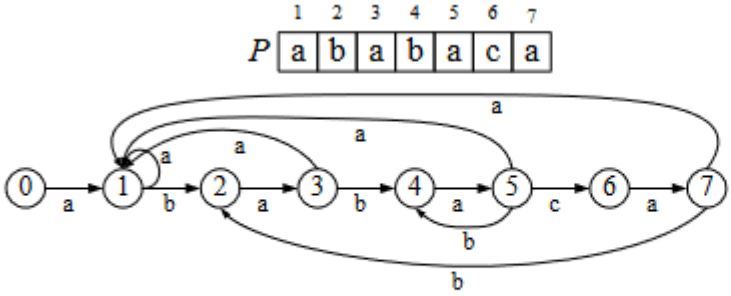
Полнотекстовый поиск по шаблону
Автоматы поиска подстрок.

Задание на курсовую работу
Реализовать программу fsmatcher (Finit State Machine string mATCHER) полнотекстового поиска по шаблону. Шаблон и имя файла (директории), в которой осуществляется поиск, передаются через аргументы командной строки в следующем порядке:

```
$ fsmatcher “g*.le” ~/mydir        #Анализ всех файлов, расположенных в ~/mydir
$ fsmatcher -r “g*.le” ~/mydir     #Рекурсивный поиск во всех директориях
                                   расположенных ниже ~/mydir
```

Указание к выполнению задания
Алгоритм поиска, использующий конечные автоматы (КА), основан на следующем принципе: для заданного шаблона P[1 ... m] строится конечный автомат $M(Q,q_0,A,\sum,\delta)$, где Q = {0, 1, 2, ..., m} – конечное множество состояний (states), $q_0 = 0$ – начальное состояние, A = m - конечное множество заключительных состояний. $\sum$ – входной алфавит, δ(q,a) – функция переходов, которая показывает, в какое состояние переходит КА, находящийся в состоянии q, при появлении символа a входного алфавита. Функция δ строится для образца, для ее построения используется суффикс-функция, рассмотренная в общем материале к данному разделу. Она определяется следующим образом: $\delta(q,a)= \sigma(P_q a)$, т.е. если КА находился в состоянии q и был обнаружен символ a, то КА переходит в состояние $\sigma(P_q a)$, т.е. определяет максимальную длину k префикса строки P[1 ... q], к которой добавлен символ ‘a’, совпадающего со своим суффиксом (k < q, P[1..k] – суффикс $P_q a]a$). Пример работы алгоритма представлен на рисунке 15:



Соответствующая графу таблица переходов:

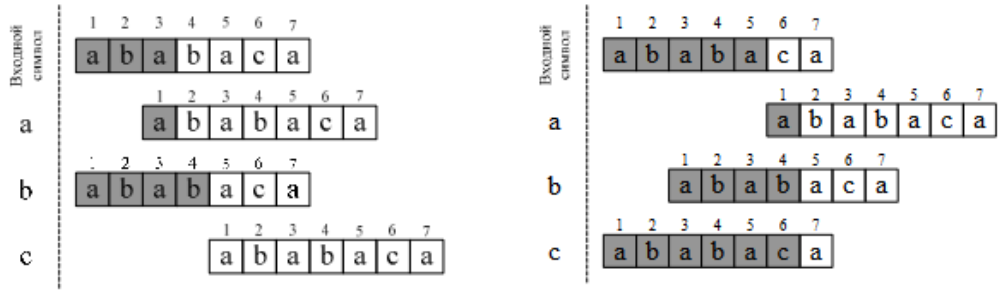
δ	0	1	2	3	4	5	6	7
a	1	1	3	1	5	1	7	1
b	0	2	0	4	0	4	0	2
c	0	0	0	0	0	6	0	0

Рисунок 15. Алгоритм поиска, использующий конечные автоматы

Этапы построения таблицы переходов:

1. Если КА находится в начальном состоянии (q = 0), то поступление любого символа, отличного от 'a' оставляет КА в исходном состоянии. Если получен символ 'a', то КА переходит в состояние 1, что означает, что в тексте найдены символы P[1 ... 1].
2. Если КА находится в состоянии q = 1 (обнаружены символы P[1 ... 1]) и поступает символ 'a', то КА остается в состоянии q = 1 (шаблон передвигается вправо на 1 позицию). Действительно, образец P = "abababaca", а прочитаны символы "aa", вхождение образца в текст возможно только со 2-й или более поздней позиции. Символ 'b' переводит КА в состояние 2, а символ 'c' – в состояние q = 0 (такие дуги не обозначены на рисунке для упрощения его чтения).

   На рисунке показаны действия, предпринимаемые при обнаружении допустимых символов, если КА находится в состоянии q = 3 и 5 (рисунок 16 а и б), а также в листинге 4 представлен псевдокод алгоритма вычисления δ(q,a).



а *б*

Рисунок 16. Пример работы КА в различных состояниях

Листинг 4. Псевдокод алгоритма вычисления $\delta(q, a)$.

```
COMPUTE_TRANSITION_F(P, ∑)
    m ← len(P)
    for q ← 0 to m do
        k ← min(m+1, q+2)
        while не ( P_k ⊐ P_q a)
            k ← k – 1
        δ(q,a) ← k
return δ
```

Анализ задачи

Конечный автомат в курсовой работе представлен таблицей переходов transitionTable, где 256 строк соответствуют максимальному числу возможных состояний автомата с входным алфавитом 256, а 256 столбцов соответствуют 256 символам таблицы ASCII. Входной алфавит представлен массивом hash длиной 256, который содержит ненулевые значения в элементах с индексами, соответствующими коду ASCII символов алфавита. Заданный шаблон и текст, по которому осуществляется поиск, заданы динамическими массивами символов, причём шаблон передаётся в качестве аргумента командной строки, а текстом является каждое имя файла или директории по отдельности, таким образом для хранения шаблона и текста объявляются динамические массивы длины 256, после определения которых память, отведённая им, перераспределяется в соответствии с их длиной.

Для создания таблицы переходов разработана функция createTable, которая заменяет в предварительно занулённой таблице состояние автомата при каком-либо символе входного алфавита на значение δ(q,a).

Для вывода на экран таблицы переходов разработана функция printTable. Для удобства заполнения строки символом горизонтальной прямой разработана функция fill.

Для рекурсивного и не рекурсивного прохождения по директориям разработаны функции recursion и nonRecursion, внутри которых предварительно проверенные на правильность имена директорий открываются на чтение. Имя каждого элемента директории поочерёдно становится строкой Text и вызываемая функция match проверяет на совпадение шаблон и подстроки Text.

Для применения конечного автомата на строке Text разработана функция match, в ней автомат переходит из q_0 в новые состояния, соответствующие символам строки Text.

Ниже приведены блок-схемы функций с пояснениями.

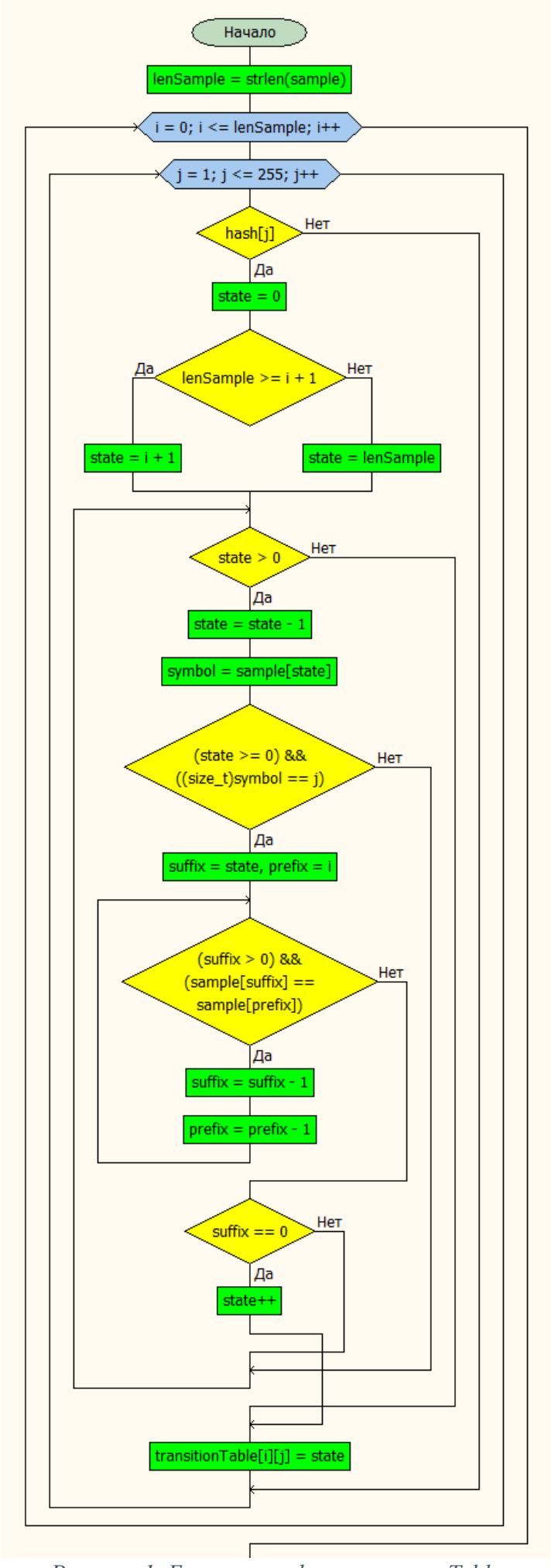


Рисунок 1. Блок-схема функции createTable

createTable

Запускается цикл for со счётчиком от нуля до числа состояний автомата (длины шаблона), внутри него запускается цикл для проверки принадлежности символа входному алфавиту. Из длины шаблона и i+1 выбирается минимальный элемент state для того, чтобы пройтись по всем значениям от 1 до lenSample.

В ещё одном вложенном цикле уменьшаем значение state до тех пор, пока суффикс и префикс, оба сравниваемые с конца, не совпадут.

Прибавляемый теоретически новый символ ‘a’ сравнивается с последним элементом префикса от 0 до j.

Координаты i, j, при которых префикс совпал с суффиксом, являются индексами элемента в таблице переходов. Его значением является state – длина максимального префикса, который является собственным суффиксом.

В функции четыре вложенных счётчика, каждый из которых в худшем случае выполняется n раз, таким образом сложность алгоритма: $O(n^4)$.

	Худший случай	Лучший случай
Сложность	$O(n^4)$	$O(n^3)$

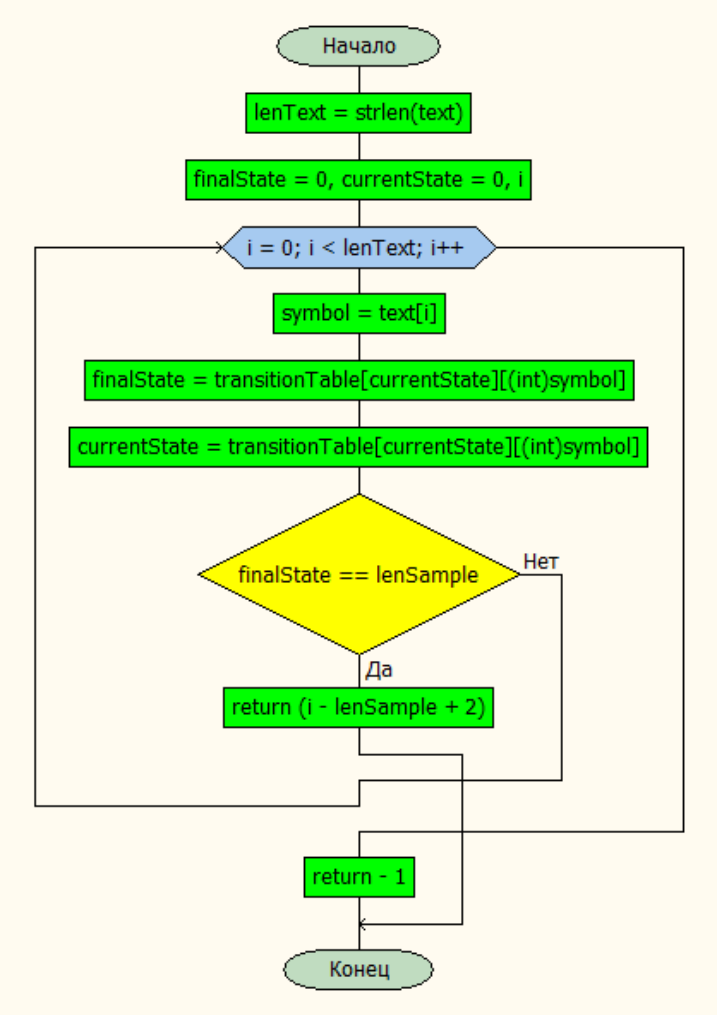


Рисунок 2. Блок-схема функции match

match

Начальное состояние автомата – 0. Запускается цикл от 0 до lenText, чтобы пройтись по всем символам строки Text. Считывается i-тый символ строки Text, значение таблицы переходов, соответствующее этому символу и предыдущему состоянию автомата, сравнивается с lenSample, которое равно заключительному состоянию. При успешном совпадении цикл покидается и функция возвращает индекс элемента строки, с которого начинается совпадение (от 0 до (lenText – lenSample)). При отсутствии совпадения возвращается -1.

nonRecursion

Открывается поток директории, при неуспешном открытии выводится сообщение об отсутствии такой директории и поток закрывается. Readdir возвращает указатель на следующий элемент в директории, создаётся строка, соответствующая имени элемента (файла или директории), выполняется функция match. Если возвращённое функцией match значение неотрицательно, то имя элемента директории при выводе на экран будет подсвечено красным, а также инкрементируется переменная findsCounter, которая подсчитывает общее число найденных подстрок. Далее в зависимости от типа элемента на экран выводится надпись file или directory. Память, выделенная под динамический массив освобождается, поток закрывается.

Recursion

Создаётся динамический массив символов fpath, открывается поток директории, также с помощью readdir указатель перемещается по элементам каталога. Строки, в которых найдены искомые подстроки также подсвечиваются красным. После обработки имени элемента обрабатывается его тип. Если элемент является директорией и не является ссылкой на домашнюю и родительскую директории, то в массив fpath копируется path, и с помощью конкатенации создаётся новый путь fpath, состоящий из “fpath + \ + fileName”, где fileName – имя директории. Рекурсивно вызывается функция recursion для директории fpath.

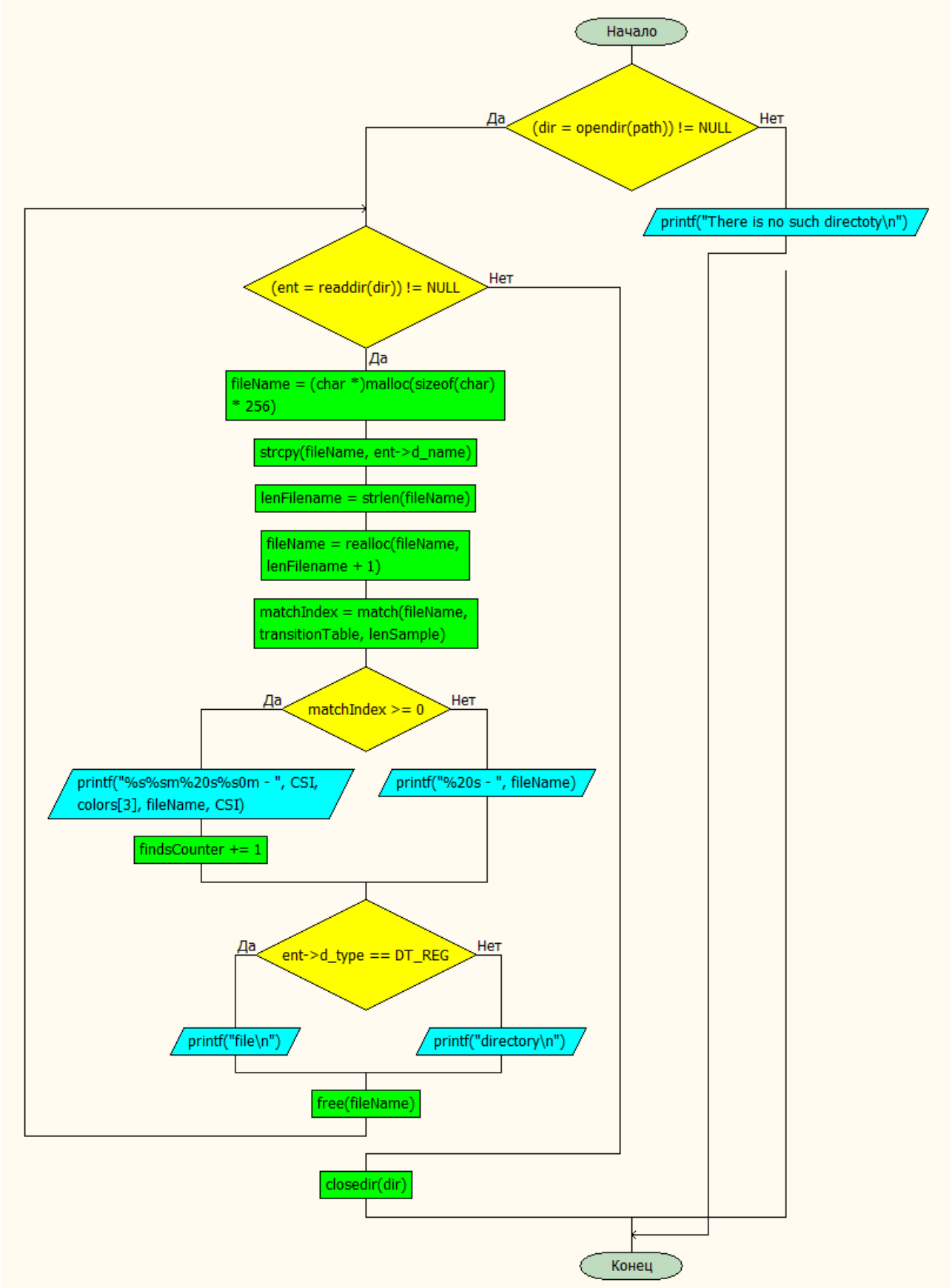


Рисунок 3. Блок-схема функции nonRecursion

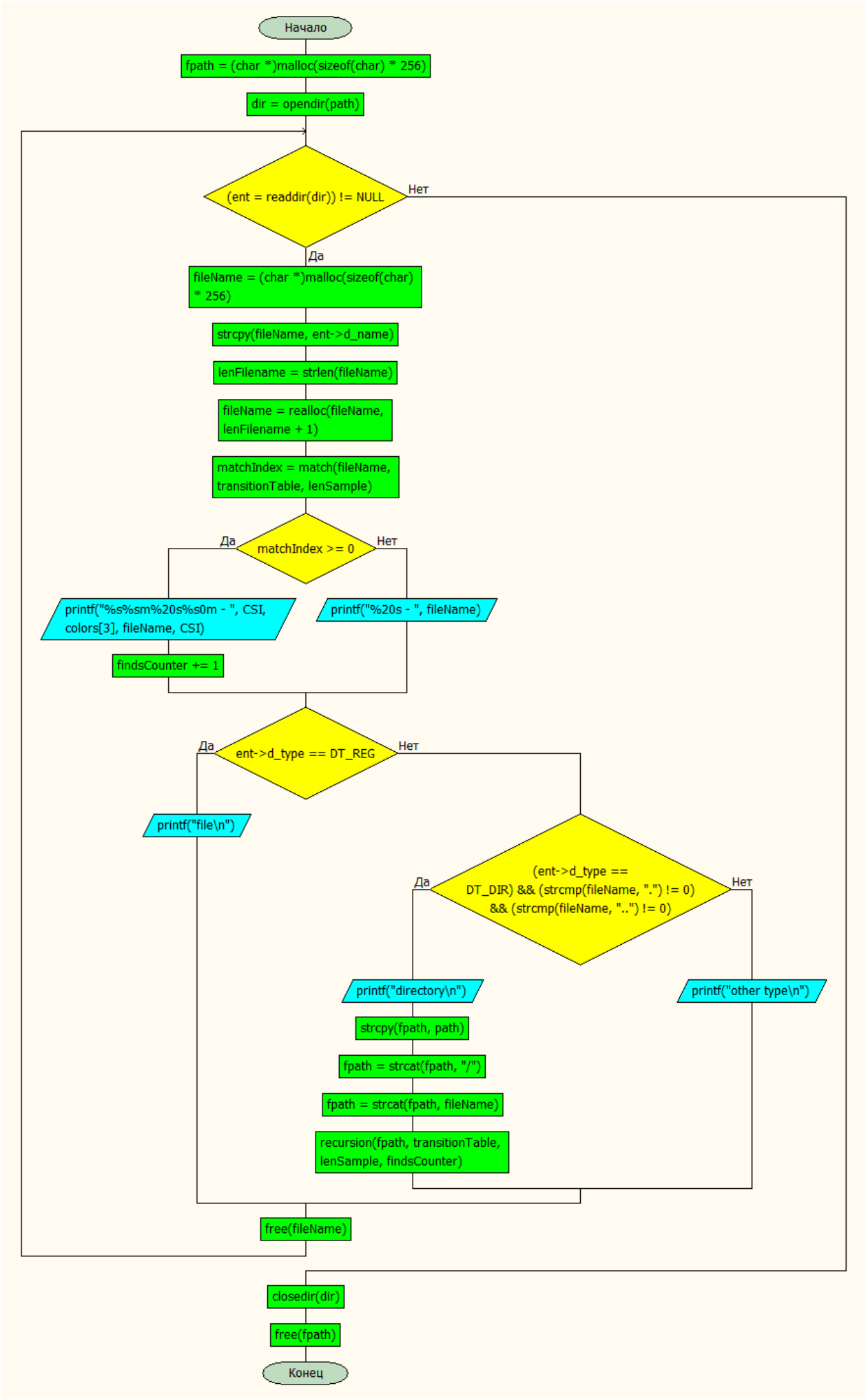


Рисунок 4. Блок-схема функции recursion

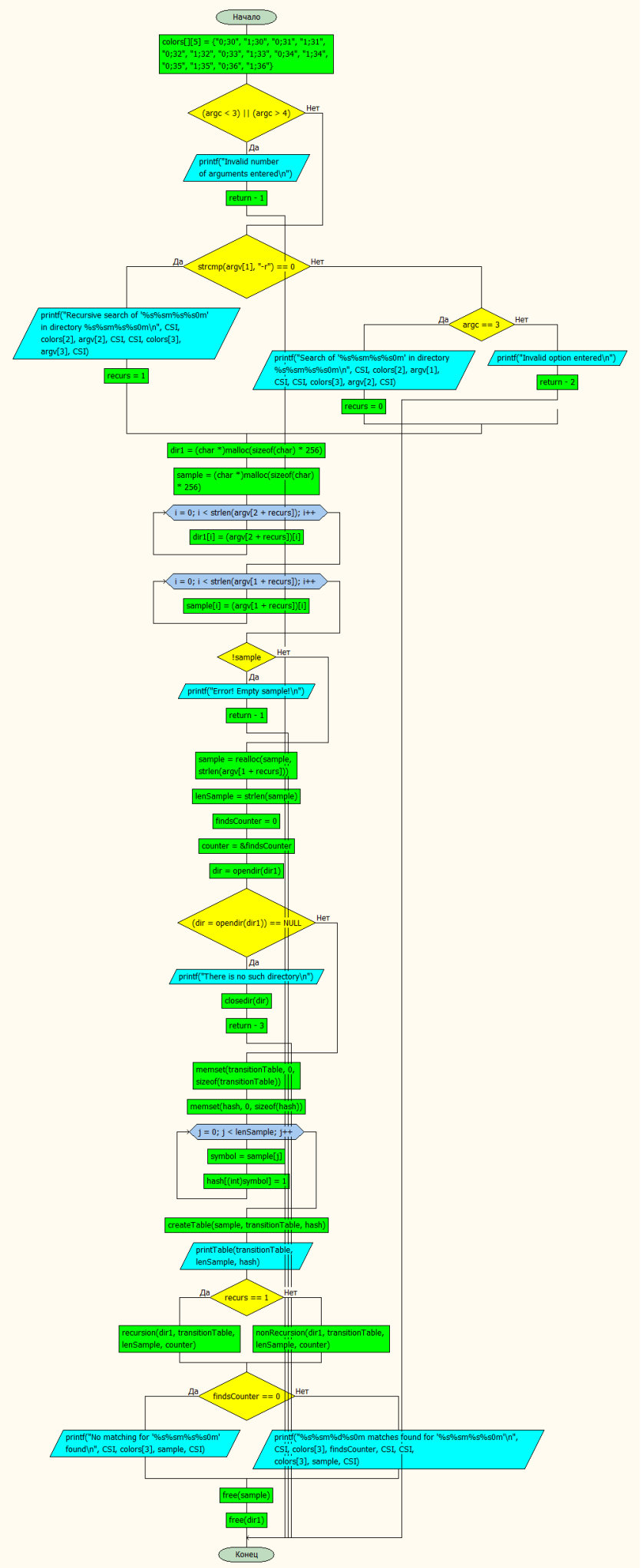


Рисунок 5. Блок-схема программы fsmatcher

Объявляются цвета, массив для входного алфавита, переменные для счётчиков и определения рекурсии.
Аргументы командной строки проверяются на правильность.
Создаются динамические массивы для хранения шаблона и пути.
Создаются переменная длины шаблона, счётчика найденных подстрок.
Объявляется таблица переходов.
Таблица переходов и массив входного алфавита зануляются, массив входного алфавита получает единицы в элементах с индексами-членами входного алфавита.
С помощью функций createTable и printable определяется и выводится на экран таблица переходов для шаблона.
Выполняется рекурсивное или не рекурсивное прохождение по директории, заданной путём.
В функциях recursion и nonRecursion выполняется функция match, по результатам которой строки с найденными подстроками подсвечиваются красным.
В зависимости от наличия найденных подстрок выводятся сообщения об их количестве или отсутствии.
Освобождается память, отведённая ранее на динамические массивы, закрывается поток. При успешном выполнении программа возвращает 0, иначе [-1..-6].

Пример работы программы с аргументами командной строки “text” и ~/prog23

```
lyudmila@hp17:~/CURS$ ./fsmatcher "text" ~/prog23
Search of 'text' in directory /home/lyudmila/prog23
Transition table:
```

δ	0	1	2	3	4
e	0	2	0	0	0
t	1	1	1	4	1
x	0	0	3	0	0

```
                curs - directory
                lab2 - directory
                lab1 - directory
        sometexts.txt - file
                lab5 - directory
                lab3 - directory
                  .. - directory
                   . - directory
1 matches found for 'text'
```

Аргументы подразумевают, что будет осуществляться не рекурсивный поиск по шаблону “text” в директории ~/prog. Составляется таблица переходов и выводится список элементов в директории. Имя, подстрокой которого является “text” подсвечивается красным. Выводится сообщение о количестве найденных вхождений подстрок.

Тестовые данные

```
lyudmila@hp17:~/CURS$ ./fsmatcher "dFo" ~/testDir
Search of 'dFo' in directory /home/lyudmila/testDir
Transition table:
```

δ	0	1	2	3
F	0	2	0	0
d	1	1	1	1
o	0	0	3	0

```
        secondFolder - directory
              folder - directory
                  .. - directory
                   . - directory
1 matches found for 'dFo'
```

Рисунок 8. Не рекурсивный поиск, путь верный, подстрока найдена

```
lyudmila@hp17:~/CURS$ ./fsmatcher "dof" ~/testDir
Search of 'dof' in directory /home/lyudmila/testDir
Transition table:
```

δ	0	1	2	3
d	1	1	1	1
f	0	0	3	0
o	0	2	0	0

```
        secondFolder - directory
              folder - directory
                  .. - directory
                   . - directory
No matching for 'dof' found
```

Рисунок 7. Не рекурсивный поиск, путь верный, подстрока не найдена

```
lyudmila@hp17:~/CURS$ ./fsmatcher "dFo" ~/test
Search of 'dFo' in directory /home/lyudmila/test
There is no such directory
```

Рисунок 9. Неверный путь

```
lyudmila@hp17:~/CURS$ ./fsmatcher -k "dFo" ~/test
Invalid option entered
```

Рисунок 6. Неверная опция

```
lyudmila@hp17:~/CURS$ ./fsmatcher -r "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaa" ~/testDir
Recursive search of 'aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa' in
 directory /home/lyudmila/testDir
Length of sample is above 256
```

Рисунок 10. Длина шаблона больше 256 символов

```
lyudmila@hp17:~/CURS$ ./fsmatcher -r "folder" ~/testDir/testDir/testDir/tes
tDir/testDir/testDir/testDir/testDir/testDir/testDir/testDir/testDir/testDi
r/testDir/testDir/testDir/testDir/testDir/testDir/testDir/testDir/testDir/t
estDir/testDir/testDir/testDir/testDir/testDir/testDir/testDir/testDir/test
Dir/testDir/testDir/testDir/testDir/testDir/testDir/testDir/testDir/testDir
/testDirv
Recursive search of 'folder' in directory /home/lyudmila/testDir/testDir/te
stDir/testDir/testDir/testDir/testDir/testDir/testDir/testDir/testDir/testD
ir/testDir/testDir/testDir/testDir/testDir/testDir/testDir/testDir/testDir/
testDir/testDir/testDir/testDir/testDir/testDir/testDir/testDir/testDir/tes
tDir/testDir/testDir/testDir/testDir/testDir/testDir/testDir/testDir/testDi
r/testDir/testDirv
Length of directory name is above 256
```

Рисунок 11. Длина пути превышает 256 символов

```
lyudmila@hp17:~/CURS$ ./fsmatcher ~/test
Invalid number of arguments entered
```

Рисунок 13. Неверное число аргументов (меньше 3)

```
lyudmila@hp17:~/CURS$ ./fsmatcher -r "dFo" "ofd" ~/test
Invalid number of arguments entered
```

Рисунок 12. Неверное число аргументов (больше 3)

```
lyudmila@hp17:~/CURS$ ./fsmatcher -r "file" ~/testDir
Recursive search of 'file' in directory /home/lyudmila/testDir
Transition table:
```

δ	0	1	2	3	4
e	0	0	0	4	0
f	1	1	1	1	1
i	0	2	0	0	0
l	0	0	3	0	0

```
       secondFolder - directory
          subfolder - directory
  subfolderfile1.txt - file
  subfolderfile2.txt - file
  subfolderfile3.txt - file
                  .. - other type
                   . - other type
         ffffff.txt - file
                  .. - other type
                   . - other type
             folder - directory
        somefile.txt - file
     filefilefile.txt - file
                  .. - other type
                   . - other type
                  .. - other type
                   . - other type
5 matches found for 'file'
```

Рисунок 14. Рекурсивный поиск, верный путь, подстрока найдена

```
lyudmila@hp17:~/CURS$ ./fsmatcher -r "ilfe" ~/testDir
Recursive search of 'ilfe' in directory /home/lyudmila/testDir
Transition table:
```

δ	0	1	2	3	4
e	0	0	0	4	0
f	0	0	3	0	0
i	1	1	1	1	1
l	0	2	0	0	0

```
       secondFolder - directory
          subfolder - directory
  subfolderfile1.txt - file
  subfolderfile2.txt - file
  subfolderfile3.txt - file
                  .. - other type
                   . - other type
         ffffff.txt - file
                  .. - other type
                   . - other type
             folder - directory
        somefile.txt - file
     filefilefile.txt - file
                  .. - other type
                   . - other type
                  .. - other type
                   . - other type
No matching for 'ilfe' found
```

Рисунок 15. Рекурсивный поиск, верный путь, подстрока не найдена

```
lyudmila@hp17:~/CURS$ ./fsmatcher -r "" ~/testDir
Recursive search of '' in directory /home/lyudmila/testDir
The sample is empty
```

Рисунок 16. Шаблон пустой