



Universidad La Salle

Compiladores

Informe de la Práctica 04

Autómatas y Expresiones Regulares

Karlo Emigdio Pacha Curimayhua

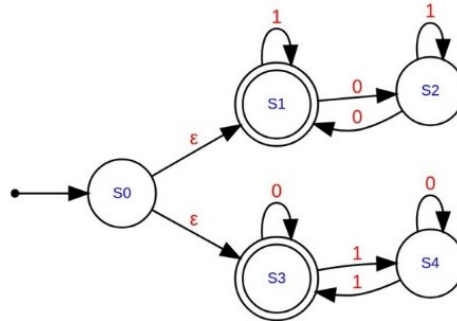
Sexto Semestre - Ingeniería de Software

2023

Ejercicio 1

Enunciado

Convierta el siguiente AFND a un AFD. Detalle todo el desarrollo, puede incluir fotos si lo hizo en papel. (7 puntos)



Solución

Tabla de transiciones AFND

States	0	1	ϵ
$\rightarrow S0$	\emptyset	\emptyset	$\{S1, S3\}$
$*S1$	S2	S1	S1
S2	S1	S2	\emptyset
$*S3$	S3	S4	S3
S4	S4	S3	\emptyset

Tabla de transiciones AFD

States	0	1
$\rightarrow * \{S0, S1, S3\}$	$\{S2, S3\}$	$\{S1, S4\}$
$* \{S2, S3\}$	$\{S1, S3\}$	$\{S2, S4\}$
$* \{S1, S4\}$	$\{S2, S4\}$	$\{S1, S3\}$
$* \{S1, S3\}$	$\{S2, S3\}$	$\{S1, S4\}$
$\{S2, S4\}$	$\{S1, S4\}$	$\{S2, S3\}$

Tabla de transiciones AFD (Renombrando Estados)

Para tener mejor control y lectura del AFD, es recomendable renombrar los estados.

States	
$\rightarrow * \{S0, S1, S3\}$	A
$* \{S2, S3\}$	B
$* \{S1, S4\}$	C
$* \{S1, S3\}$	D
$\{S2, S4\}$	E

AFD inicial

>*A	B	C
*B	D	E
*C	E	D
*D	B	C
E	C	B

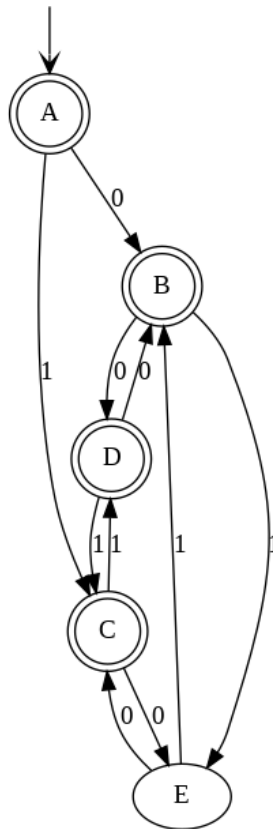


Figure 1: AFD inicial

AFD final (reducido)

Si bien el autómata parece estar terminado, se le puede aplicar una reducción. El estado D tiene transiciones a B y C que no son necesarias ya que ambos estados son estados finales o de aceptación, basta con redireccionar esas transiciones hacia A.

States	0	1
>*A	B	C
*B	A	D
*C	D	A
D	C	B

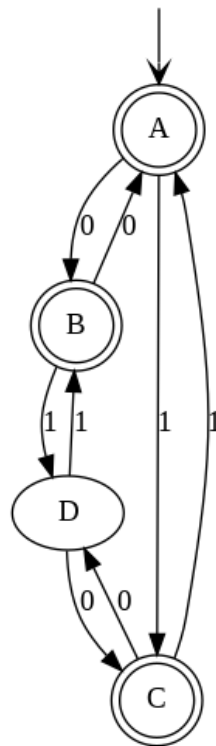


Figure 2: AFD final

Código para graficar los Autómatas

```

1 from graphviz import Digraph
2 from IPython.display import Image
3
4 def CreateAutomata(name):
5     g = Digraph()
6     g.node('invisible', style='invisible')
7     finalStates = []
8     transitions = []
9     initialStates = []
10
11     print('1-Input Initial States: ')

```

```

12 while True:
13     inputState = input('initial state: ')
14     if inputState == '': break
15     initialStates.append(inputState)
16
17 print('2-Input Final States: ')
18 while True:
19     inputState = input('final state: ')
20     if inputState == '': break
21     finalStates.append(inputState)
22
23 print('3-Input Transitions: ')
24 while True:
25     sourceState = input('source state: ')
26     destinationState = input('destination state: ')
27     inputValue = input('input value: ')
28
29     if sourceState == '' and destinationState == '' and inputValue
    == '': break
30     transitions.append([sourceState, destinationState, inputValue])
31
32 for initialState in initialStates: g.edge('invisible',
    initialState, dir='front', arrowhead='vee')
33
34 for finalState in finalStates: g.node(finalState, shape='circle',
    peripheries='2')
35
36 for transition in transitions: g.edge(transition[0], transition
    [1], label=transition[2])
37
38 g.render(name, format='png', view=True)

```

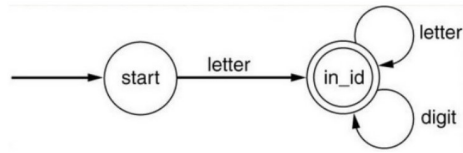
Ejercicio 2

Enunciado

Normalmente para reconocer una expresión regular se siguen los siguientes pasos:

- Convertir la expresión regular a un Automata Finito No Determinista (AFND).
- El AFND debe ser transformado a un Automata Finito Determinista (AFD).
- Finalmente este AFD, es representado mediante una tabla de transiciones.
- Se implementa un programa para reconocer las ocurrencias de una expresión regular utilizando la tabla de transiciones.

En esta caso, le brindamos el AFD para reconocer identificadores, se le pide obtener la tabla de transiciones e implementar un programa en Python que tome esta tabla (la puedes definir estáticamente) y reconozca los identificadores de un archivo de texto (similar a la función findall de python). (11 puntos)



Solución 1

Esta solución no usa Regex en las transiciones, pero cumple satisfactoriamente con lo esperado.

Código para Strings)

```

1 def Automata(A, initialState, finalState, input):
2     state = initialState
3     i = 0
4     while i < len(input):
5         try:
6             state = A[state, input[i]]
7         except KeyError:
8             return False
9         i += 1
10    if state == finalState: return True
11    return False
12
13 A = {
14    ('start', 'a'): 'in_id',
15    ('start', 'b'): 'in_id',
16    ('start', 'c'): 'in_id',
17    ('start', 'd'): 'in_id',
18    ('start', 'e'): 'in_id',
19    ('start', 'f'): 'in_id',
20    ('start', 'g'): 'in_id',
21    ('start', 'h'): 'in_id',
22    ('start', 'i'): 'in_id',
23    ('start', 'j'): 'in_id',
24    ('start', 'k'): 'in_id',
25    ('start', 'l'): 'in_id',
26    ('start', 'm'): 'in_id',
27    ('start', 'n'): 'in_id',
28    ('start', ' '): 'in_id',
29    ('start', 'o'): 'in_id',
30    ('start', 'p'): 'in_id',
31    ('start', 'q'): 'in_id',
32    ('start', 'r'): 'in_id',
33    ('start', 's'): 'in_id',
34    ('start', 't'): 'in_id',
35    ('start', 'u'): 'in_id',
36    ('start', 'v'): 'in_id',
37    ('start', 'w'): 'in_id',
38    ('start', 'x'): 'in_id',
39    ('start', 'y'): 'in_id',
40    ('start', 'z'): 'in_id',
41    ('start', 'A'): 'in_id',
42    ('start', 'B'): 'in_id',

```

```

43 ('start', 'C'): 'in_id',
44 ('start', 'D'): 'in_id',
45 ('start', 'E'): 'in_id',
46 ('start', 'F'): 'in_id',
47 ('start', 'G'): 'in_id',
48 ('start', 'H'): 'in_id',
49 ('start', 'I'): 'in_id',
50 ('start', 'J'): 'in_id',
51 ('start', 'K'): 'in_id',
52 ('start', 'L'): 'in_id',
53 ('start', 'M'): 'in_id',
54 ('start', 'N'): 'in_id',
55 ('start', ' '): 'in_id',
56 ('start', 'O'): 'in_id',
57 ('start', 'P'): 'in_id',
58 ('start', 'Q'): 'in_id',
59 ('start', 'R'): 'in_id',
60 ('start', 'S'): 'in_id',
61 ('start', 'T'): 'in_id',
62 ('start', 'U'): 'in_id',
63 ('start', 'V'): 'in_id',
64 ('start', 'W'): 'in_id',
65 ('start', 'X'): 'in_id',
66 ('start', 'Y'): 'in_id',
67 ('start', 'Z'): 'in_id',
68 #-----
69 ('in_id', '0'): 'in_id',
70 ('in_id', '1'): 'in_id',
71 ('in_id', '2'): 'in_id',
72 ('in_id', '3'): 'in_id',
73 ('in_id', '4'): 'in_id',
74 ('in_id', '5'): 'in_id',
75 ('in_id', '6'): 'in_id',
76 ('in_id', '7'): 'in_id',
77 ('in_id', '8'): 'in_id',
78 ('in_id', '9'): 'in_id',
79 #-----
80 ('in_id', 'a'): 'in_id',
81 ('in_id', 'b'): 'in_id',
82 ('in_id', 'c'): 'in_id',
83 ('in_id', 'd'): 'in_id',
84 ('in_id', 'e'): 'in_id',
85 ('in_id', 'f'): 'in_id',
86 ('in_id', 'g'): 'in_id',
87 ('in_id', 'h'): 'in_id',
88 ('in_id', 'i'): 'in_id',
89 ('in_id', 'j'): 'in_id',
90 ('in_id', 'k'): 'in_id',
91 ('in_id', 'l'): 'in_id',
92 ('in_id', 'm'): 'in_id',
93 ('in_id', 'n'): 'in_id',
94 ('in_id', ' '): 'in_id',
95 ('in_id', 'o'): 'in_id',
96 ('in_id', 'p'): 'in_id',
97 ('in_id', 'q'): 'in_id',
98 ('in_id', 'r'): 'in_id',
99 ('in_id', 's'): 'in_id',
100 ('in_id', 't'): 'in_id',

```

```

101     ('in_id', 'u'): 'in_id',
102     ('in_id', 'v'): 'in_id',
103     ('in_id', 'w'): 'in_id',
104     ('in_id', 'x'): 'in_id',
105     ('in_id', 'y'): 'in_id',
106     ('in_id', 'z'): 'in_id',
107     ('in_id', 'A'): 'in_id',
108     ('in_id', 'B'): 'in_id',
109     ('in_id', 'C'): 'in_id',
110     ('in_id', 'D'): 'in_id',
111     ('in_id', 'E'): 'in_id',
112     ('in_id', 'F'): 'in_id',
113     ('in_id', 'G'): 'in_id',
114     ('in_id', 'H'): 'in_id',
115     ('in_id', 'I'): 'in_id',
116     ('in_id', 'J'): 'in_id',
117     ('in_id', 'K'): 'in_id',
118     ('in_id', 'L'): 'in_id',
119     ('in_id', 'M'): 'in_id',
120     ('in_id', 'N'): 'in_id',
121     ('in_id', ' '): 'in_id',
122     ('in_id', 'O'): 'in_id',
123     ('in_id', 'P'): 'in_id',
124     ('in_id', 'Q'): 'in_id',
125     ('in_id', 'R'): 'in_id',
126     ('in_id', 'S'): 'in_id',
127     ('in_id', 'T'): 'in_id',
128     ('in_id', 'U'): 'in_id',
129     ('in_id', 'V'): 'in_id',
130     ('in_id', 'W'): 'in_id',
131     ('in_id', 'X'): 'in_id',
132     ('in_id', 'Y'): 'in_id',
133     ('in_id', 'Z'): 'in_id'
134 }
135
136 initialState = 'start'
137 finalState = 'in_id'
138
139 inputString = 'abcde'
140 result = Automata(A, initialState, finalState, inputString)
141 print(inputString, result)
142
143 inputString = 'a1b2c3d4e5'
144 result = Automata(A, initialState, finalState, inputString)
145 print(inputString, result)
146
147 inputString = '1a2b3c4d5e6'
148 result = Automata(A, initialState, finalState, inputString)
149 print(inputString, result)

```



```
~/4Auto$ python main2.py
abcde True
a1b2c3d4e5 True
1a2b3c4d5e6 False
~/4Auto$
```

Código para lectura de archivos(ds) y salida txt

```
1 def Automata(A, initialState, finalState, input):
2     matchs = []
3     state = initialState
4     i = 0
5     k = 0
6     l = 1
7     string = ''
8
9     while i < len(input):
10        if input[i] == '\n': l += 1
11        try:
12            state = A[state, input[i]]
13            string += input[i]
14        except KeyError:
15            if string != '':
16                matchs.append([string, [k, i - 1, l]])
17                string = ''
18            k = i + 1
19            state = initialState
20            i += 1
21    if state == finalState and string != '':
22        matchs.append([string, [k, i - 1, l]])
23    return matchs
24
25 A = {
26     ('start', 'a'): 'in_id',
27     ('start', 'b'): 'in_id',
28     ('start', 'c'): 'in_id',
29     ('start', 'd'): 'in_id',
30     ('start', 'e'): 'in_id',
31     ('start', 'f'): 'in_id',
32     ('start', 'g'): 'in_id',
33     ('start', 'h'): 'in_id',
34     ('start', 'i'): 'in_id',
35     ('start', 'j'): 'in_id',
36     ('start', 'k'): 'in_id',
37     ('start', 'l'): 'in_id',
38     ('start', 'm'): 'in_id',
39     ('start', 'n'): 'in_id',
40     ('start', ' '): 'in_id',
41     ('start', 'o'): 'in_id',
42     ('start', 'p'): 'in_id',
43     ('start', 'q'): 'in_id',
44     ('start', 'r'): 'in_id',
45     ('start', 's'): 'in_id',
46     ('start', 't'): 'in_id',
```

```

47 ('start', 'u'): 'in_id',
48 ('start', 'v'): 'in_id',
49 ('start', 'w'): 'in_id',
50 ('start', 'x'): 'in_id',
51 ('start', 'y'): 'in_id',
52 ('start', 'z'): 'in_id',
53 ('start', 'A'): 'in_id',
54 ('start', 'B'): 'in_id',
55 ('start', 'C'): 'in_id',
56 ('start', 'D'): 'in_id',
57 ('start', 'E'): 'in_id',
58 ('start', 'F'): 'in_id',
59 ('start', 'G'): 'in_id',
60 ('start', 'H'): 'in_id',
61 ('start', 'I'): 'in_id',
62 ('start', 'J'): 'in_id',
63 ('start', 'K'): 'in_id',
64 ('start', 'L'): 'in_id',
65 ('start', 'M'): 'in_id',
66 ('start', 'N'): 'in_id',
67 ('start', ' '): 'in_id',
68 ('start', 'O'): 'in_id',
69 ('start', 'P'): 'in_id',
70 ('start', 'Q'): 'in_id',
71 ('start', 'R'): 'in_id',
72 ('start', 'S'): 'in_id',
73 ('start', 'T'): 'in_id',
74 ('start', 'U'): 'in_id',
75 ('start', 'V'): 'in_id',
76 ('start', 'W'): 'in_id',
77 ('start', 'X'): 'in_id',
78 ('start', 'Y'): 'in_id',
79 ('start', 'Z'): 'in_id',
80 #-----
81 ('in_id', '0'): 'in_id',
82 ('in_id', '1'): 'in_id',
83 ('in_id', '2'): 'in_id',
84 ('in_id', '3'): 'in_id',
85 ('in_id', '4'): 'in_id',
86 ('in_id', '5'): 'in_id',
87 ('in_id', '6'): 'in_id',
88 ('in_id', '7'): 'in_id',
89 ('in_id', '8'): 'in_id',
90 ('in_id', '9'): 'in_id',
91 #-----
92 ('in_id', 'a'): 'in_id',
93 ('in_id', 'b'): 'in_id',
94 ('in_id', 'c'): 'in_id',
95 ('in_id', 'd'): 'in_id',
96 ('in_id', 'e'): 'in_id',
97 ('in_id', 'f'): 'in_id',
98 ('in_id', 'g'): 'in_id',
99 ('in_id', 'h'): 'in_id',
100 ('in_id', 'i'): 'in_id',
101 ('in_id', 'j'): 'in_id',
102 ('in_id', 'k'): 'in_id',
103 ('in_id', 'l'): 'in_id',
104 ('in_id', 'm'): 'in_id',

```

```

105     ('in_id', 'n'): 'in_id',
106     ('in_id', ' '): 'in_id',
107     ('in_id', 'o'): 'in_id',
108     ('in_id', 'p'): 'in_id',
109     ('in_id', 'q'): 'in_id',
110     ('in_id', 'r'): 'in_id',
111     ('in_id', 's'): 'in_id',
112     ('in_id', 't'): 'in_id',
113     ('in_id', 'u'): 'in_id',
114     ('in_id', 'v'): 'in_id',
115     ('in_id', 'w'): 'in_id',
116     ('in_id', 'x'): 'in_id',
117     ('in_id', 'y'): 'in_id',
118     ('in_id', 'z'): 'in_id',
119     ('in_id', 'A'): 'in_id',
120     ('in_id', 'B'): 'in_id',
121     ('in_id', 'C'): 'in_id',
122     ('in_id', 'D'): 'in_id',
123     ('in_id', 'E'): 'in_id',
124     ('in_id', 'F'): 'in_id',
125     ('in_id', 'G'): 'in_id',
126     ('in_id', 'H'): 'in_id',
127     ('in_id', 'I'): 'in_id',
128     ('in_id', 'J'): 'in_id',
129     ('in_id', 'K'): 'in_id',
130     ('in_id', 'L'): 'in_id',
131     ('in_id', 'M'): 'in_id',
132     ('in_id', 'N'): 'in_id',
133     ('in_id', ' '): 'in_id',
134     ('in_id', 'O'): 'in_id',
135     ('in_id', 'P'): 'in_id',
136     ('in_id', 'Q'): 'in_id',
137     ('in_id', 'R'): 'in_id',
138     ('in_id', 'S'): 'in_id',
139     ('in_id', 'T'): 'in_id',
140     ('in_id', 'U'): 'in_id',
141     ('in_id', 'V'): 'in_id',
142     ('in_id', 'W'): 'in_id',
143     ('in_id', 'X'): 'in_id',
144     ('in_id', 'Y'): 'in_id',
145     ('in_id', 'Z'): 'in_id'
146 }
147
148 initialState = 'start'
149 finalState = 'in_id'
150
151 data = ''
152
153 with open('example.ds', 'r') as file:
154     data = file.read()
155 tokens = Automata(A, initialState, finalState, data)
156
157 out = open("out1.txt", "w")
158 for token in tokens:
159     print(token)
160     out.write(str(token) + "\n")
161 out.close()
162

```

```

163 print('-----')
164
165 with open('example2.ds', 'r') as file:
166     data = file.read()
167 tokens = Automata(A, initialState, finalState, data)
168
169 out = open("out2.txt", "w")
170 for token in tokens:
171     print(token)
172     out.write(str(token) + "\n")
173 out.close()

```

Solución 2

Esta solución usa Regex en las transiciones y cumple satisfactoriamente con lo esperado, aunque al seguir la lógica del código proporcionado, tiene mayor gasto de recursos y tiempo.

Código para Strings

```

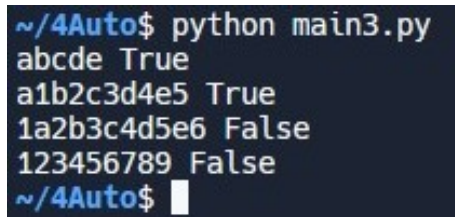
1 import re
2
3 def Automata(A, initialState, finalState, input):
4     state = initialState
5     keys = list(A.keys())
6     i = 0
7     while i < len(input):
8         j = GetKey(keys, state, input[i])
9         if j == 0: return False
10        state = A[state, keys[j-1][1]]
11        i += 1
12
13    if state == finalState: return True
14    return False
15
16 def GetKey(keys, state, input):
17     i = 0
18     for key in keys:
19         if key[0] == state and re.search(key[1], input): return i+1
20         i+=1
21     return 0
22
23 A = {
24     ('start', r'[a-zA-Z]'): 'in_id',
25     ('in_id', r'\d'): 'in_id',
26     ('in_id', r'[a-zA-Z]'): 'in_id'
27 }
28
29
30 initialState = 'start'
31 finalState = 'in_id'
32
33 inputString = 'abcde'
34 result = Automata(A, initialState, finalState, inputString)
35 print(inputString, result)
36
37 inputString = 'a1b2c3d4e5'

```

```

38 result = Automata(A, initialState, finalState, inputString)
39 print(inputString, result)
40
41 inputString = '1a2b3c4d5e6'
42 result = Automata(A, initialState, finalState, inputString)
43 print(inputString, result)
44
45 inputString = '123456789'
46 result = Automata(A, initialState, finalState, inputString)
47 print(inputString, result)

```



```

~/4Auto$ python main3.py
abcde True
a1b2c3d4e5 True
1a2b3c4d5e6 False
123456789 False
~/4Auto$

```

Código para lectura de archivos(ds) y salida txt

```

1 import re
2
3 def Automata(A, initialState, finalState, input):
4     matchs = []
5     state = initialState
6     keys = list(A.keys())
7     i = 0
8     j = 0
9     k = 0
10    l = 1
11    string = ''
12
13    while i < len(input):
14        if input[i] == '\n': l += 1
15        j = GetKey(keys, state, input[i])
16
17        if j == 0:
18            if string != '':
19                matchs.append([string, [k, i - 1, l]])
20                string = ''
21            k = i + 1
22            state = initialState
23        else:
24            state = A[state, keys[j - 1][1]]
25            string += input[i]
26
27        i += 1
28
29    if state == finalState and string != '':
30        matchs.append([string, [k, i - 1, l]])
31    return matchs
32
33
34 def GetKey(keys, state, input):

```

```

35     i = 0
36     for key in keys:
37         if key[0] == state and re.search(key[1], input): return i + 1
38         i += 1
39     return 0
40
41
42 A = {
43     ('start', r'[a-zA-Z]'): 'in_id',
44     ('in_id', r'\d'): 'in_id',
45     ('in_id', r'[a-zA-Z]'): 'in_id'
46 }
47
48 initialState = 'start'
49 finalState = 'in_id'
50
51 data = ''
52
53 with open('example.ds', 'r') as file:
54     data = file.read()
55 tokens = Automata(A, initialState, finalState, data)
56
57 out = open("out1.txt", "w")
58 for token in tokens:
59     print(token)
60     out.write(str(token) + "\n")
61 out.close()
62
63 print('-----')
64
65 with open('example2.ds', 'r') as file:
66     data = file.read()
67 tokens = Automata(A, initialState, finalState, data)
68
69 out = open("out2.txt", "w")
70 for token in tokens:
71     print(token)
72     out.write(str(token) + "\n")
73 out.close()

```

Entrada y Salida de archivos para las soluciones completas

Entrada 1

```

1 func Square (a) {
2     return a*a
3 }
4
5 func main() {
6     Var a2 = 0
7     var b3 = 10.5
8     var cCaA4 = null
9     var BB5 = "abc"
10    var S6 = 'abc'
11    var fA = false
12
13    if a < b & c == null {
14        for i = 0, b, 1 {

```

```

15         a += 1
16     }
17 }
18
19     Square(a)
20 }

```

Salida 1

```

1 ['func', [0, 3, 1]]
2 ['Square', [5, 10, 1]]
3 ['a', [13, 13, 1]]
4 ['return', [20, 25, 2]]
5 ['a', [27, 27, 2]]
6 ['a', [29, 29, 3]]
7 ['func', [34, 37, 5]]
8 ['main', [39, 42, 5]]
9 ['Var', [52, 54, 6]]
10 ['a2', [56, 57, 6]]
11 ['var', [67, 69, 7]]
12 ['b3', [71, 72, 7]]
13 ['var', [85, 87, 8]]
14 ['cCaA4', [89, 93, 8]]
15 ['null', [97, 100, 9]]
16 ['var', [106, 108, 9]]
17 ['BB5', [110, 112, 9]]
18 ['abc', [117, 119, 9]]
19 ['var', [126, 128, 10]]
20 ['S6', [130, 131, 10]]
21 ['abc', [136, 138, 10]]
22 ['var', [145, 147, 11]]
23 ['fA', [149, 150, 11]]
24 ['false', [154, 158, 12]]
25 ['if', [165, 166, 13]]
26 ['a', [168, 168, 13]]
27 ['b', [172, 172, 13]]
28 ['c', [176, 176, 13]]
29 ['null', [181, 184, 13]]
30 ['for', [196, 198, 14]]
31 ['i', [200, 200, 14]]
32 ['b', [207, 207, 14]]
33 ['a', [226, 226, 15]]
34 ['Square', [258, 263, 19]]
35 ['a', [265, 265, 19]]

```

Entrada 2

```

1 func main() {
2     print(1)
3     var a = read(2)
4     var 1MiIdentificador = 10
5     var mi_segundo_identificador = 20
6     var a1 = true
7     var 1enitero = 123
8     var 1flotante = 1.23
9     var b3ooleano = true
10    var 1nu2lo = null
11

```

```
12     return 0
13 }
```

Salida 2

```
1 ['func', [0, 3, 1]]
2 ['main', [5, 8, 1]]
3 ['print', [18, 22, 2]]
4 ['var', [31, 33, 3]]
5 ['a', [35, 35, 3]]
6 ['read', [39, 42, 3]]
7 ['var', [51, 53, 4]]
8 ['MiIdentificador', [56, 70, 4]]
9 ['var', [81, 83, 5]]
10 ['mi', [85, 86, 5]]
11 ['segundo', [88, 95, 5]]
12 ['identificador', [97, 109, 5]]
13 ['var', [120, 122, 6]]
14 ['a1', [124, 125, 6]]
15 ['true', [129, 132, 7]]
16 ['var', [138, 140, 7]]
17 ['entero', [143, 149, 7]]
18 ['var', [161, 163, 8]]
19 ['flotante', [166, 173, 8]]
20 ['var', [186, 188, 9]]
21 ['booleano', [190, 198, 9]]
22 ['true', [202, 205, 10]]
23 ['var', [211, 213, 10]]
24 ['nu2lo', [216, 220, 10]]
25 ['null', [224, 227, 11]]
26 ['return', [234, 239, 12]]
```

 Repositorio GitHub