

*Государственное образовательное учреждение высшего
профессионального образования*

**«Московский государственный технический
университет имени Н.Э. Баумана»
(МГТУ им. Н.Э. Баумана)**

РУБЕЖНЫЙ КОНТРОЛЬ №2

ПО КУРСУ «АНАЛИЗ АЛГОРИТМОВ»

Конечные автоматы и регулярные выражения

Выполнил: Сорокин А.П., гр. ИУ7-52Б

Преподаватели: Волкова Л.Л., Строганов Ю.В.

Москва, 2019 г.

Оглавление

Введение	2
1 Аналитическая часть	3
1.1 Конечные автоматы	3
1.2 Регулярные выражения	3
2 Конструкторская часть	4
2.1 Конечный автомат	4
2.2 Регулярное выражение	4
3 Технологическая часть	5
3.1 Средства реализации	5
3.2 Реализации алгоритмов	5
4 Экспериментальная часть	8
4.1 Примеры работы	8
Заключение	9
Литература	10

Введение

Цель рубежного контроля: при помощи конечного автомата и регулярного выражения написать программу для поиска критических секций в коде.

1. Аналитическая часть

В данном разделе будут описаны конечные автоматы и регулярные выражения.

1.1 Конечные автоматы

Конечный автомат — это модель вычислений, основанная на гипотетической машине состояний. В один момент времени только одно состояние может быть активным. Следовательно, для выполнения каких-либо действий машина должна менять свое состояние. Конечные автоматы обычно используются для организации и представления потока выполнения чего-либо [1].

Конечный автомат можно представить в виде графа, вершины которого являются состояниями, а ребра — переходы между ними. Каждое ребро имеет метку, информирующую о том, когда должен произойти переход.

1.2 Регулярные выражения

Регулярные выражения — это формальный язык поиска подстроки или подстрок в тексте. Для поиска используется строка-образец (паттерн, или шаблон), состоящая из символов и метасимволов (специальные символы, которые обозначают набор символов).

Это довольно мощный инструмент, который может пригодиться во многих случаях — поиск, проверка на корректность строки и т.д. Спектр его возможностей трудно уместить в одну статью [2].

2. Конструкторская часть

В данном разделе будет построены конечный автомат и регулярное выражение для решения поставленной задачи. Шаблон критической секции представлен в листинге 2.1.

Листинг 2.1: Шаблон критической секции

```
1 ::EnterCriticalSection(&obj);
2 // ...
3 ::LeaveCriticalSection(&obj);
```

2.1 Конечный автомат

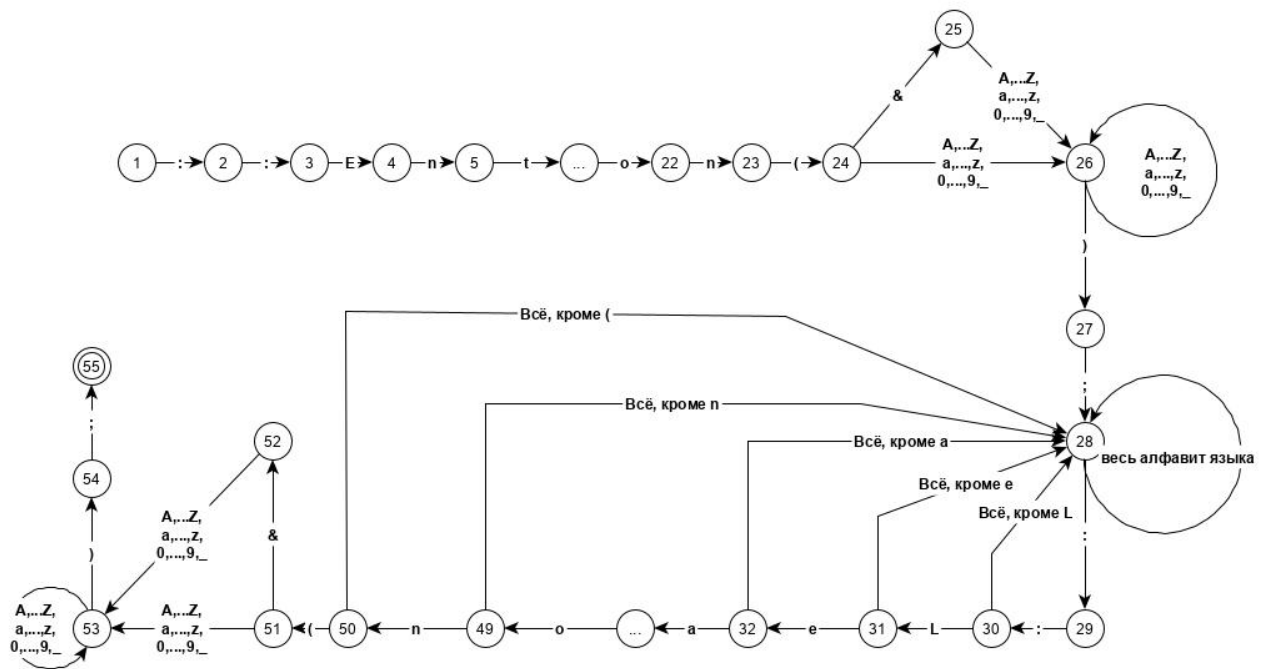


Рис. 2.1: Конечный автомат

2.2 Регулярное выражение

$:: EnterCriticalSection'('(&|\lambda)(A+...+Z+a+...+z)^+)' ; (...)^* :: LeaveCriticalSection'('(&|\lambda)(A+...+Z+a+...+z)^+)' ;$

3. Технологическая часть

3.1 Средства реализации

Для реализации программы был использован язык Python [3] (версия интерпретатора 3.7). Для измерения времени была взята функция `time.time()` из библиотеки `time`. Данный язык обусловлен тем, что функции, необходимые для реализации регулярного выражения, находятся в встроенной библиотеке `re`.

3.2 Реализации алгоритмов

На листингах 3.1 и 3.2 представлены коды реализации поиска критических секций с помощью конечных автоматов и с помощью регулярного выражения.

Листинг 3.1: Поиск с помощью конечного автомата

```
1 enter_str = "::EnterCriticalSection"
2 leave_str = "::LeaveCriticalSection"
3
4 name_symbols = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz_"
5 digits = "0123456789"
6
7
8 def find_crit(s):
9     state = 1
10    crit_section = ""
11    result = []
12
13    for value in s:
14        if state == 1:
15            if value == ' ':
16                state = 2
17            elif 2 <= state <= 22:
18                if value == enter_str[state - 1]:
19                    state += 1
20            else:
21                state = 1
22        elif state == 23:
23            if value == '(':
24                state = 24
25            else:
26                state = 1
27        elif state == 24:
28            if value == '&':
29                state = 25
```

```

30     elif value in name__symbols:
31         state = 26
32     else:
33         state = 1
34 elif state == 25:
35     if value in name__symbols + digits:
36         state = 26
37     else:
38         state = 1
39 elif state == 26:
40     if value in name__symbols + digits:
41         state = 26
42     elif value == ')':
43         state = 27
44     else:
45         state = 1
46 elif state == 27:
47     if value == ';':
48         state = 28
49     else:
50         state = 1
51
52 elif state == 28:
53     if value == ':':
54         state = 29
55
56 elif 29 <= state <= 49:
57     if value == leave__str[state - 28]:
58         state += 1
59     else:
60         state = 28
61 elif state == 50:
62     if value == '(':
63         state = 51
64     else:
65         state = 28
66 elif state == 51:
67     if value == '&':
68         state = 52
69     elif value in name__symbols:
70         state = 53
71     else:
72         state = 28
73 elif state == 52:
74     if value in name__symbols + digits:
75         state = 53
76     else:
77         state = 28
78 elif state == 53:
79     if value in name__symbols + digits:
80         state = 53
81     elif value == ')':
82         state = 54

```

```

83     else:
84         state = 28
85     elif state == 54:
86         if value == ' ';':
87             state = 55
88     else:
89         state = 28
90
91     if state == 1:
92         crit_section = ""
93     else:
94         crit_section += value
95
96     if state == 55:
97         result.append(crit_section)
98         crit_section = ""
99         state = 1
100
101 return result

```

Листинг 3.2: Поиск с помощью регулярного выражения

```

1 regexp = r'::EnterCriticalSection\((([A-Za-z&]\w*)\));' + \
2     r'((?!::EnterCriticalSection\(\1\));)[\s\S]*' + \
3     r'::LeaveCriticalSection\(\1\);'
4
5 def find_crit(text):
6     tmp = text
7
8     found = False
9     match = ""
10
11     while match is not None:
12         match = re.search(regexp, tmp)
13         if match is not None:
14             found = True
15             print(match[0])
16             tmp = tmp[match.end():]
17             print("#####")
18
19     if not found:
20         print("None")

```


4. Экспериментальная часть

4.1 Примеры работы

На рисунке 4.1 представлен пример работы программы.

```
::EnterCriticalSection(&m_obj);
do_work(1);

for (int i = 0; i < 5; i++)
{
    do_work(5);
    do_work(4);
    if (boolvar)
        do_work(7);
    else
        do_work(6);
}
do_work(4);
::LeaveCriticalSection(&m_obj);
#####
::EnterCriticalSection(&m_obj2);
do_work(1);
if (boolvar)
    do_work(7);
else
    do_work(6);
do_work(4);
::LeaveCriticalSection(&m_obj3);
#####
::EnterCriticalSection(&m_obj2);
do_work(1);
if (boolvar)
    do_work(7);
else
    do_work(6);
do_work(4);
::LeaveCriticalSection(&m_obj2);
#####
```

Рис. 4.1: Пример работы программы

Заключение

В ходе выполнения рубежного контроля были реализованы программы поиска критических секций в коде с помощью конечного автомата и регулярного выражения.

Литература

- [1] Конечные автоматы (finite-state machine) [Электронный ресурс].
<https://habr.com/ru/post/358304/>
- [2] Регулярные выражения [Электронный ресурс]. <https://habr.com/ru/company/badoo/blog/34>
- [3] Python [Электронный ресурс]. <https://www.python.org/>