

МИНОБРНАУКИ РОСИИ

*Федеральное государственное бюджетное образовательное учреждение высшего образования*

***«МИРЭА – Российский технологический университет»***

**РТУ МИРЭА**



Отчет по выполнению практического задания №6.2

**Тема:**

Поиск образца в текстеДисциплина: «Структуры и алгоритмы обработки данных»

|  |  |  |
| --- | --- | --- |
| Выполнил студент: Васильев Б.А.  Группа: ИКБО-20-23 |  |  |

Москва 2024

**СОДЕРЖАНИЕ**

[ЦЕЛЬ РАБОТЫ 3](#_Toc178190517)

[ХОД РАБОТЫ 3](#_Toc178190518)

[Формулировка задачи 3](#_Toc178190519)

[Описание подхода к решению 3](#_Toc178190520)

[Коды программы 8](#_Toc178190521)

[Результаты тестирования 12](#_Toc178190522)

[ВЫВОД 18](#_Toc178190523)

[СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ 18](#_Toc178190524)

# ЦЕЛЬ РАБОТЫ

Освоить приёмы реализации алгоритмов поиска образца в тексте.

# ХОД РАБОТЫ

### Формулировка задачи

Разработайте приложения в соответствии с заданиями в индивидуальном варианте (п.2). В отчёте в разделе «Математическая модель решения (описание алгоритма)» разобрать алгоритм поиска на примере. Подсчитать количество сравнений для успешного поиска первого вхождения образца в текст и безуспешного поиска. Определить функцию (или несколько функций) для реализации алгоритма поиска. Определить предусловие и постусловие. Сформировать таблицу тестов с указанием успешного и неуспешного поиска, используя большие и небольшие по объему текст и образец, провести на её основе этап тестирования. Оценить практическую сложность алгоритма в зависимости от длины текста и длины образца и отобразить результаты в таблицу (для отчета).

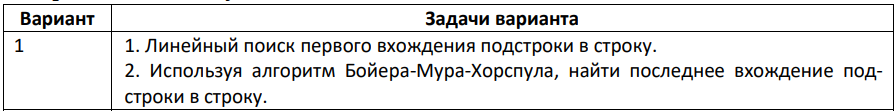


Рисунок 1 – Индивидуальный вариант задачи

### Описание подхода к решению

**Прямой поиск: описание алгоритма и пример**

Алгоритм **прямого поиска** (или наивного поиска) строк заключается в последовательном сравнении каждого символа образца с соответствующим символом подстроки в тексте. Если все символы совпадают, алгоритм завершает поиск. В противном случае, образец сдвигается на одну позицию вправо, и процедура повторяется. Если образец не найден в тексте, алгоритм завершает работу.

Рассмотрим пример на основании следующего текста и образца.

**Пример 1: успешный поиск**

**Текст (haystack):** ABCDABCDABEE  
**Образец (needle):** ABCD

1. Алгоритм начинает сравнивать образец с первой позиции текста:
   * Сравниваем A с A — совпадение.
   * Сравниваем B с B — совпадение.
   * Сравниваем C с C — совпадение.
   * Сравниваем D с D — совпадение.

Так как все символы образца совпали с подстрокой текста, алгоритм завершает поиск, и первое вхождение образца найдено на позиции 1.

**Количество сравнений:** 4.

**Пример 2: безуспешный поиск**

Теперь рассмотрим пример безуспешного поиска, где в тексте нет вхождений образца.

**Текст (haystack):** ABCDABCDABEE  
**Образец (needle):** XYZ

1. Алгоритм начинает сравнение с первой позиции текста:
   * Сравниваем A с X — не совпадает.
2. Сдвигаем образец на одну позицию вправо:
   * Сравниваем B с X — не совпадает.
3. Сдвигаем образец на одну позицию вправо:
   * Сравниваем C с X — не совпадает.

Этот процесс повторяется до тех пор, пока не будут проверены все возможные сдвиги. Так как в тексте не нашлось совпадений с образцом, поиск завершается неудачей.

**Количество сравнений:** 12 (каждый символ текста сравнивается с первым символом образца).

**Пример 3: худший случай**

Теперь рассмотрим худший случай, когда первые символы образца совпадают с текстом, но последний символ не совпадает.

**Текст (haystack):** AAAAAAAAAAAA  
**Образец (needle):** AAAB

1. Алгоритм начинает с первой позиции текста:
   * Сравниваем A с A — совпадение.
   * Сравниваем A с A — совпадение.
   * Сравниваем A с A — совпадение.
   * Сравниваем A с B — не совпадает.

Образец сдвигается на одну позицию вправо, и весь процесс повторяется до конца текста. Этот случай является худшим, так как алгоритм вынужден каждый раз проверять почти все символы образца, прежде чем обнаружить несовпадение.

**Количество сравнений:** 48 (для каждого сдвига проверяются все символы образца)

Прямой поиск может быть эффективным для небольших строк, однако его производительность в худших случаях оставляет желать лучшего, так как количество сравнений линейно зависит от длины текста и образца.

### Худший случай:

В худшем случае каждый символ образца почти полностью совпадает с подстрокой текста, кроме последнего символа. Тогда алгоритм проверяет все символы образца для каждого сдвига по тексту.

* Допустим, длина текста — n, а длина образца — m.
* На каждом шаге происходит до m сравнений, и таких шагов может быть до n - m + 1 (всего возможных сдвигов).
* Таким образом, количество сравнений в худшем случае будет около m×(n−m+1), что асимптотически равно O(n×m).

**Средний случай:**

В среднем случае предполагается, что совпадения случаются реже, и алгоритм быстрее находит несовпадения на ранних стадиях сравнения.

* В среднем ожидаемое количество сравнений будет меньше, так как чаще всего образец не совпадает с подстрокой на ранних позициях.
* Поэтому сложность алгоритма в среднем случае приближается к O(n), так как большинство сравнений происходит до первого несовпадения.

### Алгоритм Бойера-Мура-Хорспула: описание и пример

Алгоритм **Бойера-Мура-Хорспула** является упрощенной версией алгоритма Бойера-Мура. Он использует идею сдвига образца по тексту на основе анализа символов, не входящих в подстроку. Этот алгоритм эффективнее на практике по сравнению с наивным прямым поиском, особенно при работе с большими строками.

Основная идея заключается в том, что при несовпадении символа образца и текста на определенной позиции происходит сдвиг образца вправо на несколько символов, минуя некоторые позиции, где несовпадение гарантировано. Сдвиг определяется таблицей смещений, которая рассчитывается для каждого символа образца.

**Описание алгоритма**

1. Алгоритм начинает сравнение символов образца и текста справа налево.
2. Если найдено несовпадение, используется таблица смещений для сдвига образца вправо. Если символ, на котором произошло несовпадение, не входит в образец, то образец сдвигается настолько, что можно пропустить несколько символов текста.
3. Этот процесс повторяется до тех пор, пока образец не будет найден или не завершится весь текст.

**Пример 1: успешный поиск**

**Текст (haystack):** ABCDABCDABEE  
**Образец (needle):** ABCD

1. Строим таблицу смещений для символов образца. Если символ встречается в образце, сдвиг равен расстоянию до последнего вхождения этого символа. Если символа нет в образце, сдвиг равен длине образца:
   * A — 3,
   * B — 2,
   * C — 1,
   * D — 4 (длина образца).
2. Начинаем сравнение с позиции 4 в тексте:
   * Сравниваем D с D — совпадение.
   * Сравниваем C с C — совпадение.
   * Сравниваем B с B — совпадение.
   * Сравниваем A с A — совпадение.

Образец совпал с подстрокой текста на первой позиции. Поиск завершен.

**Количество сравнений:** 4.

**Пример 2: безуспешный поиск**

**Текст (haystack):** ABCDABCDABEE  
**Образец (needle):** XYZ

1. Строим таблицу смещений:
   * X — 3,
   * Y — 2,
   * Z — 1.
2. Начинаем сравнение с позиции 3 в тексте:
   * Сравниваем С с Z — не совпадает.
   * Так как С нет в образце, смещаем образец вправо на 3 позиции.
3. Повторяем процесс для всех возможных сдвигов в тексте, но совпадений не будет найдено.

Алгоритм завершит работу, когда все возможные сдвиги будут проверены.

**Количество сравнений:** значительно меньше, чем при прямом поиске, так как сдвиги позволяют пропускать части текста. В этом случае 4 сравнения.

**Сложность алгоритма**

**Средний случай**

Алгоритм Бойера-Мура-Хорспула работает быстрее на практике, так как в среднем использует меньше сравнений благодаря сдвигам. Средняя сложность приближается к O(n /m), так как сдвиги позволяют быстро пропускать символы текста.

**Худший случай**

В худшем случае, когда образец почти полностью совпадает с текстом, количество сравнений приближается к O(n×m).

### Код программы

Реализуем код приложения на языке программирования C++ (рис. 2-6)

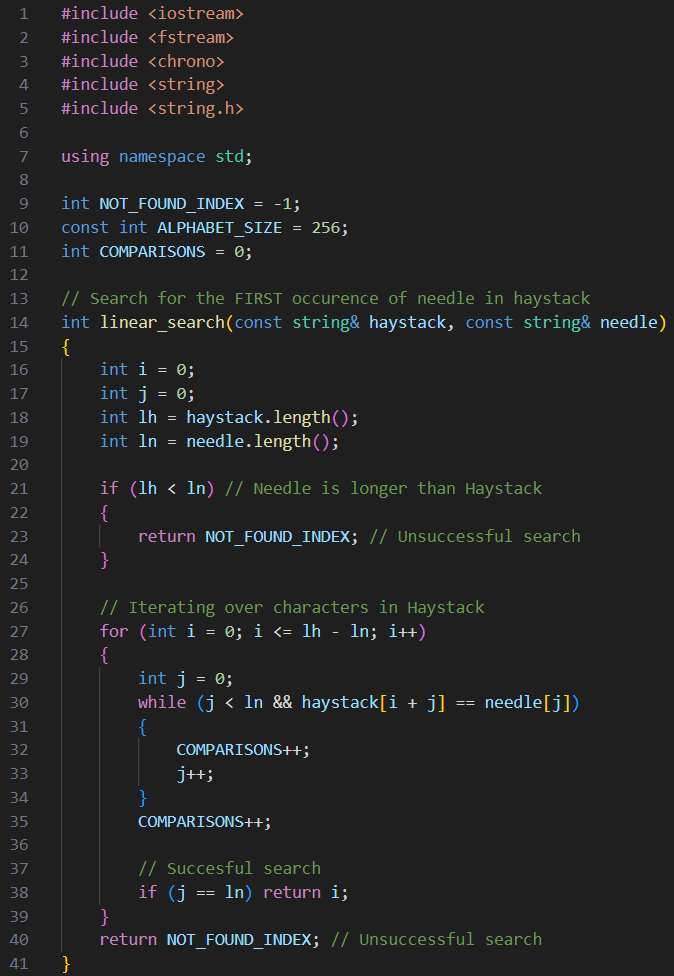


Рисунок 2 – код программы (часть 1)

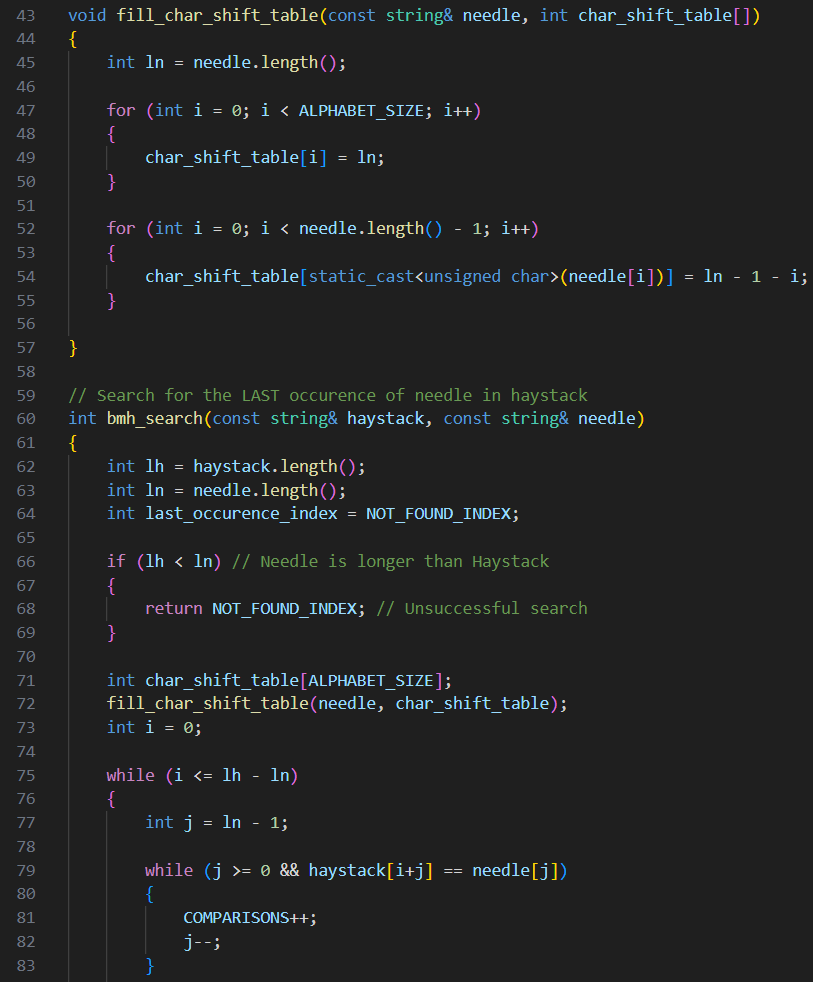


Рисунок 3 – код программы (часть 2)

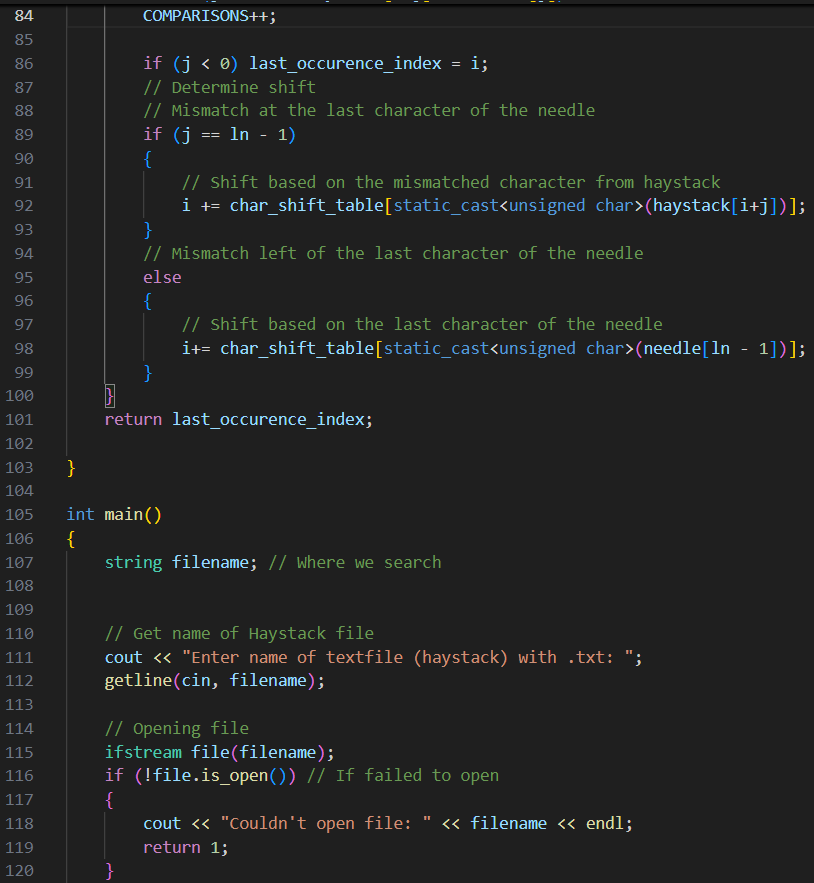


Рисунок 4 – код программы (часть 3)

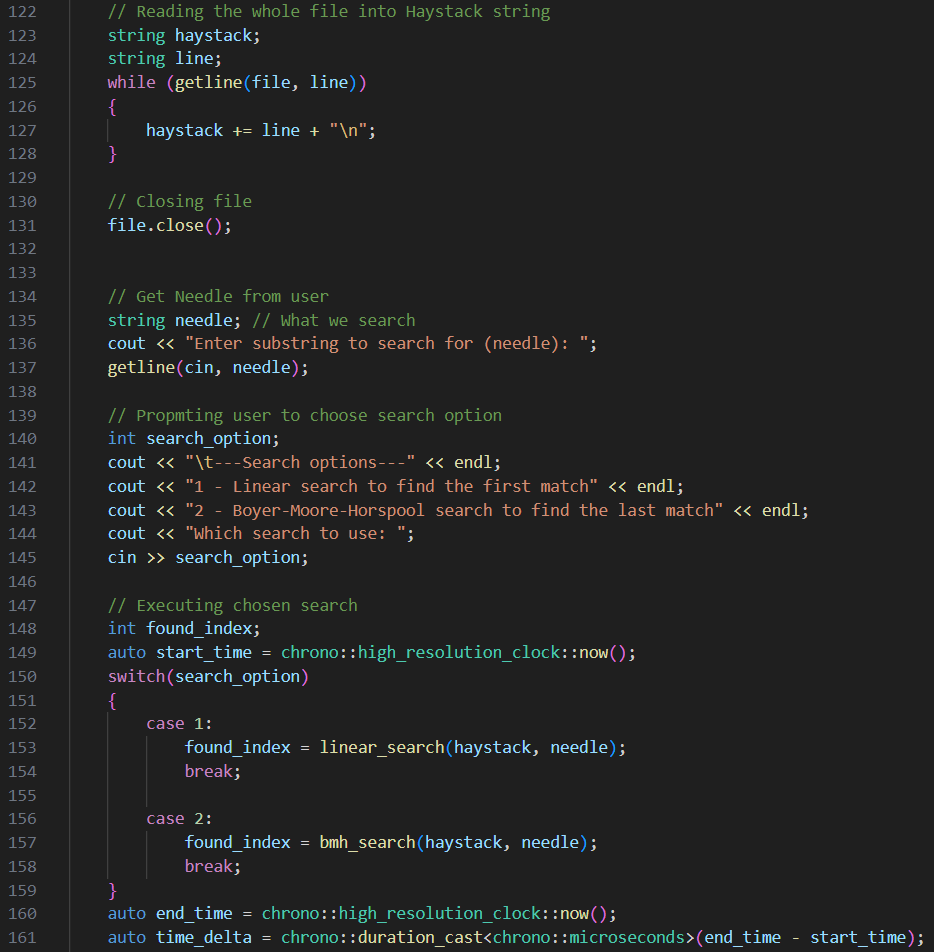


Рисунок 5 – код программы (часть 5)

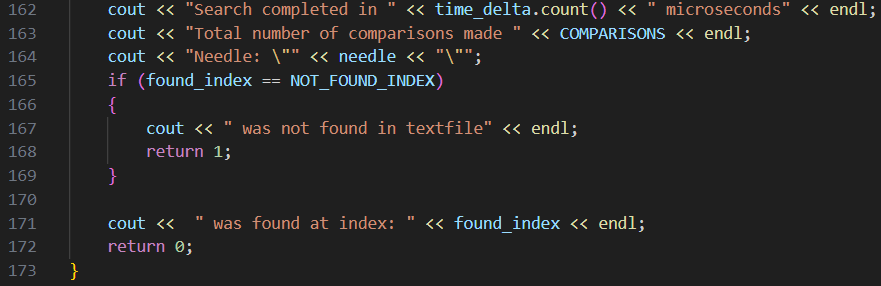
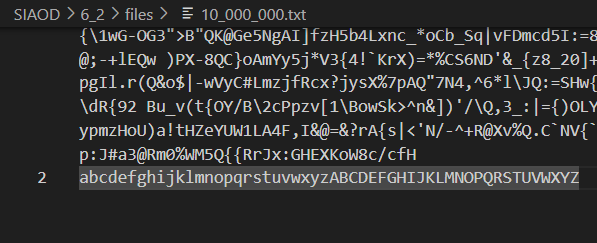
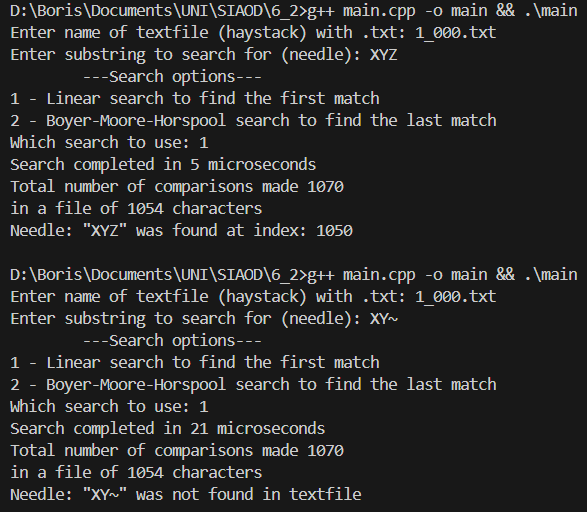


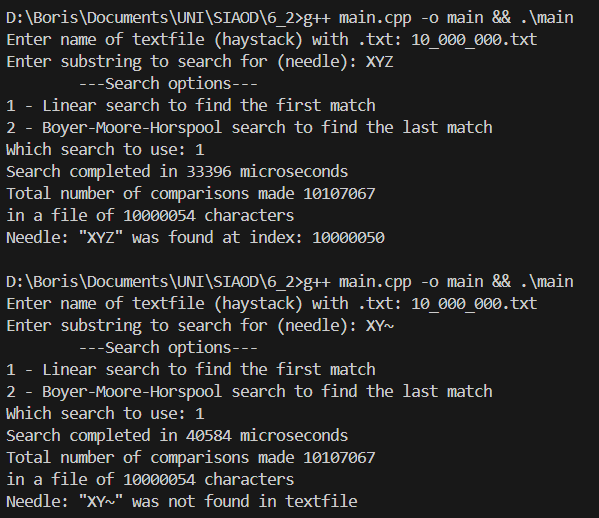
Рисунок 6 – код программы (часть 6)

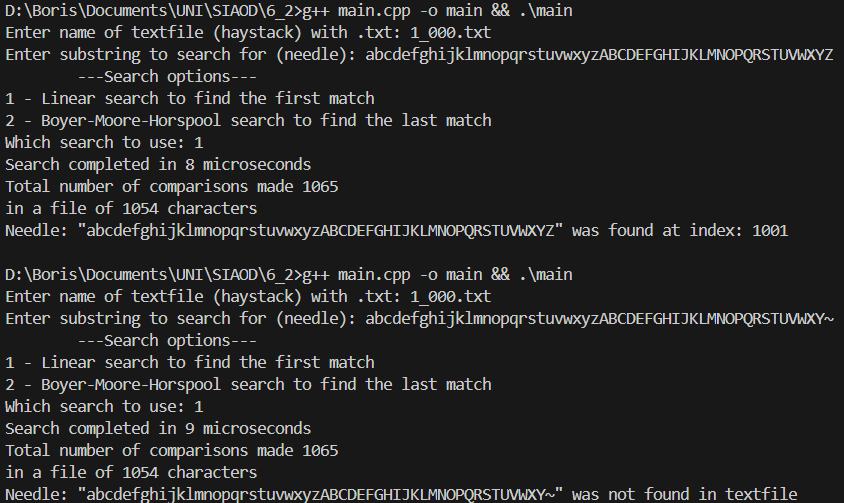
### Результаты тестирования

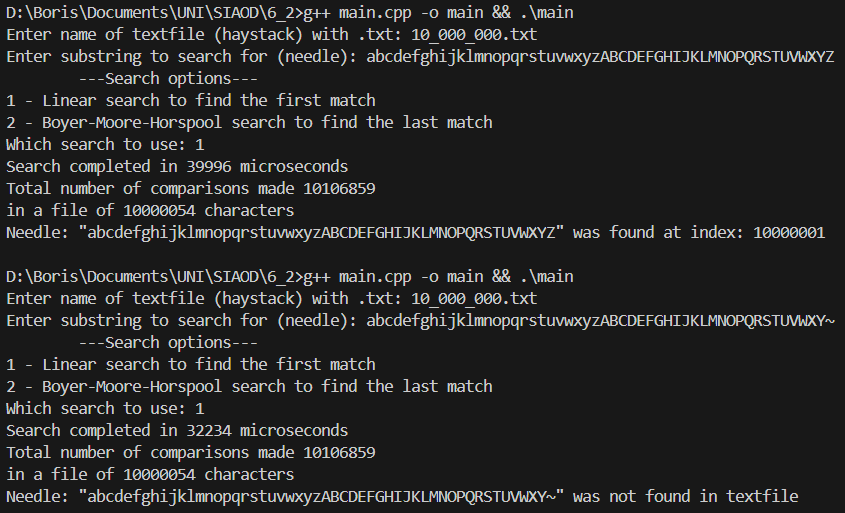
Выполним тестирование программы (рис. 7-15). Обозначим каждый из запусков четырёхзначным двоичным кодом. 0-й разряд означает успех/неуспех поиска, 1-й разряд означает длину текста (0 – короткий, 1 – длинный), 2-й разряд означает длину образца (0 – короткий, 1 – длинный), 3-й разряд означает выбранный алгоритм поиска (0 – прямой поиск, 1 – алгоритм Бойера-Мура-Хорспула). В итоге будет проведено 16 тестовых запусков. Текстовые файлы заполнены случайными символами ASCII с 32ого по 125ый, последней строкой вставлена специальная последовательность.

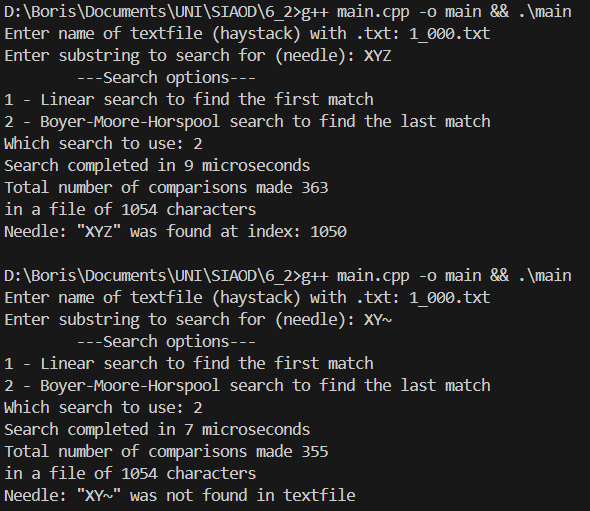
  
Рисунок 7 – Пример текстового файла

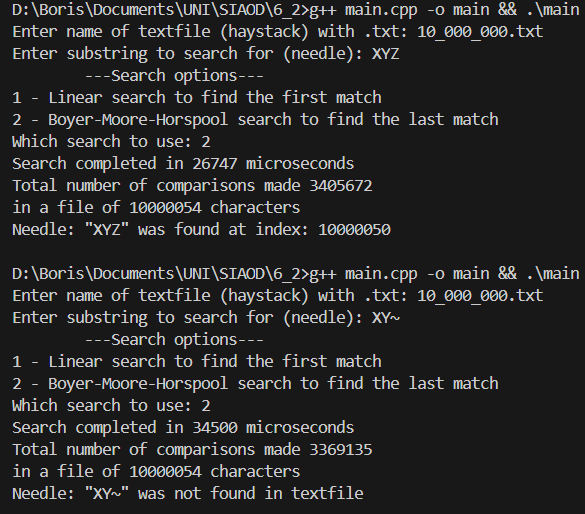
  
Рисунок 8 – Тестирование запусков 0000 и 0001

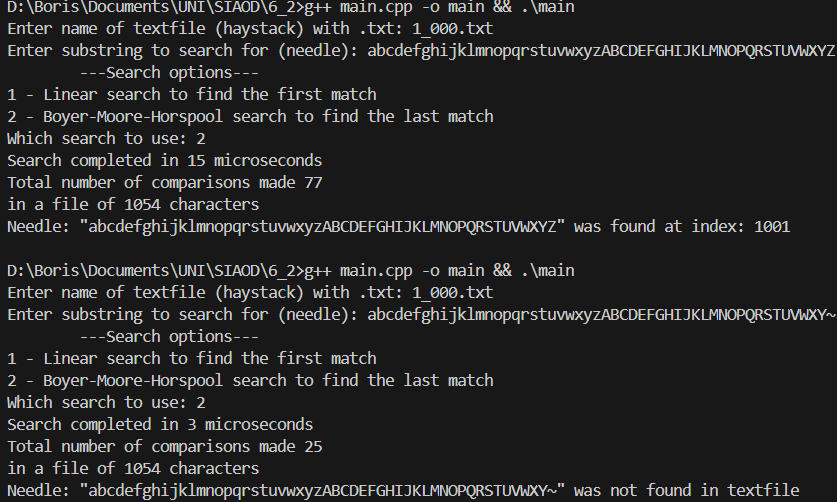
  
Рисунок 9 – Тестирование запусков 0010 и 0011

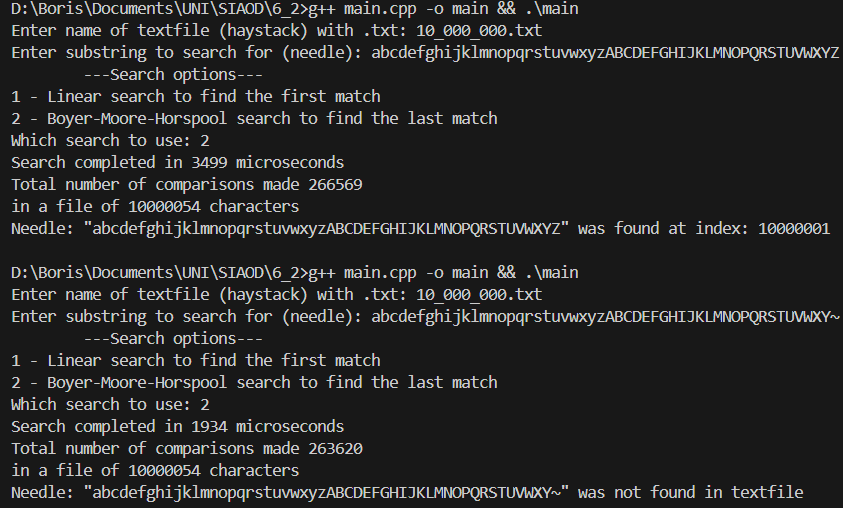
  
Рисунок 10 – Тестирование запусков 0100 и 0101

  
Рисунок 11 – Тестирование запусков 0110 и 0111

  
Рисунок 12 – Тестирование запусков 1000 и 1001

  
Рисунок 13 – Тестирование запусков 1010 и 1011

  
Рисунок 14 – Тестирование запусков 1100 и 1101

  
Рисунок 15 – Тестирование запусков 1110 и 1111

Тестирование показало, что программа работает корректно. Результаты тестирования занесём в таблицы 1, 2, 3, 4 для успешного и неуспешного прямого поиска, и для успешного и неуспешного поиска алгоритмом Бойера-Мура-Хорспула соответственно.

Таблица 1 — тестирование успешного прямого поиска

|  |  |  |
| --- | --- | --- |
| Длина текста / Длина образца | 3 символа | 52 символа |
| 1\_000 символов | 5 мкс / 1070 сравнений | 8 мкс / 1065 сравнений |
| 10\_000\_000 символов | 33396 мкс / 10107067 сравнений | 39996 мкс / 10106859 сравнений |

Таблица 2 — тестирование неуспешного прямого поиска

|  |  |  |
| --- | --- | --- |
| Длина текста / Длина образца | 3 символа | 52 символа |
| 1\_000 символов | 21 мкс / 1070 сравнений | 9 мкс / 1065 сравнений |
| 10\_000\_000 символов | 40584 мкс / 10107067 сравнений | 32234 мкс / 10106859 сравнений |

Таблица 3 — тестирование успешного поиска Бойера-Мура-Хорспула

|  |  |  |
| --- | --- | --- |
| Длина текста / Длина образца | 3 символа | 52 символа |
| 1\_000 символов | 9 мкс / 363 сравнений | 15 мкс / 77 сравнений |
| 10\_000\_000 символов | 26747 мкс / 3405672 сравнений | 3499 мкс / 266569 сравнений |

Таблица 4 — тестирование неуспешного поиска Бойера-Мура-Хорспула

|  |  |  |
| --- | --- | --- |
| Длина текста / Длина образца | 3 символа | 52 символа |
| 1\_000 символов | 7 мкс / 355 сравнений | 3 мкс / 25 сравнений |
| 10\_000\_000 символов | 34500 мкс / 3369135 сравнений | 1934 мкс / 263620 сравнений |

Получили что для текстового файла длины N, заполненного произвольными символами, и образца длины M, сложность алгоритма прямого поиска равна O(N). Сложность алгоритма Бойера-Мура-Хорспула же будет равна O(N/M).

В худшем случае оба алгоритма имеют сложность O(N\*M).

# ВЫВОД

В результате выполнения работы были освоены навыки по реализации алгоритмов поиска в тексте, в частности были изучены алгоритм прямого поиска и алгоритм Бойера-Мура-Хорспула. Последний является чаще более эффективным, особенно при работе с длинными текстами и образцами.

# СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ

1. Рысин, М. Л. Введение в структуры и алгоритмы обработки данных : учебное пособие / М. Л. Рысин, М. В. Сартаков, М. Б. Туманова. — Москва : РТУ МИРЭА, 2022 — Часть 2 : Поиск в тексте. Нелинейные структуры данных. Кодирование информации. Алгоритмические стратегии — 2022. — 111 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/310826 (дата обращения: 28.09.2024).
2. Документация по языку С++ [Электронный ресурс]. URL: https://docs.microsoft.com/ruru/cpp/cpp/ (дата обращения 28.09.2024).