

```

1: function NextWord()
2:   state  $\leftarrow s_0$ 
3:   lexeme  $\leftarrow ''$ 
4:   clear(stack)
5:   push(bad)
6:   while state  $\neq s_e$  do
7:     NextChar(char)
8:     lexeme  $\leftarrow \text{lexeme} + \text{char}$ 
9:     if state  $\in S_A$  then
10:       ▷ 一旦到达一个可接受状态就清除所有之前的状态（包括所有bad状态），
11:       ▷ 所以这是匹配最长的串
12:       clear(stack)
13:     end if
14:     push(state) ▷ 这也可能是一个bad状态
15:     cat  $\leftarrow \text{CharCat}[\text{char}]$ 
16:     state  $\leftarrow \delta[\text{state}, \text{cat}]$ 
17:   end while
18:   ▷ 如果状态是bad，因为匹配的是最长的串，所以就算之前有一个可接受状态，
19:   ▷ 但只要在到达 $s_e$ 前到达bad，bad状态也会被记录且起作用
20:   while state  $\notin S_A$  and state  $\neq \text{bad}$  do
21:     state  $\leftarrow \text{pop}()$ 
22:     truncate(lexeme)
23:     RollBack()
24:   end while
25:   if state  $\in S_A$  then
26:     return Type[state]
27:   else
28:     return invalid
29:   end if
30: end function

```