

*Государственное образовательное учреждение высшего
профессионального образования*
**«Московский государственный технический
университет имени Н.Э. Баумана»
(МГТУ им. Н.Э. Баумана)**

ЛАБОРАТОРНАЯ РАБОТА №7
ПО КУРСУ «АНАЛИЗ АЛГОРИТМОВ»

Поиск подстроки в строке

Выполнил: Сорокин А.П., гр. ИУ7-52Б

Преподаватели: Волкова Л.Л., Строганов Ю.В.

Москва, 2019 г.

Оглавление

Введение	2
1 Аналитическая часть	3
1.1 Задачи	3
1.2 Описание алгоритмов	3
1.2.1 Стандартный алгоритм	3
1.2.2 Алгоритм Кнута-Морриса-Пратта	3
1.2.3 Алгоритм Бойера-Мура	3
2 Конструкторская часть	5
3 Технологическая часть	10
3.1 Средства реализации	10
3.2 Реализации алгоритмов	10
3.3 Тесты	13
4 Экспериментальная часть	14
4.1 Примеры работы	14
4.2 Сравнение работы алгоритмов	14
Заключение	16
Литература	17

Введение

Поиск информации - одно из основных использований компьютера, и быстрый поиск точно заданной подстроки в строке является одной из самых простейших задач поиска информации. Однако эта задача является чрезвычайно важной. Данная функция встроена в различные текстовые редакторы и базы данных, что существенно ускоряет процесс поиска информации и редактирование фрагментов. В настоящее время функции поиска подстроки в строке инкапсулированы во многие высокоуровневые языки программирования. Но стоит помнить, что стандартные функции далеко не самые оптимальные и эффективные, и если основной задачей программы является нахождение подстроки в строке, то необходимо знать принципы организации функций поиска. Также не нужно забывать, что область применения функций поиска не ограничивается одними текстовыми редакторами и базами данных, наоборот, расширяется на различные сферы человеческой жизни.

1. Аналитическая часть

1.1 Задачи

Цель лабораторной работы - изучение особенностей работы алгоритмов Кнута-Морриса-Пратта и Бойера-Мура.

Для того чтобы добиться этой цели, были поставлены следующие задачи:

- применить знания программирования для реализации данных алгоритмов;
- получить практические навыки во время выполнения задания;
- экспериментально подтвердить различия во временной эффективности работы стандартного алгоритма поиска подстроки в строке, алгоритма Кнута-Морриса-Пратта и Бойера-Мура.

1.2 Описание алгоритмов

1.2.1 Стандартный алгоритм

Стандартный алгоритм начинает со сравнения первого символа текста с первым символом подстроки. Если они совпадают, то происходит переход ко второму символу текста и подстроки. При совпадении сравниваются следующие символы. Так продолжается до тех пор, пока не окажется, что подстрока целиком совпала с отрезком текста, или пока не встретятся несовпадающие символы. В первом случае задача решена, во втором мы сдвигаем указатель текущего положения в тексте на один символ и заново начинаем сравнение с подстрокой [1].

1.2.2 Алгоритм Кнута-Морриса-Пратта

Это — один из самых известных алгоритмов решения задачи поиска образца P в тексте T , имеющий временную оценку $O(n)$, т. е. в нем поиск образца осуществляется за время, пропорциональное длине текста. В какой-то мере этот результат можно считать «точкой отсчета» в стремлении специалистов по информатике создать новые алгоритмы решения данной классической задачи. Здесь образец, как и в простом алгоритме, последовательно «прикладывается» к тексту и осуществляется пошаговое сравнение символов. Но если в простом алгоритме после несовпадения в какой-то позиции осуществляется сдвиг на одну позицию, то в рассматриваемом, за счет предварительного анализа P , сдвиг выполняется в некоторых случаях более чем на один символ, в отличие от стандартного алгоритма [2].

1.2.3 Алгоритм Бойера-Мура

Алгоритм Бойера-Мура осуществляет сравнение с образцом справа налево, а не слева направо. Исследуя искомый образец, можно осуществлять более эффективные прыжки в

тексте при обнаружении несовпадения. В этом алгоритме кроме таблицы суффиксов применяется таблица стоп-символов. Она заполняется для каждого символа в алфавите. Для каждого встречающегося в подстроке символа таблица заполняется по принципу максимальной позиции символа в строке, за исключением последнего символа. При определении сдвига при очередном несовпадении строк, выбирается максимальное значение из таблицы суффиксов и стоп-символов[1].

2. Конструкторская часть

В данном разделе представлены принципы работы алгоритмов и их схемы. На рисунках 2.1 и 2.2 представлен алгоритм Кнута-Морриса-Пратта.

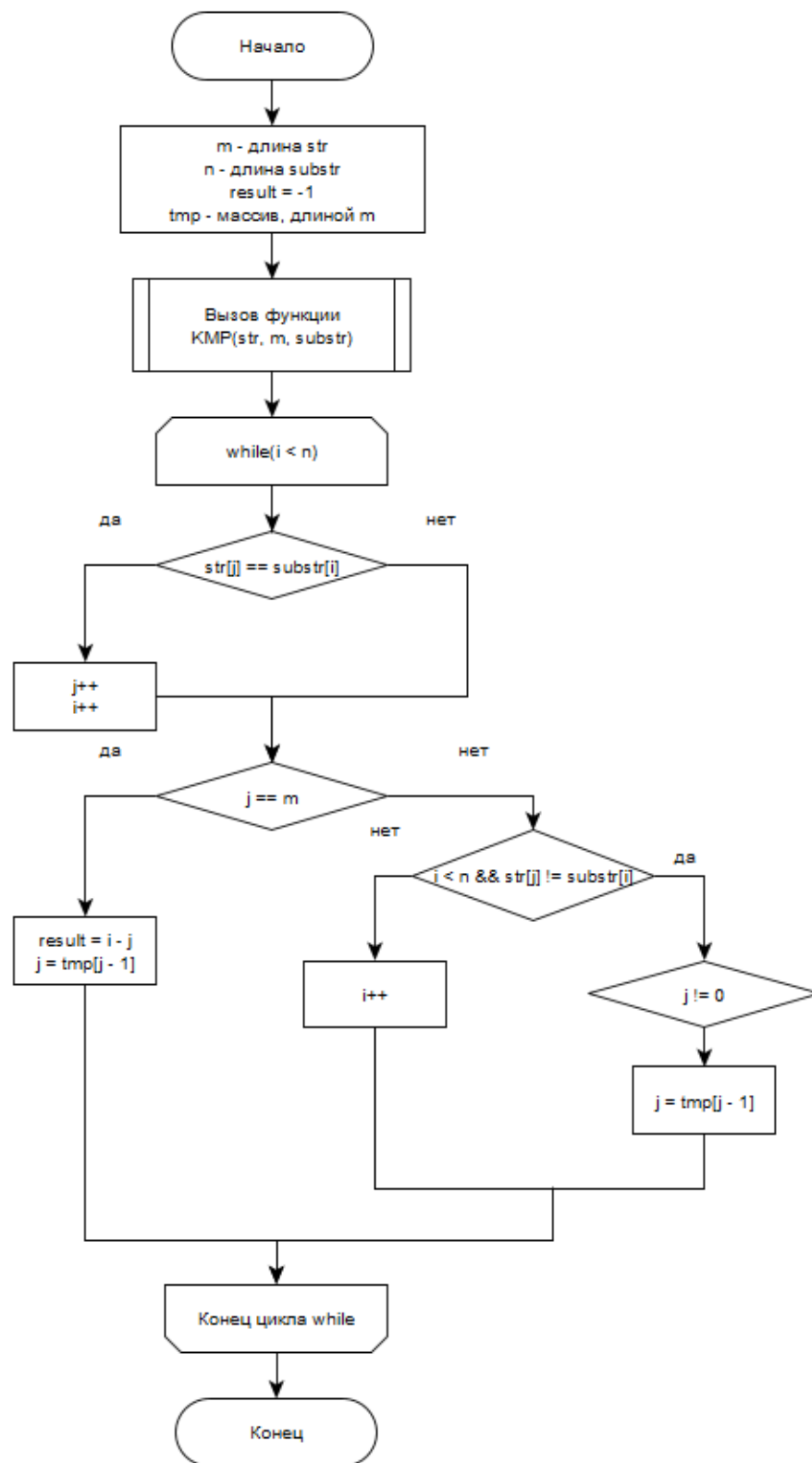


Рис. 2.1: Схема алгоритма Кнута-Морриса-Пратта

На рисунках 2.3 и 2.4 представлен алгоритм Бойера-Мура.

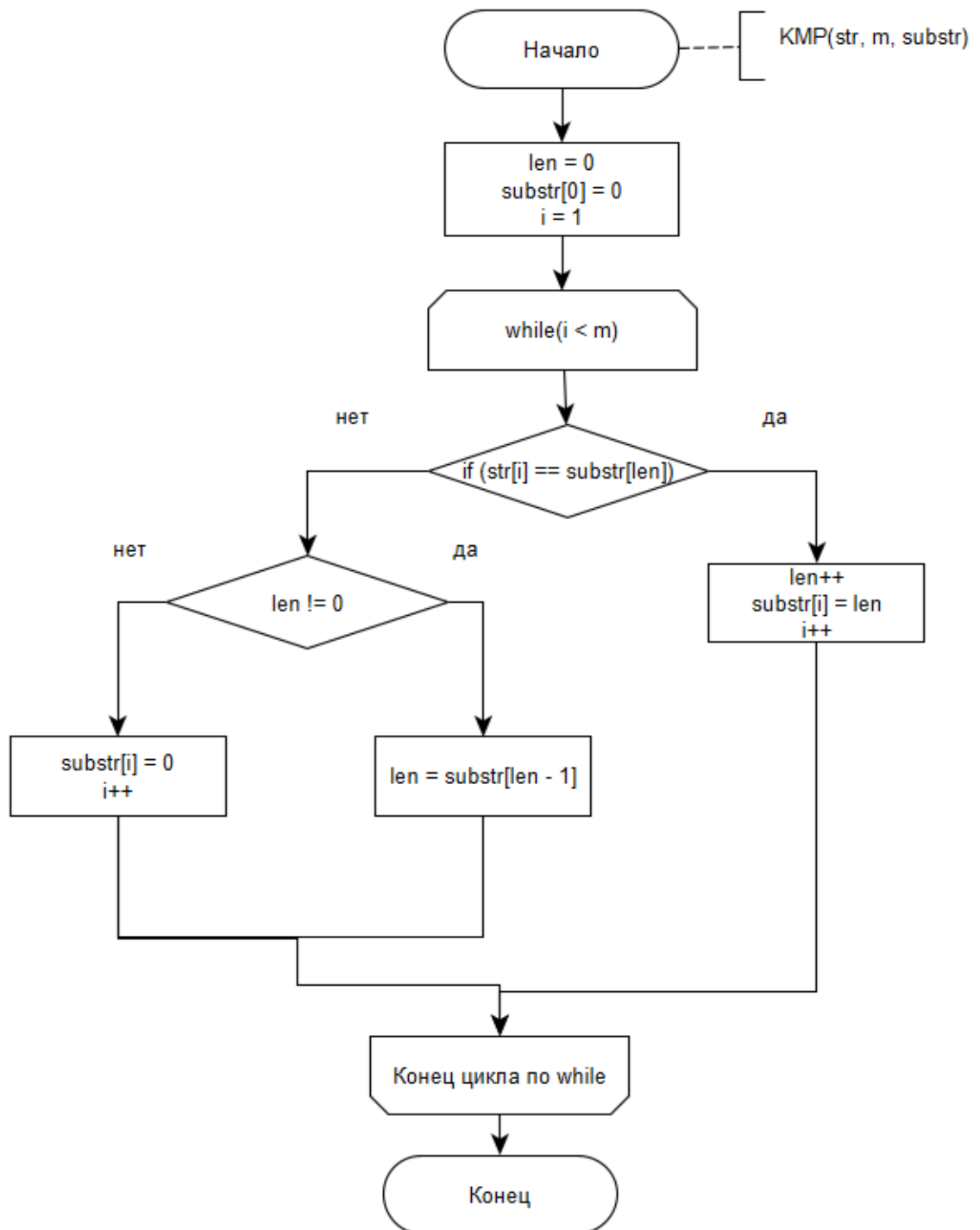


Рис. 2.2: Схема функции `kmp`

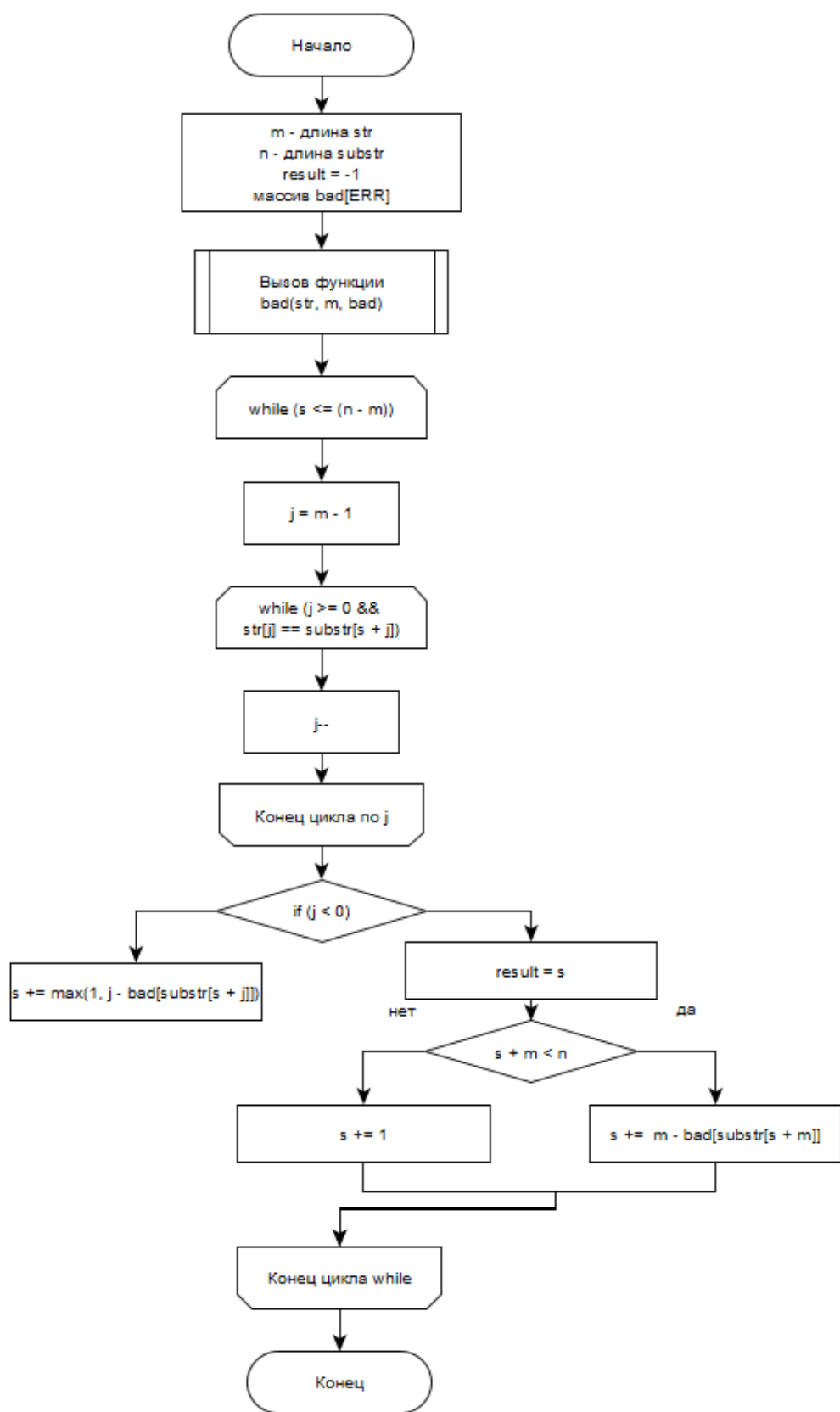


Рис. 2.3: Схема алгоритма Бойера-Мура

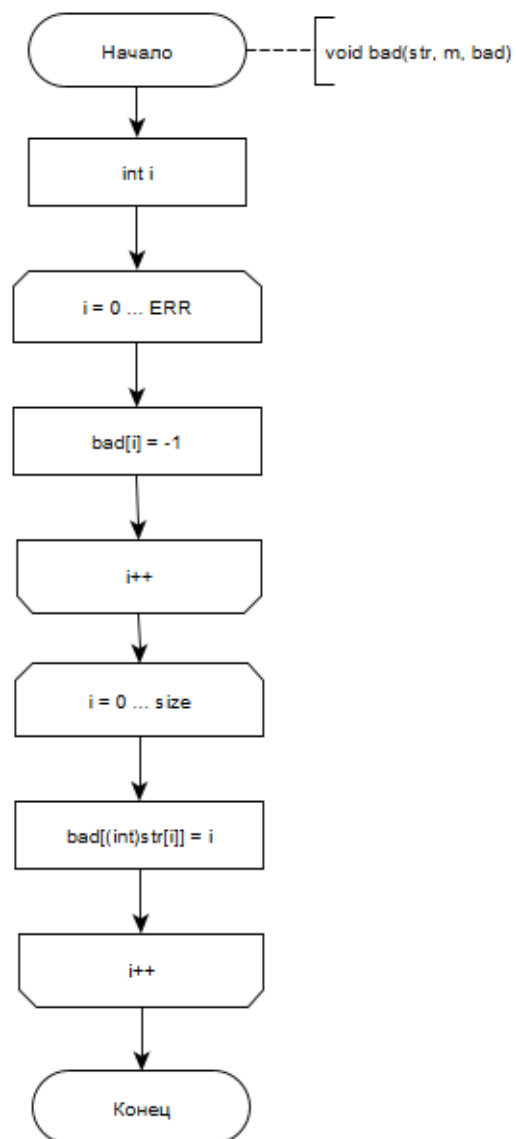


Рис. 2.4: Схема функции bad

3. Технологическая часть

В данном разделе приведены требования к программному обеспечению, средствам реализации, а также листинги кода.

3.1 Средства реализации

Для реализации программы был использован язык C++ [3]. Для замера процессорного времени была использована функция `rdtsc()` из библиотеки `stdrin.h`. Потоки реализовывались с использованием библиотеки `threads.h`.

3.2 Реализации алгоритмов

В листингах 3.1, 3.2, 3.3 представлен коды реализаций рассматриваемых в данной лабораторной работе алгоритмов поиска подстроки в строке.

Листинг 3.1: Стандартный алгоритм

```
1 int substr_std(std::string s, std::string subs)
2 {
3     int sn = s.length(), subn = subs.length();
4     int n = sn - subn + 1;
5
6     for (int i = 0; i < n; i++)
7     {
8         bool correct = true;
9         for (int j = 0; j < subn && correct; j++)
10             if (subs[j] != s[i + j])
11                 correct = false;
12         if (correct)
13             return i;
14     }
15     return -1;
16 }
```

Листинг 3.2: Алгоритм Кнута-Морриса-Пратта

```
1 void fail_compute(std::string s, int n, int *fail)
2 {
3     fail[0] = 0;
4     for (int i = 1; i < n; i++)
5     {
6         int j = fail[i - 1];
7         while (j > 0 && s[i] != s[j])
8             j = fail[j - 1];
```

```

9     if (s[i] == s[j])
10         j++;
11     fail[i] = j;
12 }
13 }
14
15 int substr_kmp(std::string s, std::string subs)
16 {
17     int sn = s.length(), subn = subs.length();
18     int *fail = new int[subn];
19
20     fail_compute(subs, subn, fail);
21
22     int j = 0;
23     for (int i = 0; i < sn; i++)
24     {
25         while (j > 0 && subs[j] != s[i])
26             j = fail[j - 1];
27         if (subs[j] == s[i])
28             j++;
29         if (j == subn)
30         {
31             delete [] fail;
32             return i - subn + 1;
33         }
34     }
35     delete [] fail;
36     return -1;
37 }

```

Листинг 3.3: Алгоритм Бойера-Мура

```

1 void bad(std::string subs, int size, int *badchar)
2 {
3     for (int i = 0; i < 256; i++)
4         badchar[i] = -1;
5     for (int i = 0; i < size; i++)
6         badchar[(int)subs[i]] = i;
7 }
8
9 int substr_bm(std::string s, std::string subs)
10 {
11     int sn = s.length(), subn = subs.length();
12     int result = -1;
13     int badchar[SIZE];
14     bad(subs, subn, badchar);
15
16     int i = 0;
17     while (i <= sn - subn)
18     {
19         int j = subn - 1;
20         while (j >= 0 && subs[j] == s[i + j])
21             j--;
22         if (j < 0)

```

```
23     {
24         result = i;
25         i += (i + subn < sn) ? subn - badchar[(int)s[i + subn]] : 1;
26     }
27     else
28         i += std::max(1, j - badchar[(int)s[i + j]]);
29 }
30 return result;
31 }
```

3.3 Тесты

Для проверки корректности работы были подготовлены функциональные тесты, представленные в таблице 3.1.

Таблица 3.1: Функциональные тесты

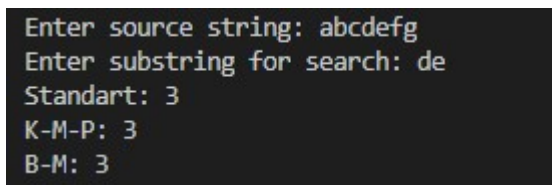
В результате проверки реализации всех алгоритмов прошли тесты.

4. Экспериментальная часть

В данном разделе будет проведён сравнительный анализ реализаций стандартного алгоритма поиска подстроки в строке, алгоритма Кнута-Морриса-Пратта и Бойера-Мура.

4.1 Примеры работы

На рисунке 4.1 представлен пример работы программы, демонстрирующий корректную работу алгоритмов.



```
Enter source string: abcdefg
Enter substring for search: de
Standart: 3
K-M-P: 3
B-M: 3
```

Рис. 4.1: Пример работы программы

4.2 Сравнение работы алгоритмов

Для сравнения времени работы алгоритмов поиска подстроки в строке были использованы случайные строки длиной от 10 до 200 с шагом 10 и подстрока фиксированной длины 2. Эксперимент для более точного результата повторялся 100 раз. Итоговый результат рассчитывался как средний из полученных результатов. Результаты измерений показаны в таблице 4.1 и на рисунке 4.2.

Таблица 4.1: Время работы алгоритмов поиска подстроки в строке в тактах процессора

Размер строки	Стандартный	Алг-м Кнута-Морриса-Пратта	Алг-м Бойера-Мура
10	267	513	1793
20	512	693	2084
30	878	1004	2565
40	953	1068	2696
50	1063	1143	2861
60	1157	1224	2999
70	1159	1216	3060
80	1349	1363	3272
90	1424	1426	3398
100	1406	1397	3663
110	2337	1500	3521
120	1636	1951	4510
130	1661	1597	3799
140	1753	1674	3925
150	1874	1768	4074
160	1954	1826	4182
170	1984	1858	4292
180	1969	1947	4435
190	2056	1875	4480
200	2245	2045	4649

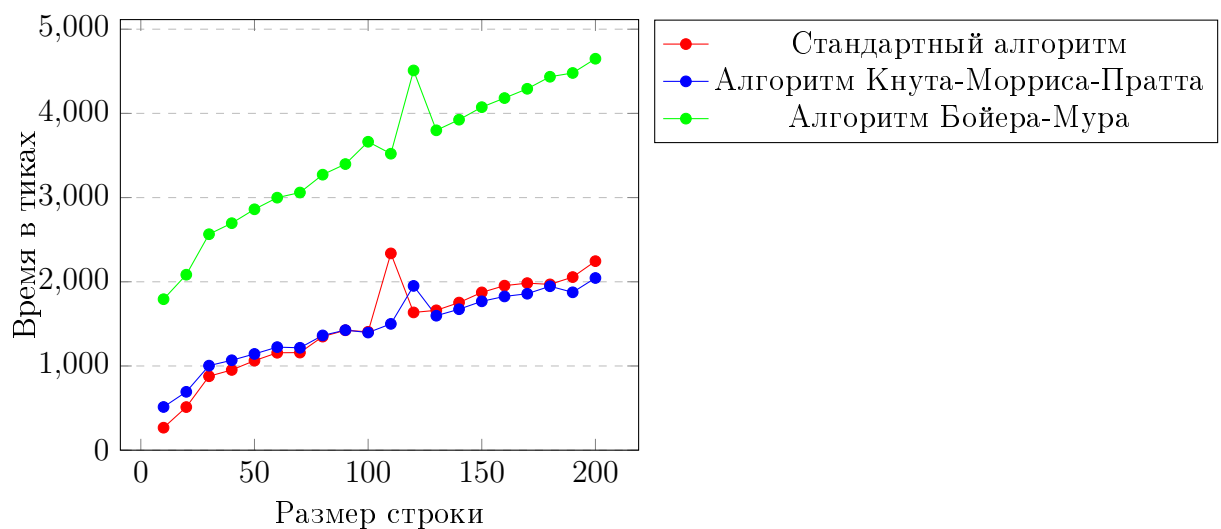


Рис. 4.2: График времени работы алгоритмов поиска подстроки в строке

Эксперименты показали, что наиболее эффективным алгоритмом из трех рассмотренных оказался стандартный алгоритм, а самым неэффективным оказался алгоритм Бойера-Мура.

Заключение

В ходе выполнения данной лабораторной работы были изучены два алгоритма для поиск подстроки в строке - Кнута-Морриса-Пратта и Бойера-Мура. Во время разработки программного обеспечения были получены практические навыки реализации указанных алгоритмов.

Литература

- [1] Дж. Макконнелл. Анализ алгоритмов. Активный обучающий подход
- [2] Окулов С.М. Алгоритмы обработки строк. – М.: БИНОМ. Лаборатория знаний, 2009.
- [3] <https://cppreference.com/> [Электронный ресурс]