Федеральное государственное бюджетное образовательное учреждение высшего образования «Сибирский государственный университет телекоммуникаций и информатики»

Факультет ИВТ

Кафедра вычислительных систем

**Курсовая работа**

на тему “Полнотекстовый поиск по шаблону”

Вариант 3.2 “Автоматы поиска подстрок”

Выполнил:

Студент группы ИВ-221

Сухаренко Илья

Проверил:

Старший преподаватель Кафедры ВС

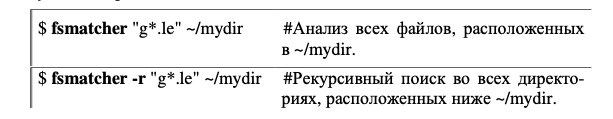
Фульман Виталий Олегович

Новосибирск, 2023

**Оглавление:**

**Исходное задание:**

Реализовать программу ***fsmatcher*** (Finit State Machine string mATCHER) полнотекстового поиска по шаблону. Шаблон и имя файла (директории), в ко- торой осуществляется поиск, передаются через аргументы командной строки в следующем порядке:



*Критерии оценки:*

Оценка **«**отлично**»**: разработанная программа обеспечивает поиск текста по шаблону рекурсивно в заданной директории. Под рекурсивным поиском понимается анализ всех текстовых файлов в текущей директории, а также во всех вложенных директориях.

Оценка **«**хорошо**»**: разработанная программа не предусматривает поиск по шаблону ИЛИ не способна выполнять рекурсивный поиск в дереве каталогов (поиск только в одном файле).

Оценка **«**удовлетворительно**»**: реализован алгоритм поиска с помощью конечного автомата.

**Анализ задачи:**

*Описание модели КА*

Конечный автомат (англ. Final State Machine) – абстрактная модель использующаяся в том числе и в программировании. Идея конечного автомата в том, что бы ввести в участок кода сотстояние. Так как автомат называется конечным, количество его состояний так же конечно. Результат работы автомата определяется по его финальному (конечному) состоянию.

В общих случаях конечный автомат следующим образом:

M = (Q, q0, F, ∑, δ) – где:

Q – множество состояний автомата.

q0 – начальное состояние автомата.

F – Множество финальных состояний автомата (автомат допустил строку).

∑ - мощность входного алфавита, формирующего цепочки символов.

δ - Функция переходов между состояниями:

Функция переходов имеет тип: δ : Q (дек х) ∑ -> Q

\*При недетерминированности формула допускает пустую строку

Функция конечного автомата может формировать таблицу переходов, наглядно показывающую переходы из состояния q1 в q2 при считывании определенного символа.

|  |  |  |
| --- | --- | --- |
|  | V1 | V2 |
| q1 | … | q1,2 |
| q2 | … | … |

δ(q1,v2) = q1,2 = q(i).

*Алгоритм работы КА.*

1. Инициализация: создание таблицы переходов, и начальных состояний КА.

2. Обработка входного символа шаблона: смена состояния по правилу заданному δ

3. Проверка состояния: автомат проверяет текущее состояние с конечным. Если они сходятся, то КА принимает строку.

4. Окончание работы.

*Пример:*

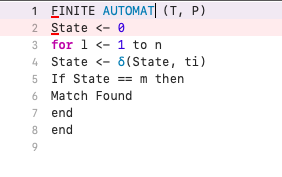
Создаем таблицу переходов, состояний и начальное состояние автомата. В данном случае у нас будет 4 состояния, по одному для каждой буквы в подстроке "abc", и состояние, которое будет означать, что мы нашли всю подстроку.

|  |  |  |  |
| --- | --- | --- | --- |
|  | A | B | C |
| Q0 | 1 | 0 | 0 |
| Q1 | 1 | 2 | 0 |
| Q2 | 1 | 0 | 3 |
| Q3 F | 1 | 2 | 0 |

Когда конечный автомат получает входной символ, он переходит в новое состояние в соответствии с правилами таблицы переходов.

Конечный автомат проверяет, является ли текущее состояние конечным состоянием. Если это так, то автомат принял входную строку. В нашем случае, если автомат находится в состоянии 3, это означает, что мы нашли всю подстроку "abc" в исходной строке.

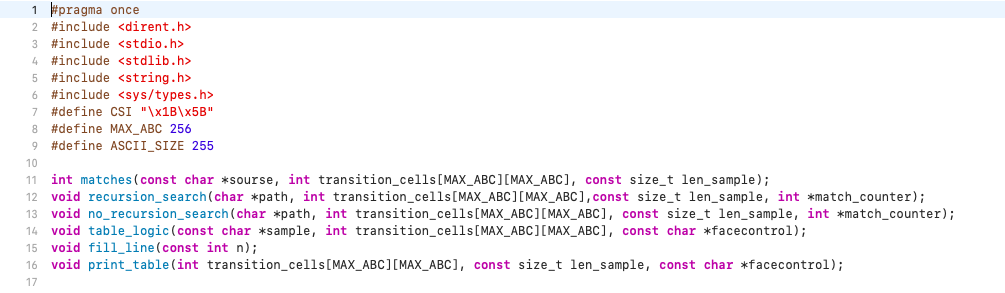
*Псевдокод:*



**Выполнение работы:**

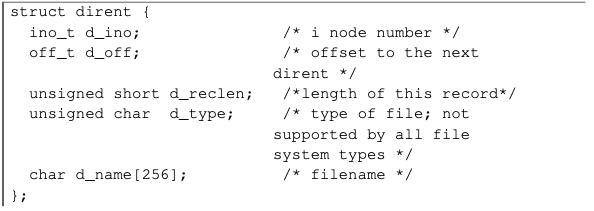
Вся программа состоит из заголовочного файла function.h, логического файла function.c и реализации в готовую программу main.c

*\*Скриншот файла function.h*

**

Библиотеки и директивы:

<dirent.h> - это библиотека для работы с директориями в операционной системе. Она содержит функции, которые позволяют просматривать содержимое директории и получать информацию о файлах внутри нее.

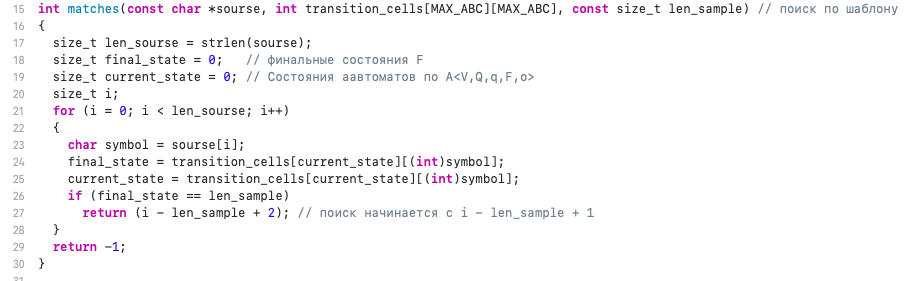


<sys/types.h> - - это библиотека, которая содержит определения типов данных, используемых в системных вызовах.

#define CSI "\x1B\x5B - директива препроцессора, которая определяет символы для управления ANSI-цветами . Символы `\x1B` и `\x5B` соответствуют ESC-последовательности, которые используются для управления цветом в терминале.

*\*Файл functions.h*

Далее по функциям и их логике:



Функция matches выполняет поиск подстроки в строке с использованием конечного автомата. Она принимает три аргумента: указатель на исходник sourse, таблицу переходов transition\_cells, двумерный массив вида …[Q – текущее состояние автомата ][V – символ из строки sourse], и размер шаблона len\_sample (равный заключительному состоянию).

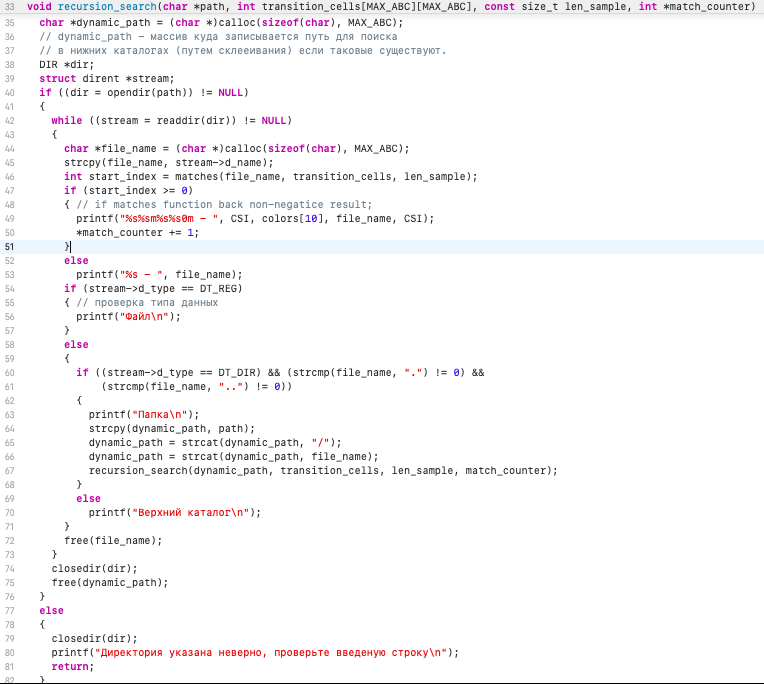
Функция определяет длину строки, устанавливает переменные по FSM (q0, F) – текущее состояние куда изначально ставится нуль, и финальное состояние. Цикл for перебирает все символы получаемой строки. Символ (char-тип), в процессе формирования состояний КА, кастуется как тип int для успешной работы с int-транзитивной-таблицей. Строки 24-25 устанавливают последнее состояние автомата, которое может быть финальным состоянием, и хранения текущего состояния автомата соответственно. Если final\_state становиться равным длине шаблона то автомат «пропустил» строку – возвращается индекс начала совпадения. Если же в результате цикла подстрока так и не была найдена функция возвращает -1.

Пример работы функции. Шаблон «abc» Sourse {x,y,z,a,b,c} – индекс i проходящийся по данному массиву в определенный момент встанет на позицию I = 3. При наилучшем варианте таблица переходов будет заполнена как:

State 0 -> 1 -> 2 -> 3

Len\_sample === 3 Следовательно функция перехода δ(2->3,с) = F.

Char ‘a’ -> ’b’ -> ‘c’



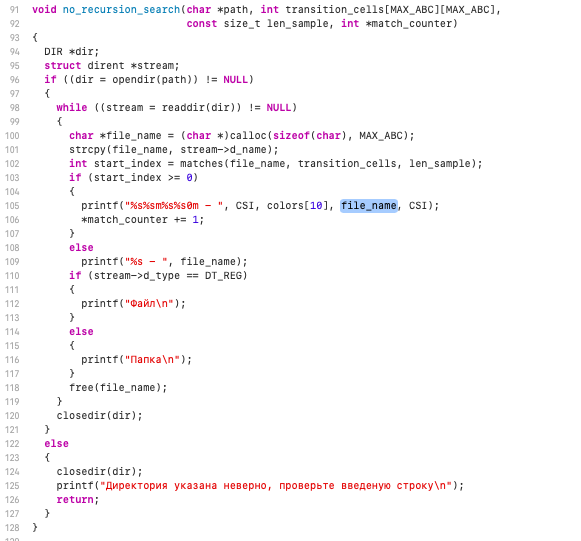
recursion search – одна из двух функций отвечающих за поиск файлов в указанных директориях. Функция вызывается при установке ключа -r(ecursion) в терминал. Принимает указатель на строку path являющуюся путем поиска, таблицу переходов, длину шаблона и счетчик совпадений match\_counter.

Функцией calloc задается динамический массив dynamic\_path. Он назван так, потому что в процессе подкаталогового поиска будет дополнять строку поиска path путем конкатенации. Библиотека dirrent.h содержит ряд функций для работы с директориями и файлами:

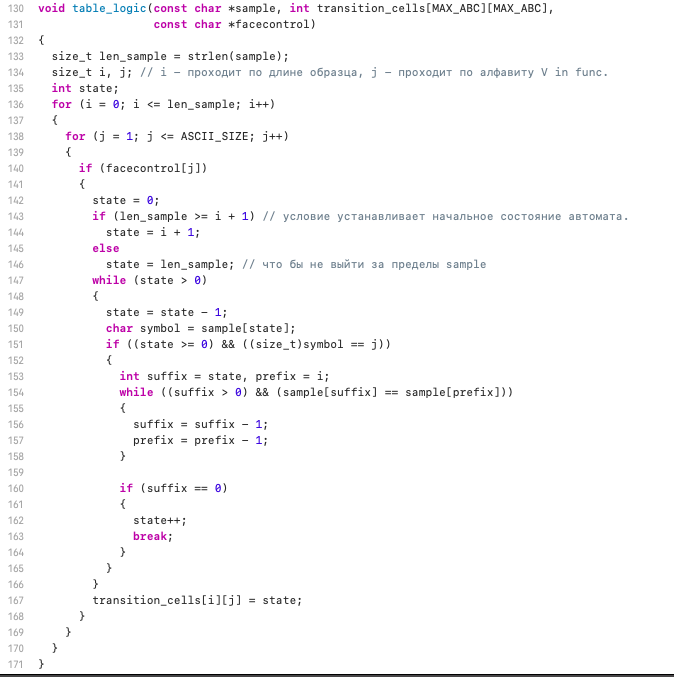
Opendir (возвращает указатель на структуру DIR) открывает путь path. Если path не может быть открыт функция выводит сообщение об ошибке. Readdir (возвращает указатель на следующий элемент директории) читает и записывает файлы из представленной директории в поток stream. Если stream получает не информацию о файле, a NULL – директория закрывается, динамический массив освобождается.

Имя каждого элемента записанного в stream записывается в еще один ДМ file\_name. Переменная start\_index получает индекс начала подстроки (искомого шаблона) в строке (имени файла) с помощью функции matches. Переменная starе\_index нужна для неотрицательной проверки, которая инкриминирует счетчик совпадений (используется для наглядности результатов поиска).

Файл в названии которого найден искомый шаблон подкрашивается, проверяются типы файлов структуре dirrent stream, выводятся из типы – папка, файл или родительский каталог. Если родительские каталоги были найдены, dynamic\_path пополняется новым направлением поиска, функция рекурсивно вызывается вновь. В конце отчищаем память.



Функция no\_recursion\_search работает проще чем своя рекурсивная версия. Здесь не выделяется массив под динамический путь, не отображаются родительские каталоги. Все элементы кода не связанные с рекурсией и пополнением пути поиска остаются на месте, и работают также как уже было описано выше.



table\_logic – функция принимает в качестве аргументов строку образец, таблицу переходов и массив facecontrol размерностью равной мощности алгоритма. Ее задача – продумать логику для таблицы переходов которая наглядно демонстрирует возможное множество Q – состояний, путь к F – финальному состоянию и установку начального состояния q0 в случае прерывания верной последовательности символов.

Для данной функции также вводится понятие состояние, которое позже будет записано на пересечении элементов таблицы переходов.

Внешний и вложенный циклы for отвечают за прохождение по числу состояний автомата (len\_sample), и входному алфавиту соответсвенно. Для каждой пары (i, j) вычисляется значение transition\_cells[i][j], которое является состоянием автомата при переходе из i-того состояния в j-тый символ.

Происходит проверка вхождения переменной j в алфавит, представленный массивом facecontrol[256].

Массив помогает оптимизировать проверку символа. Без него в цикле пришлось бы проверять вхождение каждого символа в алфавит что замедляет скорость выполнения.

\*Если j !принадлежит facecontrol -> transition\_cells[i][j] = 0;

State приравнивается к нулю.

Далее выбирается минимальный элемент из i +1 и len\_sample.

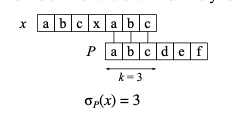
1а. Если текущая позиция во входной строке меньше или равна длине образца то state приравнивается к увеличенному на один индексу позициии образце (автомат должен находиться в состоянии, соответствующем позиции во входной строке, до тех пор, пока эта позиция не достигнет длины образца)

ИЛИ

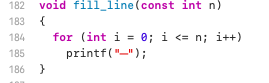
1б. В других случаях (i > len\_sample) состояние автомата приравнивается к длине образца. Это связано с тем, что после достижения конца образца, автомат должен оставаться в конечном состоянии, пока входная строка не закончится.

В дополнительном вложенном цикле значение state уменьшается до тех пор, пока суффикс и префикс

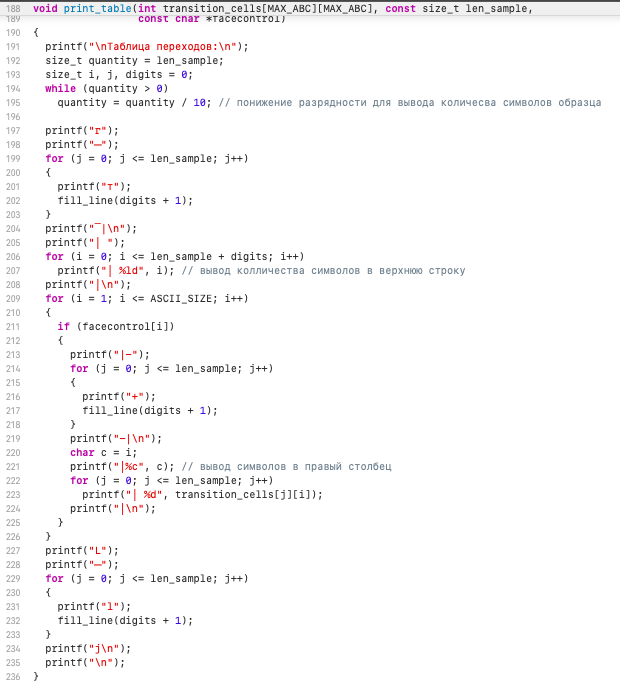
не совпадут\*



Координаты i и j при которых суффикс совпал с с префиксом дают результат в виде n вхождения образца в строку, это индекс элемента вошедшего в таблицу переходВ этом случае значение `state` увеличивается на 1, чтобы указать на следующую позицию в образце, и цикл завершается с помощью оператора `break`. Заполняется таблица переходов.



Void fill\_line вспомогательная функция которая чертит линию длиною в n-символов.



Print\_table это функция для вывода в формате ASCII таблицы переходов для конечного автомата, который используется для поиска подстроки в строке. Она принимает в качестве аргументов двумерный массив переходов transition\_cells, длину образца (en\_sample и строку алфавита facecontrol.

Длина образца вычисляется для выравнивания угловых сторон в таблице (кол-во символов). Функция прописана так что бы первый столбец носил значения символов алфавита, верхняя строка длину образца. Начинается она с вывода левого верхнего угла.

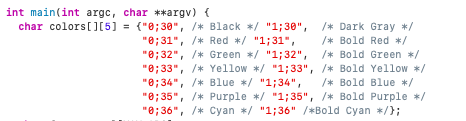
Для каждого символа алфавита, который присутствует в образце, выводим разделительную линию функцией fill\_line и символ алфавита в правом столбце.

Переменная quantity служит для вывода правильного числа блоков для всех символов образца. Каждое повторение цикла while понижает разрядность.

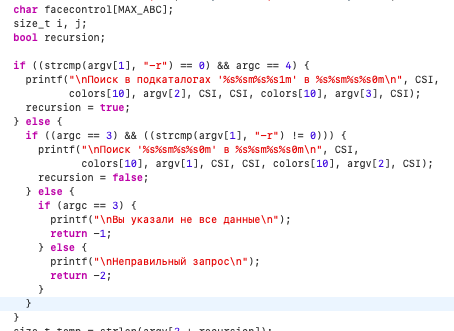
Вывод значений таблицы: выводим верхнюю строку таблицы, которая содержит количество символов в образце c помощью разделительных символов и цикла for.

Для каждого символа алфавита, который присутствует в образце, мы выводим разделительную линию, символ алфавита в правом столбце и значения переходов в ячейках таблицы. В цикле for идет перебор всех символов алфавита до 255. Если символ прошел «фейсконтроль» выводятся разделительные символы и символы присутствующие в искомом образце. Затем выводится символ алфавита в правом столбце и значения переходов в ячейках таблицы, используя цикл for и обращаясь к элементам массива transition\_cells через спецификатор %d.

*main.c*

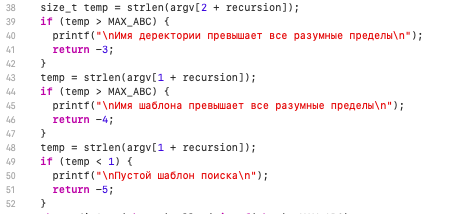
**

Главная функция начинается с объявления массива цветов. 0/1; определяет будет ли шрифт жирным. 30/37; на цвет текста (фиолетовый неон в нашем случае).

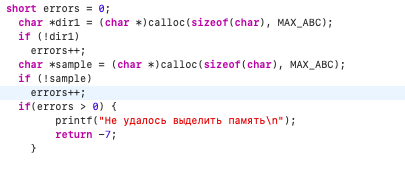


Создаются вспомогательные size\_t переменные и массив-алфавит. Как уже было упомянуто рекурсия — это опциональный параметр программы. Поэтому ключ -r будет влиять на bool переменную recursion.

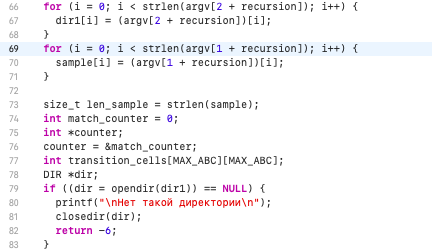
Далее программа выбирает тип поиска. Подсвечивает строчки при форматированном выводе. Тут argv[2] содержит имя каталога argv[3] ключ -r/файл. При использовании if вызывается strcmp проверяющая записался ли в определенный аргумент нужный ключ.

Функция наглядно показывает пользователю что и в каком каталоге она ищет. Также проверяется формат входных данных формат входных данных.

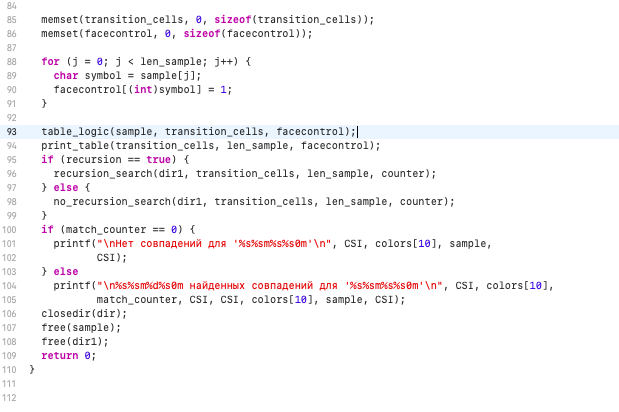
Этот блок кода отвечает за проверку аргументов, переданных командной строке на размерность (не превышает размер алфавита, и не равна нулю – пустому шаблону). Если условия не выполняются то выводятся соответствующее сообщения.



Здесь мы выделяем память под строку, в которую будет записана директория и образец. Если по какой-то причине этого не происходит выводится сообщение об ошибке.



Для копирования аргументов строки в соответствующее им массивы (в которых производится поиск) созданы два цикла. Далее устанавливаются значение аргументов используемых при вызове функций, открывается соответствующая директория (если нет выводится ошибка). Поток ввода директории закрывается функцией closedir.



Функцией memset заполняются нулями массивы transition\_cells и facecontrol. Это необходимо для правильной работы с таблицей переходов и контроль-алфавитом в функциях.

Используя цикл for для прохождения по символам искомого шаблона для каждого символа происходит установка соответствующего флага в массиве facecontrol с помощью функции (int)symbol, которая возвращает ASCII-код символа.

\*face\_control определяется как массив булевых значений при такой установке.

Далее:

1) Последовательный вызов table\_logic & print\_table для создания и вывода таблицы переходов.

2) В зависимости от bool recursion вызывается либо функция recursion\_search, либо no\_recursion\_search.

3)Внутри блока 2 запускаются функции matches, возвращаемый результат которых представлен как match\_counter. В зависимости от его значения выбирается формат вывода.

4) Служебные функции файла <dirrent.h> закрывают директорию. Отчищается выделенная память.

Мы получаем результат выполнения команды в терминале!