Пермский национальный исследовательский политехнический университет ПНИПУ

Отчёт по лабораторной работе на тему “Поиски”

Выполнил студент группы РИС-23-3Б:

Смирнов Андрей Сергеевич

Проверила:

Доцент кафедры ИТАС

О.А. Полякова

2024

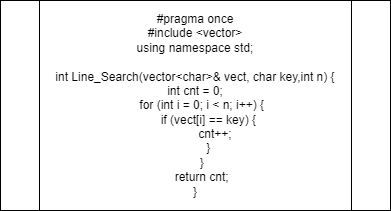
1. Постановка задачи:

Реализовать поиск подстроки в заданной строке, используя алгоритм Линейного поиска

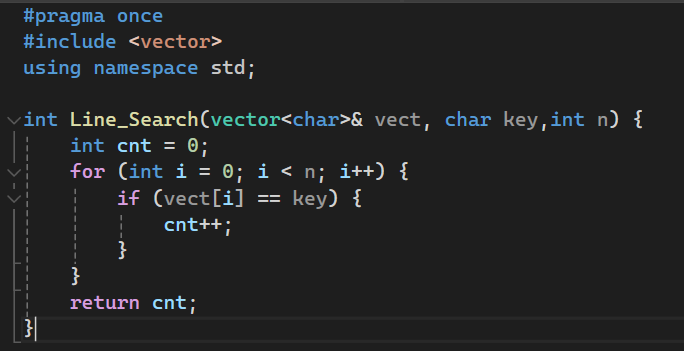
1. Анализ задачи:

* Каждый символ сравнивается с нужным, пока строка не подойдёт к концу

1. Блок-схема:

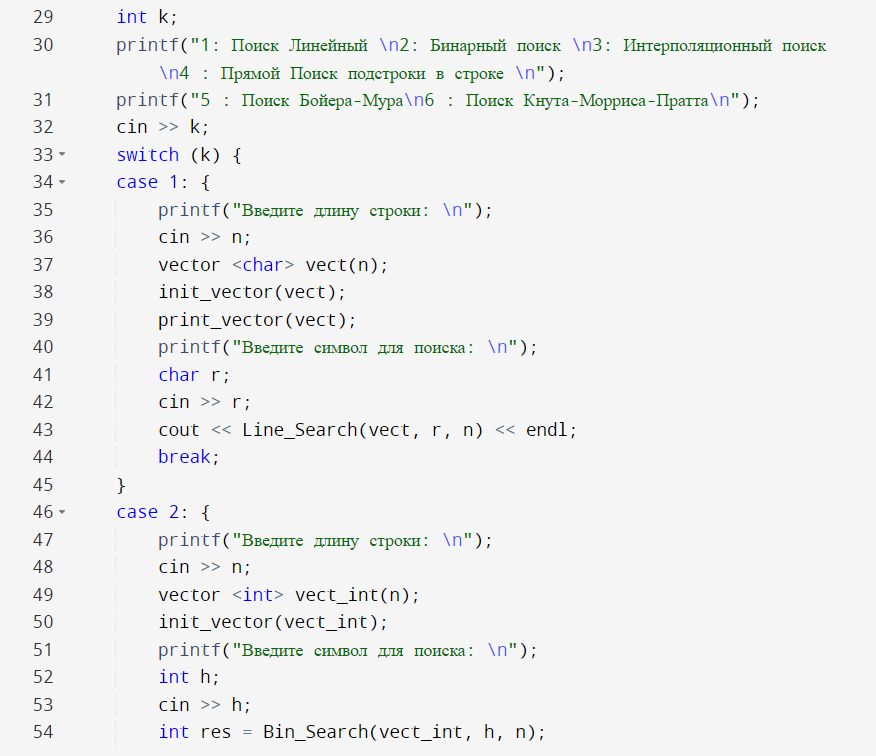


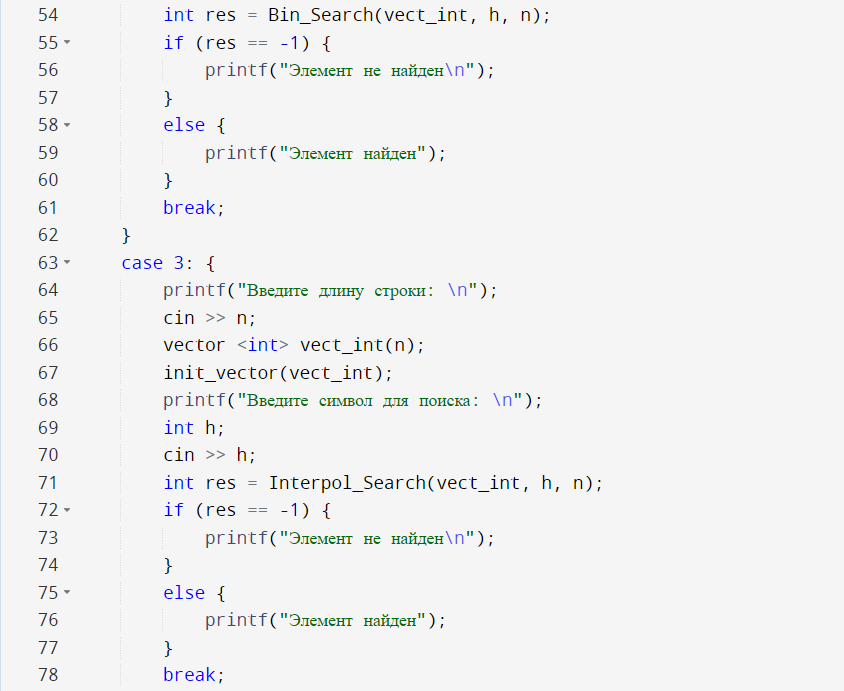
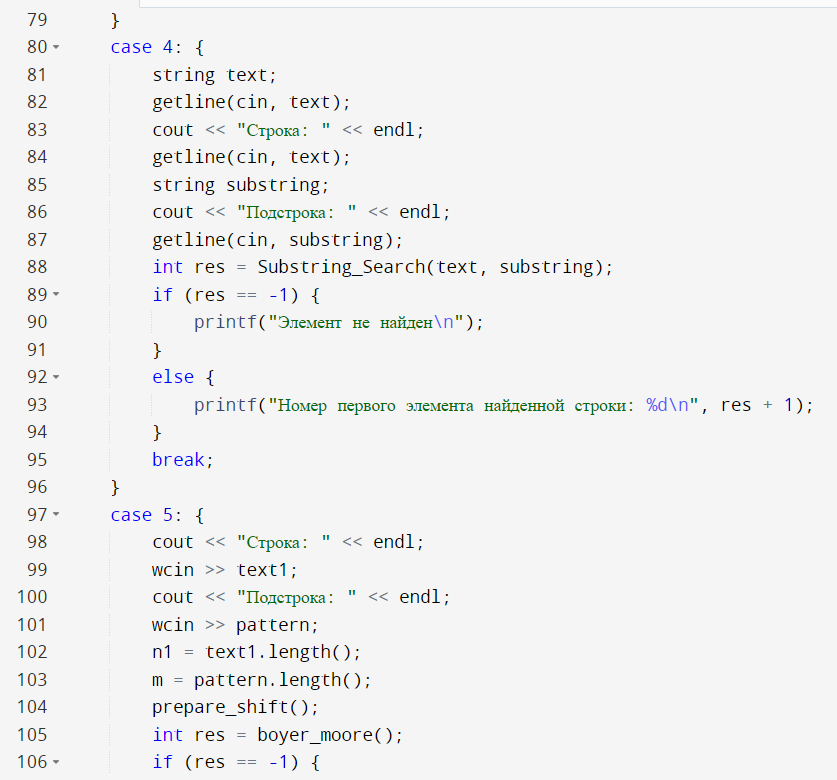
1. Код:

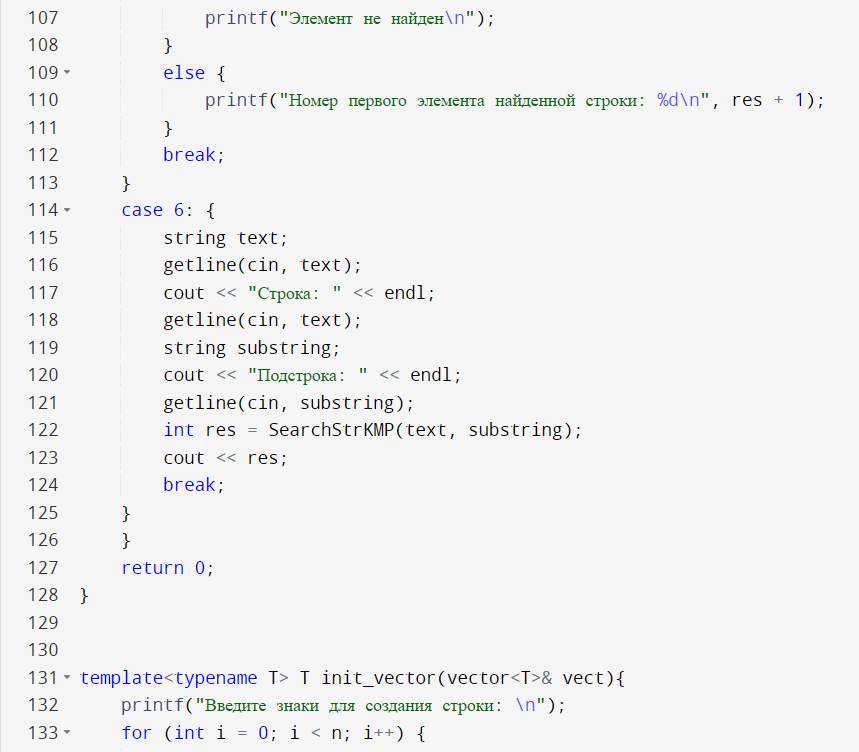


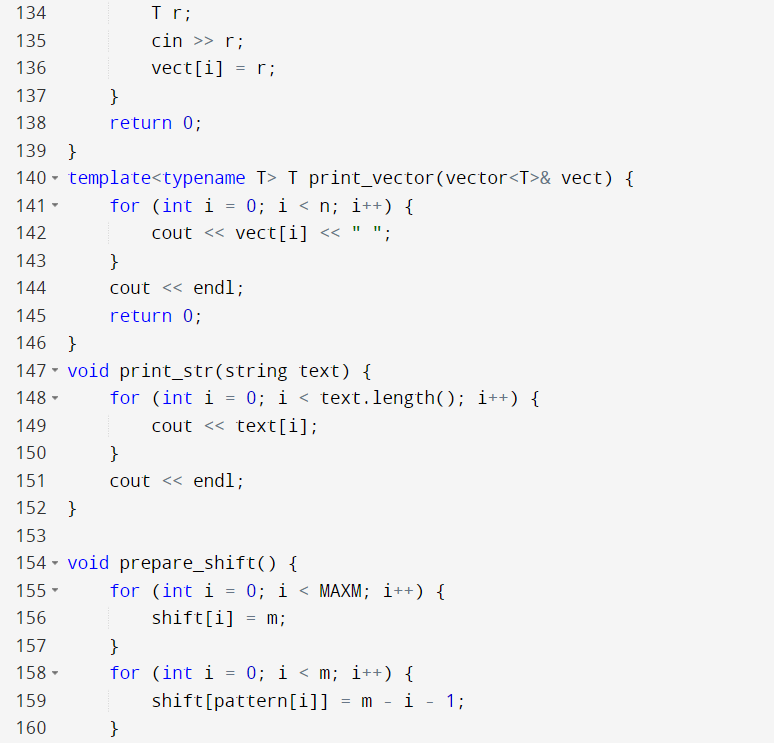
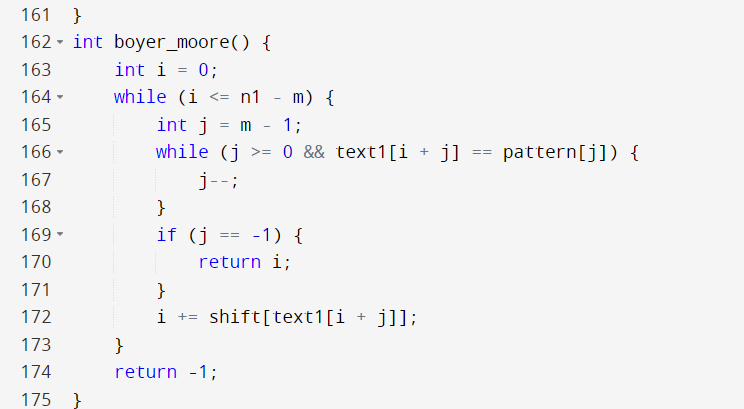
Функция main():



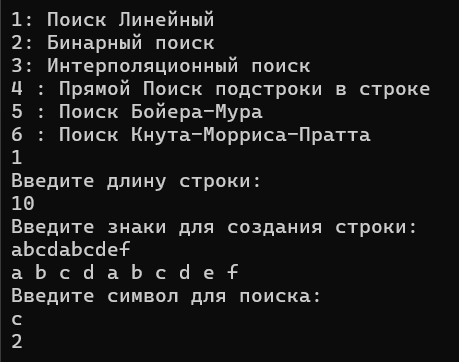




1. Результат работы:



1. Вывод:

Прямой поиск работает стабильно, но очень медленно т.к. нужно при каждом нахождении первого символа в строке проверять все последующие

1. GitHub:

<https://github.com/Andr0medA007/Labs/tree/main/Labs_1-12>

Бинарный поиск

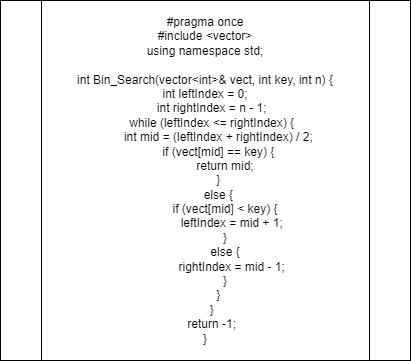
1. Постановка задачи:

Реализовать поиск подстроки в заданной строке, используя алгоритм Бинарногопоиска

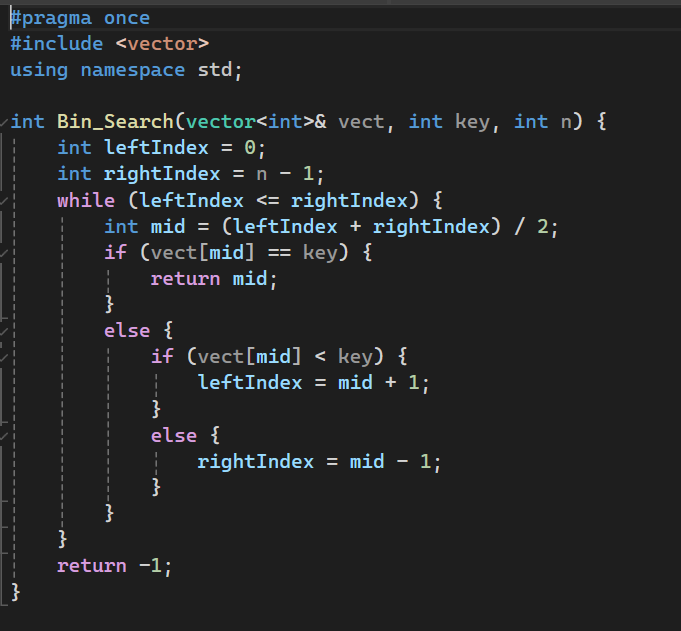
1. Анализ задачи:

* Находится середина строк
* Т.к. строка – упорядоченный набор чисел, то все последующие перемещения середины выполняются либо в правую половину, либо в левую, пока не находится символ
* Иначе выполнение прекращается

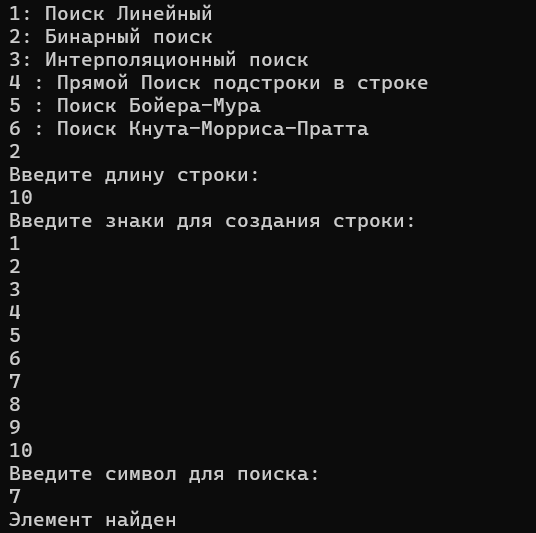
1. Блок-схема:



1. Код:



1. Результат работы:



1. Вывод:

Алгоритм работает, но количество операций недостаточно малы для совершенной работы.

1. GitHub:

<https://github.com/Andr0medA007/Labs/tree/main/Labs_1-12>

Интерполяционный поиск

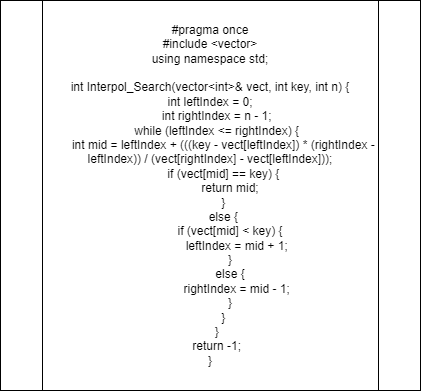
1. Постановка задачи:

Реализовать поиск подстроки в заданной строке, используя алгоритм Интерполяционного поиска

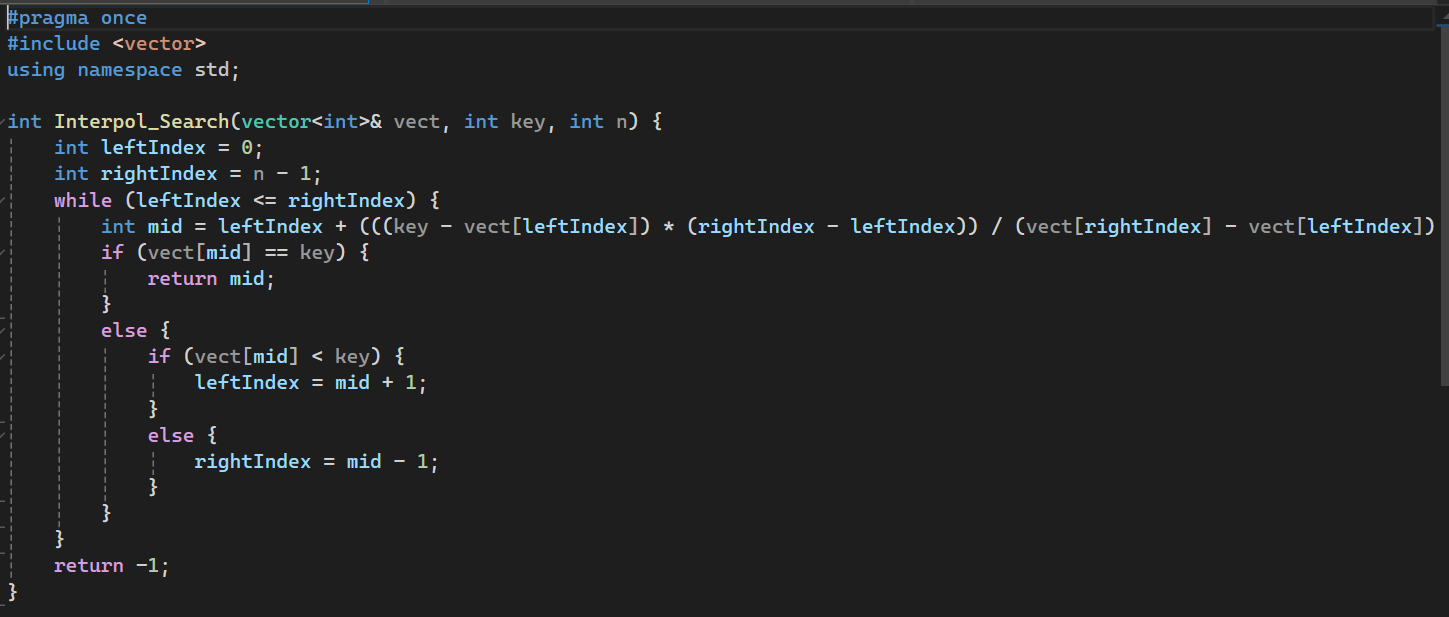
1. Анализ задачи:

* Алгоритм схож с бинарным поиском, но серединное число находится по специальной формуле, что немного ускоряет алгоритм.

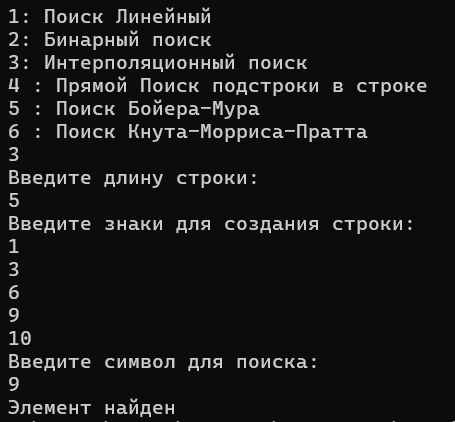
1. Блок-схема:



1. Код:



1. Результат работы:



1. Вывод:

Время выполнения работы интерполяционного поиска чуть меньше, чем у бинарного, за счёт использования специальной формулы. Но алгоритм также требует сортировку строки, что сильно увеличивает время выполнения и его сложность.

1. GitHub:

<https://github.com/Andr0medA007/Labs/tree/main/Labs_1-12>

Прямой поиск

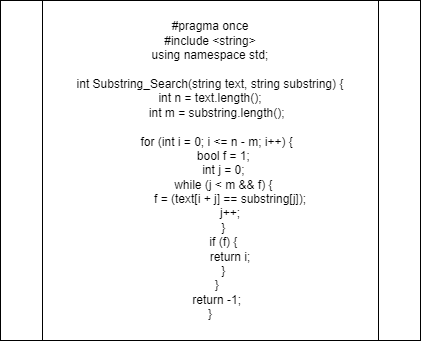
1. Постановка задачи:

Реализовать поиск подстроки в заданной строке, используя алгоритм Поиска подстроки в строке

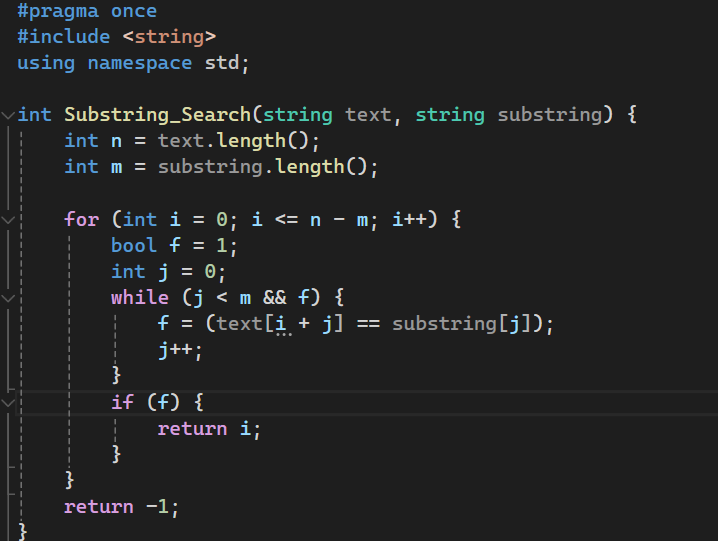
1. Анализ задачи:

* Если в строке находится первый символ подстроки, то начинают проверяться все последующие символы подстроки
* При отличном от требуемого символе алгоритм перемещает указатель подстроки на начало и проверяет все символы строки подстроки заново

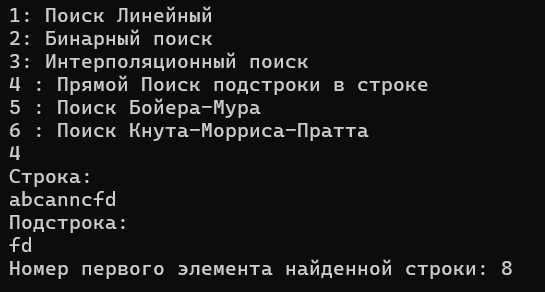
1. Блок-схема:



1. Код:



1. Результат работы:



1. Вывод:

Алгоритм поиска подстроки в строке стабилен, но работает очень медленно из-за проверки каждого символа строки с каждым символом подстроки

1. GitHub:

<https://github.com/Andr0medA007/Labs/tree/main/Labs_1-12>

Сортировка Бойера-Мура

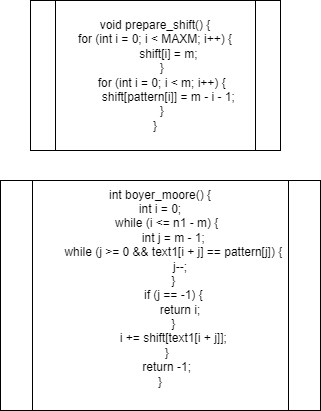
1. Постановка задачи:

Реализовать поиск подстроки в заданной строке, используя алгоритм Поиска Бойера-Мура

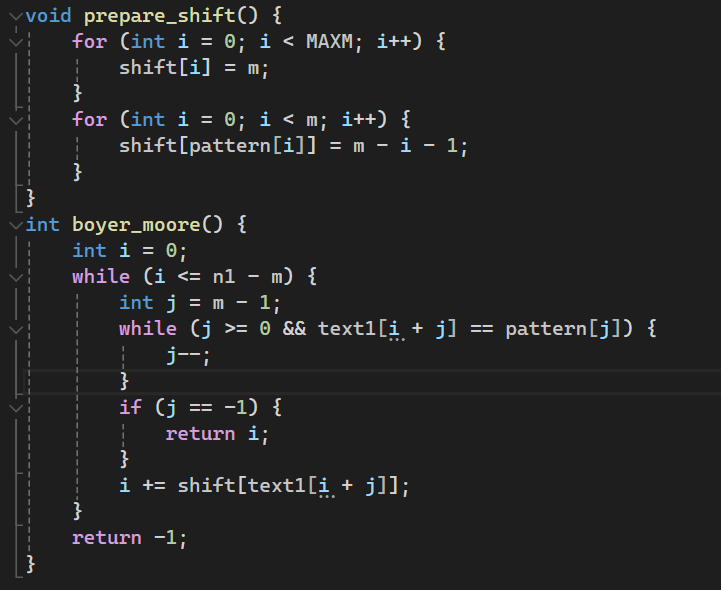
1. Анализ задачи:

* Строится таблица смещений для каждого символа по принципу его позиции от конца подстроки
* Если при прохождении по строке не был найден символ, то маркер переносится на длину подстроки
* Так при полном прохождении таблицы смещений, выводится результат

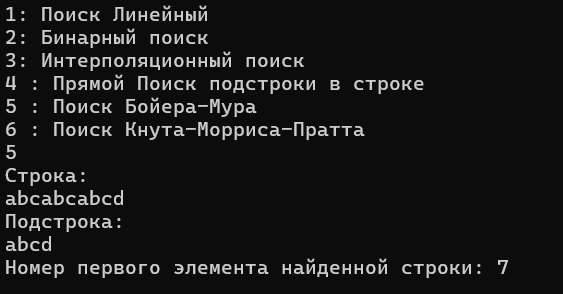
1. Блок-схема:



1. Код:



1. Результат работы:



1. Вывод:

Алгоритм Бойера-Мура превосходит другие алгоритмы по скорости, что делает его одним из самых удобных в поиске.

1. GitHub:

<https://github.com/Andr0medA007/Labs/tree/main/Labs_1-12>

Сортировка КМП

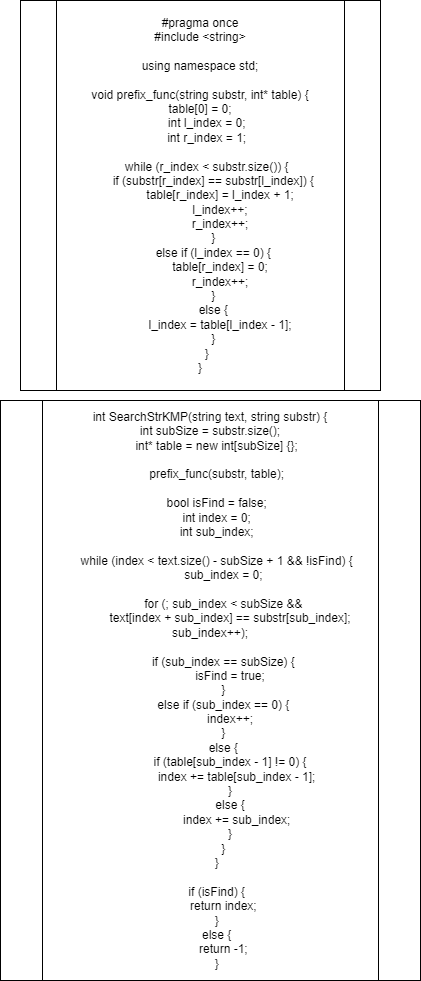
1. Постановка задачи:

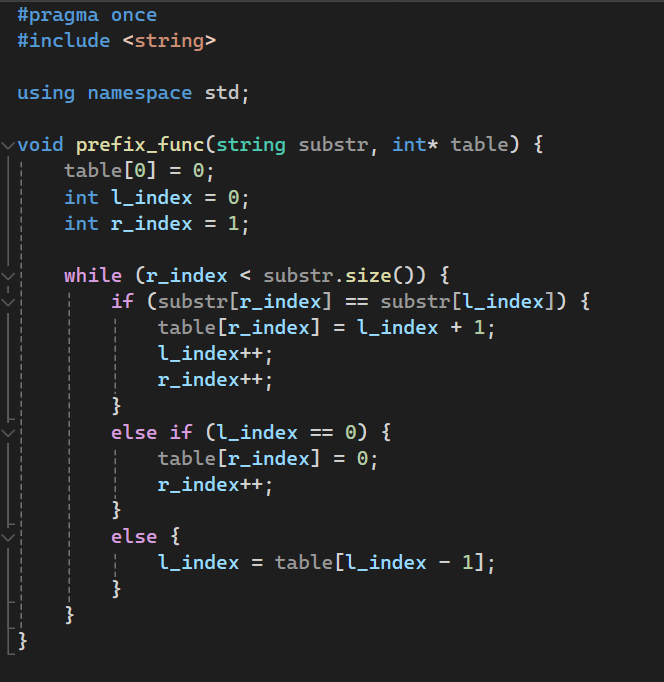
Реализовать поиск подстроки в заданной строке, используя алгоритм Поиска Кнута-Морриса-Пратта

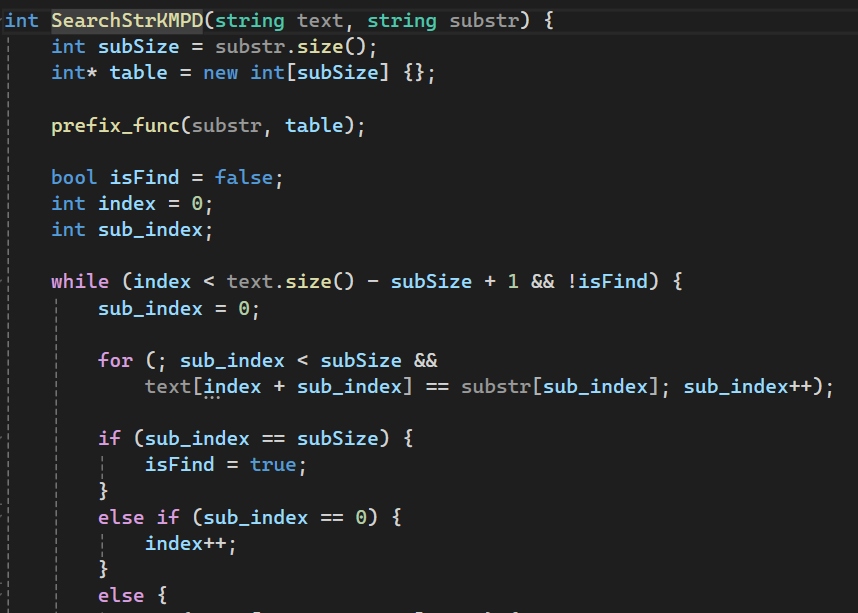
1. Анализ задачи:

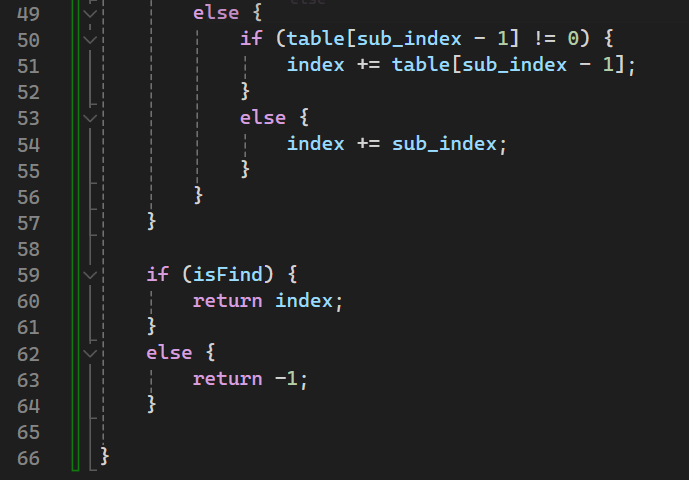
* Создаётся функция по нахождению таблицы префиксов и суффиксов подстроки, т.е. первая строка – номер символа в подстроке, вторая строка таблицы – максимальная длина префиксов в строке.
* При совпадении префикса и суффикса значение префикса записывается для нужного символа в таблицу
* Проверяется следующий символ, префикс для которого уже найден и происходит проверка на совмещение с подстрокой

1. Блок-схема:

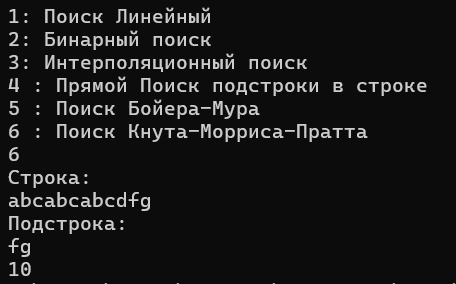


1. Код: 





1. Результат выполнения:



(10 – индекс первого вхождения подстроки в строку)

1. Вывод:

Алгоритм Кнута-Морриса-Пратта эффективнее остальных алгоритмов поиска за счёт своей сложности. Он может использоваться в текстах с большим объёмом букв и длинной подстрокой и время выполнения никак не изменится в худшую сторону

1. GitHub:

<https://github.com/Andr0medA007/Labs/tree/main/Labs_1-12>