Compiladores

Aula 7

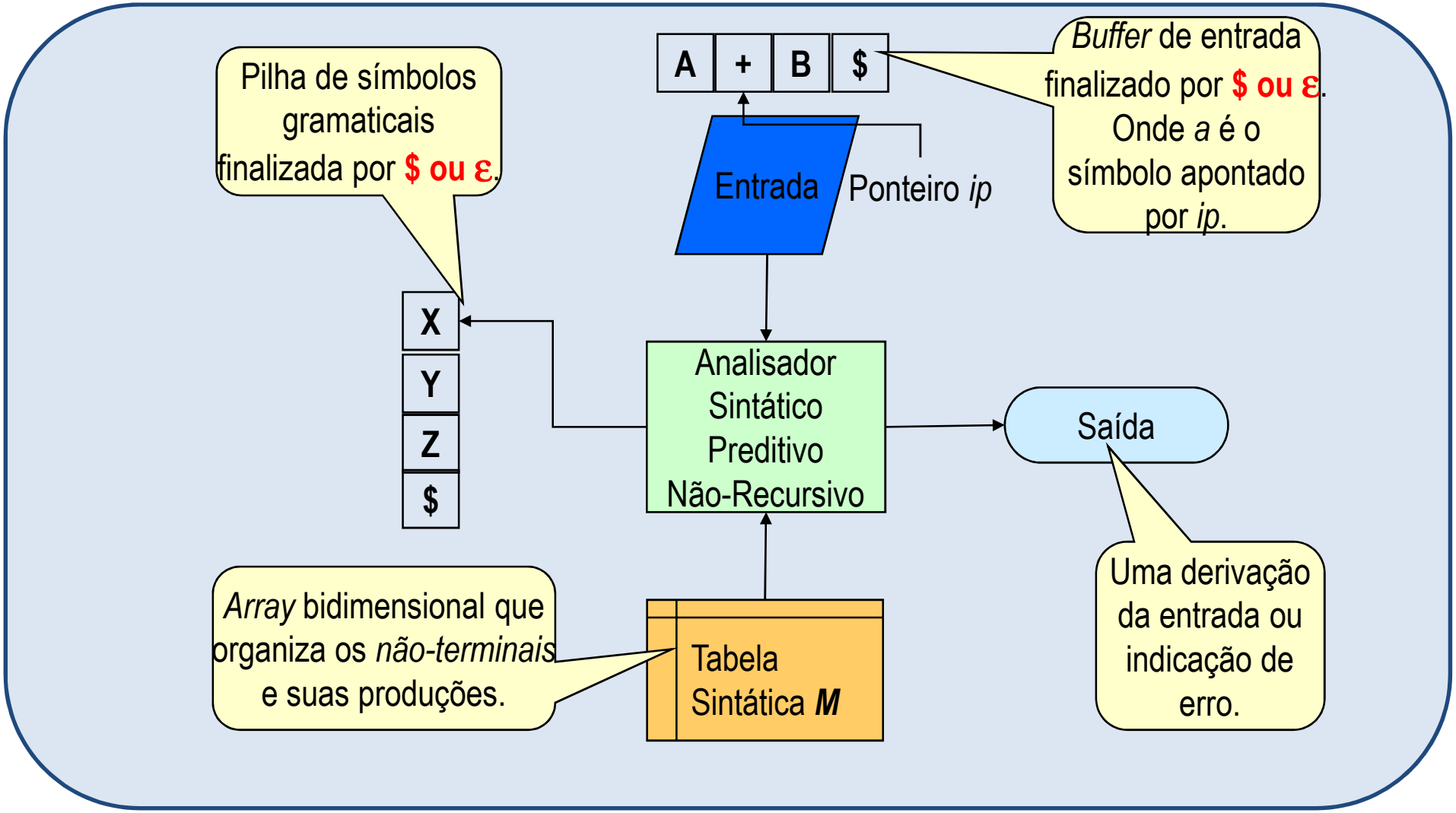
Celso Olivete Júnior

olivete@fct.unesp.br

Na aula de hoje

- Tratamento de erros na análise sintática descendente não-recursiva (tabela + pilha) e recursiva (procedimentos recursivos)

Análise sintática descendente não-recursiva



ASD não-recursiva

- Relembrando a construção da tabela de análise sintática descendente

Algoritmo para construção da tabela de análise sintática

- Para cada produção A → α da gramática faça:

1. Para cada terminal *a* em FIRST(A), inclua A→ α em M[A,a]

2. Se FIRST(α) inclui a palavra vazia, então adicione A→ ε a M[A,b] para cada b em FOLLOW(A)

3. Entrada indefinida indica ERRO

Não Terminal	Símbolo de Entrada		
	a	b	ε
S	S→AB 1.	S→AB 1.	S → AB 1.
A	A → aA 1.	A → ε 2.	A → ε 2.
B	ERRO 3.	B → bB	B → ε 2.

G1

S → AB
A → aA | ε
B → bB | ε
First(S) = { b, a, ε}
First(A) = {a, ε}
First(B) = {b, ε }
Follow(S)={$}
Follow(A)={b,$}
Follow(B)={$}

Algoritmo para construção da tabela de análise sintática

- Para cada produção A → α da gramática faça:

1. Para cada terminal *a* em FIRST(A), inclua A→ α em M[A,a]

2. Se FIRST(α) inclui a palavra vazia, então adicione A→ ε a M[A,b] para cada b em FOLLOW(A)

3. Entrada indefinida indica ERRO

Não Terminal	Símbolo de Entrada		
	a	b	ε
S	S→AB 1.	S→AB 1.	S → AB 1.
A	A → aA 1.	A → ε 2.	A → ε 2.
B	ERRO 3.	B → bB	B → ε 2.

Análise da sentença "aabbb"
- mostrando a pilha, entrada e regra usada

G1

S → AB
A → aA | ε
B → bB | ε
First(S) = { b, a, ε}
First(A) = {a, ε}
First(B) = {b, ε }
Follow(S)={$}
Follow(A)={b,$}
Follow(B)={$}

Análise da sentença aabbb utilizando a tabela de análise sintática

Pilha	Entrada	Regra
Sε	aabbbε	S→AB

Não Terminal	Símbolo de Entrada		
	a	b	ε
S	S→AB 1.	S→AB 1.	S → AB 1.
A	A → aA 1.	A → ε 2.	A → ε 2.
B	ERRO 3.	B → bB	B → ε 2.

G1
S → AB A → aA | ε B → bB | ε

Análise da sentença aabbb utilizando a tabela de análise sintática

Pilha	Entrada	Regra
Sε	aabbbε	S→AB
ABε	aabbbε	A→aA

Não Terminal	Símbolo de Entrada		
	a	b	ε
S	S→AB 1.	S→AB 1.	S → AB 1.
A	A → aA 1.	A → ε 2.	A → ε 2.
B	ERRO 3.	B → bB	B → ε 2.

G1
S → AB A → aA | ε B → bB | ε

Análise da sentença aabbb utilizando a tabela de análise sintática

Pilha	Entrada	Regra
Sε	aabbbε	S→AB
ABε	aabbbε	A→aA
aABε	aabbbε	Casa a
ABε	abbbε	A→aA
aABε	abbbε	Casa a
ABε	bbbε	A→ε
Bε	bbbε	B→bB
bBε	bbbε	Casa b
Bε	bbε	B→bB
bBε	bbε	Casa b
Bε	bε	B→bB
bBε	bε	Casa b
Bε	ε	B→ε
ε	ε	**aceitou**

Não Terminal	Símbolo de Entrada		
	a	b	ε
S	S→AB 1.	S→AB 1.	S → AB 1.
A	A → aA 1.	A → ε 2.	A → ε 2.
B	ERRO 3.	B → bB	B → ε 2.

G1
S → AB A → aA | ε B → bB | ε

Análise da sentença ababb utilizando a tabela de análise sintática

Pilha	Entrada	Regra
Sε	ababbε	S→AB
ABε	ababbε	A→aA
aABε	ababbε	Casa a

Não Terminal	Símbolo de Entrada		
	a	b	ε
S	S→AB 1.	S→AB 1.	S → AB 1.
A	A → aA 1.	A → ε 2.	A → ε 2.
B	ERRO 3.	B → bB	B → ε 2.

G1
S → AB A → aA | ε B → bB | ε

Análise da sentença aba utilizando a tabela de análise sintática

Pilha	Entrada	Regra
Sε	ababbε	S→AB
ABε	ababbε	A→aA
aABε	ababbε	Casa a
ABε	babbε	A→ε
Bε	babbε	B → bB
bBε	babbε	Casa b
Bε	abbε	ERRO

Não Terminal	Símbolo de Entrada		
	a	b	ε
S	S→AB 1.	S→AB 1.	S → AB 1.
A	A → aA 1.	A → ε 2.	A → ε 2.
B	ERRO 3.	B → bB	B → ε 2.

Como prosseguir a análise?

G1
S → AB A → aA | ε B → bB | ε

Recuperação de erros na análise sintática

- A pilha (no topo) de um analisador preditivo não-recursivo torna claro os terminais e não-terminais que ele espera reconhecer com o restante da entrada
- Dessa forma, sabe-se que ocorre um erro quando:
  - o símbolo corrente da entrada não corresponde ao terminal contido no topo da pilha **OU**
  - o símbolo corrente da entrada não possui produção correspondente a partir do não-terminal contido no topo da pilha.

Recuperação de erros na análise sintática

- Tais erros podem ser recuperados através da **modalidade do desespero (pânico)**, por exemplo, através do descarte de símbolos da entrada até a localização de um *token* de **sincronização**.
- Veremos como esses tokens são incluídos a seguir...

Não Terminal	Símbolo de Entrada					
	Id	+	*	(	)	$
E	E::=TE'			E::=TE'	SINC	SINC
E'		E'::=+TE'			E'::=ε	E'::=ε
T	T::=FT'	SINC		T::=FT'	SINC	SINC
T'		T'::= ε	T'::=*FT'		T'::= ε	T'::= ε
F	F::=**id**	SINC	SINC	F::=(E)	SINC	SINC

1.

Recuperação de erros na análise sintática

- A eficiência da recuperação de erros desta forma depende da escolha de um conjunto adequado de *tokens* de sincronização.

- A recuperação de erros é provável na maioria dos casos embora não possa ser assegurada completamente.

Não Terminal	Símbolo de Entrada					
	Id	+	*	(	)	$
E	E::=TE'			E::=TE'	SINC	SINC
E'		E'::=+TE'			E'::=ε	E'::=ε
T	T::=FT'	SINC		T::=FT'	SINC	SINC
T'		T'::= ε	T'::=*FT'		T'::= ε	T'::= ε
F	F::=**id**	SINC	SINC	F::=(E)	SINC	SINC

1.

Recuperação de erros na análise sintática

❑ Modalidade desespero

- Considerando a ocorrência de um erro durante a expansão do não-terminal A:
  - ✓ Como *tokens* de *sincronização* usam-se todos os símbolos em Follow(A).
  - ✓ Descartam-se os símbolos de entrada até encontrar-se um elemento de Follow(A), quando também descartamos A. É provável que a análise sintática possa prosseguir.

Mensagens de erro

- Todas as mensagens de erro produzidas por um compilador devem ser informativas o suficiente para permitir a rápida localização e correção do erro

- Sugere-se que cada entrada vazia da tabela sintática preditiva contenha apontadores para rotinas de tratamento de erro onde mensagens apropriadas serão fornecidas conforme o erro que as provocou

Algoritmo para construção da tabela de análise sintática

- Para cada produção A → α da gramática faça:

  1. Para cada terminal *a* em FIRST(A), inclua A→ α em M[A,a]

  2. Se FIRST(α) inclui a palavra vazia, então adicione A→ ε a M[A,b] para cada b em FOLLOW(A)

Não Terminal	Símbolo de Entrada					
	Id	+	*	(	)	ε
E	E::=TE' 1.			E::=TE' 1.	SINC	SINC
E'		E'::=+TE' 1.			E'::=ε 2.	E'::=ε 2.
T	T::=FT' 1.	SINC		T::=FT' 1.	SINC	SINC
T'		T'::= ε 2.	T'::=*FT' 1.		T'::= ε 2.	T'::= ε 2.
F	F::=**id** 1.	SINC	SINC	F::=(E) 1.	SINC	SINC

E → TE’	First(E)= First(*T*) = {*(, id*}	Follow(E)= FIRST(*)*) U {$} = {*)*,$}
E’ → +TE’ | **ε**	First(E’)= First(*+*) U First(*ε*) = {*+, ε*}	Follow(E’)= Follow(E) = {*)*,$}
T → FT’	First(T)= First(*F*) = {*(, id*}	Follow(T)= First(E’) U Follow(E’) = {+,*)*,$}
T’ → *FT’ | **ε**	First(T’)= First(*) U First(*ε*) = {*, ε*}	Follow(T’)= Follow(T) = {+,*)*,$}
F → (E) | **id**	First(F)= First(*(*) U First(*id*) = {*(, id*}	Follow(F)= First(T’) U Follow(T’) = {*,+,*)*,$}

- Olhando para os conjuntos *Follows,* obtêm-se os tokens de sincronismo (**SINC**) para cada um dos não-terminais em análise. Por exemplo: em T tem-se o SINC para os terminais {+,),$}. O mesmo processo deve ser repetido para os demais não-terminais, seguindo os conjuntos Follows

-

Não Terminal	Símbolo de Entrada					
	Id	+	*	(	)	$
E	E::=TE’ 1.			E::=TE’ 1.	**SINC**	**SINC**
E’		E’::=+TE’ 1.			E’::=ε 2.	E’::=ε 2.
T	T::=FT’ 1.	**SINC**		T::=FT’ 1.	**SINC**	**SINC**
T’		T’::= ε 2.	T’::=*FT’ 1.		T’::= ε 2.	T’::= ε 2.
F	F::=**id** 1.	**SINC**	**SINC**	F::=(E) 1.	**SINC**	**SINC**

1. Se o analisador sintático procurar pela entrada M[A,*a*] e encontrar que a mesma está em **branco**, então o símbolo *a* é **ignorado** (pula).
2. Se a entrada for SINC então o não-terminal ao topo da pilha é removido numa tentativa de se retomar a análise sintática.
3. Se um token ao topo da pilha não casar com o símbolo de entrada, então é removido da pilha.

Considerando a entrada errônea

+ id * + id

Pilha	Entrada	Ação	Comentário
E$	+ id * + id$		REGRA 1 Erro (+) pula

Não Terminal	Símbo					
	Id	+	*	(	)	$
E	E::=TE’			E::=TE’	SINC	SINC
E’		E’::=+TE’			E’::=ℰ	E’::=ℰ
T	T::=FT’	SINC		T::=FT’	SINC	SINC
T’		T’::= ℰ	T’::=*FT’		T’::= ℰ	T’::= ℰ
F	F::=**id**	SINC	SINC	F::=(E)	SINC	SINC

E →TE’
E’ →+TE’|ε
T →FT’
T’ →*FT’|ε
F → (E)|**id**

1.

1. Se o analisador sintático procurar pela entrada M[A,*a*] e encontrar que a mesma está em **branco**, então o símbolo *a* é **ignorado**(pula)
2. Se a entrada for SINC então o não-terminal ao topo da pilha é removido numa tentativa de se retomar a análise sintática.
3. Se um token ao topo da pilha não casar com o símbolo de entrada, então é removido da pilha.

Considerando a entrada errônea

+ id * + id

Pilha	Entrada	Ação	Comentário
E$	+ id * + id$		REGRA 1 Erro (+) pula
E$	id * + id$	E→TE '	id está em First(E)

Não Terminal	Símbo					
	Id	+	*	(	)	$
E	E::=TE'			E::=TE'	SINC	SINC
E'		E'::=+TE'			E'::=ℰ	E'::=ℰ
T	T::=FT'	SINC		T::=FT'	SINC	SINC
T'		T'::= ℰ	T'::=*FT'		T'::= ℰ	T'::= ℰ
F	F::=id	SINC	SINC	F::=(E)	SINC	SINC

E →TE'
E' →+TE' | ε
T →FT'
T' →*FT' | ε
F → (E) | id

1.

1. Se o analisador sintático procurar pela entrada M[A,*a*] e encontrar que a mesma está em **branco**, então o símbolo *a* é **ignorado**.
2. Se a entrada for SINC então o não-terminal ao topo da pilha é removido numa tentativa de se retomar a análise sintática.
3. Se um token ao topo da pilha não casar com o símbolo de entrada, então é removido da pilha.

Considerando a entrada errônea

+ id * + id

Pilha	Entrada	Ação	Comentário
E$	+ id * + id$		REGRA 1 Erro (+) pula
E$	id * + id$	E→TE '	id está em First(E)
TE '$	id * + id$	T→FT '	

Não Terminal	Símbo					
	Id	+	*	(	)	$
E	E::=TE'			E::=TE'	SINC	SINC
E'		E'::=+TE'			E'::=ℰ	E'::=ℰ
T	T::=FT'	SINC		T::=FT'	SINC	SINC
T'		T'::= ℰ	T'::=*FT'		T'::= ℰ	T'::= ℰ
F	F::=id	SINC	SINC	F::=(E)	SINC	SINC

E →TE'
E' →+TE' | ε
T →FT'
T' →*FT' | ε
F → (E) | id

1.

1. Se o analisador sintático procurar pela entrada M[A,*a*] e encontrar que a mesma está em **branco**, então o símbolo *a* é **ignorado**.
2. Se a entrada for SINC então o não-terminal ao topo da pilha é removido numa tentativa de se retomar a análise sintática.
3. Se um token ao topo da pilha não casar com o símbolo de entrada, então é removido da pilha.

Considerando a entrada errônea

+ id * + id

Pilha	Entrada	Ação	Comentário
E$	+ id * + id$		REGRA 1 Erro (+) pula
E$	id * + id$	E→TE '	id está em First(E)
TE '$	id * + id$	T→FT '	
FT ' E' $	id * + id$	F→id	

Não Terminal	Símbo					
	Id	+	*	(	)	$
E	E::=TE'			E::=TE'	SINC	SINC
E'		E'::=+TE'			E'::=ℰ	E'::=ℰ
T	T::=FT'	SINC		T::=FT'	SINC	SINC
T'		T'::= ℰ	T'::=*FT'		T'::= ℰ	T'::= ℰ
F	F::=id	SINC	SINC	F::=(E)	SINC	SINC

E →TE'
E' →+TE'|ε
T →FT'
T' →*FT'|ε
F → (E)|id

1.

1. Se o analisador sintático procurar pela entrada M[A,*a*] e encontrar que a mesma está em **branco**, então o símbolo *a* é **ignorado**.
2. Se a entrada for SINC então o não-terminal ao topo da pilha é removido numa tentativa de se retomar a análise sintática.
3. Se um token ao topo da pilha não casar com o símbolo de entrada, então é removido da pilha.

Considerando a entrada errônea

+ id * + id

Pilha	Entrada	Ação	Comentário
E$	+ id * + id$		REGRA 1 Erro (+) pula
E$	id * + id$	E→TE '	id está em First(E)
TE '$	id * + id$	T→FT '	
FT ' E' $	id * + id$	F→id	
idT ' E' $	id * + id$	Casa id	

Não Terminal	Símbo					
	Id	+	*	(	)	$
E	E::=TE'			E::=TE'	SINC	SINC
E'		E'::=+TE'			E'::=ℰ	E'::=ℰ
T	T::=FT'	SINC		T::=FT'	SINC	SINC
T'		T'::= ℰ	T'::=*FT'		T'::= ℰ	T'::= ℰ
F	F::=id	SINC	SINC	F::=(E)	SINC	SINC

E →TE'
E' →+TE' | ε
T →FT'
T' →*FT' | ε
F → (E) | id

1.

1. Se o analisador sintático procurar pela entrada M[A,*a*] e encontrar que a mesma está em **branco**, então o símbolo *a* é **ignorado**.
2. Se a entrada for SINC então o não-terminal ao topo da pilha é removido numa tentativa de se retomar a análise sintática.
3. Se um token ao topo da pilha não casar com o símbolo de entrada, então é removido da pilha.

Considerando a entrada errônea

+ id * + id

Pilha	Entrada	Ação	Comentário
E$	+ id * + id$		REGRA 1 Erro (+) pula
E$	id * + id$	E→TE '	id está em First(E)
TE '$	id * + id$	T→FT '	
FT ' E' $	id * + id$	F→id	
idT ' E' $	id * + id$	Casa id	
T ' E' $	* + id$	T'→*FT '	

Não Terminal	Símbo					
	Id	+	*	(	)	$
E	E::=TE'			E::=TE'	SINC	SINC
E'		E'::=+TE'			E'::=ℰ	E'::=ℰ
T	T::=FT'	SINC		T::=FT'	SINC	SINC
T'		T'::= ℰ	T'::=*FT'		T'::= ℰ	T'::= ℰ
F	F::=id	SINC	SINC	F::=(E)	SINC	SINC

E →TE'
E' →+TE' | ε
T →FT'
T' →*FT' | ε
F → (E) | id

1.

1. Se o analisador sintático procurar pela entrada M[A,*a*] e encontrar que a mesma está em **branco**, então o símbolo *a* é **ignorado**.
2. Se a entrada for SINC então o não-terminal ao topo da pilha é removido numa tentativa de se retomar a análise sintática.
3. Se um token ao topo da pilha não casar com o símbolo de entrada, então é removido da pilha.

Considerando a entrada errônea

+ id * + id

Pilha	Entrada	Ação	Comentário
E$	+ id * + id$		REGRA 1 Erro (+) pula
E$	id * + id$	E→TE '	id está em First(E)
TE '$	id * + id$	T→FT '	
FT ' E' $	id * + id$	F→id	
idT ' E' $	id * + id$	Casa id	
T ' E' $	* + id$	T'→*FT '	
*FT ' E' $	* + id$	Casa *	

Não Terminal	Símbo					
	Id	+	*	(	)	$
E	E::=TE'			E::=TE'	SINC	SINC
E'		E'::=+TE'			E'::=ℰ	E'::=ℰ
T	T::=FT'	SINC		T::=FT'	SINC	SINC
T'		T'::= ℰ	T'::=*FT'		T'::= ℰ	T'::= ℰ
F	F::=id	SINC	SINC	F::=(E)	SINC	SINC

E →TE'
E' →+TE' | ε
T →FT'
T' →*FT' | ε
F → (E) | id

1.

1. Se o analisador sintático procurar pela entrada M[A,*a*] e encontrar que a mesma está em **branco**, então o símbolo *a* é **ignorado**.
2. Se a entrada for SINC então o não-terminal ao topo da pilha é removido numa tentativa de se retomar a análise sintática.
3. Se um token ao topo da pilha não casar com o símbolo de entrada, então é removido da pilha.

Considerando a entrada errônea

+ id * + id

Pilha	Entrada	Ação	Comentário
E$	+ id * + id$		REGRA 1 Erro (+) pula
E$	id * + id$	E→TE '	id está em First(E)
TE '$	id * + id$	T→FT '	
FT ' E' $	id * + id$	F→id	
idT ' E' $	id * + id$	Casa id	
T ' E' $	* + id$	T'→*FT '	
*FT ' E' $	* + id$	Casa *	
FT ' E' $	+ id$		REGRA 2 SINC

Não Terminal	Símbo					
	Id	+	*	(	)	$
E	E::=TE'			E::=TE'	SINC	SINC
E'		E'::=+TE'			E'::=ℰ	E'::=ℰ
T	T::=FT'	SINC		T::=FT'	SINC	SINC
T'		T'::= ℰ	T'::=*FT'		T'::= ℰ	T'::= ℰ
F	F::=id	SINC	SINC	F::=(E)	SINC	SINC

E →TE'
E' →+TE' | ε
T →FT'
T' →*FT' | ε
F → (E) | id

1.

1. Se o analisador sintático procurar pela entrada M[A,*a*] e encontrar que a mesma está em **branco**, então o símbolo *a* é **ignorado**.
2. Se a entrada for SINC então o não-terminal ao topo da pilha é removido numa tentativa de se retomar a análise sintática.
3. Se um token ao topo da pilha não casar com o símbolo de entrada, então é removido da pilha.

Considerando a entrada errônea

+ id * + id

Pilha	Entrada	Ação	Comentário
E$	+ id * + id$		REGRA 1 Erro (+) pula
E$	id * + id$	E→TE '	id está em First(E)
TE '$	id * + id$	T→FT '	
FT ' E' $	id * + id$	F→id	
idT ' E' $	id * + id$	Casa id	
T ' E' $	* + id$	T'→*FT '	
*FT ' E' $	* + id$	Casa *	
FT ' E' $	+ id$		REGRA 2 SINC
T ' E' $	+ id$	T'→ε	F foi removido

Não Terminal	Símbo					
	Id	+	*	(	)	$
E	E::=TE'			E::=TE'	SINC	SINC
E'		E'::=+TE'			E'::=ε	E'::=ε
T	T::=FT'	SINC		T::=FT'	SINC	SINC
T'		T'::= ε	T'::=*FT'		T'::= ε	T'::= ε
F	F::=id	SINC	SINC	F::=(E)	SINC	SINC

E →TE'
E' →+TE' | ε
T →FT'
T' →*FT' | ε
F → (E) | id

1.

1. Se o analisador sintático procurar pela entrada M[A,*a*] e encontrar que a mesma está em **branco**, então o símbolo *a* é **ignorado**.
2. Se a entrada for SINC então o não-terminal ao topo da pilha é removido numa tentativa de se retomar a análise sintática.
3. Se um token ao topo da pilha não casar com o símbolo de entrada, então é removido da pilha.

Considerando a entrada errônea

+ id * + id

Pilha	Entrada	Ação	Comentário
E$	+ id * + id$		REGRA 1 Erro (+) pula
E$	id * + id$	E→TE '	id está em First(E)
TE '$	id * + id$	T→FT '	
FT ' E' $	id * + id$	F→id	
idT ' E' $	id * + id$	Casa id	
T ' E' $	* + id$	T'→*FT '	
*FT ' E' $	* + id$	Casa *	
FT ' E' $	+ id$		REGRA 2 SINC
T ' E' $	+ id$	T'→ε	F foi removido
E' $	+ id$	E→+TE '	

Não Terminal	Símbo					
	Id	+	*	(	)	$
E	E::=TE'			E::=TE'	SINC	SINC
E'		E'::=+TE'			E'::=ε	E'::=ε
T	T::=FT'	SINC		T::=FT'	SINC	SINC
T'		T'::= ε	T'::=*FT'		T'::= ε	T'::= ε
F	F::=id	SINC	SINC	F::=(E)	SINC	SINC

E →TE'
E' →+TE'|ε
T →FT'
T' →*FT'|ε
F → (E)|id

1.

1. Se o analisador sintático procurar pela entrada M[A,*a*] e encontrar que a mesma está em **branco**, então o símbolo *a* é **ignorado**.
2. Se a entrada for SINC então o não-terminal ao topo da pilha é removido numa tentativa de se retomar a análise sintática.
3. Se um token ao topo da pilha não casar com o símbolo de entrada, então é removido da pilha.

Considerando a entrada errônea

+ id * + id

Pilha	Entrada	Ação	Comentário
E$	+ id * + id$		REGRA 1 Erro (+) pula
E$	id * + id$	E→TE '	id está em First(E)
TE '$	id * + id$	T→FT '	
FT ' E' $	id * + id$	F→id	
idT ' E' $	id * + id$	Casa id	
T ' E' $	* + id$	T'→*FT '	
*FT ' E' $	* + id$	Casa *	
FT ' E' $	+ id$		REGRA 2 SINC
T ' E' $	+ id$	T'→ε	F foi removido
E' $	+ id$	E→+TE '	
+TE' $	+ id$	Casa +	

Não Terminal	Símbo					
	Id	+	*	(	)	$
E	E::=TE'			E::=TE'	SINC	SINC
E'		E'::=+TE'			E'::=ℰ	E'::=ℰ
T	T::=FT'	SINC		T::=FT'	SINC	SINC
T'		T'::= ℰ	T'::=*FT'		T'::= ℰ	T'::= ℰ
F	F::=id	SINC	SINC	F::=(E)	SINC	SINC

E →TE'
E' →+TE'|ε
T →FT'
T' →*FT'|ε
F → (E)|id

1.

1. Se o analisador sintático procurar pela entrada M[A,*a*] e encontrar que a mesma está em **branco**, então o símbolo *a* é **ignorado**.
2. Se a entrada for SINC então o não-terminal ao topo da pilha é removido numa tentativa de se retomar a análise sintática.
3. Se um token ao topo da pilha não casar com o símbolo de entrada, então é removido da pilha.

Considerando a entrada errônea

+ id * + id

Pilha	Entrada	Ação	Comentário
E$	+ id * + id$		REGRA 1 Erro (+) pula
E$	id * + id$	E→TE '	id está em First(E)
TE '$	id * + id$	T→FT '	
FT ' E' $	id * + id$	F→id	
idT ' E' $	id * + id$	Casa id	
T ' E' $	* + id$	T'→*FT '	
*FT ' E' $	* + id$	Casa *	
FT ' E' $	+ id$		REGRA 2 SINC
T ' E' $	+ id$	T'→ε	F foi removido
E' $	+ id$	E→+TE '	
+TE' $	+ id$	Casa +	
TE' $	id$	T→FT '	

Não Terminal	Símbo					
	Id	+	*	(	)	$
E	E::=TE'			E::=TE'	SINC	SINC
E'		E'::=+TE'			E'::=ℰ	E'::=ℰ
T	T::=FT'	SINC		T::=FT'	SINC	SINC
T'		T'::= ℰ	T'::=*FT'		T'::= ℰ	T'::= ℰ
F	F::=id	SINC	SINC	F::=(E)	SINC	SINC

E →TE'
E' →+TE'|ε
T →FT'
T' →*FT'|ε
F → (E)|id

1.

1. Se o analisador sintático procurar pela entrada M[A,*a*] e encontrar que a mesma está em **branco**, então o símbolo *a* é **ignorado**.
2. Se a entrada for SINC então o não-terminal ao topo da pilha é removido numa tentativa de se retomar a análise sintática.
3. Se um token ao topo da pilha não casar com o símbolo de entrada, então é removido da pilha.

Considerando a entrada errônea

+ id * + id

Pilha	Entrada	Ação	Comentário
E$	+ id * + id$		REGRA 1 Erro (+) pula
E$	id * + id$	E→TE '	id está em First(E)
TE '$	id * + id$	T→FT '	
FT ' E' $	id * + id$	F→id	
idT ' E' $	id * + id$	Casa id	
T ' E' $	* + id$	T'→*FT '	
*FT ' E' $	* + id$	Casa *	
FT ' E' $	+ id$		REGRA 2 SINC
T ' E' $	+ id$	T'→ε	F foi removido
E' $	+ id$	E→+TE '	
+TE' $	+ id$	Casa +	
TE' $	id$	T→FT '	
F T' E' $	id$	F→id	

Não Terminal	Símbo					
	Id	+	*	(	)	$
E	E::=TE'			E::=TE'	SINC	SINC
E'		E'::=+TE'			E'::=ℰ	E'::=ℰ
T	T::=FT'	SINC		T::=FT'	SINC	SINC
T'		T'::= ℰ	T'::=*FT'		T'::= ℰ	T'::= ℰ
F	F::=id	SINC	SINC	F::=(E)	SINC	SINC

E →TE'
E' →+TE'|ε
T →FT'
T' →*FT'|ε
F → (E)|id

1.

1. Se o analisador sintático procurar pela entrada M[A,*a*] e encontrar que a mesma está em **branco**, então o símbolo *a* é **ignorado**.
2. Se a entrada for SINC então o não-terminal ao topo da pilha é removido numa tentativa de se retomar a análise sintática.
3. Se um token ao topo da pilha não casar com o símbolo de entrada, então é removido da pilha.

Considerando a entrada errônea

+ id * + id

Pilha	Entrada	Ação	Comentário
E$	+ id * + id$		REGRA 1 Erro (+) pula
E$	id * + id$	E→TE '	id está em First(E)
TE '$	id * + id$	T→FT '	
FT ' E' $	id * + id$	F→id	
idT ' E' $	id * + id$	Casa id	
T ' E' $	* + id$	T'→*FT '	
*FT ' E' $	* + id$	Casa *	
FT ' E' $	+ id$		REGRA 2 SINC
T ' E' $	+ id$	T'→ε	F foi removido
E' $	+ id$	E→+TE '	
+TE' $	+ id$	Casa +	
TE' $	id$	T→FT '	
F T' E' $	id$	F→id	
idT' E' $	id$	Casa id	

Não Terminal	Símbo					
	Id	+	*	(	)	$
E	E::=TE'			E::=TE'	SINC	SINC
E'		E'::=+TE'			E'::=ℰ	E'::=ℰ
T	T::=FT'	SINC		T::=FT'	SINC	SINC
T'		T'::= ℰ	T'::=*FT'		T'::= ℰ	T'::= ℰ
F	F::=id	SINC	SINC	F::=(E)	SINC	SINC

E →TE'
E' →+TE'|ε
T →FT'
T' →*FT'|ε
F → (E)|id

1.

1. Se o analisador sintático procurar pela entrada M[A,*a*] e encontrar que a mesma está em **branco**, então o símbolo *a* é **ignorado**.
2. Se a entrada for SINC então o não-terminal ao topo da pilha é removido numa tentativa de se retomar a análise sintática.
3. Se um token ao topo da pilha não casar com o símbolo de entrada, então é removido da pilha.

Considerando a entrada errônea

+ id * + id

Pilha	Entrada	Ação	Comentário
E$	+ id * + id$		REGRA 1 Erro (+) pula
E$	id * + id$	E→TE '	id está em First(E)
TE '$	id * + id$	T→FT '	
FT ' E' $	id * + id$	F→id	
idT ' E' $	id * + id$	Casa id	
T ' E' $	* + id$	T'→*FT '	
*FT ' E' $	* + id$	Casa *	
FT ' E' $	+ id$		REGRA 2 SINC
T ' E' $	+ id$	T'→ε	F foi removido
E' $	+ id$	E→+TE '	
+TE' $	+ id$	Casa +	
TE' $	id$	T→FT '	
F T' E' $	id$	F→id	
idT' E' $	id$	Casa id	
T' E' $	$	T'→ε	

Não Terminal	Símbo					
	Id	+	*	(	)	$
E	E::=TE'			E::=TE'	SINC	SINC
E'		E'::=+TE'			E'::=ℰ	E'::=ℰ
T	T::=FT'	SINC		T::=FT'	SINC	SINC
T'		T'::= ℰ	T'::=*FT'		T'::= ℰ	T'::= ℰ
F	F::=id	SINC	SINC	F::=(E)	SINC	SINC

E →TE'
E' →+TE'|ε
T →FT'
T' →*FT'|ε
F → (E)|id

1.

1. Se o analisador sintático procurar pela entrada M[A,*a*] e encontrar que a mesma está em **branco**, então o símbolo *a* é **ignorado**.
2. Se a entrada for SINC então o não-terminal ao topo da pilha é removido numa tentativa de se retomar a análise sintática.
3. Se um token ao topo da pilha não casar com o símbolo de entrada, então é removido da pilha.

Considerando a entrada errônea

+ id * + id

Pilha	Entrada	Ação	Comentário
E$	+ id * + id$		REGRA 1 Erro (+) pula
E$	id * + id$	E→TE '	id está em First(E)
TE '$	id * + id$	T→FT '	
FT ' E' $	id * + id$	F→id	
idT ' E' $	id * + id$	Casa id	
T ' E' $	* + id$	T'→*FT '	
*FT ' E' $	* + id$	Casa *	
FT ' E' $	+ id$		REGRA 2 SINC
T ' E' $	+ id$	T'→ε	F foi removido
E' $	+ id$	E→+TE '	
+TE' $	+ id$	Casa +	
TE' $	id$	T→FT '	
F T' E' $	id$	F→id	
idT' E' $	id$	Casa id	
T' E' $	$	T'→ε	
E' $	$	E'→ε	

Não Terminal	Símbo					
	Id	+	*	(	)	$
E	E::=TE'			E::=TE'	SINC	SINC
E'		E'::=+TE'			E'::=ℰ	E'::=ℰ
T	T::=FT'	SINC		T::=FT'	SINC	SINC
T'		T'::= ℰ	T'::=*FT'		T'::= ℰ	T'::= ℰ
F	F::=id	SINC	SINC	F::=(E)	SINC	SINC

E →TE'
E' →+TE'|ε
T →FT'
T' →*FT'|ε
F → (E)|id

1.

1. Se o analisador sintático procurar pela entrada M[A,*a*] e encontrar que a mesma está em **branco**, então o símbolo *a* é **ignorado**.
2. Se a entrada for SINC então o não-terminal ao topo da pilha é removido numa tentativa de se retomar a análise sintática.
3. Se um token ao topo da pilha não casar com o símbolo de entrada, então é removido da pilha.

Considerando a entrada errônea

+ id * + id

Pilha	Entrada	Ação	Comentário
E$	+ id * + id$		REGRA 1 Erro (+) pula
E$	id * + id$	E→TE'	id está em First(E)
TE'$	id * + id$	T→FT'	
FT'E'$	id * + id$	F→id	
idT'E'$	id * + id$	Casa id	
T'E'$	* + id$	T'→*FT'	
*FT'E'$	* + id$	Casa *	
FT'E'$	+ id$		REGRA 2 SINC
T'E'$	+ id$	T'→ε	F foi removido
E'$	+ id$	E→+TE'	
+TE'$	+ id$	Casa +	
TE'$	id$	T→FT'	
FT'E'$	id$	F→id	
idT'E'$	id$	Casa id	
T'E'$	$	T'→ε	
E'$	$	E'→ε	
$	$		FIM

Não Terminal	Símbo					
	Id	+	*	(	)	$
E	E::=TE'			E::=TE'	SINC	SINC
E'		E'::=+TE'			E'::=ε	E'::=ε
T	T::=FT'	SINC		T::=FT'	SINC	SINC
T'		T'::= ε	T'::=*FT'		T'::= ε	T'::= ε
F	F::=id	SINC	SINC	F::=(E)	SINC	SINC

E →TE'
E' →+TE' | ε
T →FT'
T' →*FT' | ε
F → (E) | id

1.

Exercício

1. Como prosseguir esta análise?

Pilha	Entrada	Regra
Sε	ababbε	S→AB
ABε	ababbε	A→aA
aABε	ababbε	Casa a
ABε	babbε	A→ε
Bε	babbε	B → bB
bBε	babbε	Casa b
Bε	abbε	**ERRO**

Não Terminal	Símbolo de Entrada		
	a	b	ε
S	S→AB 1.	S→AB 1.	S → AB 1.
A	A → aA 1.	A → ε 2.	A → ε 2.
B	**ERRO** 3.	B → bB	B → ε 2.

1. Se o analisador sintático procurar pela entrada M[A,*a*] e encontrar que a mesma está em **branco**, então o símbolo *a* é **ignorado**.

Exercício

1. Como prosseguir esta análise?

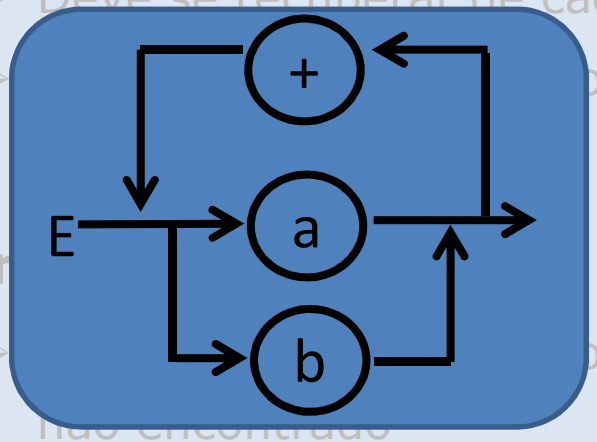
Pilha	Entrada	Regra	Comentário
Sε	ababbε	S→AB	
ABε	ababbε	A→aA	
aABε	ababbε	Casa a	
ABε	babbε	A→ε	
Bε	babbε	B → bB	
bBε	babbε	Casa b	
Bε	abbε	branco	Ignora a
Bε	bbε	B → bB	
bBε	bbε	Casa b	
Bε	bε	B → bB	
bBε	bε	Casa b	
Bε	ε	B→ε	
ε	ε	FIM	

Não Terminal	Símbolo de Entrada		
	a	b	ε
S	S→AB 1.	S→AB 1.	S → AB 1.
A	A → aA 1.	A → ε 2.	A → ε 2.
B	ERRO 3.	B → bB	B → ε 2.

1. Se o analisador sintático procurar pela entrada M[A,*a*] e encontrar que a mesma está em **branco**, então o símbolo *a* é **ignorado**.

Analisador sintático descendente recursivo
tratamento de erros sintáticos

- **Funções**
  - Deve relatar a presença de
  - Deve se recuperar de cada
  -
- **Er**
  - om do não encontrado




```
procedimento E
begin
  se (símbolo='a') ou (símbolo='b') então
      obter_próximo()
  senão ERRO;
      enquanto (símbolo='+') faça
          obter_próximo();
          se (símbolo='a') ou (símbolo='b') então
                obter_próximo()
          senão ERRO;
end
```

- A realização efetiva do tratamento de erros pode ser uma tarefa difícil

Analisador sintático descendente recursivo

tratamento de erros sintáticos

- **Funções**
  - Deve relatar a presença de erros de forma clara e precisa
  - Deve se recuperar de cada erro para continuar a análise do programa
  - Pode reparar alguns erros

- **Erro sintático**
  - Programa não condiz com a gramática da linguagem: símbolo esperado não encontrado

- A realização efetiva do tratamento de erros pode ser uma tarefa difícil

ASD recursivo
tratamento de erros sintáticos

- Dados referentes a erros sintáticos
  - 80% dos enunciados contém apenas um erro; 13% tem dois
  - 90% são erros em um único *token*
  - 60% são erros de pontuação: por exemplo, uso do ponto e vírgula (;)
  - 20% são erros de operadores e operandos: por exemplo, omissão de : no símbolo :=
  - 15% são erros de palavras-chave: por exemplo, erros ortográficos (wrteln)

Analisador sintático descendente
tratamento de erros sintáticos

- O tratamento inadequado de erros pode introduzir novos erros, que não foram cometidos pelo programador, mas pelo tratamento de erros realizado

- Tratamento de erros deve ser cauteloso e selecionar os tipos de erros que podem ser tratados para se obter um processamento eficiente

Analisador sintático descendente
tratamento de erros sintáticos

- Muitas técnicas para o tratamento de erros
  - Nenhuma delas se mostrou universalmente aceitável
  - Poucos métodos têm ampla aplicabilidade
  - Muitas vezes o processo é artesanal

- Estratégias de tratamento de erro
  - Modalidade do desespero → mais utilizado
  - Recuperação de frases
  - Produções de erro
  - Correção global

Analisador sintático descendente
tratamento de erros sintáticos

- Modalidade do desespero
  - Método mais simples e fácil de implementar → usado pela maioria dos analisadores sintáticos
  - Ao encontrar um erro
    1. Relata-se o erro
    2. Pulam-se *tokens* até que um *token* de sincronização seja encontrado

- *Tokens* de sincronização: pontos do programa (palavras-chave, delimitadores, etc.) em que é possível se retomar a análise sintática com certa segurança. Esses *tokens* precisam ser determinados pelo projetista do compilador

Analisador sintático descendente
tratamento de erros sintáticos

- Exemplo

```
while (x<2 do
readln(y)...
```

- Ao notar a falta do fecha parênteses, relata-se a falta do mesmo e se consome tudo até que o *token* ***readln*** seja encontrado, a partir de onde se recomeça a análise

Analisador sintático descendente
tratamento de erros – modo de pânico

- Modalidade do desespero: pulam-se *tokens* até que se encontre um a partir do qual se possa retomar a análise

```
program P;
var x: integer;
begin
...
end.
```

<PROGRAMA> ::= program <ID> ; <CORPO> .
<CORPO> ::= <DC_V> begin <COMANDOS> end
<DC_V> ::= **var** <VARIAVEIS> : <TIPO_VAR> ; <DC_V> | ε
<TIPO_VAR> ::= integer
<VARIAVEIS> ::= <ID> <MAIS_VAR>
...
<ID> ::= identificador

- Diante da ausência/erro de **var**, de onde recomeçar a análise? Quem é esse símbolo de recomeço?

Analisador sintático descendente
tratamento de erros – modo de pânico

- Modalidade do desespero: pulam-se *tokens* até que se encontre um a partir do qual se possa retomar a análise

```
program P;
var x: integer;
begin
...
end.
```

<PROGRAMA> ::= program <ID> ; <CORPO> .
<CORPO> ::= <DC_V> begin <COMANDOS> end
<DC_V> ::= **var** <VARIAVEIS> : <TIPO_VAR> ; <DC_V> | ε
<TIPO_VAR> ::= integer
<VARIAVEIS> ::= **<ID>** <MAIS_VAR>
...
<ID> ::= **identificador**

- Diante da ausência/erro de **var**, de onde recomeçar a análise? Quem é esse símbolo de recomeço?

identificador, ***FOLLOW*** do terminal **var**

Analisador sintático descendente
tratamento de erros – modo de pânico

- Modalidade do desespero: pulam-se *tokens* até que se encontre um a partir do qual se possa retomar a análise

```
program P;
var x: integer;
begin
...
end.
```

<PROGRAMA> ::= program <ID> ; <CORPO> .
<CORPO> ::= <DC_V> begin <COMANDOS> end
<DC_V> ::= **var** <VARIAVEIS> : <TIPO_VAR> ; **<DC_V>** | ε
<TIPO_VAR> ::= integer
<VARIAVEIS> ::= **<ID>** <MAIS_VAR>
...
<ID> ::= **identificador**

- Diante da ausência/erro de um trecho grande de código, de onde recomeçar a análise? Quem é esse símbolo de recomeço?

Símbolo de **begin**, ***FOLLOW*** do não terminal de **<DC_V>**

Analisador sintático descendente
tratamento de erros – modo de pânico

- Modalidade do desespero: pulam-se *tokens* até que se encontre um a partir do qual se possa retomar a análise

```
program P;
...
begin
readln(x);
writeln(x+2);
...
end.
```

<comandos> ::= <cmd> ; <comandos> |ε
<cmd> ::= readln(<variaveis>) |
writeln (<variaveis>) |
if <condicao> then <cmd> <pfalsa> |
<id> := <expressao>

- Diante da ausência/erro de um ponto-e-vírgula (;), de onde recomeçar a análise? Quem é esse símbolo de recomeço?

Analisador sintático descendente
tratamento de erros – modo de pânico

- Modalidade do desespero: pulam-se *tokens* até que se encontre um a partir do qual se possa retomar a análise

```
program P;
...
begin
readln(x);
writeln(x+2);
...
end.
```

<comandos> ::= <cmd> ; <comandos> |ε
<cmd> ::= readln(<variaveis>) |
writeln (<variaveis>) |
if <condicao> then <cmd> <pfalsa> |
<id> := <expressao>

- Diante da ausência/erro de um ponto-e-vírgula (;), de onde recomeçar a análise? Quem é esse símbolo de recomeço?

Símbolo de **writeln**, ***FIRST*** do não terminal **<cmd>**

Analisador sintático descendente
tratamento de erros – modo de pânico

- Modalidade do desespero: pulam-se *tokens* até que se encontre um a partir do qual se possa retomar a análise
- Símbolos de sincronização para um símbolo qualquer A sendo consumido
  - Inicialmente, utilizam-se os ***FOLLOWS***(A)
  - Acrescentam-se os símbolos ***FOLLOWS*** do pai de A, isto é, de quem o acionou
    - ✓ Útil quando tudo dentro de A deu errado
  - Símbolos de sincronização extras, para garantir que muito da entrada não seja consumido
    - ✓ Determinados pelo projetista do compilador

Analisador sintático descendente
tratamento de erros – modo de pânico

- Exemplo

```
<comandos> ::= <cmd> ; <comandos> |ε
<cmd> ::=  readln( <variaveis> ) |
           writeln ( <variaveis> ) |
           while (<condição>) do <cmd> |
           if ...
```

- Se acontecer um erro dentro de **<condição>**, buscam-se seus ***FOLLOWS*** para se retomar a análise: **)**, por exemplo

Analisador sintático descendente
tratamento de erros – modo de pânico

- Exemplo

```
<comandos> ::= <cmd> ; <comandos> |ε
<cmd> ::=  readln( <variaveis> ) |
           writeln ( <variaveis> ) |
           while (<condicao>) do <cmd> |
           if ...
```

- Se acontecer um erro dentro de **<condição>**, e o programador omitiu os ***FOLLOWS*** deste, buscam-se os ***FOLLOWS*** do pai para continuar a análise: ponto e vírgula (**;**), por exemplo

Analisador sintático descendente
tratamento de erros – modo de pânico

- Exemplo

```
<comandos> ::= <cmd> ; <comandos> |ε
<cmd> ::=  readln( <variaveis> ) |
           writeln ( <variaveis> ) |
           while (<condicao>) do <cmd>  |
           if ...
```

- Ao se buscar pelos ***FOLLOWS***, pode-se perder a análise de grande parte do programa nesse caso (a análise de **<cmd>**, por exemplo). Em pontos críticos do programa, buscam-se, portanto, *tokens* a partir de onde seja seguro retomar a análise: **readln**, **writeln** e **if**, por exemplo

Analisador sintático descendente
tratamento de erros

- Nos analisadores sintáticos descendentes recursivos, a busca por *tokens* de sincronização é feita sempre que um *token* esperado não é encontrado – no ponto de ERRO dos procedimentos recursivos

- Nos analisadores sintáticos não-recursivos: quando não é possível seguir em frente com a análise, por falta de produção na tabela sintática ou por incompatibilidade de terminais na pilha e na cadeia de entrada

ASD recursivo
algoritmo do procedimento ERRO

```
procedimento ERRO(num_erro, conjunto_simb_sincr)
begin
  imprimir mensagem de erro relativa ao erro de número num_erro;
  enquanto (token_corrente não pertence a conjunto_simb_sincr)
    faça  obter_símbolo(); //retorna um símbolo para ser consumido
end;
```

- **Observações**
  - Tomar o cuidado de verificar quando se chegou ao fim do programa-fonte
  - Opcionalmente, a mensagem de erro pode ser impressa antes da chamada do procedimento ERRO
  - Verificar quem é o ***FOLLOW*** encontrado

ASD recursivo
algoritmo do procedimento ERRO

❑ Exemplo: **<programa> ::= program id ; <bloco> .**

```
procedimento programa(S)
begin
      se (simb=program) então obter_símbolo() //ALéxico
       senão
            imprimir("Erro: program esperado");
            ERRO({id}+S); //consome todos até encontrar id
       se (simb=id) então obter_símbolo()
       senão
            imprimir("Erro: identificador de programa esperado");
            ERRO({;}+S); //consome todos até encontrar ;
       se (simb=simb_pv) então obter_símbolo()
       senão
            imprimir("Erro: ponto e vírgula esperado");
            ERRO(First(bloco)+S); //consome todos até encontrar um sincronizador
       bloco();
       se (simb=ponto) então obter_símbolo()
       senão imprimir("Erro: ponto esperado");
end;
```

ID, Seguidor de program

Conjunto sincronizador {var|procedure|id|while|repeat|if|begin}

```
<programa> ::= program <id> ; <bloco> .
<bloco> ::= <dc_v> begin <comandos> end
<dc_v> ::= var <variaveis> : <tipo_var> ; <dc_v> | ε
```

Analisador sintático descendente
algoritmo do procedimento ERRO

Exemplo: **<bloco> ::= <dc> begin <comandos> end**

```
procedimento corpo(S)
begin
    dc();
    se (simb=begin) então obter_símbolo()
    senão
        imprimir("Erro: begin esperado");
        ERRO(First(comandos)+S);
    comandos();
    se (simb=end) então obter_símbolo()
    senão
        imprimir("Erro: end esperado");
        ERRO(S);
end;
```

Analisador sintático descendente
algoritmo do procedimento ERRO

- Exercício: faça o procedimento recursivo com tratamento de erro para o não terminal <dc_v>.

<dc_v> ::= var <variaveis> : <tipo_var> ; <dc_v>|ε

Continuação da gramática...

<TIPO_VAR> ::= integer | real

<VARIAVEIS> ::= <ID> <MAIS_VAR>

<MAIS_VAR> ::= , <VARIAVEIS> | ε

Analisador sintático descendente

<dc_v> ::= var <variaveis> : <tipo_var> ; <dc_v>|ε

```
procedimento dc_v(S)
begin
    se (simb=var) então obter_símbolo()
    senão
        imprimir("Erro: var esperado");
        ERRO(Primeiro(variaveis)+S); //consome até encontrar ID
    variaveis();
    se (simb=simb_dp) então obter_símbolo()
    senão
        imprimir("Erro: ':' esperado");
        ERRO(Primeiro(tipo_var)+S); //consome até encontrar integer ou real
    tipo_var();
    se (simb=simb_pv) então obter_símbolo()
    senão
        imprimir("Erro: ';' esperado");
        ERRO(Primeiro(dc_v)+S); //consome até encontrar ;
    dc_v();
end;
```

```
<DC_V> ::= var <VARIAVEIS> : <TIPO_VAR> ; <DC_V> | ε
<TIPO_VAR> ::= integer | real
<VARIAVEIS> ::= <ID> <MAIS_VAR>
<MAIS_VAR> ::= , <VARIAVEIS> | ε
```

Analisador sintático descendente

- Exercício:

1. Faça os procedimentos para o restante da gramática.
2. Implemente o analisador sintático: recursivo ou não recursivo – demais informações na página