**Київський національний університет імені Тараса Шевченка**

**факультет комп’ютерних наук та кібернетики**

**кафедра інформаційних систем**

**Лабораторна робота № 8**

**“Реалізація різних алгоритмів пошуку шаблона в тексті”**

**Виконав студент 2 курсу**

**групи К-29**

**Аксой Денис Гюралпович**

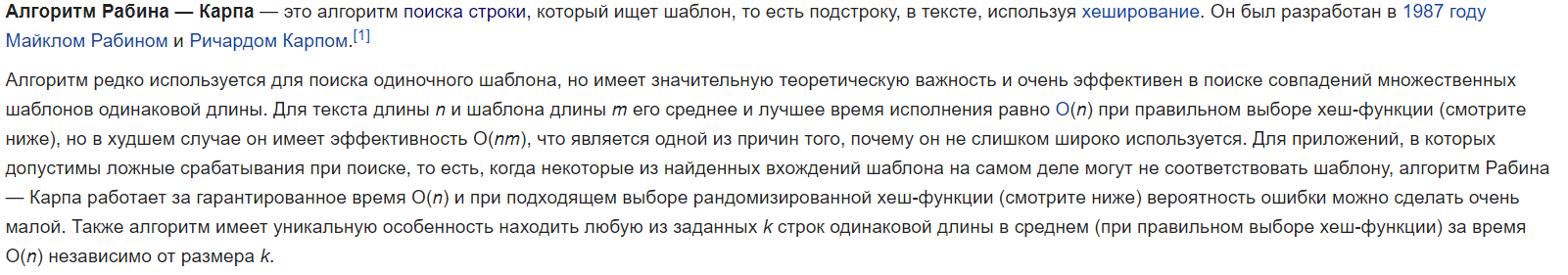
**2018**

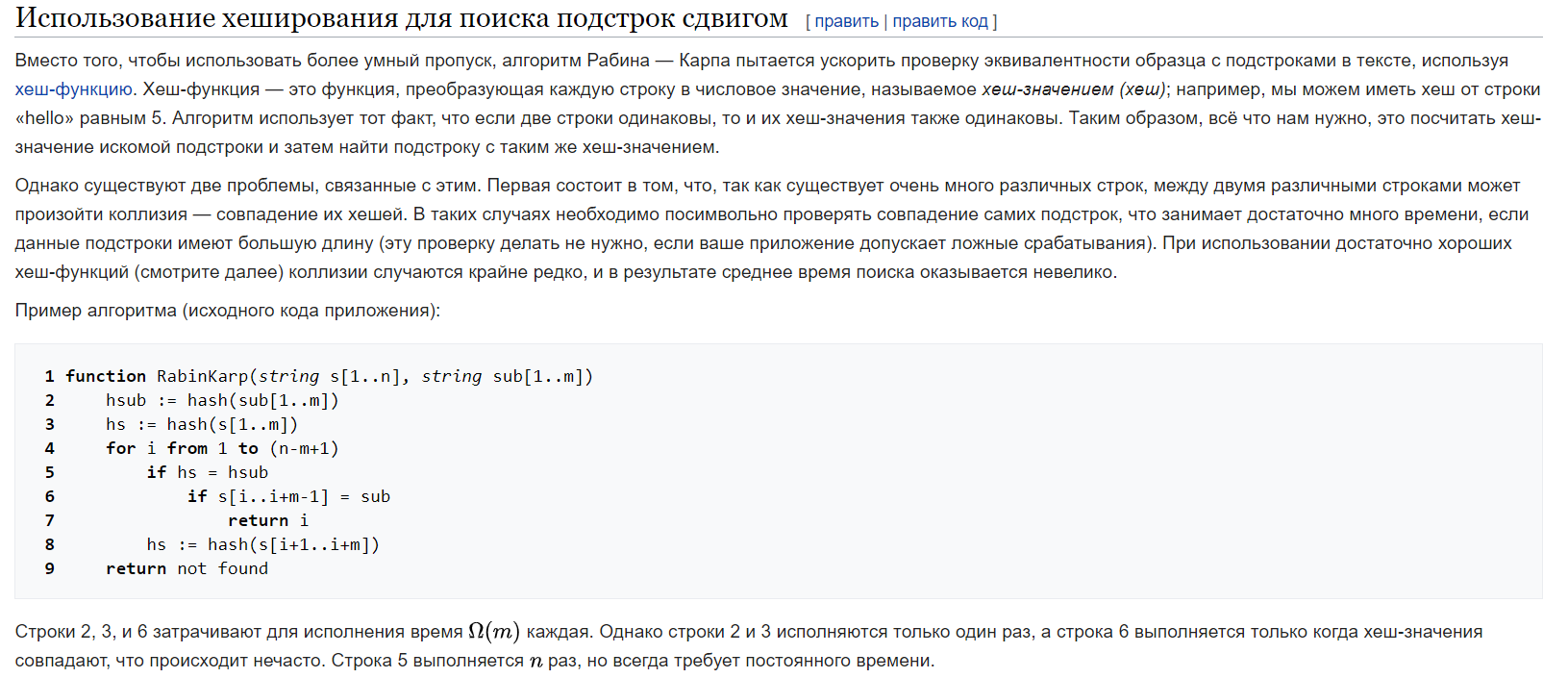
1. Зміст завдання та уточнення умови

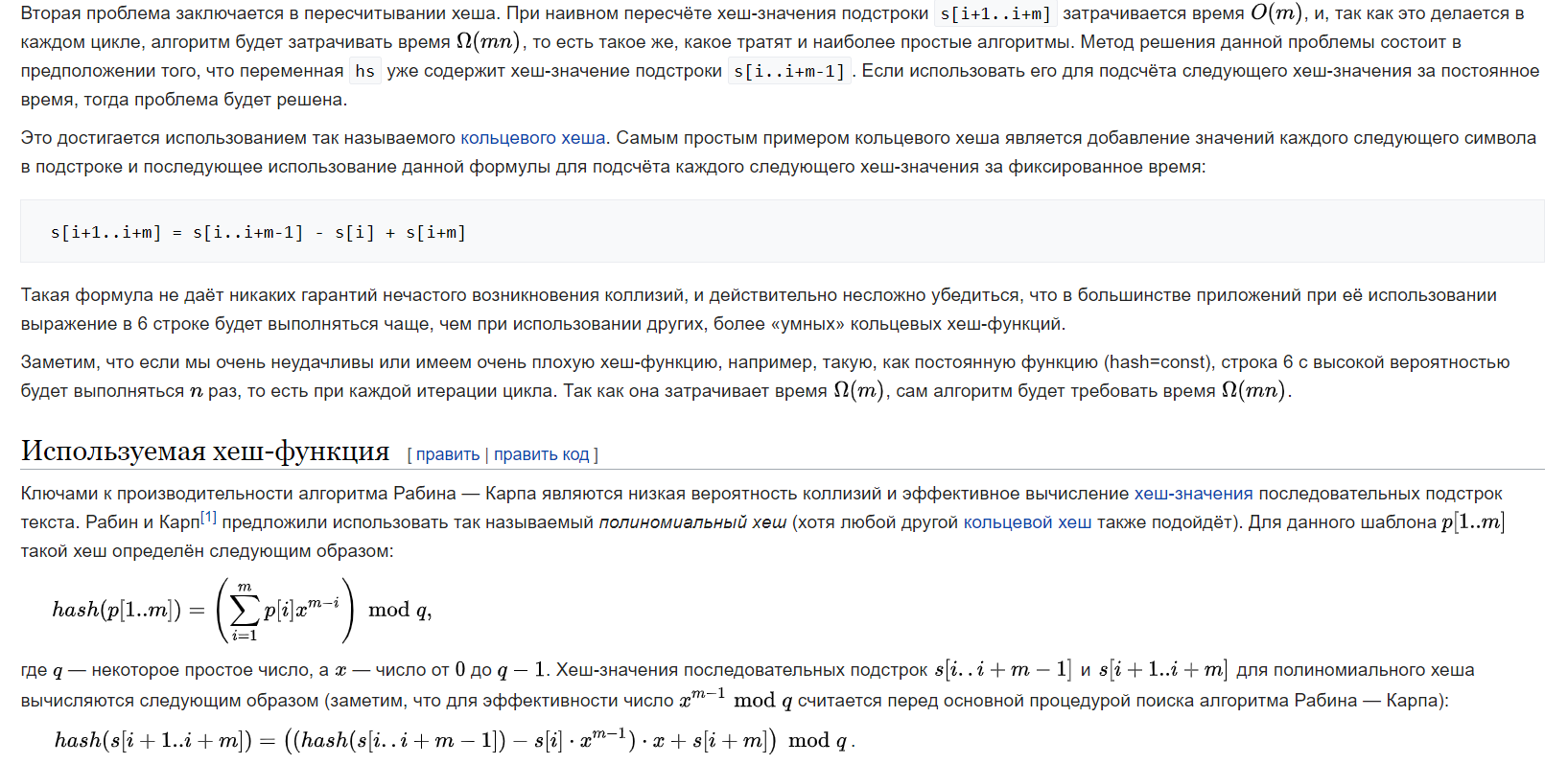
Реалізувати такі алгоритми пошуку шаблона в тексті:

1. Рабіна-Карпа
2. Боєра-Мура
3. Боєра-Мура-Хорспула
4. Кнута-Моріса-Прата
5. Опис алгоритму

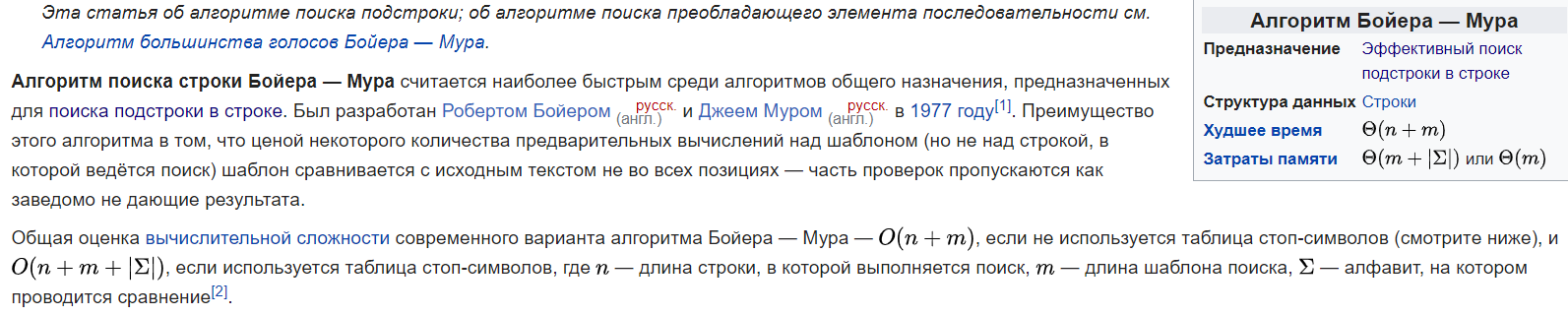
1)Алгоритм Рабіна-Карпа

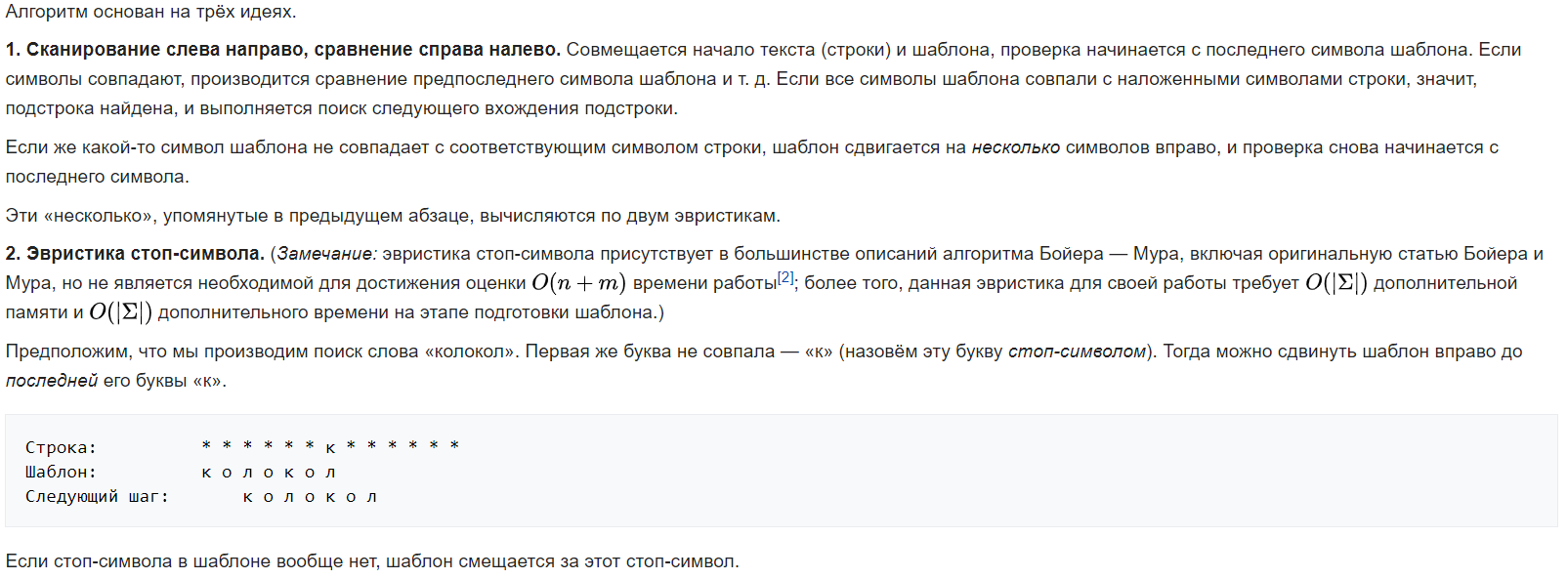


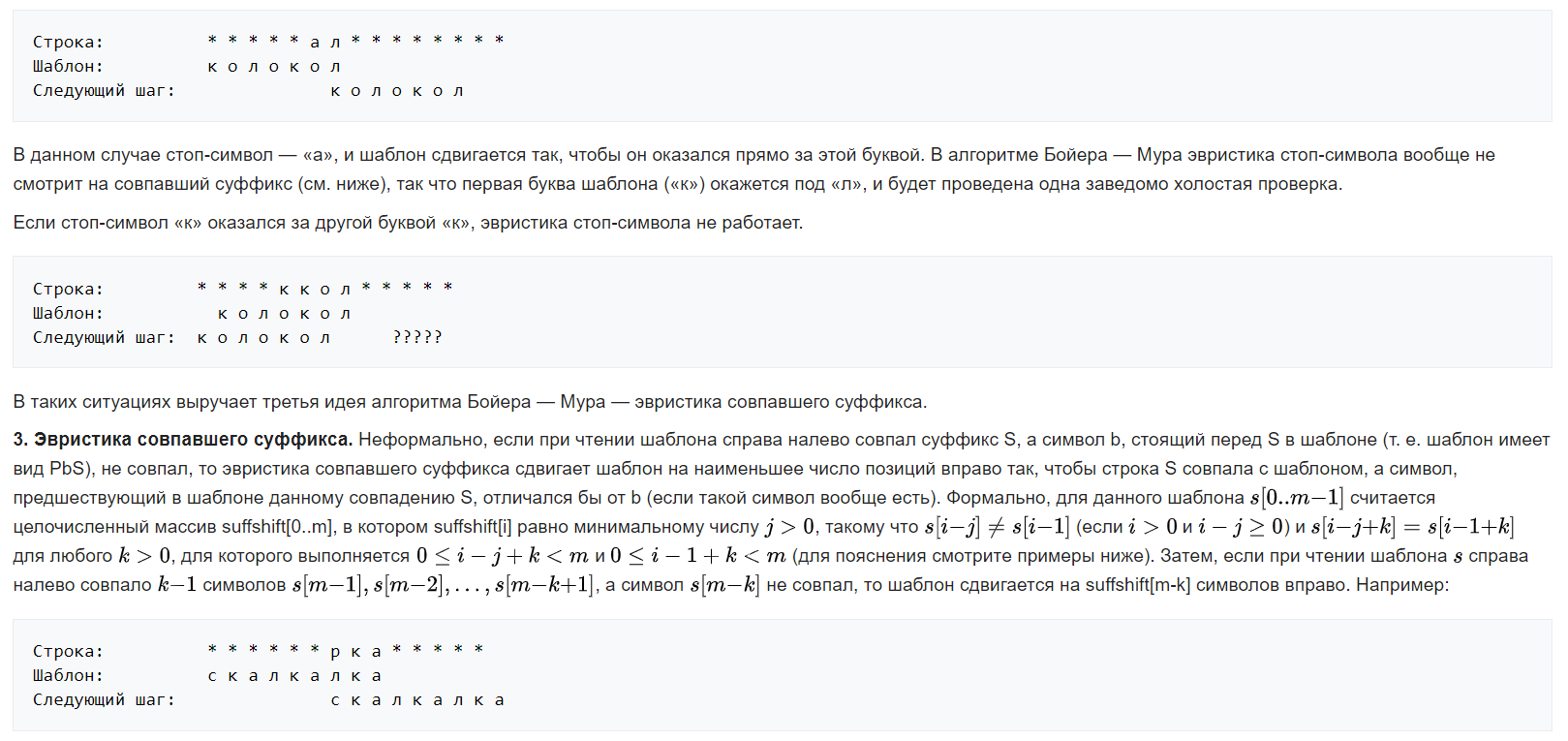


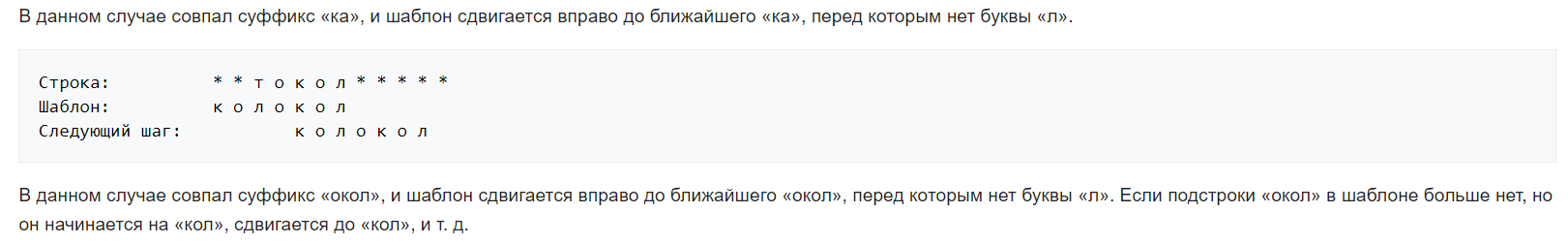


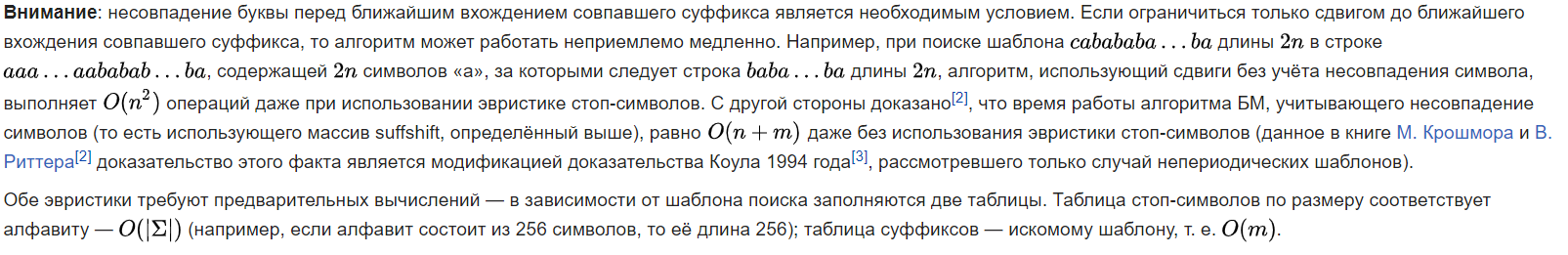
2)Алгоритм Боєра-Муєра

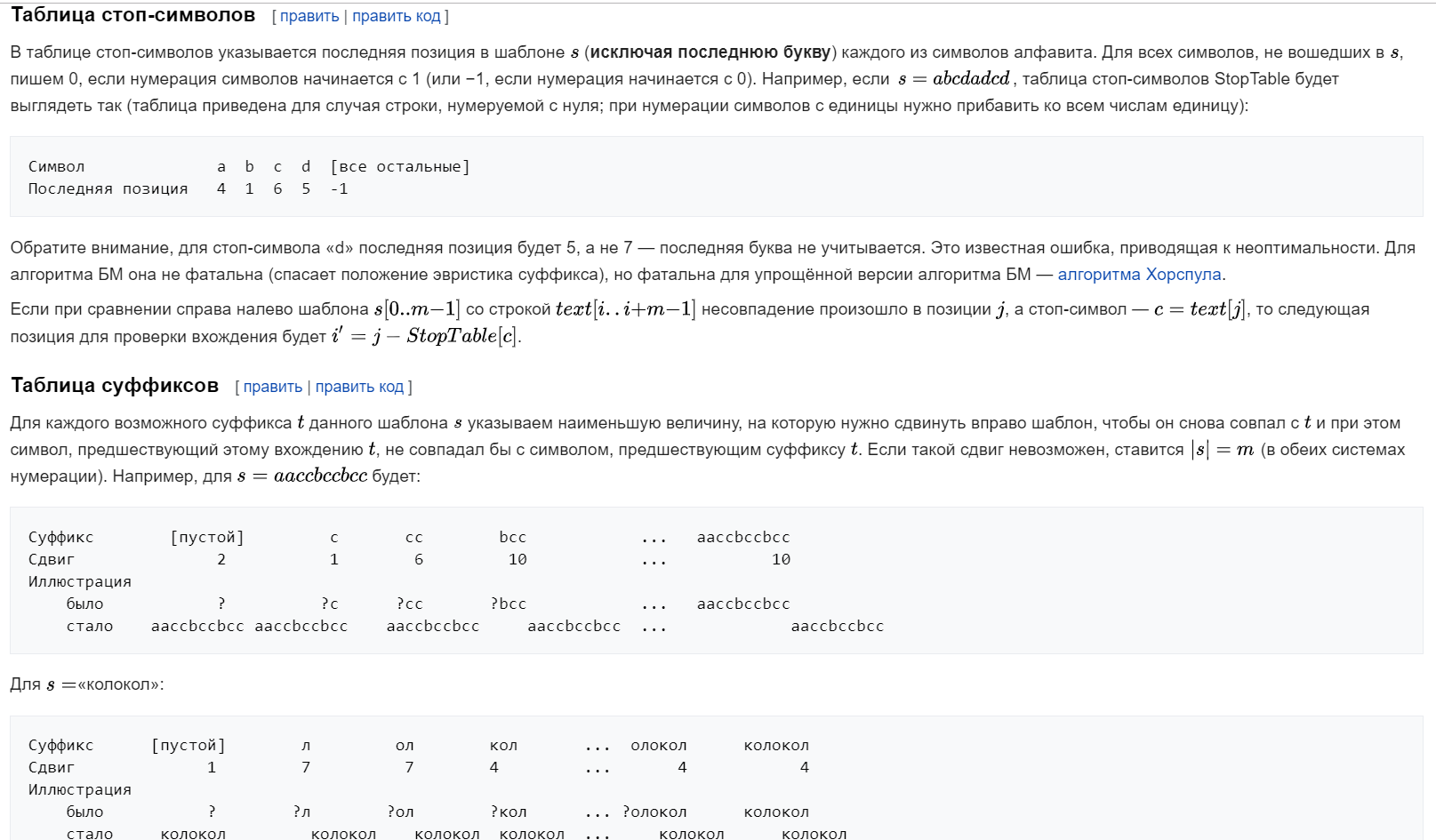




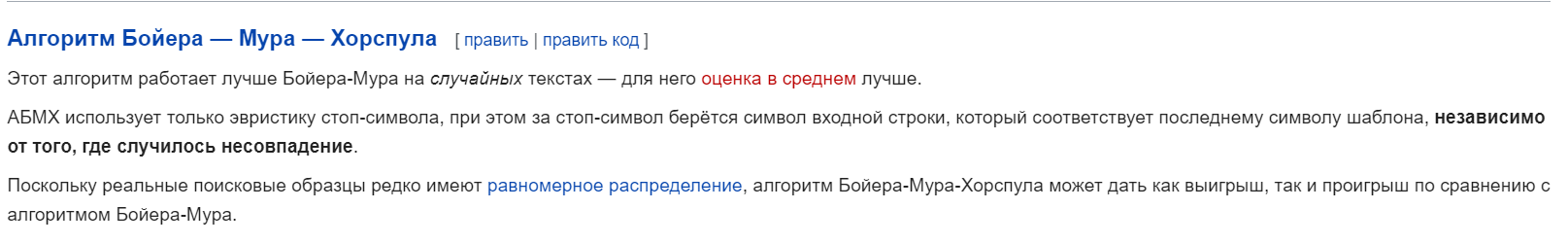




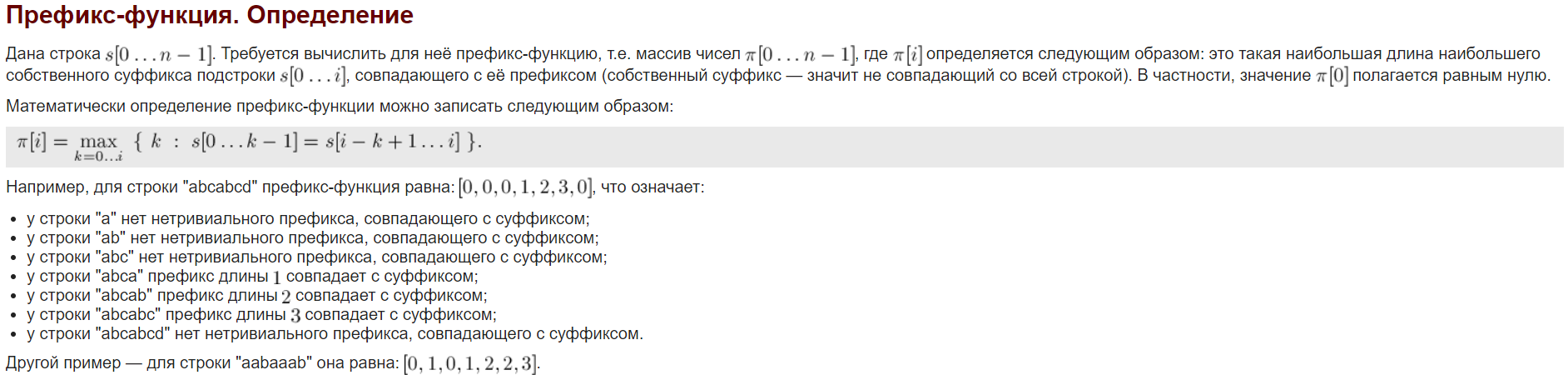


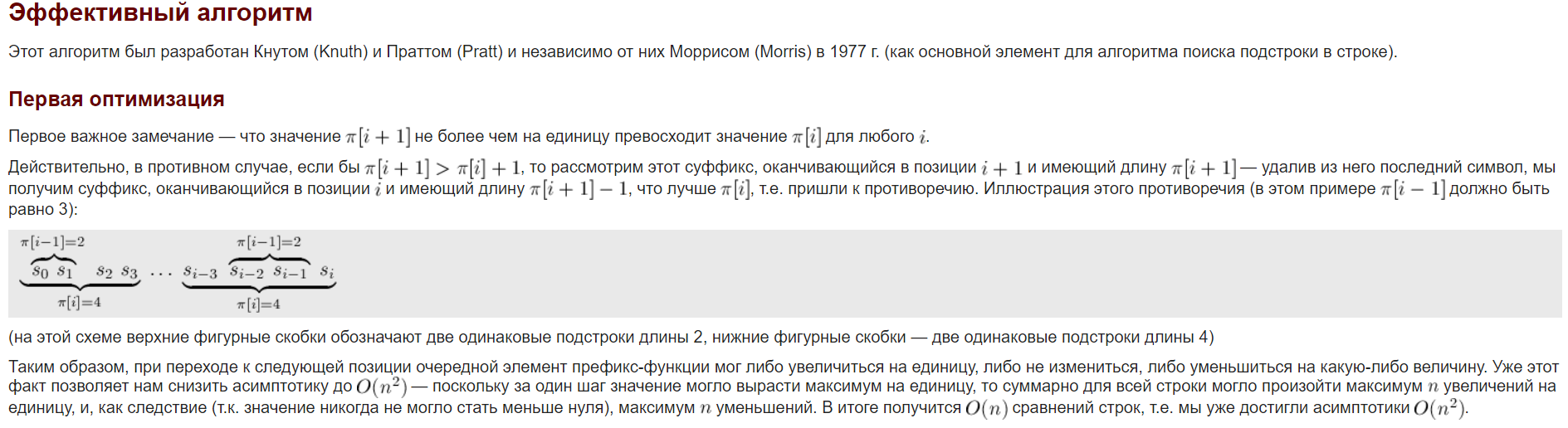


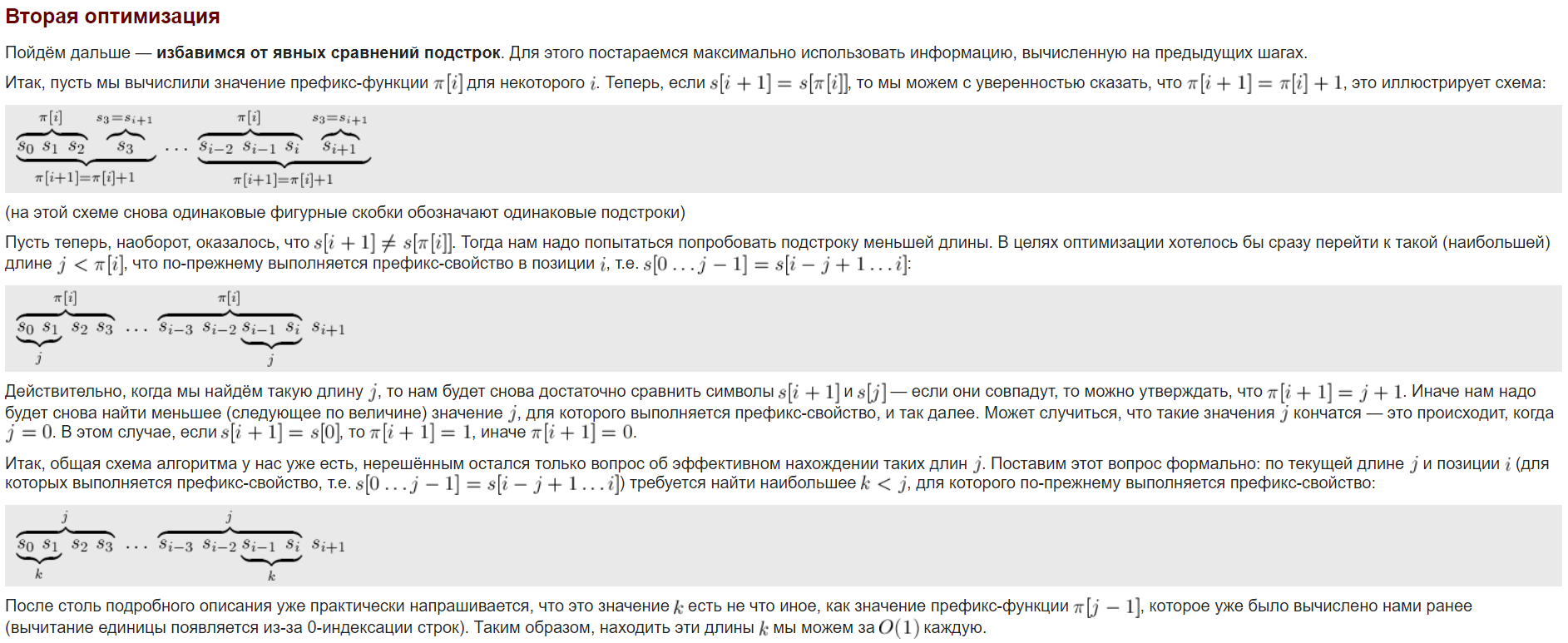
3)Алгоритм Бойера-Мура-Хорспула

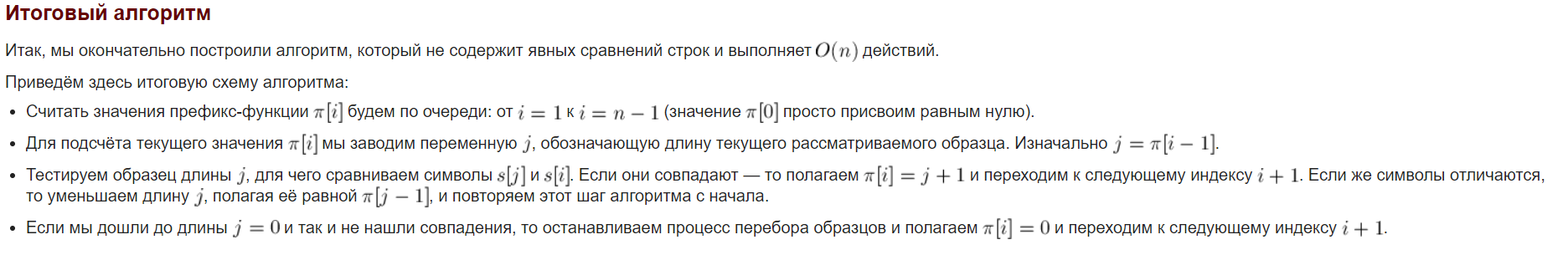


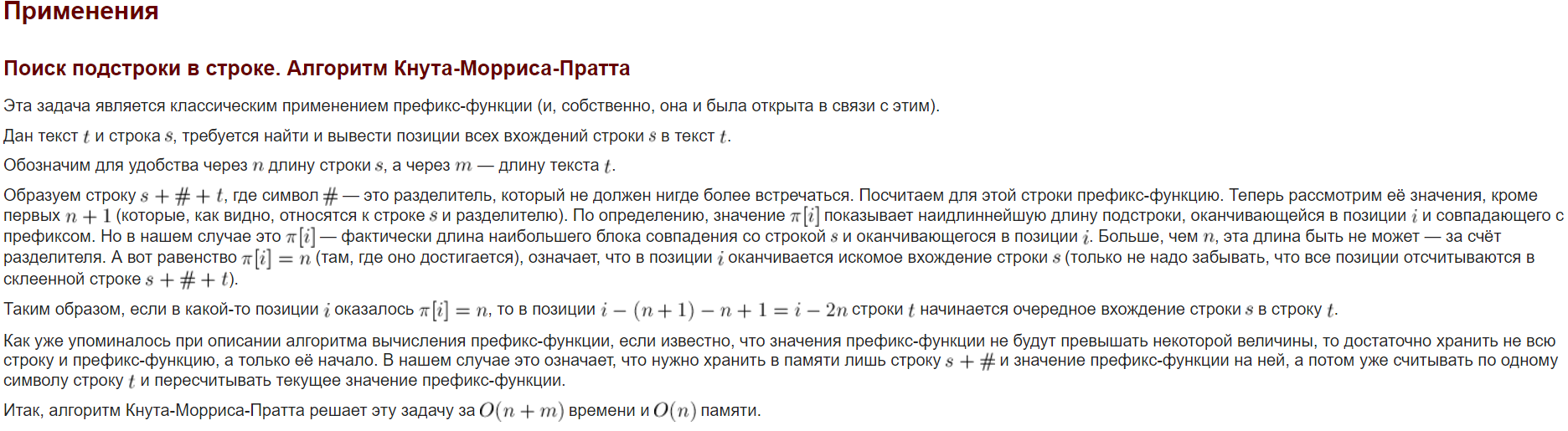
4)Алгоритм Кнута-Морріса-Прата





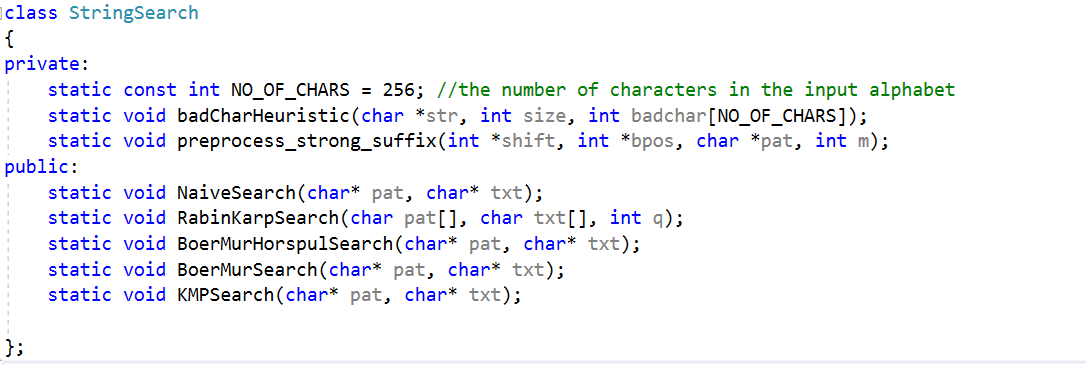






1. Основні модулі програми:

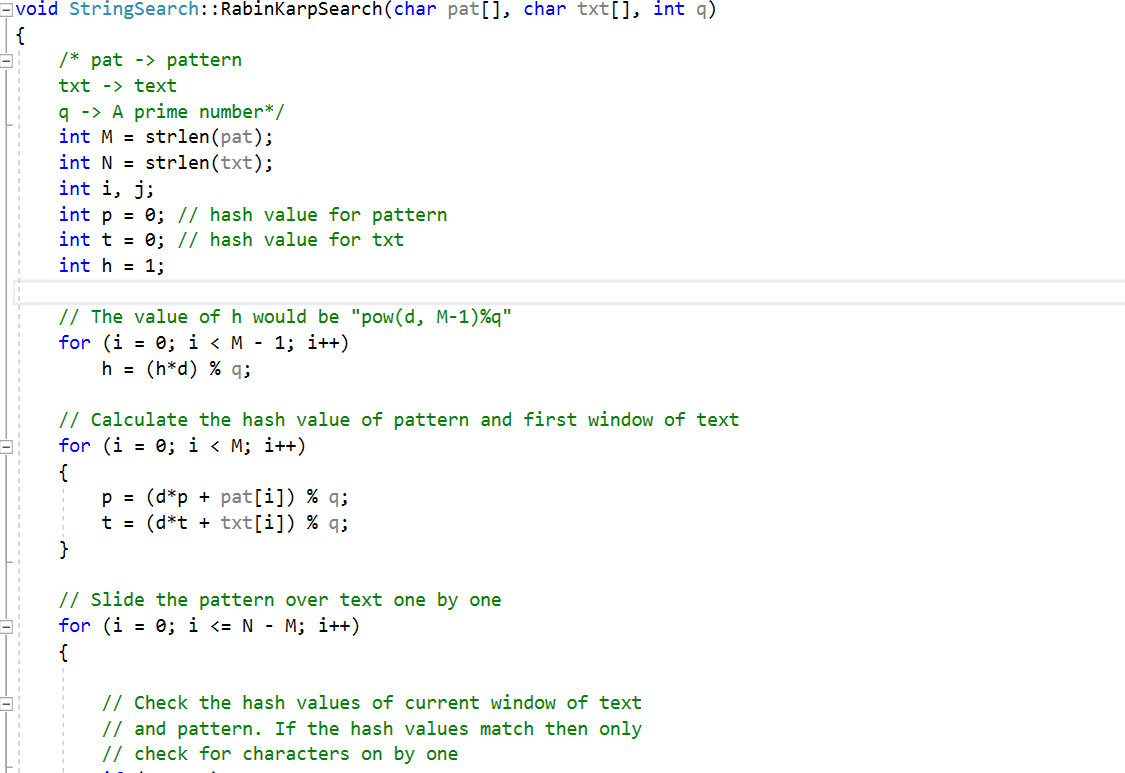
Клас StringSearch з реалізованими алгоритмами пошуку шаблона в тексті. Код згрупований в клас для зручності використання. Далі з основної програми викликаємо по черзі кожен з алгоритмів та фіксуємо час виконання, що виводимо на екран разом з позиціями, на яких був знайдений шаблон в тектсі.

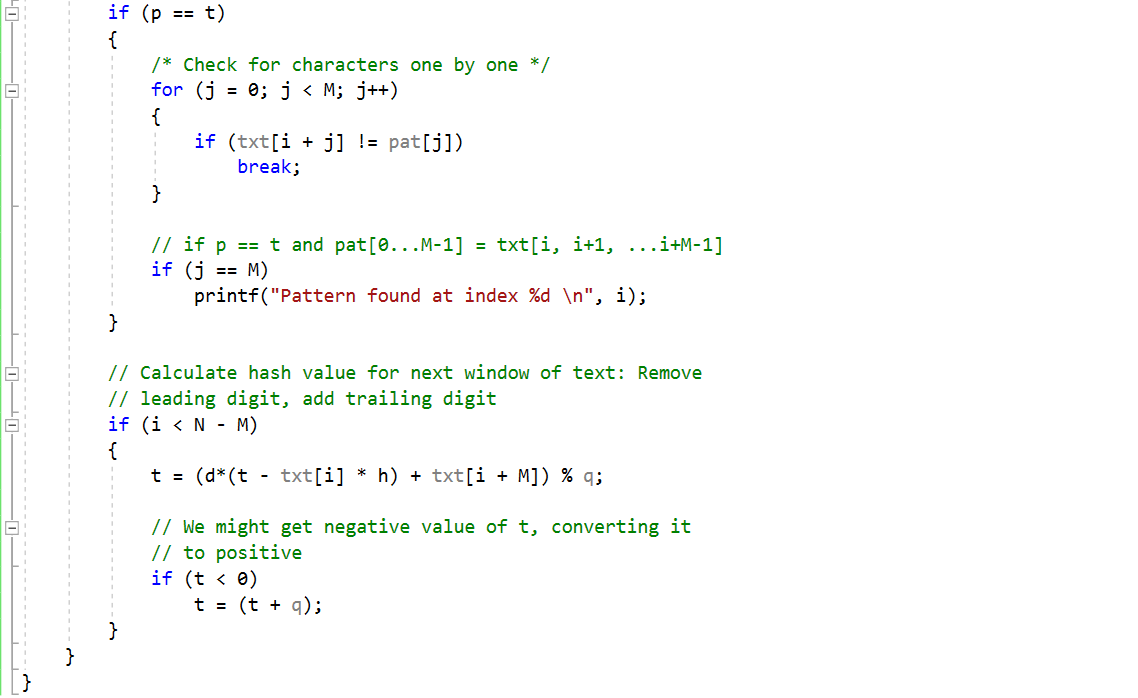


1)Алгоритм Рабіна-Карпа

Спочатку ми підраховуємо хеш-функцію для pat[] і txt[0,m-1]

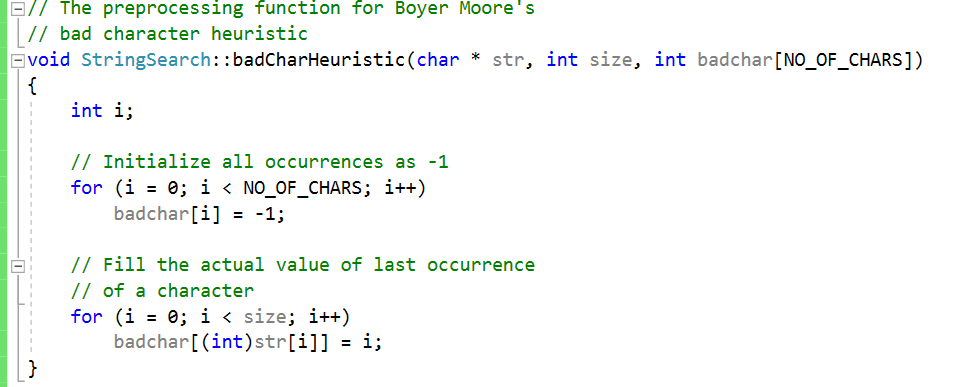
Якщо на i-му(i від 0 до N-M кроці хеші співпадають, то це або колізія(хеші рівні, але рядки ні) або співпадіння з pattern. Тому порівняємо посимвольно text[i...i+M-1] і pat[0,M-1]

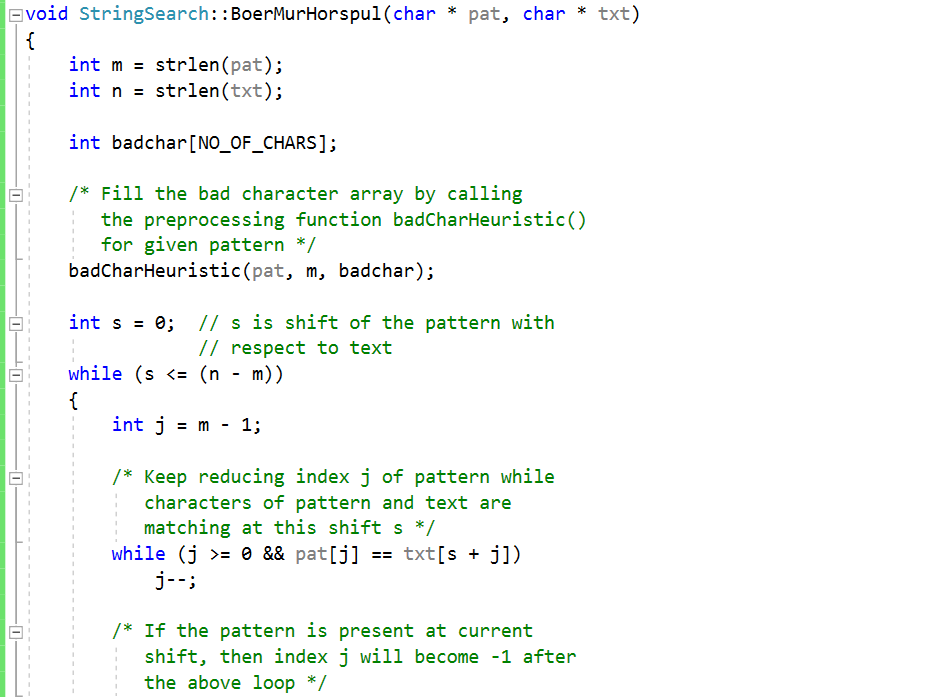
В кінці кожної(окрім останньої) ітерації перераховуємо хеш-значення функції для підрядка txt[i+1…i+M].

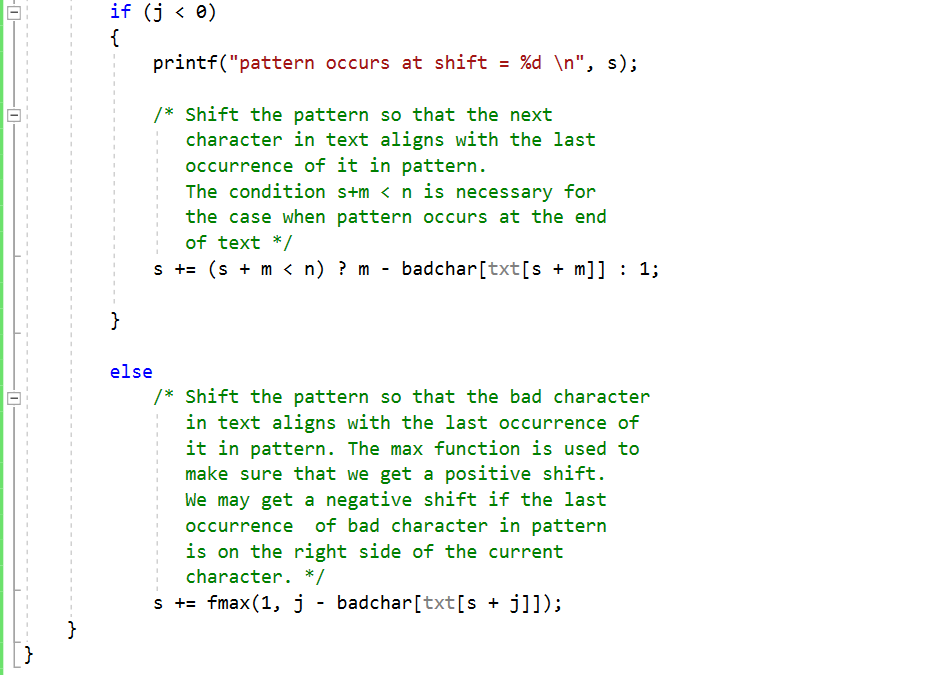


2)Алгоритм Боєра-Мура-Хорспула

Складається з двох частин: препроцесинг – побудова таблиці bad character[NO\_OF\_CHARS] та пошуку - порівнюємо символи шаблону справа наліво з текстом. Після повного збігу шаблон зміщується так, щоб наступний символ в тексті(після шаблону) порівнявся з останньою появою цього ж символа в шаблоні. Після невідповідності на певному символі шаблон зміщується так, щоб mismatch символ порінявся з останньою появою його в шаблоні.



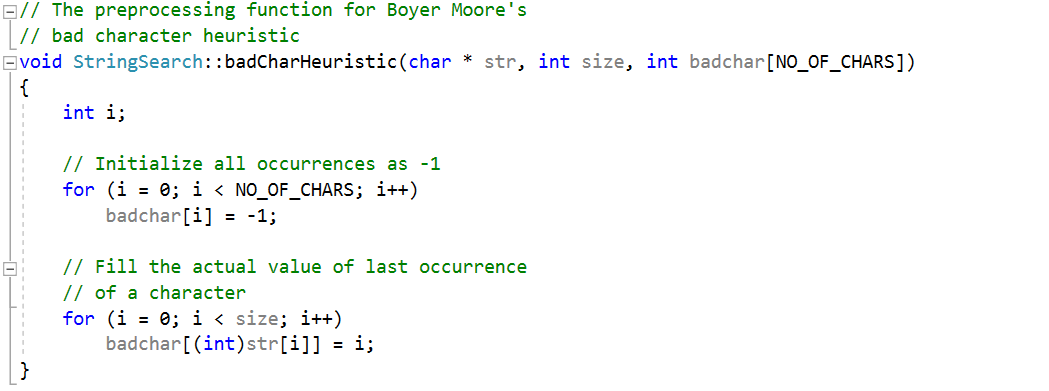


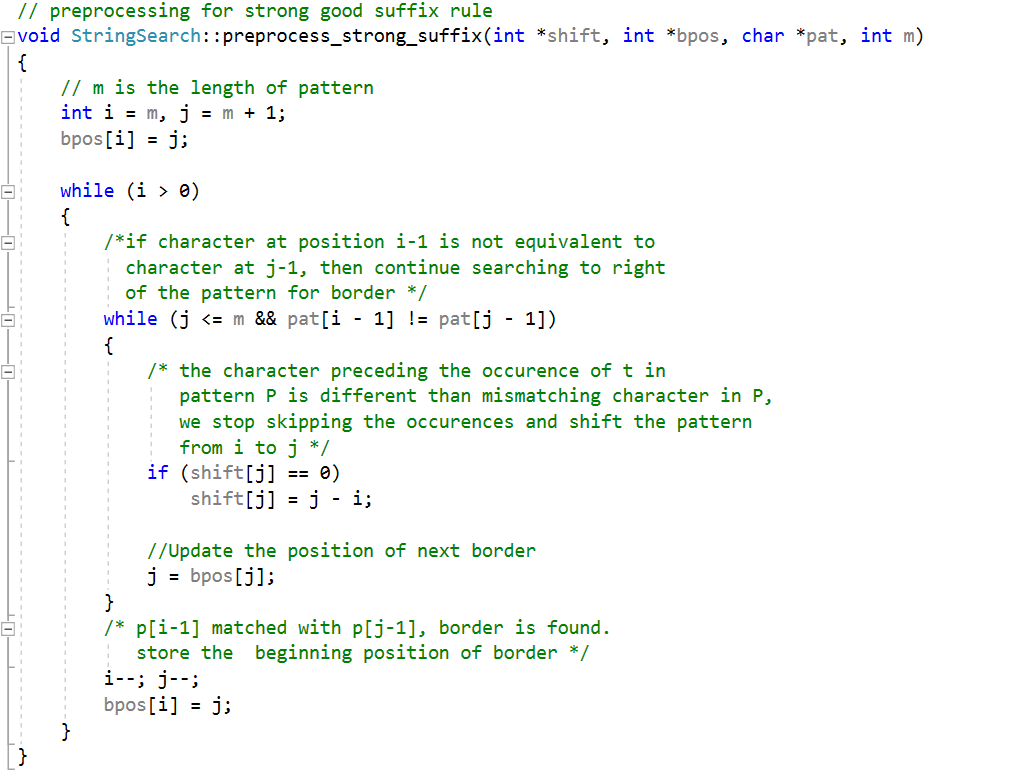


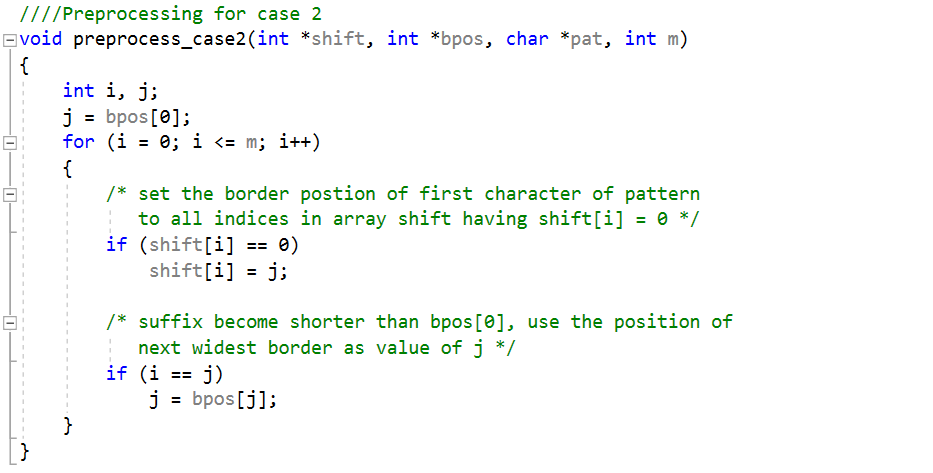
3) Алгоритм Боєра-Мура

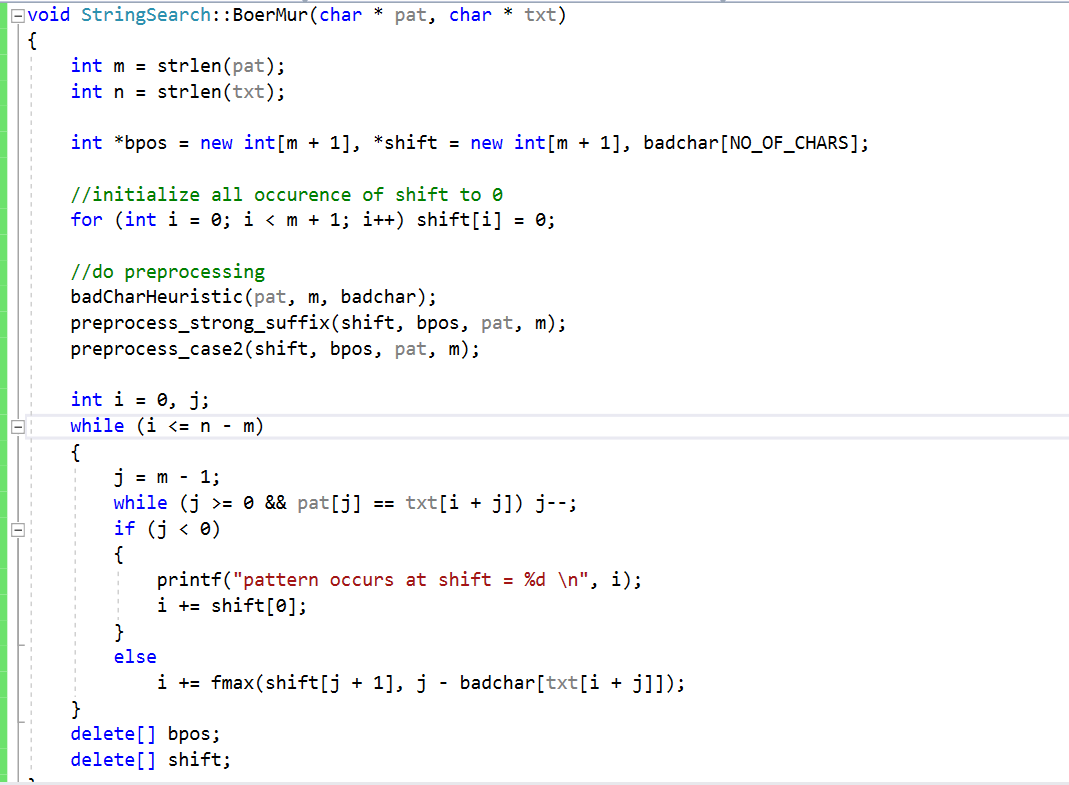
Алгоритм препроцесингу алгоритму Бойєра-Мура складається з попередньої побудови таблиці для bad character та обидвох випадків для good suffix.

Алгоритм пошуку порівнює символи шаблону справа наліво з текстом. Після повного збігу шаблон зміщується залежно від того, наскільки це дозволяє його найширша межа(shift[0]). Після невідповідності шаблон зміщується на максимум значень, заданих good suffix та bad character евристиками.





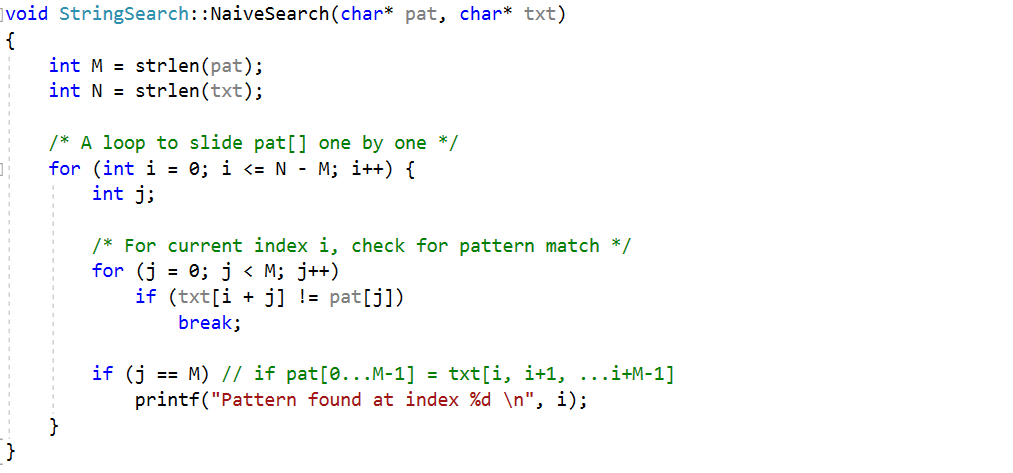




4)Наївний алгоритм пошуку

Не потребує особливих пояснень. Перебираємо всі підстроки text[0…M-1], txt[1…M]…txt[N-M…N-1] на співпадіння з pattern[0…M-1] посимвольно.

У випадку неспівпадіння на певному символі переходимо до наступнії підстроки.



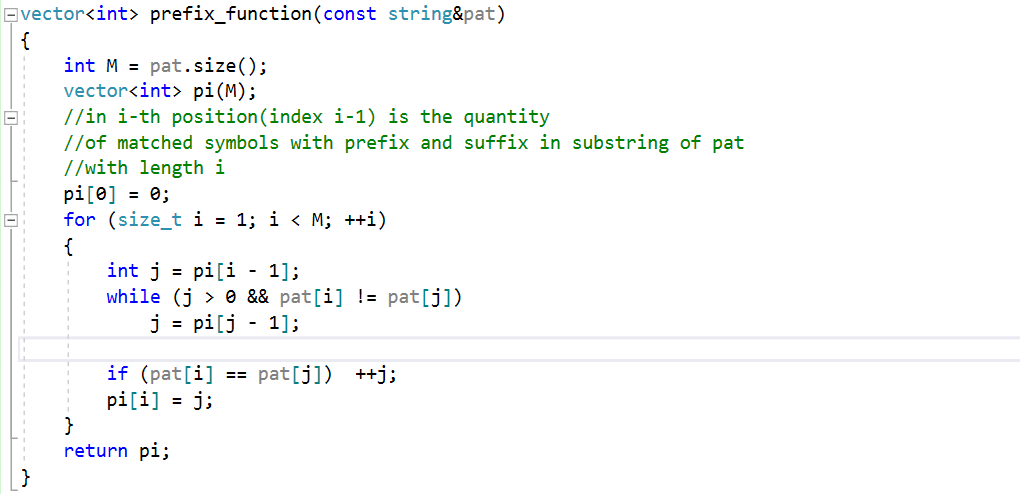
5)Алгоритм Кнута-Морріса-Прата

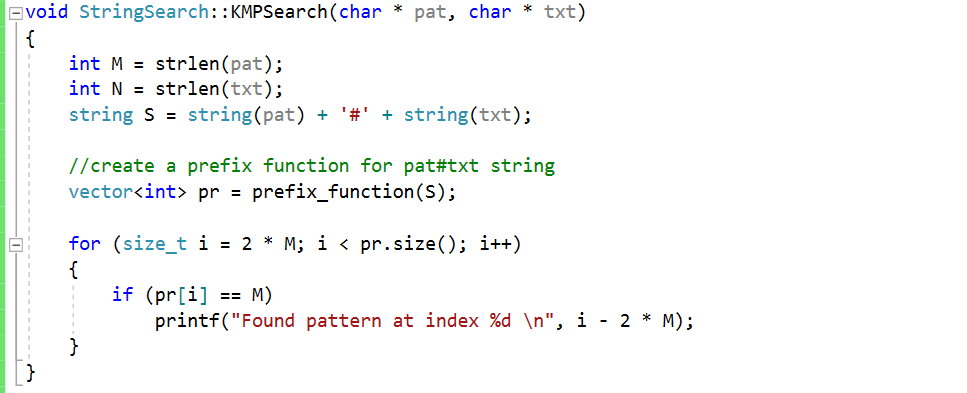
Спочатку будуємо префікс-функцію для рядка pattern#text

Тоді якщо її значення на i-му кроці рівне довжині pattern, то це означає, що на

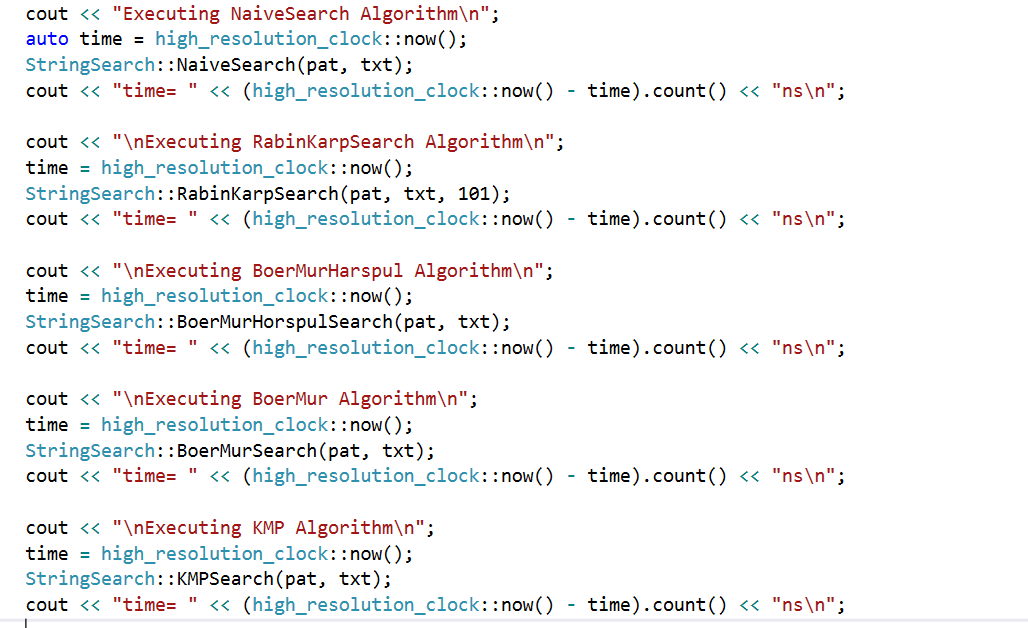
Позиції [i-2\*pat.size()…i] в text знаходиться pattern

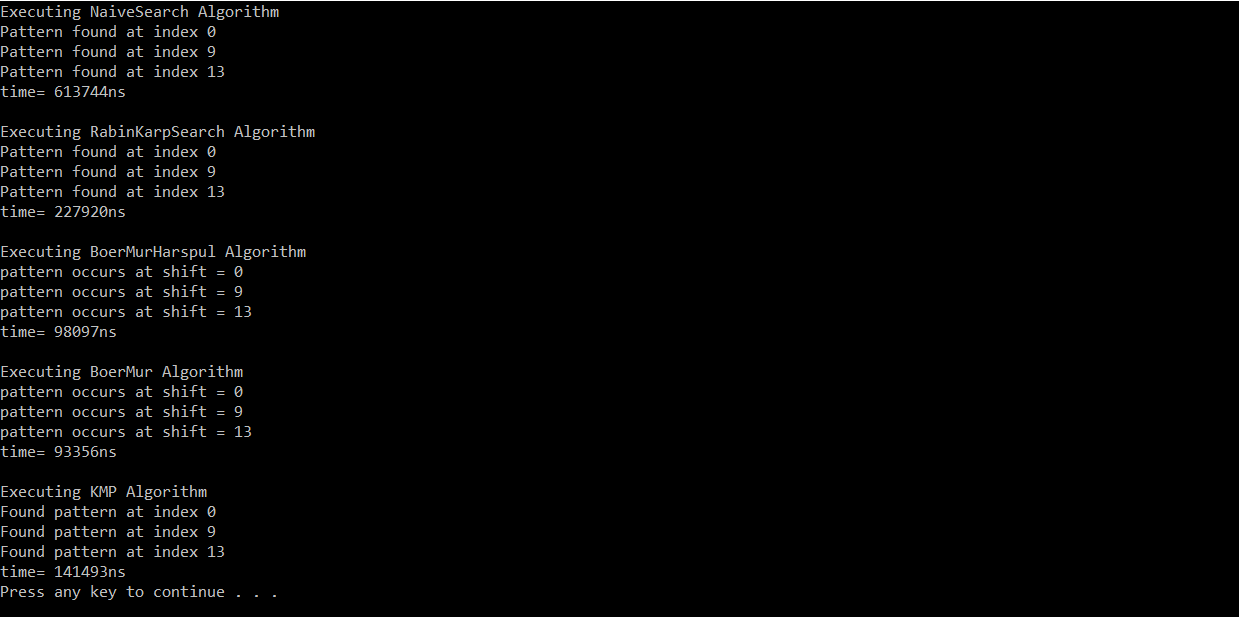
Зміщення йде через розділовий символ #(символ, якого немає ні pattern, ні в text) та pattern, що приписаний перед text

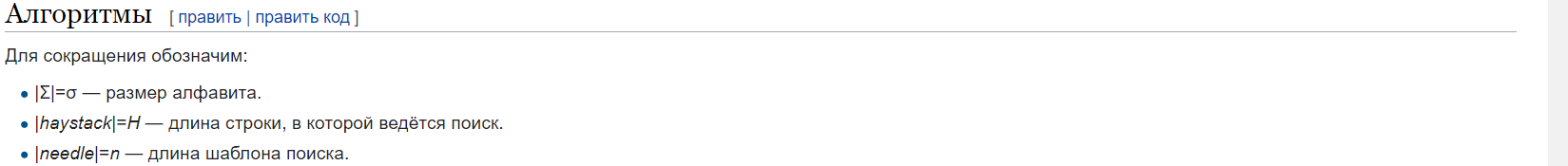


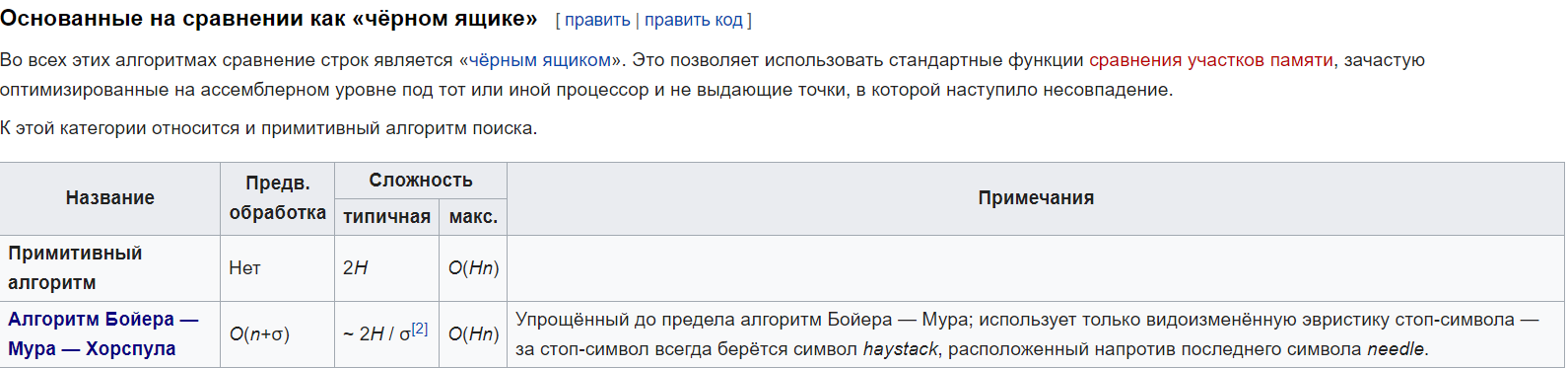


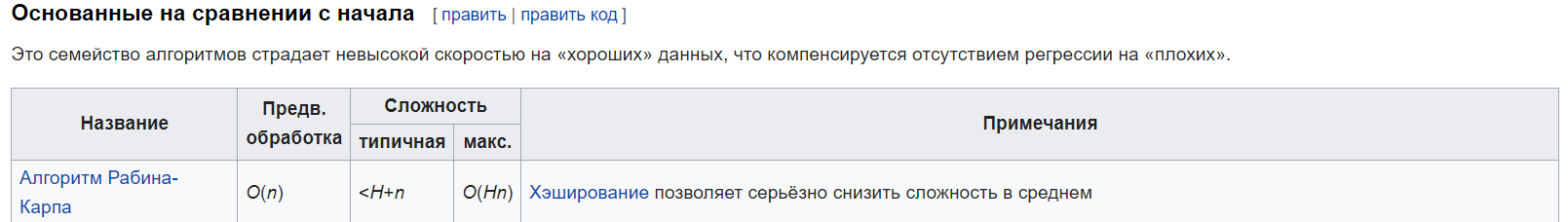
1. Порівняльна характеристика

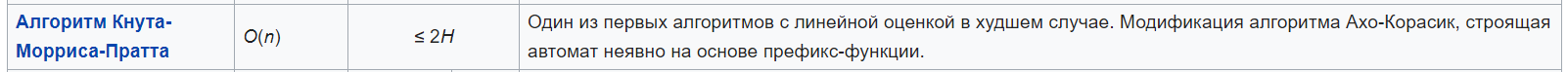


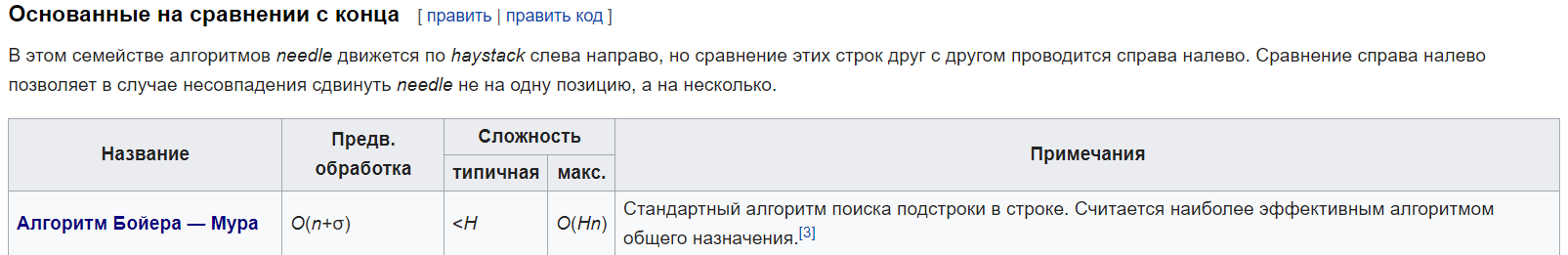












1. Використані джерела:

<https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%B8%D1%81%D0%BA_%D0%BF%D0%BE%D0%B4%D1%81%D1%82%D1%80%D0%BE%D0%BA%D0%B8>

[**https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC\_%D0%91%D0%BE%D0%B9%D0%B5%D1%80%D0%B0\_%E2%80%94\_%D0%9C%D1%83%D1%80%D0%B0**](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%91%D0%BE%D0%B9%D0%B5%D1%80%D0%B0_%E2%80%94_%D0%9C%D1%83%D1%80%D0%B0)

[**https://www.geeksforgeeks.org/naive-algorithm-for-pattern-searching/**](https://www.geeksforgeeks.org/naive-algorithm-for-pattern-searching/)

[**https://www.geeksforgeeks.org/rabin-karp-algorithm-for-pattern-searching/**](https://www.geeksforgeeks.org/rabin-karp-algorithm-for-pattern-searching/)

[**https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC\_%D0%A0%D0%B0%D0%B1%D0%B8%D0%BD%D0%B0\_%E2%80%94\_%D0%9A%D0%B0%D1%80%D0%BF%D0%B0**](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%A0%D0%B0%D0%B1%D0%B8%D0%BD%D0%B0_%E2%80%94_%D0%9A%D0%B0%D1%80%D0%BF%D0%B0)

[**http://www.iti.fh-flensburg.de/lang/algorithmen/pattern/bmen.htm**](http://www.iti.fh-flensburg.de/lang/algorithmen/pattern/bmen.htm)

[**https://www.geeksforgeeks.org/boyer-moore-algorithm-for-pattern-searching/**](https://www.geeksforgeeks.org/boyer-moore-algorithm-for-pattern-searching/)

[**https://www.geeksforgeeks.org/boyer-moore-algorithm-good-suffix-heuristic/**](https://www.geeksforgeeks.org/boyer-moore-algorithm-good-suffix-heuristic/)

<http://e-maxx.ru/algo/export_prefix_function>

**https://www.geeksforgeeks.org/kmp-algorithm-for-pattern-searching/**