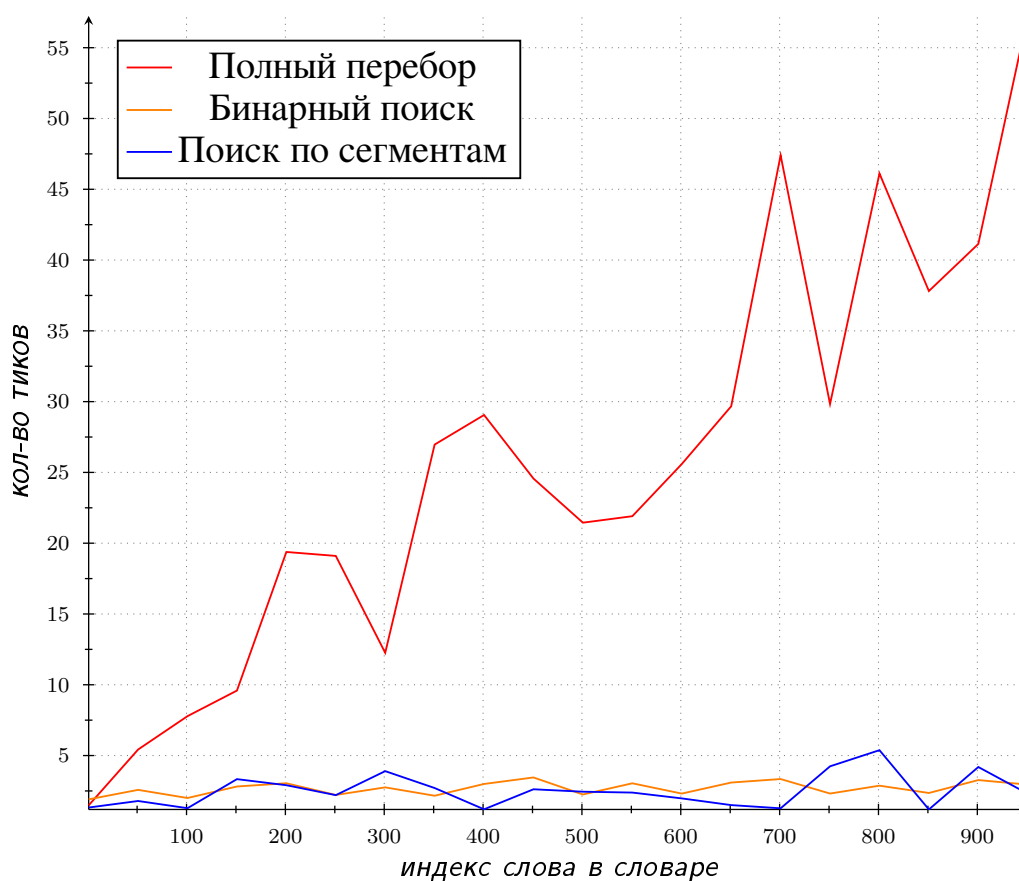


Рисунок 5 – Результаты замеров процессорного времени.

Вывод

По результатам тестирования выяснилось, что бинарный поиск наиболее стабилен по времени, но поиск по сегментам в некоторых случаях работает быстрее. По графику алгоритма поиска по сегментам можно увидеть резкие скачки. Это происходит из-за того, что слово находится в конце сегмента. Самым медленным оказался полный перебор, у которого время работы пропорционально индексу слова в словаре.

5 Заключение

В ходе выполнения данной лабораторной работы были изучены три алгоритмы поиска по словарю. Были описаны все алгоритмы и реализованы. Также были изучены способы хранения слов, то есть в случаях первого и второго алгоритмов слова хранились в массивах, а в третьем хранилось по сегментам. Сравнили время работы алгоритмов, в результате которого стало понятно, что наиболее стабильным по времени оказался бинарный поиск, но в некоторых случаях поиск по сегментам может быть быстрее.

Литература

- 1) Бьерн Страуструп. Язык программирования C++. -URL:

https://codernet.ru/books/c_plus/bern_straustруп_yazyk_programmirovaniya_c_specialnoe_izdanie/

(дата обращения: 01.10.2020). Текст: электронный.

- 2) Qt. -URL:

<https://www.qt.io/> (дата обращения: 01.10.2020). Текст: электронный.

- 3) Функция *clock*. -URL:

<https://docs.microsoft.com/ru-ru/cpp/c-runtime-library/reference/clock?view=vs-2019> (дата обращения: 01.10.2020). Текст: электронный.

- 4) Полный перебор. -URL:

https://spravochnick.ru/informatika/algoritmizaciya/algoritm_polnogo_perebora/ (дата обращения: 28.12.2020). Текст: электронный.

- 5) Бинарный поиск. -URL:

<https://prog-cpp.ru/search-binary/> (дата обращения: 28.12.2020). Текст: электронный.

- 6) Словарь. -URL:

http://gramota.ru/slovari/types/17_2 (дата обращения: 28.12.2020). Текст: электронный.