

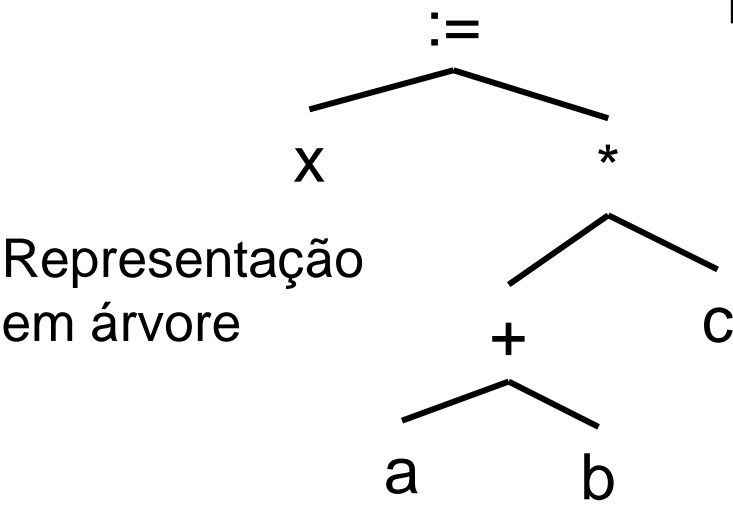
# Cap. 5 Tradução Dirigida pela Sintaxe

Tradução de um programa fonte em um programa objeto é feita em 2 passos e uma representação intermediária é construída

Essa representação intermediária é tipicamente encontrada em duas formas alternativas:

- Árvore sintática
- ou Seqüência de comandos (código de três endereços)

Ex.  $x := (a + b) * c$

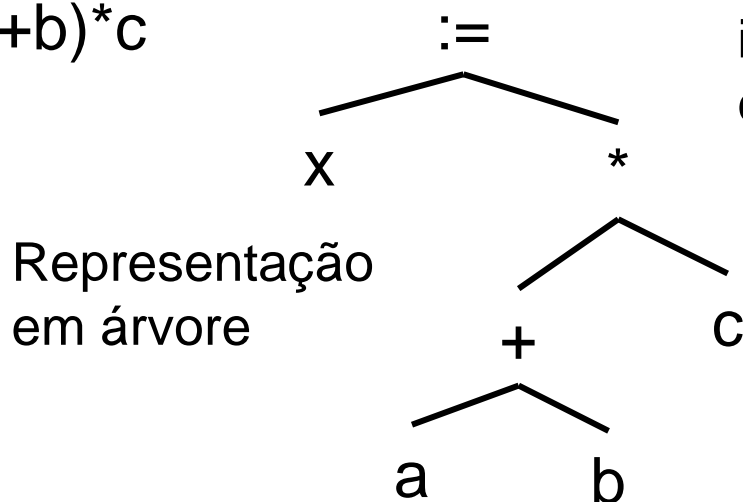


Representação em seqüência de instruções

+	a	b	t1
*	t1	c	t2
:=	x	t1	...

# Cap. 5 Tradução Dirigida pela Sintaxe

Ex.  $x := (a+b)*c$



Representação em seqüência de instruções, com código de 3 endereços

+	a	b	t1
*	t1	c	t2
:=	x	t1	...

**Para os dois casos de representação  
A transformação em código de máquina é SIMPLES.**

**Iremos abordar inicialmente a representação por código de 3 endereços, chamado de quádruplas**



# Cap. 5 Tradução Dirigida pela Sintaxe

Iremos abordar inicialmente a representação por código de 3 endereços, chamado de quádruplas, seguindo um exemplo de implementação.

Quádruplas (código de 3 endereços)

Quádruplas são geradas em Tabela ou Arquivo

Cada Entrada na Tabela contém 4 Campos para representar uma quádrupla:

operador	operando1	operando2	operando3
----------	-----------	-----------	-----------

0 : [ ADD 4 5 50 ]
4 : [ ADD 1 2 51 ]
5 : [ STO 51 0 ]

Quem  
preenche  
Quádrupla  
é **gera** ou  
**remenda**

Sintaxe	Ações semânticas com uso de \$ para atributos (Yacc-like)
0. $S' \rightarrow S$	{encerra();}
1. $S \rightarrow V = E$	{gera(STO, \$3.indiceSimb,\$1.indiceSimb,NADA)}
2. $S \rightarrow I ( E )$	{tabSimb[\$1.indiceSimb].entidade = Procedimento; gera (PARAM,\$3.indiceSimb,NADA,NADA); gera(CALL, \$1.indiceSimb, NADA, NADA)}
3. $E \rightarrow E + V$	{\$\$ .indiceSimb = temp(); gera ( ADD, \$1.indiceSimb, \$3.indiceSimb, \$\$ .indiceSimb ) }
$\rightarrow V$	{\$\$ .indiceSimb = \$1.indiceSimb;}
$\rightarrow V = I ( E )$	{tabSimb[\$3.indiceSimb].entidade = Funcao; gera (PARAM,\$5.indiceSimb,NADA,NADA); gera(CALL, \$3.indiceSimb, NADA, NADA); gera(POP, \$1.indiceSimb, NADA, NADA) ; \$\$ .indiceSimb=\$1.indiceSimb;}
$\rightarrow I$	{\$\$ .indiceSimb = \$1.indiceSimb;}
$\rightarrow a$	{\$\$ .indiceSimb = incluiSimbTab (\$1, Variavel)}

- 0 : [ ADD 4 5 50 ]
- 1 : [ PARAM 50 ]
- 2 : [ CALL 3 ]
- 3 : [ POP 2 ]
- 4 : [ ADD 1 2 51 ]
- 5 : [ STO 51 0 ]

{+,...,12}{V,2,16}{=,....,14}{l,3,18}{(,....,13}{E,4,17}{+,...,12}{V,5,16}	0 : [ ADD 4 5 50 ]
{+,...,12}{V,2,16}{=,....,14}{l,3,18}{(,....,13}{E,50,17}	
{+,...,12}{V,2,16}{=,....,14}{l,3,18}{(,....,13}{E,50,17}{),....,19}	
{+,...,12}{V,2,16}{=,....,5}{E,1,7}{+,...,12}{V,2,16}	1 : [ PARAM 50 ] 2 : [ CALL 3 ] 3 : [ POP 2 ]
{+,...,12}{V,2,16}{=,....,5}{E,51,7}	4 : [ ADD 1 2 51 ]
{+,...,12}{S,....,1}	5 : [ STO 51 0 ]

Sintaxe	Ações semânticas com uso de \$ para atributos (Yacc-like)
0. $S' \rightarrow S$	{encerra();}
1. $S \rightarrow V = E$	{gera(STO, \$3.indiceSimb,\$1.indiceSimb,NADA)}
2. $S \rightarrow I ( E )$	{tabSimb[\$1.indiceSimb].entidade = Procedimento; gera (PARAM,\$3.indiceSimb,NADA,NADA); gera(CALL, \$1.indiceSimb, NADA, NADA)}
3. $E \rightarrow E + V$	{\$\$ .indiceSimb = temp(); gera ( ADD, \$1.indiceSimb, \$3.indiceSimb, \$\$ .indiceSimb ) }
4. $E \rightarrow V$	{\$\$ .indiceSimb = \$1.indiceSimb;}
5. $V \rightarrow V = I ( E )$	{tabSimb[\$3.indiceSimb].entidade = Funcao; gera (PARAM,\$5.indiceSimb,NADA,NADA); gera(CALL, \$3.indiceSimb, NADA, NADA); gera(POP, \$1.indiceSimb, NADA, NADA) ; \$\$ .indiceSimb=\$1.indiceSimb;}
6. $V \rightarrow I$	{\$\$ .indiceSimb = \$1.indiceSimb;}
7. $I \rightarrow a$	{\$\$ .indiceSimb = incluiSimbTab (\$1, Variavel)}

{,,,,,0}{V,0,2}{=,,,,,5}{E,1,7}{+,,,,,,12}{V,2,16}{=,,,,,14}{I,3,18}{(,,,,,13){E,4,17}{+,,,,,,12}{V,5,16}	
{,,,,,0}{V,0,2}{=,,,,,5}{E,1,7}{+,,,,,,12}{V,2,16}{=,,,,,14}{I,3,18}{(,,,,,13){E,50,17}	0 : [ ADD 4 5 50 ]
{,,,,,0}{V,0,2}{=,,,,,5}{E,1,7}{+,,,,,,12}{V,2,16}{=,,,,,14}{I,3,18}{(,,,,,13){E,50,17}{,,,,,19}	
{,,,,,0}{V,0,2}{=,,,,,5}{E,1,7}{+,,,,,,12}{V,2,16}	1 : [ PARAM 50 ] 2 : [ CALL 3 ] 3 : [ POP 2 ]
{,,,,,0}{V,0,2}{=,,,,,5}{E,51,7}	4 : [ ADD 1 2 51 ]
{,,,,,0}{S,,,,,1}	5 : [ STO 51 0 ]

## Cap. 5 Tradução Dirigida pela Sintaxe

Entrada na tabela para representar 1 quádrupla:

operador	operando1	operando2	operando3
----------	-----------	-----------	-----------

Para criar uma entrada na tabela, pode ser usada uma rotina chamada **gera** (**op**, **op1**, **op2**, **op3**)

Uma variável global chamada **prox** é incrementada a cada chamada **gera**, e sempre aponta para a próxima entrada (quádrupla) a ser gerada.

Referência a uma entrada (instrução) pode ser pela posição na tabela

```

void gera (
    Operador codop,int end1,int end2,int end3) {
    quadrupla [prox].op = codop;
    quadrupla [prox].operando1 = end1;
    quadrupla [prox].operando2 = end2;
    quadrupla [prox].operando3 = end3;
    prox++;
}

```

	gera(CALL, \$1.indiceSimb, NADA, NADA)}
3. $E \rightarrow E + V$	{ $$$.$ indiceSimb = temp(); gera ( ADD, \$1.indiceSimb, \$3.indiceSimb, $$$.$ indiceSimb ) } 
$A \rightarrow A + V$	$$$$.$ indiceSimb = \$1.indiceSimb



## Cap. 5 Tradução Dirigida pela Sintaxe

**Precisamos de variáveis temporárias para guardar valores intermediários no cálculo de expressões**

**uma função temporária `temp()` devolve o endereço da nova temporária criada na tabela de símbolos.**

**Atenção ! variáveis temporárias têm o mesmo status de variáveis definidas pelo programador.**

**Elas têm tipo, valor, endereço de memória.**

Sintaxe	Ações semânticas com uso de \$ para atributos (Yacc-like)
0. $S' \rightarrow S$	{encerra();}
1. $S \rightarrow V = E$	{gera(STO, \$3.indiceSimb,\$1.indiceSimb,NADA)}
2. $S \rightarrow I ( E )$	{tabSimb[\$1.indiceSimb].entidade = Procedimento; gera (PARAM,\$3.indiceSimb,NADA,NADA); gera(CALL, \$1.indiceSimb, NADA, NADA)}
3. $E \rightarrow E + V$	{\$\$ .indiceSimb = temp(); gera ( ADD, \$1.indiceSimb, \$3.indiceSimb, \$\$ .indiceSimb ) }
4. $E \rightarrow V$	{\$\$ .indiceSimb = \$1.indiceSimb;}
5. $V \rightarrow V = I ( E )$	{tabSimb[\$3.indiceSimb].entidade = Funcao; gera (PARAM,\$5.indiceSimb,NADA,NADA); gera(CALL, \$3.indiceSimb, NADA, NADA); gera(POP, \$1.indiceSimb, NADA, NADA) ; \$\$ .indiceSimb=\$1.indiceSimb;}
6. $V \rightarrow I$	{\$\$ .indiceSimb = \$1.indiceSimb;}
7. $I \rightarrow a$	{\$\$ .indiceSimb = incluiSimbTab (\$1, Variavel)}

{,,,,,0}{V,0,2}{=,,,,,5}{E,1,7}{+,,,,,,12}{V,2,16}{=,,,,,14}{I,3,18}{(,,,,,13){E,4,17}{+,,,,,,12}{V,5,16}	
{,,,,,0}{V,0,2}{=,,,,,5}{E,1,7}{+,,,,,,12}{V,2,16}{=,,,,,14}{I,3,18}{(,,,,,13){E,50,17}	0 : [ ADD 4 5 50 ]
{,,,,,0}{V,0,2}{=,,,,,5}{E,1,7}{+,,,,,,12}{V,2,16}{=,,,,,14}{I,3,18}{(,,,,,13){E,50,17}{),,,,,,19}	
{,,,,,0}{V,0,2}{=,,,,,5}{E,1,7}{+,,,,,,12}{V,2,16}	1 : [ PARAM 50 ] 2 : [ CALL 3 ] 3 : [ POP 2 ]
{,,,,,0}{V,0,2}{=,,,,,5}{E,51,7}	4 : [ ADD 1 2 51 ]
{,,,,,0}{S,,,,,1}	5 : [ STO 51 0 ]

3.  $E \rightarrow E + V$

{ $$$.$ indiceSimb = temp();

gera ( ADD, \$1.indiceSimb, \$3.indiceSimb,  $$$.$ indiceSimb ) }

{,,,,,0}{V,0,2}{=,,,,,5}{E,1,7}{+,,,,,,12}{V,2,16}{=,,,,,14}{I,3,18}{(,,,,,13){E,4,17}{+,,,,,,12}{V,5,16}	
{,,,,,0}{V,0,2}{=,,,,,5}{E,1,7}{+,,,,,,12}{V,2,16}{=,,,,,14}{I,3,18}{(,,,,,13){E,50,17}	0 : [ ADD 4 5 50 ]
{,,,,,0}{V,0,2}{=,,,,,5}{E,1,7}{+,,,,,,12}{V,2,16}{=,,,,,14}{I,3,18}{(,,,,,13){E,50,17}{),,,,,,19}	
{,,,,,0}{V,0,2}{=,,,,,5}{E,1,7}{+,,,,,,12}{V,2,16}	1 : [ PARAM 50 ] 2 : [ CALL 3 ] 3 : [ POP 2 ]
{,,,,,0}{V,0,2}{=,,,,,5}{E,51,7}	4 : [ ADD 1 2 51 ]
{,,,,,0}{S,,,,,1}	5 : [ STO 51 0 ]

Tabela de Símbolos – observar que os atributos na pilha serão preenchidos com o índice da tabela

Índice	nome	entidade	Valor da Variável ou Endereço de Chamada para Procedimento e Função
0	x	Variavel	
1	y	Variavel	
2	z	Variavel	
3	f	Funcao	
4	r	Variavel	
5	t	Variavel	
...			
50	t1	Temporario	
51	t2	Temporario	

## Cap. 5 Tradução Dirigida pela Sintaxe

### Exemplos de quádruplas (instruções) , numa geração de código intermediário

+	a	b	c
---	---	---	---

`c = a + b`

*	a	b	c
---	---	---	---

`c = a * b`

:=	a	b	_
----	---	---	---

`b := a`

JF	a	b	_
----	---	---	---

`se a é falso, jump para b`

J	b	_	_
---	---	---	---

`jump para b`

# ATRIBUIÇÃO DE ENDEREÇOS EFETIVOS ÀS VARIÁVEIS USADAS NO PROGRAMA E TEMPORÁRIAS

Variáveis declaradas em um bloco, ganham espaço apenas na ativação do bloco

O endereço da variável se torna uma fórmula que leva em conta o endereço inicial do espaço de alocação e seu deslocamento

Essas decisões são tomadas na geração de código de máquina, assim, no código intermediário, variáveis de programa e temporárias são referenciadas da tabela de símbolo.

- Endereço de id pode ser obtido pela função `simb (id)`
- Endereço de temporária pode ser obtido por `temp ()`

Nos exemplos, a função `simb` foi substituída pela `incluirSimbTab`  
Essa função de entrada na tabela de símbolos deve devolver a posição na tabela. Ela serve tanto para busca como para inclusão.

Tabela de Símbolos – observar que os atributos na pilha serão preenchidos com o índice da tabela

Índice	nome ou valor	entidade: Constante ou Variavel	Valor da Variável ou Constante
0	3	Constante	
1	x	Variavel	
2	4	Constante	
3	y	Variavel	
4	0	Constante	
5			
...			

## Cap. 5 Tradução Dirigida pela Sintaxe

Nos exemplos, a função **simb** foi substituída pela **incluirSimbTab**

```
void incluirSimbTab(YSTYPE simb, Entidade vOUc) {
    int retorno;
    retorno = buscaSimbTab (simb, vOUc, Alocar);
    if (retorno == FRACASSO ) { //simb deve ser incluído
        if (vOUc == Constante)
            tabSimb [topTab].valor = simb.i; //vOUc eh Constante
        else //variável, função, procedimento
            strcpy (tabSimb [topTab].nome, simb.cadeia);
        tabSimb [topTab].entidade = vOUc;
        retorno = topTab;
        topTab++;
    } // de FRACASSO, incluído novo simbolo
    return (retorno)
```

## Cap. 5 Tradução Dirigida pela Sintaxe

Implementação da função **temp** , que aloca uma entrada na tabela de símbolo para a variável temporária

```
int temp () {  
    char [4] nomeTemporaria;  
    strcpy(nomeTemporaria, "t");//prefixo  
    strcat(nomeTemporaria, itoa (topTemp-50));// sufixo  
    strcpy(tabSimb [topTemp].nome, nomeTemporaria);  
    tabSimb [topTemp].entidade = Temporario;  
    topTemp++;  
    return (topTemp-1);  
};
```



## Cap. 5 Tradução Dirigida pela Sintaxe

**Atributos sintetizados e herdados,**  
durante a **compilação**



Hein ?

cada **símbolo da gramática** pode estar **representado** por  
uma **estrutura** com **campos** para **manter informações**

**Uma dessas informações será um atributo da entidade** do  
programa **que** corresponde ao **símbolo da gramática,**  
**que pode estar** representando:

uma **cadeia** ou

um **tipo** ou

uma **localização de memória** em que se encontra

- um **valor** referenciado por uma **variável/cte**
- uma **instrução**
- um **procedimento** ou **função**

## **Atributos sintetizados e herdados,** durante a **compilação**

Um **valor** para um **atributo**  
está em um **nó da árvore gramatical**  
e é definido por uma **regra semântica**  
associada à **regra gramatical** usada naquele **nó**

O **valor** de um **atributo Sintetizado** é computado dos  
**valores dos atributos dos filhos** daquele **nó da árvore**  
**gramatical**

O **valor** de um **atributo Herdado** é computado a partir de  
**atributos dos irmãos e pai** daquele **nó da árvore**  
**gramatical**

## Cap. 5 Tradução Dirigida pela Sintaxe

**Cálculo de atributos sintetizados, supondo a regra  $A \rightarrow X1 X2 \dots Xm$  e um método ascendente de análise sintática, tipo laLR(1), que é o usado em ferramenta Yacc**

**Na redução, é necessário o seguinte tratamento com os atributos dos símbolos da regra:**

**Na hora de dar o pop do símbolo  $X_i$ , guardar o atributo da unidade removida da pilha**

**Quando todos os símbolos do lado direito tiverem sido desempilhados, os seus atributos estarão acessíveis, assim, o atributo para o símbolo  $A$  do lado esquerdo poderá ser calculado**

**Atributos comuns associados aos símbolos são os seguintes: endereço, nome, valor, tamanho, localização**

**As regras semânticas estabelecem dependência entre os atributos**

**Na avaliação de cada regra semântica é calculado o valor do atributo do símbolo do lado esquerdo, durante a análise de uma dada cadeia**

```
exp:NUM {$$ = $1;}  
    |exp '+' exp {$$ = $1 + $3;}  
    |exp '-' exp {$$ = $1 - $3;}  
    |exp '*' exp {$$ = $1 * $3;}  
    |exp '/' exp {$$ = $1 / $3;}  
    |'-' exp %prec NEG {$$ = -$2;}  
    |exp '^' exp {$$ = pow($1,$3);}  
    |'(' exp ')' {$$ = $2;}  
;  
%%
```

Cada **regra de produção** pode estar associada a um trecho de **código (ações)**, que pode ser **executado** quando a **regra for reduzida**.

Atenção ! Algumas **ações** se referem à **própria compilação** e outras à **geração de código**.

Não existe um limite imposto do que esse código deve fazer e esse código é compilado juntamente com o analisador

Ele pode imprimir mensagem, para a **atividade de análise** ou tratar as **estruturas de dados do compilador**

As **operações** que executam em combinação com as **regras** são chamadas de **ações semânticas**, porque elas agem **além da sintaxe**, sendo muitas relacionadas ao **significado do programa**

Quando uma **ação semântica** é efetuada para a **regra**  $A \rightarrow X_1 \dots X_n$ , os **valores semânticos associados** a cada **símbolo**  $X_i$  da **regra** podem ser acessados

Num analisador de baixo para cima, os **valores semânticos** de  $X_1 \dots X_n$  são disponíveis no momento em que a **regra de produção** está para ser **reduzida a A** e **as ações semânticas** preparam um **valor** para **A**

Num analisador de cima para baixo, um **valor** para **A** é disponível quando a **regra de produção** é **expandida** e os **valores** dos **atributos** de  $X_1 \dots X_n$  vão sendo **preenchidos** à medida que vão retornando no **caminho da árvore**

Quando uma **ação semântica** é efetuada para a **regra**

$A \rightarrow X_1 \dots X_n$ , os **valores semânticos associados** a cada **símbolo  $X_i$**  da **regra** podem ser acessados

Num analisador de baixo para cima, os **valores semânticos** de  $X_1 \dots X_n$  são disponíveis no momento em que a **regra de produção** está para ser **reduzida** a **A** e **as ações semânticas** preparam um **valor** para **A**

```
/* Grammar follows */
%%
input: /* empty string */
      |input line
;

line:  '\n'
|exp '\n'{printf("\t%.10g\n",$1);}
;

exp:NUM { $$ = $1; }
|exp '+' exp { $$ = $1 + $3; }
|exp '-' exp { $$ = $1 - $3; }
|exp '*' exp { $$ = $1 * $3; }
|exp '/' exp { $$ = $1 / $3; }
| '-' exp %prec NEG { $$ = -$2; }
|exp '^' exp { $$ = pow($1,$3); }
| '(' exp ')' { $$ = $2; }
;
%%
```

Ações  
semânticas

# Cap. 5 Tradução Dirigida pela Sintaxe

## Técnicas Básicas para geração de quádruplas

Supondo o seguinte:

Tratamento das declarações já feito

Existe uma **tabela de símbolos**, na qual podem ser encontrados os **endereços das entidades: variáveis ou constantes**, chamando, p/ex., `incluiSimbTab($1, Variavel`, que devolve o endereço do nome, na tabela.

A função **gera**, gera as quádruplas na tabela de quádruplas e atualiza a global **prox** (endereço da quádrupla)

Os atributos sintetizados, que encontram-se na pilha podem ser endereço, valor, nome de variável, tipo, ...

*Em vários momentos, a seguir, vamos usar a notação tipo Yacc, que indica os símbolos pela posição e chama o campo de atributo de \$i para os símbolos à direita da regra e de \$\$ o da esquerda da regra.*



# ***Itens importantes que devem ser levados em conta na tradução em compiladores de subida como os LR(1)***

Considerando a regra genérica, a seguir:  $X \rightarrow X_1 \dots X_i \dots X_n$

**A cada transição**, é armazenada uma unidade na pilha associada ao **símbolo  $X_i$**  (**terminal** ou **não terminal**).

A **estrutura dessa unidade** contém **3 membros**

{	1) <b><math>X_i</math></b> símbolo da gramática
	2) <b>atributo</b> associado ao <b><math>X_i</math></b>
	3) <b>estado do autômato</b>

**No momento da transição de redução da regra**, a **tradução** providencia o **pop** de cada **símbolo  $X_i$**  que está **empilhado**, começando pelo **topo** (**símbolo  $X_n$**  mais à direita) até chegar ao  **$X_1$** .

A cada  **$X_i$**  o **membro atributo da estrutura** da **unidade de pilha** é guardado na variável  **$\$i$** .

O cálculo do valor de  **$\$i$** , que é uma dentre as várias **ações**, na **redução**, deve ser codificado pelo programador do compilador.

Esse **código** é associado à **regra**, e será executado no momento dessa **transição**, na compilação.

O valor de  **$\$i$**  será guardado no **atributo** da unidade do **símbolo  $X$** .

Lembrar que o **símbolo  $X$**  está à **esquerda da regra**.

Quando a **estrutura estiver preenchida**, com os **campos símbolo, atributo e estado**, ela deve ser **empilhada**.

Por isso o nome **redução, pop dos símbolos da direita e push da esquerda**

## ***Itens importantes a serem levados em conta nas ferramentas tipo Yacc***

Considerando a regra genérica:

$$X \rightarrow X_1 \dots X_i \dots X_n$$

```
typedef union {  
    char        cadeia [20];  
    int         i;  
    Logico      b;  
    Operador    opcode;  
    tipoIntOUBool tipo;  
    int         indiceSimb;  
    int         indiceQuadrupla;  
} YYSTYPE;
```

Devemos lembrar que ao especificarmos um compilador, usando o yacc, apenas estamos declarando os seus parâmetros.

Assim, variáveis externas serão necessárias, como as \$\$ e \$i.

As variáveis \$i para cada Xi e a \$\$ para X são do tipo YYSTYPE, Esse tipo é definido como uma union em C.

Uma variável declarada com esse tipo pode assumir vários tipos, sendo os mais comuns: os de cadeia, de endereço de tabela de símbolo, de endereço de quádrupla, de tipo.

# LEX e YACC

## *Itens importantes que devem ser levados em conta*

Considerando a regra genérica, a seguir:  $X \rightarrow X_1 \dots X_i \dots X_n$

A variável **prox** indica a próxima posição disponível na tabela de quádruplas (código),  
e tem uma grande importância nas várias ações semânticas,  
que auxiliam ou vão fazer parte de desvios de código nas estruturas de controle nos comandos if, for, while, repeat

# Técnicas Básicas de Geração de Quádruplas

Regras de declaração	Ações Semânticas na Redução, em Yacc
<b>DV</b> → <b>DV</b> , <u><b>IDENT</b></u>	{entra (\$3.cadeia, \$1.tipo); \$\$.tipo = \$1.tipo;}
<b>DV</b> → <b>TIPO</b> <u><b>IDENT</b></u>	{entra (\$2.cadeia, \$1.tipo); \$\$.tipo = \$1.tipo;}
<b>TIPO</b> → int	{\$\$.tipo = i;}
<b>TIPO</b> → char	{\$\$.tipo = c;}

As regras:

**TIPO** → int ou **TIPO** → char serão reduzidas em primeiro lugar, e suas ações semânticas guardarão o tipo em \$.tipo, que será o \$1.tipo da regra **DV** → **TIPO** **IDENT** cujas ações semânticas guardarão esse \$1.tipo em \$.tipo, que será o \$1.tipo da regra **DV** → **DV** , **IDENT** que assim terá para os próximos **IDENT**'s que aparecerem no programa fonte o tipo guardado em \$1.tipo.

# Técnicas Básicas de Geração de Quádruplas

Exemplo para a gramática de uma linguagem de programação simples :

Sintaxe

Ações semânticas com uso de \$ para atributos (Yacc-like)

0. $S' \rightarrow S$	{ <u>encerra();</u> }
1. $S \rightarrow V = E$	{ <u>gera</u> ( <u>STO</u> , <u>\$3.indiceSimb</u> , <u>\$1.indiceSimb</u> , <u>NADA</u> )}
2. $S \rightarrow I ( E )$	{ <u>tabSimb</u> [ <u>\$1.indiceSimb</u> ]. <u>entidade</u> = <u>Procedimento</u> ; <u>gera</u> ( <u>PARAM</u> , <u>\$3.indiceSimb</u> , <u>NADA</u> , <u>NADA</u> ); <u>gera</u> ( <u>CALL</u> , <u>\$1.indiceSimb</u> , <u>NADA</u> , <u>NADA</u> )}
3. $E \rightarrow E + V$	{ <u>\$\$</u> . <u>indiceSimb</u> = <u>temp()</u> ; <u>gera</u> ( <u>ADD</u> , <u>\$1.indiceSimb</u> , <u>\$3.indiceSimb</u> , <u>\$\$</u> . <u>indiceSimb</u> ) }
4. $E \rightarrow V$	{ <u>\$\$</u> . <u>indiceSimb</u> = <u>\$1.indiceSimb</u> ;} <u>gera</u> ( <u>ADD</u> , <u>\$1.indiceSimb</u> , <u>\$3.indiceSimb</u> , <u>\$\$</u> . <u>indiceSimb</u> ) }
5. $V \rightarrow V = I ( E )$	{ <u>tabSimb</u> [ <u>\$3.indiceSimb</u> ]. <u>entidade</u> = <u>Funcao</u> ; <u>gera</u> ( <u>PARAM</u> , <u>\$5.indiceSimb</u> , <u>NADA</u> , <u>NADA</u> ); <u>gera</u> ( <u>CALL</u> , <u>\$3.indiceSimb</u> , <u>NADA</u> , <u>NADA</u> ); <u>gera</u> ( <u>POP</u> , <u>\$1.indiceSimb</u> , <u>NADA</u> , <u>NADA</u> ) ; <u>\$\$</u> . <u>indiceSimb</u> = <u>\$1.indiceSimb</u> ;} <u>gera</u> ( <u>ADD</u> , <u>\$1.indiceSimb</u> , <u>\$3.indiceSimb</u> , <u>\$\$</u> . <u>indiceSimb</u> ) }
6. $V \rightarrow I$	{ <u>\$\$</u> . <u>indiceSimb</u> = <u>\$1.indiceSimb</u> ;} <u>gera</u> ( <u>ADD</u> , <u>\$1.indiceSimb</u> , <u>\$3.indiceSimb</u> , <u>\$\$</u> . <u>indiceSimb</u> ) }
7. $I \rightarrow a$	{ <u>\$\$</u> . <u>indiceSimb</u> = <u>incluiSimbTab</u> ( <u>\$1</u> , <u>Variavel</u> )}

# ***Técnicas Básicas de Geração de Quádruplas***

## **Regras de Expressão**

Para as regras  $E \rightarrow T$ ,  $T \rightarrow F$  e  $F \rightarrow (E)$ , a situação é semelhante a de  $E \rightarrow V$  da gramática anterior:

$E \rightarrow T$              $\{\$$.indiceSimb = \$1.indiceSimb\}$

$T \rightarrow F$              $\{\$$.indiceSimb = \$1.indiceSimb\}$

$F \rightarrow ( E )$          $\{\$$.indiceSimb = \$2.indiceSimb\}$

Para as regras  $E \rightarrow E + T$  e  $T \rightarrow T * F$ , a situação é semelhante a de  $E \rightarrow E + V$  da gramática anterior:

$E \rightarrow E + T$     $\{\$$.indiceSimb = temp ( );$   
                   $gera(ADD, \$1.indiceSimb, \$3.indiceSimb, \$$.indiceSimb)\}$

$T \rightarrow T * F$      $\{\$$.indiceSimb = temp ( );$   
                   $gera(MULT, \$1.indiceSimb, \$3.indiceSimb, \$$.indiceSimb)\}$

## Sintaxe original

## Sintaxe com terminais nomeados por 1 letra p facilitar a implementação

## Ações semânticas com uso de \$ para atributos (Yacc-like)

Alterações de regras p acerto de desvios

0. $L' \rightarrow L$	0. $L' \rightarrow L$	{encerra();}
1. $L \rightarrow S ; L$	1. $L \rightarrow S ; L$	
2. $L \rightarrow S$	2. $L \rightarrow S$	
3. $S \rightarrow v = E$	3. $S \rightarrow v = E$	{ $\$1$ .indSimb = incluiSimbTab ( $\$1$ , Variavel); gera(STO, $\$3$ .indSimb, $\$1$ .indSimb, NADA)}
4. $S \rightarrow \text{if } E \text{ then } T$	4. $S \rightarrow s E t M T$	{remenda ( $\$4$ .indQuadr, JF, $\$2$ .indSimb, prox)}
5. $S \rightarrow \text{while } E \text{ do } T$	5. $S \rightarrow w N E M d T$	{gera (J, $\$2$ .ind Quadr, NADA, NADA); remenda( $\$4$ .indQuadr, JF, $\$3$ .indSimb,prox,NADA)}
6. $E \rightarrow v$	6. $E \rightarrow v$	{ $\$ \$$ .indSimb = incluiSimbTab ( $\$1$ , Variavel)}
7. $E \rightarrow n$	7. $E \rightarrow n$	{ $\$ \$$ .indSimb = incluiSimbTab ( $\$1$ , Constante)}
	8. $M \rightarrow \epsilon$	{ $\$ \$$ .indQuadr = prox; prox++;}
	9. $N \rightarrow \epsilon$	{ $\$ \$$ .indQuadr = prox;}
8. $T \rightarrow \{ L \}$	10. $T \rightarrow \{ L \}$	

Terminais associados	Programa fonte edentado na sintaxe original	Programa fonte edentado na sintaxe com os terminais resumidos
<b>Terminais associados</b> while      w do          d if          s then        t	x = 3; y = 4; while x do { if y then { x = 0 }; y = 0 }	x = 3; y = 4; w x d { s y t { x = 0 }; y = 0 }

aAs variáveis **x**, **y** correspondem ao terminal **v** e as constantes numéricas **3**, **4** e **0** correspondem ao terminal **n**



Sintaxe original	Sintaxe com terminais nomeados por 1 letra p facilitar a implementação	Ações semânticas com uso de \$ para atributos (Yacc-like)
------------------	--	---

0. $L' \rightarrow L$	0. $L' \rightarrow L$	{encerra();}
1. $L \rightarrow S ; L$	1. $L \rightarrow S ; L$	
2. $L \rightarrow S$	2. $L \rightarrow S$	
3. $S \rightarrow v = E$	3. $S \rightarrow v = E$	{ $\$1$ .indSimb = incluiSimbTab ( $\$1$ , Variavel); gera(STO, $\$3$ .indSimb, $\$1$ .indSimb,NADA)}
4. $S \rightarrow \text{if } E \text{ then } T$	4. $S \rightarrow s E t M T$	{remenda ( $\$4$ .indQuadr, JF, $\$2$ .indSimb, prox)}
5. $S \rightarrow \text{while } E \text{ do } T$	5. $S \rightarrow w N E M d T$	{gera (J, $\$2$ .ind Quadr, NADA, NADA); remenda( $\$4$ .indQuadr, JF, $\$3$ .indSimb,prox,NADA)}
6. $E \rightarrow v$	6. $E \rightarrow v$	{ $\$ \$$ .indSimb = incluiSimbTab ( $\$1$ , Variavel)}
7. $E \rightarrow n$	7. $E \rightarrow n$	{ $\$ \$$ .indSimb = incluiSimbTab ( $\$1$ , Constante)}
	8. $M \rightarrow \epsilon$	{ $\$ \$$ .indQuadr = prox; prox++;}
	9. $N \rightarrow \epsilon$	{ $\$ \$$ .indQuadr = prox;}
8. $T \rightarrow \{ L \}$	10. $T \rightarrow \{ L \}$	



Regras que precisam mudanças para ajudar nas ações

**COM** → **IF COND THEN LISTACOM ENDIF**

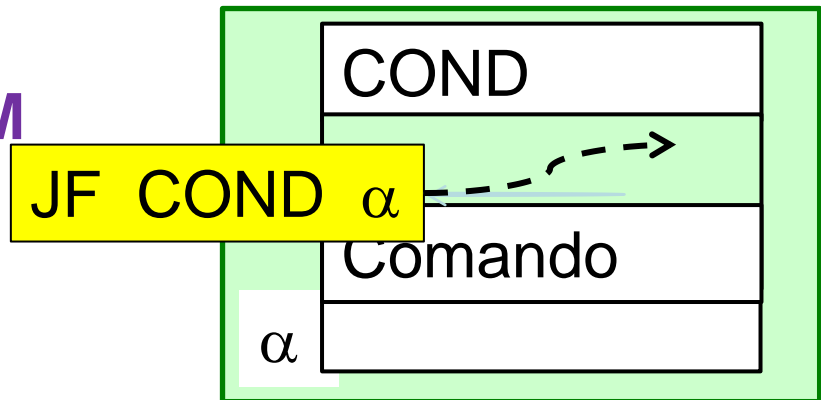
Esta regra vai mudar na especificação para o compilador, por causa da necessidade de se abrir espaço para a quádrupla que será gerada, quando se souber o endereço da instrução, para onde o jump desviará, como, a seguir:

**COM** → **IF COND **M** THEN LISTACOM ENDIF**

**M** →  $\epsilon$

Com essa mudança, as ações em **M**

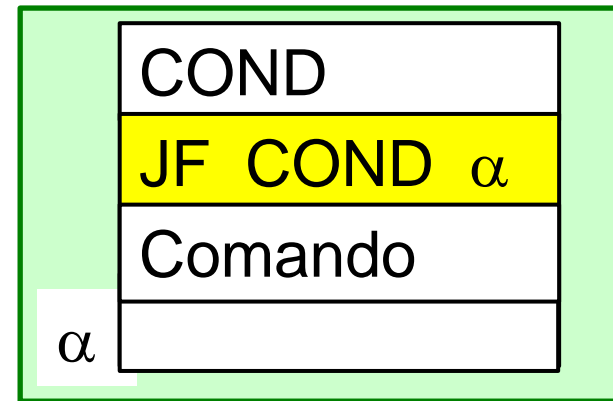
```
{  
  $$ . indiceQuadrupla = prox;  
  prox++;  
}
```



- guardam a posição da quádrupla na tabela. Essa quádrupla receberá a instrução “JF COND α”, no momento da redução da regra **if**. Essa posição estará guardada no atributo de **M** (\$3).
- abrem o espaço, incrementando a variável **prox**.

## Regras que precisam mudanças para ajudar nas ações

```
M  $\rightarrow \epsilon$  {  
  $$.indiceQuadrupla = prox;  
  prox++;  
}
```



Observe que essa ação de **M** vai

- na primeira instrução: guardar a posição da tabela aonde a quádrupla para “**JF** . . .”, a ser gerada no final da análise de **if**, deverá ser inserida contando com o procedimento **remenda**
- **abrir o espaço**, ao incrementar a variável **prox**, para permitir essa inserção, que pode ser vista abaixo, na ação da regra **if**.

**COM**  $\rightarrow$  **IF** **COND** **M** **THEN** **LISTACOM** **ENDIF**

{**remenda** (**\$3**.**indiceQuadrupla**, **JF**, **\$2**.**indiceSimb**, **prox**, **NADA**)}

**\$3**.**indiceQuadrupla** (posição da quádrupla a ser remendada)

**JF** (jump se falso)

**\$2**.**indiceSimb** (variável que guarda a condição false/true)

**prox** (endereço da próxima instrução fora do bloco do if)

Regras que precisam mudanças para ajudar nas ações

**COM** → **IF COND THEN LISTACOM ELSE LISTACOM ENDIF**

Esta regra muda, por causa da necessidade de se abrir espaço para as quádruplas a serem remendadas, quando se souber o endereço da instrução, para onde cada jump desviará, como, a seguir:

**COM** → **IF COND **M** THEN LISTACOM **M** ELSE LISTACOM ENDIF**

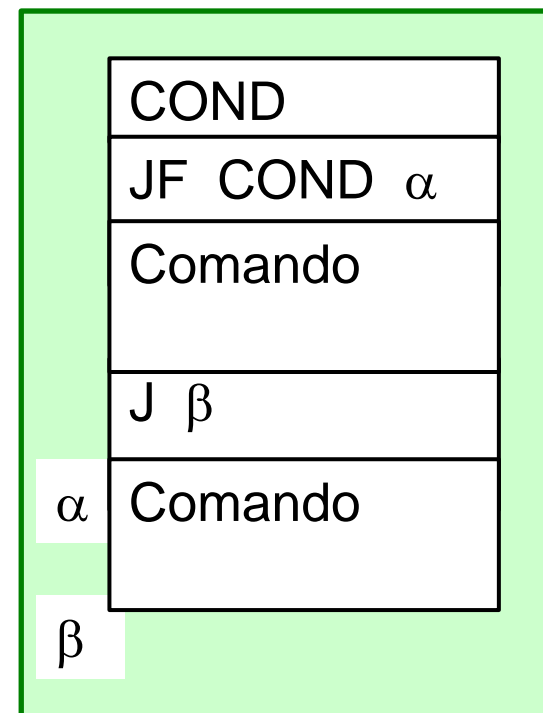
**M** →  $\epsilon$

Assim, com **M** suas ações

```
{  
$.indiceQuadrupla = prox;  
prox++;  
}
```

permitem **abrir os dois espaços**, para **guardar os locais da tabela** aonde as quádruplas para “**JF COND ?**” e “**J ?**”

serão remendadas no final da análise de **IF**.

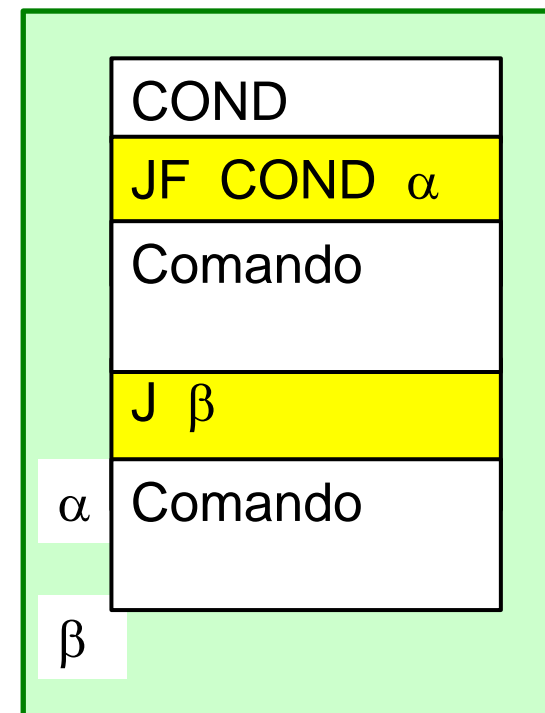


# Regras que precisam mudanças para ajudar nas ações

Aqui nessa situação de regra também há a necessidade de empregar a regra M

```
COM → IF COND M THEN LISTACOM M ELSE LISTACOM ENDIF
{
  remenda (
    $3.indiceQuadrupla, JF, $2.indiceSimb,
    $6.indiceQuadrupla+1, NADA) ;

  remenda (
    $6.indiceQuadrupla, J, prox, NADA, NADA) ;
}
```



# Regras que precisam mudanças para ajudar nas ações

**COM** → **WHILE COND DO LISTACOM ENDDO**

Esta regra vai mudar na especificação para o compilador, por causa da necessidade de:

- guardar a posição da quádrupla antes de COND para que no fim do WHILE seja gerado um jump incondicional para este endereço
- abrir um espaço para a quádrupla, que será gerada quando se souber o endereço da instrução para onde o jump condicional desviará, caso COND seja falsa;

como, a seguir:

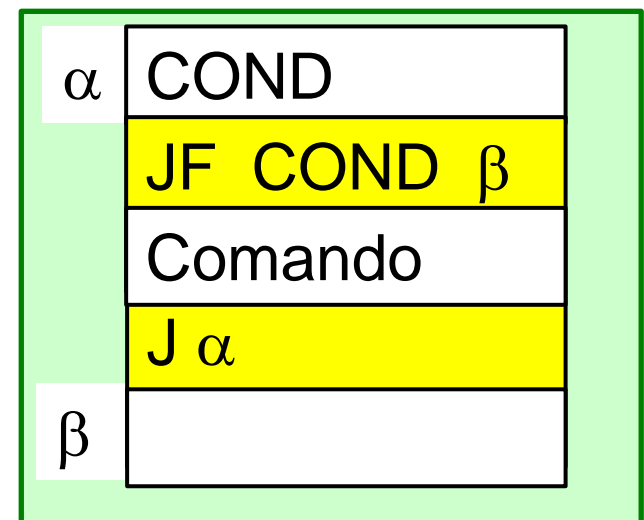
**COM** → **WHILE N COND M DO LISTACOM ENDDO**

**M** →  $\varepsilon$

**N** →  $\varepsilon$

Ações na redução de **M** →  $\varepsilon$   
{ **\$\$**.**indiceQuadrupla** = **prox**;  
**prox++**; }

Ações na redução de **N** →  $\varepsilon$   
{**\$\$**.**indiceQuadrupla** = **prox**;} }



# Regras que precisam mudanças para ajudar nas ações

**COM**  $\rightarrow$  **WHILE** **N** **COND** **M** **DO** LISTACOM **ENDDO**

**M**  $\rightarrow \epsilon$

**N**  $\rightarrow \epsilon$

Ações na redução de **N**  $\rightarrow \epsilon$   
{**\$\$**.**indiceQuadrupla** = **prox**;} }

Ações na redução de **M**  $\rightarrow \epsilon$   
{**\$\$**.**indiceQuadrupla** = **prox**;  
**prox++**; }

$\alpha$	COND
	JF COND $\beta$
	Comando
	J $\alpha$
$\beta$	

Ações na redução de **COM**  $\rightarrow$  **WHILE** **N** **COND** **M** **DO** LISTACOM **ENDDO**  
{  
**gera** (**J**, **\$2**.**indiceQuadrupla**, **NADA**, **NADA**) ;  
  
**remenda** (**\$4**.**indiceQuadrupla**, **JF**, **\$3**.**indiceSimb**, **prox**, **NADA**) ;  
}

Os autômatos de pilha ascendentes vistos no cap. 3, trataram somente com a informação do símbolo gramatical (terminal e não terminal) na unidade da pilha.

Vindo de autômato de pilha reconhecido por cadeia vazia e pilha vazia.

Agora, são acrescentadas, à estrutura da unidade de pilha, as informações de estado (obtido pelo LR(1) e seus derivados e atributo.

Assim, a unidade vira uma estrutura (símbolo, atributo, estado)

Transições para os exercícios de chinês a seguir:

1) Se  $\delta(p, a) = q$ , empilha a estrutura : (a, atributo, q)

2) Se  $\delta(p, a) = R_i$  e regra i é  $B \rightarrow \gamma$   
então

reduz  $|\gamma|$  (nº de pop's)

verifica na tabela de estados, o estado do topo com B

$\delta(\text{topo}, B) = q$ ,

empilha a estrutura (B, atributo, q) (\$\$ é o atributo do lado esq.)

3) Quando chegar a  $R_0$ , como resultado da transição, então encerra a análise.

Sintaxe original

Sintaxe com  
terminais nomeados  
por 1 letra p facilitar  
a implementação

Ações semânticas com uso de \$ para atributos (Yacc-like)

0. $L' \rightarrow L$	0. $L' \rightarrow L$	{encerra();}
1. $L \rightarrow S ; L$	1. $L \rightarrow S ; L$	
2. $L \rightarrow S$	2. $L \rightarrow S$	
3. $S \rightarrow v = E$	3. $S \rightarrow v = E$	{ $\$1.indSimb = incluiSimbTab (\$1, Variavel);$ $gera(STO, \$3.indSimb, \$1.indSimb, NADA)$ }
4. $S \rightarrow \text{if } E \text{ then } T$	4. $S \rightarrow s E t M T$	{ $remenda (\$4.indQuadr, JF, \$2.indSimb, prox)$ }
5. $S \rightarrow \text{while } E \text{ do } T$	5. $S \rightarrow w N E M d T$	{ $gera (J, \$2.indQuadr, NADA, NADA);$ $remenda(\$4.indQuadr, JF, \$3.indSimb, prox, NADA)$ }
6. $E \rightarrow v$	6. $E \rightarrow v$	{ $\$$.indSimb = incluiSimbTab (\$1, Variavel)$ }
7. $E \rightarrow n$	7. $E \rightarrow n$	{ $\$$.indSimb = incluiSimbTab (\$1, Constante)$ }
	8. $M \rightarrow \epsilon$	{ $\$$.indQuadr = prox;$ $prox++;$ }
	9. $N \rightarrow \epsilon$	{ $\$$.indQuadr = prox;$ }
8. $T \rightarrow \{ L \}$	10. $T \rightarrow \{ L \}$	

Terminais associados	Programa fonte edentado na sintaxe original	Programa fonte edentado na sintaxe com os terminais resumidos
Terminais associados while      w do          d if          s then        t	x = 3; y = 4; while x do { if y then { x = 0 }; y = 0 }	x = 3; y = 4; w x d { s y t { x = 0 }; y = 0 }

aAs variáveis  $x$ ,  $y$  correspondem ao terminal  $v$  e as constantes numéricas 3, 4 e 0 correspondem ao terminal  $n$



tabela laLR(1) para a G3

	L	S	E	M	N	T	;	v	=	s	w	n	{	}	t	d	\$	
0	1	2						3		4	5							0
1																	R <sub>0</sub>	1
2							6							R <sub>2</sub>			R <sub>2</sub>	2
3									7									3
4			8					9				10						4
5					11			R <sub>9</sub>				R <sub>9</sub>						5
6	12	2						3		4	5							6
7			13					9				10						7
8															14			8
9							R <sub>6</sub>							R <sub>6</sub>	R <sub>6</sub>	R <sub>6</sub>	R <sub>6</sub>	9
10							R <sub>7</sub>							R <sub>7</sub>	R <sub>7</sub>	R <sub>7</sub>	R <sub>7</sub>	10
11			15					9				10						11
12														R <sub>1</sub>			R <sub>1</sub>	12
13							R <sub>3</sub>							R <sub>3</sub>			R <sub>3</sub>	13
14				16									R <sub>8</sub>					14
15				17												R <sub>8</sub>		15
16						18							19					16
17																20		17
18							R <sub>4</sub>							R <sub>4</sub>			R <sub>4</sub>	18
19	21	2						3		4	5							19
20						22							19					20
21														23				21
22							R <sub>5</sub>							R <sub>5</sub>			R <sub>5</sub>	22
23							R <sub>10</sub>							R <sub>10</sub>			R <sub>10</sub>	23
	L	S	E	M	N	T	;	v	=	s	w	n	{	}	t	d	\$	

```

int incluiSimbTab(
YYSTYPE simb,Entidade vOUc){
int retorno;
/*se busca encontrar, retorna posição da tabela*/
retorno = buscaSimbTab (simb, vOUc, Alocar);
if (retorno == FRACASSO ) { //simb deve ser incluído
    if (vOUc == Constante)
        tabSimb [topTab].valor = simb.i;//Constante
    else//variável
        strcpy (tabSimb [topTab].nome, simb.cadeia);
        tabSimb [topTab].entidade = vOUc;
    retorno = topTab; // de FRACASSO, incluído novo simbolo
    topTab++;
}
return (retorno)
}

```

```

x = 3;
y = 4;
while x do {
    if y then {
        x = 0
    };
    y = 0
}

```

tabela			
Índice	nome/valor	entidade: Cte/Variavel	Valor
0	3	Constante	
1	x	Variavel	
2	4	Constante	
3	y	Variavel	
4	0	Constante	
5			
...			

pilha sintática		entrada do prog.fonte	prox	i:[op op1 op2 op3]	G3
$x = 3; y = 4; w \ x \ d\{s \ y \ t\{x = 0\}; y = 0\}\$$					
1	{..., ..., 0}	x=3; y=4; ...	0		
2	{..., ..., 0}{v, x, 3}	=3; y=4; ...			
3	{..., ..., 0}{v, x, 3}{=, ..., 7}	3; y=4; ...			
4	{..., ..., 0}{v, x, 3}{=, ..., 7}{n, 3, 10}	; y=4; w...			
5	{..., ..., 0}{v, x, 3}{=, ..., 7}{E, 0, 13}				
6	{..., ..., 0}{S, ..., 2}		1	0:[STO, 0, 1]	
7	{..., ..., 0}{S, ..., 2}{;, ..., 6}	y=4; w...			
8	{..., ..., 0}{S, ..., 2}{;, ..., 6}{v, y, 3}	=4; w...			
9	{..., ..., 0}{S, ..., 2}{;, ..., 6}{v, y, 3}{=, ..., 7}	4; w x d			
10	{..., ..., 0}{S, ..., 2}{;, ..., 6}{v, y, 3}{=, ..., 7}{n, 4, 10}	; w x d			
11	{..., ..., 0}{S, ..., 2}{;, ..., 6}{v, y, 3}{=, ..., 7}{E, 2, 13}				
12	{..., ..., 0}{S, ..., 2}{;, ..., 6}{S, ..., 2}		2	1:[STO, 2, 3]	
13	{..., ..., 0}{S, ..., 2}{;, ..., 6}{S, ..., 2}{;, ..., 6}	w x d{			
14	{..., ..., 0}{S, ..., 2}{;, ..., 6}{S, ..., 2}{;, ..., 6}{w, ..., 5}	x d{			
estranho ? aqui vai haver a redução da regra 9. $N \rightarrow \epsilon$ , em que o valor do prox é guardado no atributo de N.					
15	{..., ..., 0}{S, ..., 2}{;, ..., 6}{S, ..., 2}{;, ..., 6}{w, ..., 5}{N, 2, 11}				
16	{..., ..., 0}{S, ..., 2}{;, ..., 6}{S, ..., 2}{;, ..., 6}{w, ..., 5}{N, 2, 11}{v, x, 9}	d{syt{x=0}; y=0}\$			
17	{..., ..., 0}{S, ..., 2}{;, ..., 6}{S, ..., 2}{;, ..., 6}{w, ..., 5}{N, 2, 11}{E, 1, 15}				
estranho ? aqui vai haver a redução da regra 8. $M \rightarrow \epsilon$ , em que o valor do prox é guardado no atributo de M e prox é incrementado para abrir espaço p remendar o JF nesta quádrupla.					
18	{..., ..., 0}{S, ..., 2}{;, ..., 6}{S, ..., 2}{;, ..., 6}{w, ..., 5}{N, 2, 11}{E, 1, 15}{M, 2, 17}		3		
19	{..., ..., 0}{S, ..., 2}{;, ..., 6}{S, ..., 2}{;, ..., 6}{w, ..., 5}{N, 2, 11}{E, 1, 15}{M, 2, 17}{d, ..., 20} retomada na l. 44	{syt{x=0}; y=0}\$			
20	{..., ..., 19}	syt{x=0}; y=0}\$			
21	{..., ..., 19}{s, ..., 4}{v, y, 9}	t{x=0}; y=0}\$			
22	{..., ..., 19}{s, ..., 4}{E, 3, 8}				
23	{..., ..., 19}{s, ..., 4}{E, 3, 8}{t, ..., 14}	{x=0}; y=0}\$			
estranho ? aqui vai haver a redução da regra 8. $M \rightarrow \epsilon$ , em que o valor do prox é guardado no atributo de M e prox é incrementado para abrir espaço p remendar o JF nesta quádrupla.					
24	{..., ..., 19}{s, ..., 4}{E, 3, 8}{t, ..., 14}{M, 3, 16}		4		

pilha sintática		x = 3; y = 4; w x d{s y t{x = 0}; y = 0}\$	entrada do prog.fonte	prox	i:[op op1 op2 op3]
1	{..., ..., 0}	<p>Transições para os exercícios de chinês a seguir:</p> <p>1) Se <math>\delta(p, a) = q</math>, empilha a estrutura : (a, atributo, q)</p> <p>2) Se <math>\delta(p, a) = R_i</math> e regra i é <math>B \rightarrow \gamma</math> então  reduz <math> \gamma </math> (nº de pop's)  verifica na tabela de estados, o estado do topo com B  <math>\delta(\text{topo}, B) = q</math>,  empilha a estrutura (B, atributo, q) (\$\$ é o atributo do lado esq.)</p> <p>3) Quando chegar a <math>R_0</math>, como resultado da transição, então encerra a análise.</p>	x=3; y=4; ...	0	
2	{..., ..., 0}{v, x, 3}		=3; y=4; ...		
3	{..., ..., 0}{v, x, 3}{=, ..., 7}		3; y=4; ...		
4	{..., ..., 0}{v, x, 3}{=, ..., 7}{n, 3, 10}		; y=4; w...		
5	{..., ..., 0}{v, x, 3}{=, ..., 7}{E, 0, 13}				
6	{..., ..., 0}{S, ..., 2}			1	0:[STO, 0, 1]
7	{..., ..., 0}{S, ..., 2}{;, ..., 6}		y=4; w...		
8	{..., ..., 0}{S, ..., 2}{;, ..., 6}{v, y, 3}		=4; w...		
9	{..., ..., 0}{S, ..., 2}{;, ..., 6}{v, y, 3}{=, ..., 7}		4; w x d		
10	{..., ..., 0}{S, ..., 2}{;, ..., 6}{v, y, 3}{=, ..., 7}{n, 4, 10}		; w x d		
11	{..., ..., 0}{S, ..., 2}{;, ..., 6}{v, y, 3}{=, ..., 7}{E, 2, 13}				
12	{..., ..., 0}{S, ..., 2}{;, ..., 6}{S, ..., 2}			2	1:[STO, 2, 3]
13	{..., ..., 0}{S, ..., 2}{;, ..., 6}{S, ..., 2}{;, ..., 6}		w x d{		
14	{..., ..., 0}{S, ..., 2}{;, ..., 6}{S, ..., 2}{;, ..., 6}{w, ..., 5}		x d{		
	estranho ? aqui vai haver a redução da regra 9. $N \rightarrow \epsilon$ , em que o valor do prox é guardado no atributo de N.				
15	{..., ..., 0}{S, ..., 2}{;, ..., 6}{S, ..., 2}{;, ..., 6}{w, ..., 5}{N, 2, 11}				
16	{..., ..., 0}{S, ..., 2}{;, ..., 6}{S, ..., 2}{;, ..., 6}{w, ..., 5}{N, 2, 11}{v, x, 9}		d{syt{x=0}; y=0}\$		
17	{..., ..., 0}{S, ..., 2}{;, ..., 6}{S, ..., 2}{;, ..., 6}{w, ..., 5}{N, 2, 11}{E, 1, 15}				
	estranho ? aqui vai haver a redução da regra 8. $M \rightarrow \epsilon$ , em que o valor do prox é guardado no atributo de M e prox é incrementado para abrir espaço p remendar o JF nesta quádrupla.				
18	{..., ..., 0}{S, ..., 2}{;, ..., 6}{S, ..., 2}{;, ..., 6}{w, ..., 5}{N, 2, 11}{E, 1, 15}{M, 2, 17}			3	
19	{..., ..., 0}{S, ..., 2}{;, ..., 6}{S, ..., 2}{;, ..., 6}{w, ..., 5}{N, 2, 11}{E, 1, 15}{M, 2, 17}{d, ..., 20} retomada na l. 44		{syt{x=0}; y=0}\$		
20	{..., ..., 19}		syt{x=0}; y=0}\$		
21	{..., ..., 19}{s, ..., 4}{v, y, 9}		t{x=0}; y=0}\$		
22	{..., ..., 19}{s, ..., 4}{E, 3, 8}				
23	{..., ..., 19}{s, ..., 4}{E, 3, 8}{t, ..., 14}		{x=0}; y=0}\$		
	estranho ? aqui vai haver a redução da regra 8. $M \rightarrow \epsilon$ , em que o valor do prox é guardado no atributo de M e prox é incrementado para abrir espaço p remendar o JF nesta quádrupla.				
24	{..., ..., 19}{s, ..., 4}{E, 3, 8}{t, ..., 14}{M, 3, 16}			4	

25	$\{\{s, \dots, 19\} \{s, \dots, 4\} \{E, 3, 8\} \{t, \dots, 14\} \{M, 3, 16\} \{\{s, \dots, 19\}\}$	$x=0; y=0\}$		
26	$\{\{s, \dots, 19\} \{s, \dots, 4\} \{E, 3, 8\} \{t, \dots, 14\} \{M, 3, 16\} \{\{s, \dots, 19\} \{v, x, 3\}\}$	$=0; y=0\}$		
27	$\{\{s, \dots, 19\} \{s, \dots, 4\} \{E, 3, 8\} \{t, \dots, 14\} \{M, 3, 16\} \{\{s, \dots, 19\} \{v, x, 3\} \{=, \dots, 7\}\}$	$0; y=0\}$		
28	$\{\{s, \dots, 19\} \{s, \dots, 4\} \{E, 3, 8\} \{t, \dots, 14\} \{M, 3, 16\} \{\{s, \dots, 19\} \{v, x, 3\} \{=, \dots, 7\} \{n, 0, 10\}\}$	$\}; y=0\}$		
29	$\{\{s, \dots, 19\} \{s, \dots, 4\} \{E, 3, 8\} \{t, \dots, 14\} \{M, 3, 16\} \{\{s, \dots, 19\} \{v, x, 3\} \{=, \dots, 7\} \{E, 4, 13\}\}$			
30	$\{\{s, \dots, 19\} \{s, \dots, 4\} \{E, 3, 8\} \{t, \dots, 14\} \{M, 3, 16\} \{\{s, \dots, 19\} \{S, \dots, 2\}\}$		5	4:[STO, 4, 1]
31	$\{\{s, \dots, 19\} \{s, \dots, 4\} \{E, 3, 8\} \{t, \dots, 14\} \{M, 3, 16\} \{\{s, \dots, 19\} \{L, \dots, 21\}\}$			
32	$\{\{s, \dots, 19\} \{s, \dots, 4\} \{E, 3, 8\} \{t, \dots, 14\} \{M, 3, 16\} \{\{s, \dots, 19\} \{L, \dots, 21\} \{ \}, \dots, 23\}\}$	$; y=0\}$		
33	$\{\{s, \dots, 19\} \{s, \dots, 4\} \{E, 3, 8\} \{t, \dots, 14\} \{M, 3, 16\} \{T, \dots, 18\}$ aqui vai haver a redução da regra 4. $S \rightarrow s E t M T$ com os efeitos das ações semânticas, na coluna de quádruplas.			
34	$\{\{s, \dots, 19\} \{S, \dots, 2\}\}$			3:[JF, 3, 5, NADA]
35	$\{\{s, \dots, 19\} \{S, \dots, 2\} \{ \}, \dots, 6\}$	$y=0\}$		
36	$\{\{s, \dots, 19\} \{S, \dots, 2\} \{ \}, \dots, 6\} \{v, y, 3\}$	$=0\}$		
37	$\{\{s, \dots, 19\} \{S, \dots, 2\} \{ \}, \dots, 6\} \{v, y, 3\} \{=, \dots, 7\}$	$0\}$		
38	$\{\{s, \dots, 19\} \{S, \dots, 2\} \{ \}, \dots, 6\} \{v, y, 3\} \{=, \dots, 7\} \{n, 0, 10\}$	$\}\}$		
39	$\{\{s, \dots, 19\} \{S, \dots, 2\} \{ \}, \dots, 6\} \{v, y, 3\} \{=, \dots, 7\} \{E, 4, 13\}$			
40	$\{\{s, \dots, 19\} \{S, \dots, 2\} \{ \}, \dots, 6\} \{S, \dots, 2\}$		6	5:[STO, 4, 3, NADA]
41	$\{\{s, \dots, 19\} \{S, \dots, 2\} \{ \}, \dots, 6\} \{L, \dots, 12\}$			
42	$\{\{s, \dots, 19\} \{L, \dots, 21\}\}$			
43	$\{\{s, \dots, 19\} \{L, \dots, 21\} \{ \}, \dots, 23\}$ aqui haverá a redução da regra 10. $T \rightarrow \{ L \}$ , cujo T será empilhado continuando o que tinha na pilha desde a linha 19, ver abaixo, deixando a regra while completa.	$\}$		
44	$\{\{ \dots, \dots, 0\} \{S, \dots, 2\} \{ \}, \dots, 6\} \{S, \dots, 2\} \{ \}, \dots, 6\} \{w, \dots, 5\} \{N, 2, 11\} \{E, 1, 15\} \{M, 2, 17\} \{d, \dots, 20\} \{T, \dots, 22\}$ aqui vai haver a redução da regra 5. $S \rightarrow w N E M d T$ com os efeitos das ações semânticas, na coluna de quádruplas e do prox.			
45	$\{\{ \dots, \dots, 0\} \{S, \dots, 2\} \{ \}, \dots, 6\} \{S, \dots, 2\} \{ \}, \dots, 6\} \{S, \dots, 2\}$		7	6:[J, 2, NADA, NADA] 2:[JF, 1, 7, NADA]
46	$\{\{ \dots, \dots, 0\} \{S, \dots, 2\} \{ \}, \dots, 6\} \{S, \dots, 2\} \{ \}, \dots, 6\} \{L, \dots, 12\}$			
47	$\{\{ \dots, \dots, 0\} \{S, \dots, 2\} \{ \}, \dots, 6\} \{L, \dots, 12\}$			
48	$\{\{ \dots, \dots, 0\} \{L, \dots, 1\}\}$			
49	$\{\{ \dots, \dots, 0\} \{L', \dots, \dots\}$ reduz a regra 0, encerra a análise			



Estado 0		
$S' \rightarrow \cdot S$	S	1
$S \rightarrow \cdot V = E$	V	2
$S \rightarrow \cdot I(E)$	I	3
$V \rightarrow \cdot V = I(E)$	V	2
$V \rightarrow \cdot I$	I	3
$I \rightarrow \cdot a$	a	4

follow

S	\$
E	\$ ) +
V	= \$ ) +
I	( = \$ ) +

Estado 1		
$S' \rightarrow S \cdot$	$\pi$	0

Estado 2

$S \rightarrow V \cdot = E$	=	5
$V \rightarrow V \cdot = I(E)$	=	5

Estado 3

$S \rightarrow I \cdot (E)$	(	6
$V \rightarrow I \cdot$	$\pi$	6

Estado 4

$I \rightarrow a \cdot$	$\pi$	7
-------------------------	-------	---

Estado 5

$S \rightarrow V = \cdot E$	E	7
$V \rightarrow V = \cdot I(E)$	I	8
$E \rightarrow \cdot E + V$	E	7
$E \rightarrow \cdot V$	V	9
$V \rightarrow \cdot V = I(E)$	V	9
$V \rightarrow \cdot I$	I	8
$I \rightarrow \cdot a$	a	4

Estado 6

$S \rightarrow I ( \cdot E )$	E	10
$E \rightarrow \cdot E + V$	E	10
$E \rightarrow \cdot V$	V	9
$V \rightarrow \cdot V = I(E)$	V	9
$V \rightarrow \cdot I$	I	11
$I \rightarrow \cdot a$	a	4

Estado 16

$E \rightarrow E + V \cdot$	$\pi$	3
$V \rightarrow V \cdot = I(E)$	=	14

Estado 17

$V \rightarrow V = I ( E \cdot )$	)	19
$E \rightarrow E \cdot + V$	+	12

Estado 18

$V \rightarrow V = I \cdot (E)$	(	13
---------------------------------	---	----

Estado 7

$S \rightarrow V = E \cdot$	$\pi$	1
$E \rightarrow E \cdot + V$	+	12

Estado 8

$V \rightarrow V = I \cdot (E)$	(	13
$V \rightarrow I \cdot$	$\pi$	6

Estado 9

$E \rightarrow V \cdot$	$\pi$	4
$V \rightarrow V \cdot = I(E)$	=	14

Estado 10

$S \rightarrow I ( E \cdot )$	)	15
$E \rightarrow E \cdot + V$	+	12

Estado 11

$V \rightarrow I \cdot$	$\pi$	6
-------------------------	-------	---

Estado 19

$V \rightarrow V = I ( E ) \cdot$	$\pi$	5
-----------------------------------	-------	---

Estado 12

$E \rightarrow E + \cdot V$	V	16
$V \rightarrow \cdot V = I(E)$	V	16
$V \rightarrow \cdot I$	I	11
$I \rightarrow \cdot a$	a	4

Estado 13

$V \rightarrow V = I ( \cdot E )$	E	17
$E \rightarrow \cdot E + V$	E	17
$E \rightarrow \cdot V$	V	9
$V \rightarrow \cdot V = I(E)$	V	9
$V \rightarrow \cdot I$	I	11
$I \rightarrow \cdot a$	a	4

Estado 14

$V \rightarrow V = \cdot I(E)$	I	18
$I \rightarrow \cdot a$	a	4

Estado 15

$S \rightarrow I ( E ) \cdot$	$\pi$	2
-------------------------------	-------	---

	S	E	V	I	(	)	+	a	\$
0									
1									$\pi_0$
2				5					
3				$\pi_6$	6	$\pi_6$	$\pi_6$		$\pi_6$
4				$\pi_7$	$\pi_7$	$\pi_7$	$\pi_7$		$\pi_7$
5		7	9	8					4
6		10	9	11					4
7								12	$\pi_1$
8				$\pi_6$	13	$\pi_6$	$\pi_6$		$\pi_6$
9				14		$\pi_4$	$\pi_4$		$\pi_4$
10						15	12		
11				$\pi_6$		$\pi_6$	$\pi_6$		$\pi_6$
12			16	11					4
13			17	9	11				4
14				18					4
15									$\pi_2$
16				14		$\pi_3$	$\pi_3$		$\pi_3$
17						19	12		
18					13				
19				$\pi_5$		$\pi_5$	$\pi_5$		$\pi_5$

Gramática G2 sLR(1)

GRAMÁTICA

0.  $S' \rightarrow S$
1.  $S \rightarrow V = E$
2.  $S \rightarrow I(E)$
3.  $E \rightarrow E + V$
4.  $E \rightarrow V$
5.  $V \rightarrow V = I(E)$
6.  $V \rightarrow I$
7.  $I \rightarrow a$

## Gramática G2 sLR(1)

GRAMÁTICA

0.  $S' \rightarrow S$

1.  $S \rightarrow V = E$

2.  $S \rightarrow I(E)$

3.  $E \rightarrow E + V$

4.  $E \rightarrow V$

5.  $V \rightarrow V = I(E)$

6.  $V \rightarrow I$

7.  $I \rightarrow a$

	S E V I = ( ) + a \$									
0	1	2	3						4	
1										$\pi_0$
2					5					
3					$\pi_6$	6	$\pi_6$	$\pi_6$		$\pi_6$
4					$\pi_7$	$\pi_7$	$\pi_7$	$\pi_7$		$\pi_7$
5		7	9	8					4	
6		10	9	11					4	
7								12		$\pi_1$
8					$\pi_6$	13	$\pi_6$	$\pi_6$		$\pi_6$
9					14		$\pi_4$	$\pi_4$		$\pi_4$
10							15	12		
11					$\pi_6$		$\pi_6$	$\pi_6$		$\pi_6$
12			16	11					4	
13		17	9	11					4	
14				18					4	
15										$\pi_2$
16					14		$\pi_3$	$\pi_3$		$\pi_3$
17							19	12		
18						13				
19					$\pi_5$		$\pi_5$	$\pi_5$		$\pi_5$

Sintaxe	Ações semânticas com uso de \$ para atributos (Yacc-like)
0. $S' \rightarrow S$	{encerra();}
1. $S \rightarrow V = E$	{gera(STO, \$3.indiceSimb,\$1.indiceSimb,NADA)}
2. $S \rightarrow I ( E )$	{tabSimb[\$1.indiceSimb].entidade = Procedimento; gera (PARAM,\$3.indiceSimb,NADA,NADA); gera(CALL, \$1.indiceSimb, NADA, NADA)}
3. $E \rightarrow E + V$	{\$\$indiceSimb = temp(); gera ( ADD, \$1.indiceSimb, \$3.indiceSimb, \$\$indiceSimb ) }
4. $E \rightarrow V$	{\$\$indiceSimb = \$1.indiceSimb;}
5. $V \rightarrow V = I ( E )$	{tabSimb[\$3.indiceSimb].entidade = Funcao; gera (PARAM,\$5.indiceSimb,NADA,NADA); gera(CALL, \$3.indiceSimb, NADA, NADA); gera(POP, \$1.indiceSimb, NADA, NADA) ; \$\$indiceSimb=\$1.indiceSimb;}
6. $V \rightarrow I$	{\$\$indiceSimb = \$1.indiceSimb;}
7. $I \rightarrow a$	{\$\$indiceSimb = incluiSimbTab (\$1, Variavel)}

Tabela sLR(1)										
	S	E	V	I	=	(	)	+	a	\$
0	1		2	3					4	
1										R <sub>0</sub>
2					5					
3					R <sub>8</sub>	6	R <sub>8</sub>	R <sub>8</sub>		R <sub>8</sub>
4					R <sub>7</sub>	R <sub>7</sub>	R <sub>7</sub>	R <sub>7</sub>		R <sub>7</sub>
5		7	9	8					4	
6		10	9	11					4	
7								12		R <sub>1</sub>
8					R <sub>8</sub>	13	R <sub>8</sub>	R <sub>8</sub>		R <sub>8</sub>
9					14		R <sub>4</sub>	R <sub>4</sub>		R <sub>4</sub>
10							15	12		
11					R <sub>8</sub>		R <sub>8</sub>	R <sub>8</sub>		R <sub>8</sub>
12			16	11					4	
13		17	9	11					4	
14				18					4	
15										R <sub>2</sub>
16					14		R <sub>3</sub>	R <sub>3</sub>		R <sub>3</sub>
17							19	12		
18						13				
19					R <sub>5</sub>		R <sub>5</sub>	R <sub>5</sub>		R <sub>5</sub>



# Gramática G2

Sintaxe	Ações semânticas com uso de \$ para atributos (Yacc-like)
0. $S' \rightarrow S$	{ <u>encerra();</u> }
1. $S \rightarrow V = E$	{ <u>gera(STO, \$3.indiceSimb,\$1.indiceSimb,NADA)</u> }
2. $S \rightarrow I ( E )$	{ <u>tabSimb[\$1.indiceSimb].entidade = Procedimento;</u> <u>gera (PARAM,\$3.indiceSimb,NADA,NADA);</u> <u>gera(CALL, \$1.indiceSimb, NADA, NADA)</u> }
3. $E \rightarrow E + V$	{ <u>\$\$.<u>indiceSimb</u> = temp();</u> <u>gera ( ADD, \$1.indiceSimb, \$3.indiceSimb, \$\$.<u>indiceSimb</u> ) }</u> }
4. $E \rightarrow V$	{ <u>\$\$.<u>indiceSimb</u> = \$1.<u>indiceSimb</u>;</u> }
5. $V \rightarrow V = I ( E )$	{ <u>tabSimb[\$3.indiceSimb].entidade = Funcao;</u> <u>gera (PARAM,\$5.indiceSimb,NADA,NADA);</u> <u>gera(CALL, \$3.indiceSimb, NADA, NADA);</u> <u>gera(POP, \$1.indiceSimb, NADA, NADA) ;</u> <u>\$\$.<u>indiceSimb</u>=\$1.<u>indiceSimb</u>;</u> }
6. $V \rightarrow I$	{ <u>\$\$.<u>indiceSimb</u> = \$1.<u>indiceSimb</u>;</u> }
7. $I \rightarrow a$	{ <u>\$\$.<u>indiceSimb</u> = incluiSimbTab (\$1, Variavel)</u> }

## Gramática G2

Tabela de símbolos, preenchida conforme o aparecimento dos identificadores na análise de:  $x=y+z=f(r+t)$  \$

Tabela de Símbolos – observar que os atributos na pilha serão preenchidos com o índice da tabela

Índice	nome	entidade	Valor da Variável ou Endereço de Chamada para Procedimento e Função
0	x	Variavel	
1	y	Variavel	
2	z	Variavel	
3	f	Funcao	
4	r	Variavel	
5	t	Variavel	
...			
50	t1	Temporario	
51	t2	Temporario	

# pilha prox, e quádruplas na análise de: $x=y+z=f(r+t) \$$

	pilha sintática	Transições para os exercícios de chinês a seguir:	entrada do prog.fonte	prox	i:[op op1 op2 op3]
1	{...,0}	1) Se $\delta(p, a) = q$ , empilha a estrutura : (a, atributo, q)	$x=y+z=f(r+t) \$$	0	
2	{...,0}{a,x,4}	2) Se $\delta(p, a) = R_i$ e regra i é $B \rightarrow \gamma$	$=y+z=f(r+t) \$$		
3	{...,0}{l,0,3}	então			
4	{...,0}{V,0,2}	reduz $ \gamma $ (nº de pop's)			
5	{...,0}{V,0,2}{=,5}	verifica na tabela de estados, o estado do topo com B	$y+z=f(r+t) \$$		
6	{...,0}{V,0,2}{=,5}{a,y,4}	$\delta(\text{topo}, B) = q$ ,	$+z=f(r+t) \$$		
7	{...,0}{V,0,2}{=,5}{l,1,8}	empilha a estrutura (B, atributo, q) (\$\$ é o atributo do			
8	{...,0}{V,0,2}{=,5}{V,1,9}	lado esq.)			
9	{...,0}{V,0,2}{=,5}{E,1,7}	3) Quando chegar a $R_0$ , como resultado da transição,			
10	{...,0}{V,0,2}{=,5}{E,1,7}{+,12}	então encerra a análise.	$z=f(r+t) \$$		
11	{...,0}{V,0,2}{=,5}{E,1,7}{+,12}{a,z,4}		$=f(r+t) \$$		
12	{...,0}{V,0,2}{=,5}{E,1,7}{+,12}{l,2,11}				
13	{...,0}{V,0,2}{=,5}{E,1,7}{+,12}{V,2,16}				
14	{...,0}{V,0,2}{=,5}{E,1,7}{+,12}{V,2,16}{=,14}		$f(r+t) \$$		
15	{...,0}{V,0,2}{=,5}{E,1,7}{+,12}{V,2,16}{=,14}{a,f,4}		$(r+t) \$$		
16	{...,0}{V,0,2}{=,5}{E,1,7}{+,12}{V,2,16}{=,14}{l,3,18}				
17	{...,0}{V,0,2}{=,5}{E,1,7}{+,12}{V,2,16}{=,14}{l,3,18}{(,13}		$r+t) \$$		
18	{...,0}{V,0,2}{=,5}{E,1,7}{+,12}{V,2,16}{=,14}{l,3,18}{(,13}{a,r,4}		$+t) \$$		
19	{...,0}{V,0,2}{=,5}{E,1,7}{+,12}{V,2,16}{=,14}{l,3,18}{(,13}{l,4,11}				
20	{...,0}{V,0,2}{=,5}{E,1,7}{+,12}{V,2,16}{=,14}{l,3,18}{(,13}{V,4,9}				
21	{...,0}{V,0,2}{=,5}{E,1,7}{+,12}{V,2,16}{=,14}{l,3,18}{(,13}{E,4,17}				
22	{...,0}{V,0,2}{=,5}{E,1,7}{+,12}{V,2,16}{=,14}{l,3,18}{(,13}{E,4,17}{+,12}		$t) \$$		
23	{...,0}{V,0,2}{=,5}{E,1,7}{+,12}{V,2,16}{=,14}{l,3,18}{(,13}{E,4,17}{+,12}{a,t,4}		$) \$$		
24	{...,0}{V,0,2}{=,5}{E,1,7}{+,12}{V,2,16}{=,14}{l,3,18}{(,13}{E,4,17}{+,12}{l,5,11}				
25	{...,0}{V,0,2}{=,5}{E,1,7}{+,12}{V,2,16}{=,14}{l,3,18}{(,13}{E,4,17}{+,12}{V,5,16}				
26	{...,0}{V,0,2}{=,5}{E,1,7}{+,12}{V,2,16}{=,14}{l,3,18}{(,13}{E,50,17}			1	0 : [ ADD 4 5 50 ]
27	{...,0}{V,0,2}{=,5}{E,1,7}{+,12}{V,2,16}{=,14}{l,3,18}{(,13}{E,50,17}{),19}		$\$$		
28	{...,0}{V,0,2}{=,5}{E,1,7}{+,12}{V,2,16}			2	1 : [ PARAM 50 ]
				3	2 : [ CALL 3 ]
				4	3 : [ POP 2 ]
29	{...,0}{V,0,2}{=,5}{E,51,7}			5	4 : [ ADD 1 2 51 ]
30	{...,0}{S',1}			6	5 : [ STO 51 0 ]
31	{...,0}{S',...} a redução da regra 0 encerra a análise				

## *Exemplo para a gramática da linguagem da calculadora calc:*

```
/*
Calculadora de notação
infix - calc
*/

%{
#define YYSTYPE double
#include <math.h>
%}

/* BISON Declarations */
%token NUM
%left '-' '+'
%left '*' '/'
/*negation unary minus */
%left NEG
'^'/*exponentiation*/
%right '^'/*exponentiation*/
```

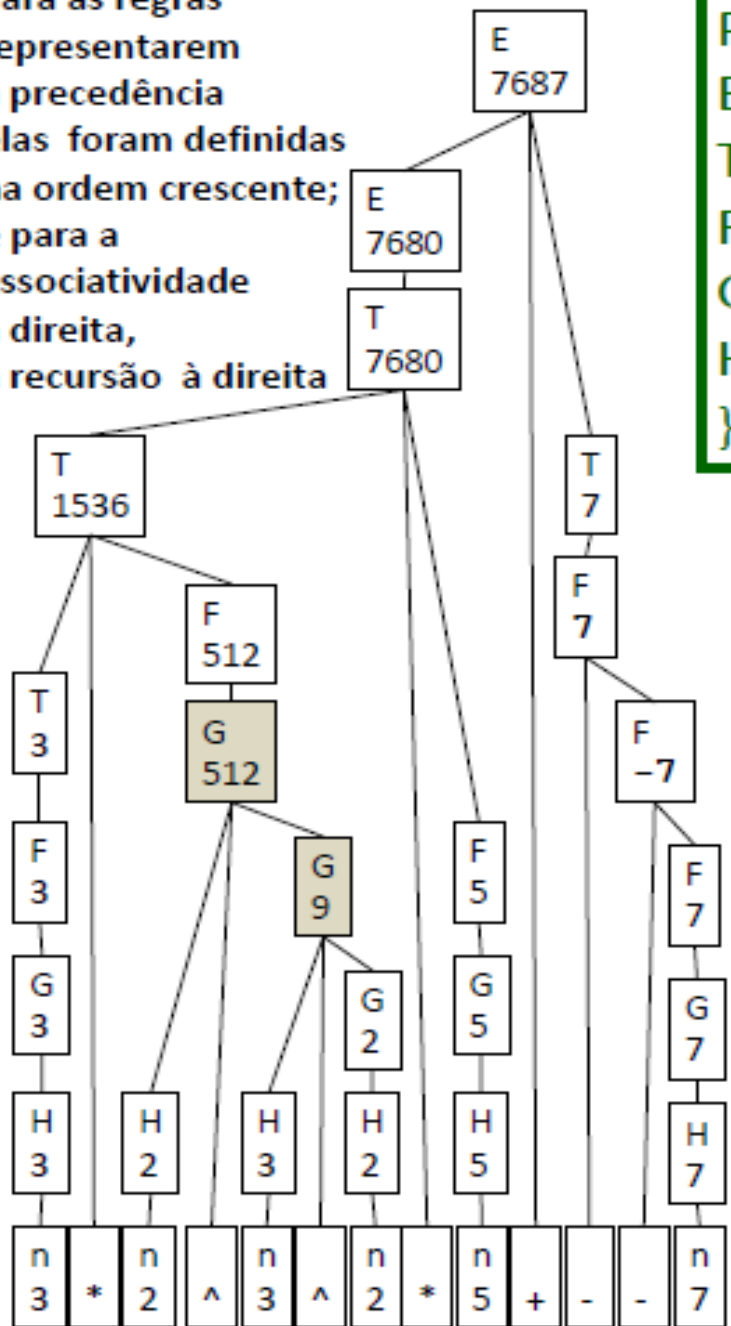
```
/* Grammar follows */
%%
input: /* empty string */
      |input line
      ;

line:  '\n'
|exp '\n'{printf("\t%.10g\n", $1); }
;

exp:NUM {$$ = $1;}
    |exp '+' exp {$$ = $1 + $3;}
    |exp '-' exp {$$ = $1 - $3;}
    |exp '*' exp {$$ = $1 * $3;}
    |exp '/' exp {$$ = $1 / $3;}
    | '-' exp %prec NEG {$$ = -$2;}
    |exp '^' exp {$$ = pow($1, $3);}
    | '(' exp ')' {$$ = $2;}
;
%%
```

Árvore de derivação para a expressão :  $3 * 2 ^ 3 ^ 2 * 5 + - - 7$   
 Para as regras representarem a precedência elas foram definidas na ordem crescente; e para a associatividade à direita, a recursão à direita

$P = \{$   
 $E \rightarrow E + T \mid T$   
 $T \rightarrow T * F \mid F$   
 $F \rightarrow -F \mid G$   
 $G \rightarrow H ^ G \mid H$   
 $H \rightarrow (E) \mid n$   
 $\}$



Observar os seguintes nós G 9 e G 512 resultantes de operação de potência ^ com a maior precedência e associatividade à direita, foram obtidos na seguinte ordem:

1º G 9    2º G 512

**Transformação da gramática para tirar a ambiguidade e definir a precedência e associatividade diretamente pelas regras.**

```
/* Grammar follows */
%%
input: /* empty string */
      |input line
;

line:  '\n'
|exp '\n'{printf("\t%.10g\n", $1); }
;

exp:NUM {$$ = $1;}
    |exp '+' exp {$$ = $1 + $3;}
    |exp '-' exp {$$ = $1 - $3;}
    |exp '*' exp {$$ = $1 * $3;}
    |exp '/' exp {$$ = $1 / $3;}
    |'-' exp %prec NEG {$$ = -$2;}
    |exp '^' exp {$$ = pow($1, $3);}
    |'(' exp ')' {$$ = $2;}
;
%%
```