Análise Descendente (Predictive Parsing)

```
void S() { E(); eat(EOF); }

void E() {
    switch (tok) {
        case ?: E(); eat(PLUS); T(); break;
        case ?: E(); eat(MINUS); T(); break;
        case ?: T(); break;
        default: error(); }
}

void T() {
    switch (tok) {
        case ?: T(); eat(TIMES); F(); break;
        case ?: T(); eat(DIV); F(); break;
        case ?: F(); break;
        default: error(); }
}
```

Funciona ???

$S \rightarrow E\ \$$
$E \rightarrow E + T$
$E \rightarrow E - T$
$E \rightarrow T$
$T \rightarrow T * F$
$T \rightarrow T / F$
$T \rightarrow F$
$F \rightarrow$ id
$F \rightarrow$ num
$F \rightarrow (E)$

Análise Descendente (Predictive Parsing)

$S \rightarrow E\,\$$
$E \rightarrow T\,E'$
$E' \rightarrow +\,T\,E'$
$E' \rightarrow -\,T\,E'$
$E' \rightarrow$
$T \rightarrow F\,T'$
$T' \rightarrow *\,F\,T'$
$T' \rightarrow /\,F\,T'$
$T' \rightarrow$
$F \rightarrow id$
$F \rightarrow num$
$F \rightarrow (E)$

	+	*	id	(	)	$
S			$S \rightarrow E\$$	$S \rightarrow E\$$		
E			$E \rightarrow TE'$	$E \rightarrow TE'$		
E'	$E' \rightarrow +TE'$				$E' \rightarrow$	$E' \rightarrow$
T			$T \rightarrow FT'$	$T \rightarrow FT'$		
T'	$T' \rightarrow$	$T' \rightarrow *FT'$			$T' \rightarrow$	$T' \rightarrow$
F			$F \rightarrow$ id	$F \rightarrow (E)$		

*** Algumas colunas da tabela foram omitidas**

Análise Descendente (Predictive Parsing)

$S \rightarrow E\,\$$
$E \rightarrow T\,E'$
$E' \rightarrow +\,T\,E'$
$E' \rightarrow -\,T\,E'$
$E' \rightarrow$
$T \rightarrow F\,T'$
$T' \rightarrow *\,F\,T'$
$T' \rightarrow /\,F\,T'$
$T' \rightarrow$
$F \rightarrow id$
$F \rightarrow num$
$F \rightarrow (E)$

	+	*	id	(	)	$
S			$S \rightarrow E\$$	$S \rightarrow E\$$		
E			$E \rightarrow TE'$	$E \rightarrow TE'$		
E'	$E' \rightarrow +TE'$				$E' \rightarrow$	$E' \rightarrow$
T			$T \rightarrow FT'$	$T \rightarrow FT'$		
T'	$T' \rightarrow$	$T' \rightarrow *FT'$			$T' \rightarrow$	$T' \rightarrow$
F			$F \rightarrow$ id	$F \rightarrow (E)$		

* Algumas colunas da tabela foram omitidas

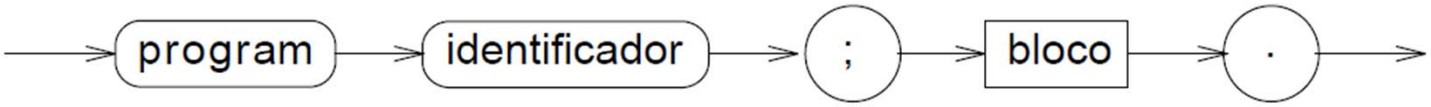
```
void T() {
    switch (tok) {
     case ID:
     case NUM:
     case LPAREN: F(); Tprime(); break;
     default: print("expected id, num, or left-paren");
}}

void Tprime() {
 switch (tok) {
     case PLUS: break;
     case TIMES: eat(TIMES); F(); Tprime(); break;
     case RPAREN: break;
     case EOF: break;
     default: print("expected +, *, right-paren, or end-of-file");
}}
```

PASCAL

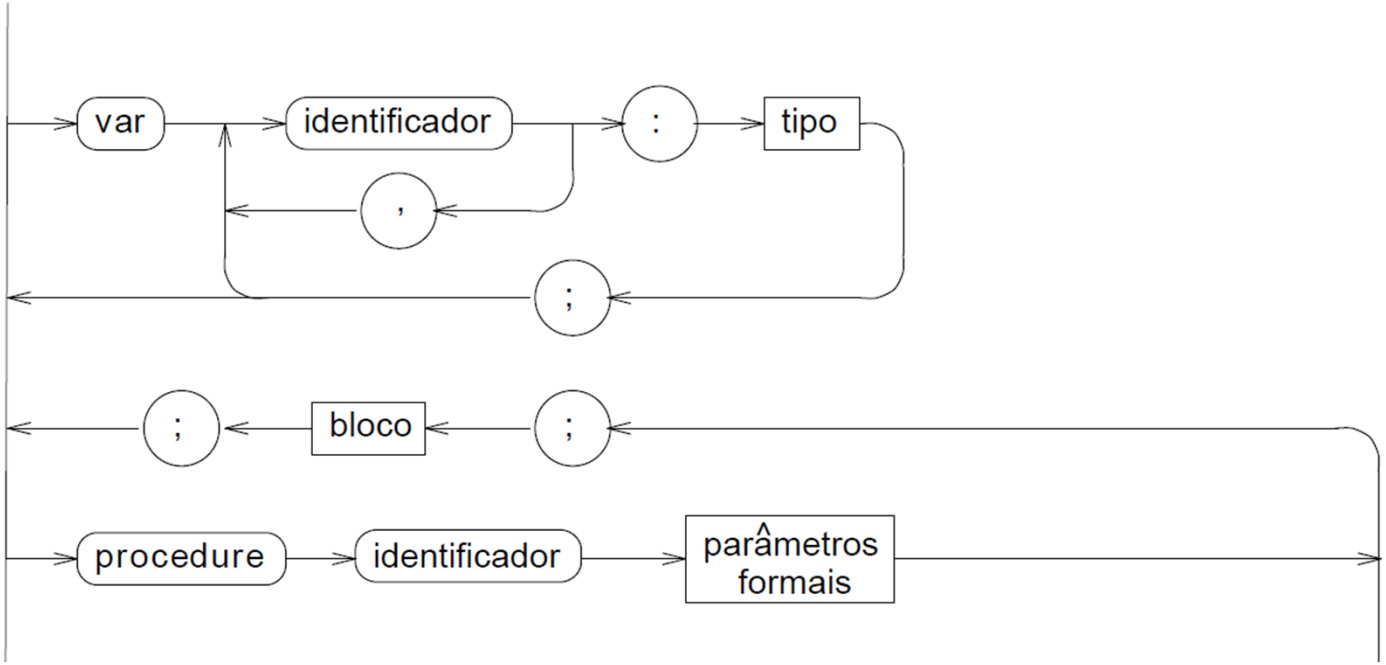
```
program ex;
    var m: integer;
function F(n:integer; var k:integer):integer;
var p,q:integer;
begin
    if n<2 then
    begin
        F:=n;
        k:=0
    end
    else
    begin
        F:=F(n-1,p)+F(n-2,q);
        k:=p+q+1
    end;
    write(n,k)
end
begin
    write(F(3,m),m);
end.
```

PASCAL - Cartas Sintáticas

programa:

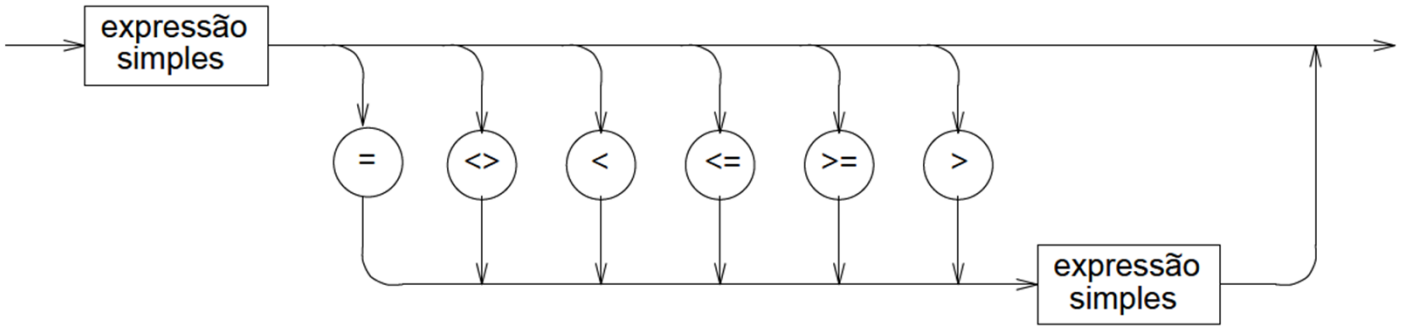


bloco:



PASCAL - Cartas Sintáticas

expressão:




```
expressao → expressao_simples
expressao → expressao =  expressao_simples
expressao → expressao <> expressao_simples
expressao → expressao <  expressao_simples
expressao → expressao <= expressao_simples
expressao → expressao >= expressao_simples
expressao → expressao >  expressao_simples
```