

Analisi sintattico top-down  
metodo del "discendente ricorsivo"

# Uno pseudo-codice per descrivere programmi e algoritmi

Istruzioni:

- assegnazione:  $\leftarrow$  (esempio:  $A \leftarrow B$ )
- espressioni numeriche e booleane
- dichiarazione e chiamata di funzione (metodo) *anche ricorsive*:  
**nome** (*param 1, param 2, ...*)  
**nome()** (se senza parametri)
- ritorno da una funzione: **return** (**valore**) (opzionale, se non c'è valore da ritornare)

Dati composti:

- i-esimo elemento array A: **A[i]**
- liste definite come **<m1, m2, .. >**. Funzioni **cons(m, l)** e **nil** (stringa vuota)
- funzioni hd(lista) (primo elemento) e tl(lista) (secondo elemento)

struttura di blocco (sequenza): indentazione, oppure { }, oppure **begin end**

costrutto condizionale:

```
if (condizione)
    istruzione
[elseif (condizione) istruzione]
[elseif (condizione) istruzione]
...
[else istruzione]
```

costrutto iterativi:

```
while (condizione)
    istruzione
```

## Esempi

Fattoriale (chamata tipo: fact(10) )

```
fact (n)
  if (n=0) f ← 1
  else f ← n * fact(n-1)
return f
```

Lista dei primi numeri pari fino a n (chiamata tipo plist(10) )

```
plist (n)
  if (n = 0) lista ← <0>
  elseif (n % 2 ≠ 0) lista ← plist(n-1)
  else lista ← cons(n, plist(n-1))
return lista
```

# Notazioni in pseudo-codice

Nella slide successiva viene utilizzato lo pseudo-codice per descrivere l'algoritmo di parsificazione. Si assume che il programma abbia accesso agli insiemi guida della grammatica LL(1) da parsificare.

- cc contiene il primo simbolo dell'input, quello su cui verranno prese le decisioni;
- PROSS è una funzione che restituisce simbolo sotto la testina di lettura e avanza la testina stessa;

# Analizzatore a "discesa ricorsiva": idea base

Esempio

acb&c

$S \rightarrow PQ$

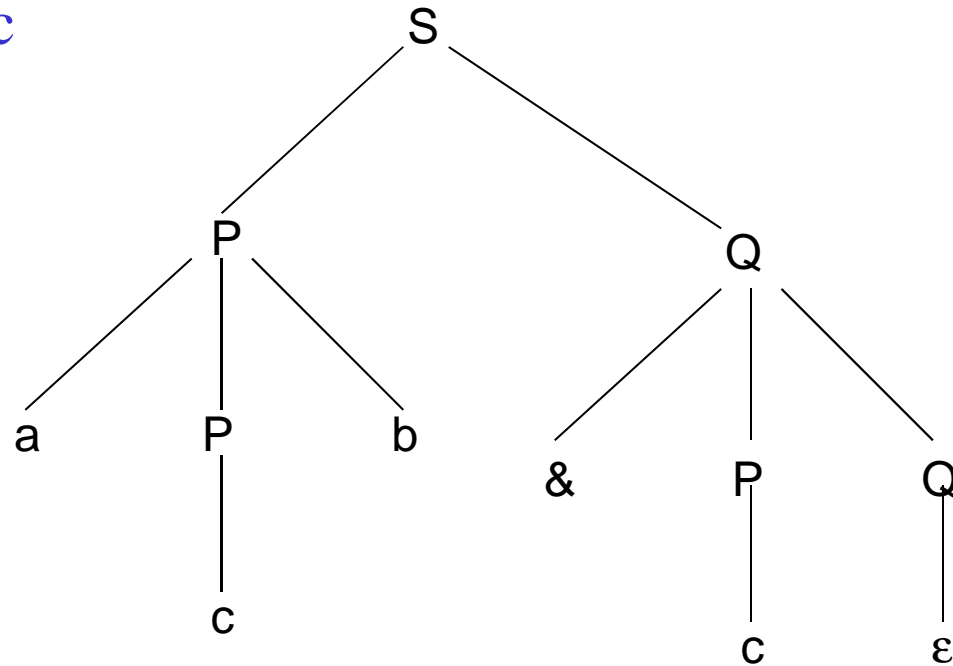
$Q \rightarrow \&PQ$

$Q \rightarrow \epsilon$

$P \rightarrow aPb$

$P \rightarrow bPa$

$P \rightarrow c$



-Si può associare ad *ogni variabile* A una procedura (ricorsiva) in modo che l'albero (implicito) delle chiamate ricorsive corrisponda all'albero sintattico della stringa da parsificare.

-Idea base: una procedura associata ad una variabile A "riconosce" una stringa generabile da A

## Analizzatore a "discesa ricorsiva":idea base

In particolare la procedura associata ad una variabile A è tale che:

- la porzione della stringa di input contenuta tra la posizione iniziale (compresa) del contatore cc (quando A viene chiamata
- e la posizione finale (esclusa) del contatore cc (quando si esegue il ritorno *dalla stessa chiamata* di A ) è una stringa generabile da A nella grammatica.

Nota: il puntatore cc è globale (statico) e visibile da tutte le funzioni.

# Analizzatore a Discesa Dicorsiva

Consideriamo una grammatica LL(1)

*Ad ogni variabile A con produzioni*

$A \rightarrow \alpha_1$

$A \rightarrow \alpha_2$

$\dots$   
 $A \rightarrow \alpha_k$

si associa una procedura:

function A()

begin

if (cc  $\in$  Gui ( $A \rightarrow \alpha_1$ )) body ( $\alpha_1$ )

else if (cc  $\in$  Gui( $A \rightarrow \alpha_2$ )) body ( $\alpha_2$ )

$\dots$

else if (cc  $\in$  Gui( $A \rightarrow \alpha_k$ )) body ( $\alpha_k$ )

else ERRORE (...)

end

## Parsificazione top-down: analizzatore a discesa ricorsiva

Se  $\alpha = \varepsilon$ ,  $\text{body}(\varepsilon) = \underline{\text{do nothing}}$

Se  $\alpha = X_1, \dots, X_m$ ,  $\text{body}(X_1, \dots, X_m)$  è così definito:

$\text{body}(X_1, \dots, X_m) = \text{act}(X_1) \text{ act}(X_2) \dots \text{act}(X_m)$

$$\text{act}(X) = \begin{cases} X() & \text{se } X \in V \\ \underline{\text{if}} (cc = X) \text{ cc} \leftarrow \text{PROSS} & \text{se } X \in \Sigma \\ \underline{\text{else}} \text{ERRORE}(\dots) & \end{cases}$$

Alcune azioni possono essere raggruppate in modo intuitivo

```
main() //Program discesa_ricorsiva
begin cc ← PROSS
    S()
    if (cc = '$') "stringa accettata"
    else ERRORE(...)
end
```



## Parsificazione top-down: analizzatore a discesa ricorsiva

### Grammatica

1.  $S \rightarrow PQ$
2.  $Q \rightarrow \&PQ$
3.  $Q \rightarrow \varepsilon$
4.  $P \rightarrow aPb$
5.  $P \rightarrow bPa$
6.  $P \rightarrow c$

### Insieme guida

- {a, b, c}
- {&}
- {\$}
- {a}
- {b}
- {c}

// Program discesa\_ricorsiva

main()

begin cc := PROSS

S()

if cc = '\$' "stringa accettata"

else ERRORE (...)

end

function S()

begin if cc = 'a' or cc = 'b' or cc = 'c'

P()

Q()

else ERRORE (...)

end

## Parsificazione top-down: analizzatore a discesa ricorsiva

```
function P()  
begin if cc = 'a' cc ← PROSS  
      P()  
      if cc = 'b' cc ← PROSS  
      else ERRORE(...)  
else if cc = 'b' cc ← PROSS  
      P()  
      if cc = 'a' cc ← PROSS  
      else ERRORE(...)  
else if cc = 'c' cc ← PROSS  
      else ERRORE(...)  
end
```

```
function Q()  
begin if cc = '&' cc ← PROSS  
      P()  
      Q()  
      else if cc = '$' do nothing  
      else ERRORE (...)  
end
```

# Parsificazione top-down: analizzatore a discesa ricorsiva

a a c b b & b c a \$

[ ][ ][ ][ ][S]

[ ][ ][P][Q]

[ ][P][b][Q] 1

[P][b][b][Q] 2

[ ][b][b][Q]

[ ][ ][b][Q]

[ ][ ][ ][Q] 3

[ ][ ][P][Q]

[ ][P][a][Q] 4

[ ][ ][a][Q]

[ ][ ][ ][Q]

[ ][ ][ ][ ][ ]

if cc = 'a' or cc = 'b' or cc = 'c'

P()

if cc = 'a' cc ← PROSS

P() (punto 1)

if cc = 'a' cc ← PROSS

P() (punto 2)

if cc = 'c' cc ← PROSS

if cc = 'b' cc ← PROSS  
if cc = 'b' cc ← PROSS

Q() (punto 3)

if cc = '&' cc ← PROSS

P()

if cc = 'b' cc ← PROSS

P() (punto 4)

if cc = 'c' then cc ← PROSS

if cc = 'a' cc ← PROSS

Q()

if cc = '\$' do nothing

S → PQ

Q → &PQ


Q → ε

P → aPb

P → bPa

P → c

## Parsificazione top-down: analizzatore a discesa ricorsiva, altro esempio

Grammatica	Insieme guida	
$S \rightarrow ( S )$	$\{ ( \}$	 LL(1)
$S \rightarrow [ S ]$	$\{ [ \}$	
$S \rightarrow < S >$	$\{ < \}$	
$S \rightarrow \varepsilon$	$\{ ), ], >, \$ \}$	

```
main()  // Program discesa_ricorsiva
  begin cc ← PROSS
        S()
        if (cc = '$') "stringa accettata"
        else ERRORE(...)
  end
```

## Parsificazione top-down: analizzatore a discesa ricorsiva

```
function S
  begin if (cc = '(') cc ← PROSS
      S()
      if (cc = ')') cc ← PROSS
      else ERRORE(...)
  else if (cc = '[') cc ← PROSS
      S()
      if (cc = ']') cc ← PROSS
      else ERRORE(...)
  else if (cc = '<') cc ← PROSS
      S()
      if (cc = '>') cc ← PROSS
      else ERRORE(...)
  else if (cc = ') or cc = ']' or cc = '>' or cc = '$')
      do nothing
  else ERRORE (...)
end
```

## Parsificazione top-down: analizzatore a discesa ricorsiva, altro esempio

Grammatica

1  $S \rightarrow a B b$

2  $S \rightarrow B S$

3  $B \rightarrow b$

4  $B \rightarrow c$

Insieme guida

{a}

{b, c}

{b}

{c}

 LL(1)

```
main()    // Program discesa_ricorsiva
begin cc ← PROSS
          S()
          if (cc = '$') "stringa accettata"
          else ERRORE(...)
end
```

## Parsificazione top-down: analizzatore a discesa ricorsiva, altro esempio

```
function S()  
  begin  if (cc = 'a')  
          cc ← PROSS  
          B()  
          if (cc = 'b') cc ← PROSS  
          else ERRORE(...)  
  else if (cc = 'b' or cc = 'c')  
          B()  
          S()  
  else ERRORE(...)  
end  
  
  function B()  
    begin  if (cc = 'b') cc ← PROSS  
            else if (cc = 'c') cc ← PROSS  
            else ERRORE(...)  
    end
```