

Relatrio__Construo_de_um_compilador-Paulo_Bittencourt_E_Amadeu_Martim

Professora

- Layse Santos Souza

Grupo

- Amadeu Martim
- Paulo Bittencourt

Instruções

- Baixe o arquivo com o nome ME_Amadeu_Paulo e extraia a pasta contida nele no seu diretório onde está o antlr4
 - Link para download dos arquivos

https://github.com/amadeu100401/analizador_lexico_amadeu_paulo



O arquivo Lex é a parte léxica e o arquivo MeAL é a gramática

- Em seguida abra o terminal dentro da pasta chamada ME
- Com o terminal aberto utilize o seguinte comando:




grun MeAL prog -gui

- Caso deseje gerar os arquivos pelo antlr4 do zero na máquina, siga esse passo a passo.
 1. Crie uma pasta e coloque os arquivos MeAL e Lex
 2. Abra o terminal na pasta e siga essa ordem de comandos:
 - a. antlr4 [nome-do-arquivo].g4

b. `compile [nomedoarquivo]*.java`

ps: substitua [nome-do-arquivo] por MeAL

3. Após rodar esses dois comandos execute o seguinte comando:

 `grun MeAL prog -gui`

Analizador Léxico

Na parte léxica, foram criadas outras regras fora as existentes no documento proposto

CTE → Para números

CADEIA → Para cadeia de letras

IDENTIFIER → Para identificadores com letras e números

BOOLEAN → Para atribuir valores de verdadeiro ou falso

WS → Para ignorar quebra de linha, vazio e afins

COMENTARIO → Para que fosse possível acrescentar comentários sem interferir no código

CTE: `[0-9]+ | '-'[0-9]+ ;`

CADEIA : `""('A'..'Z'|'a'..'z'|'0'..'9'|'-'|'_')+ "";`

IDENTIFIER : `[a-zA-Z][a-zA-Z0-9];`

BOOLEAN : `'TRUE' | 'FALSE';`

OPAD : `'+' | '-';`

OPMULT : `"|'/';`

OPLOG : `'OR'|'AND';`

OPNEG : `'~';`

OPREL : `'<'|'<='|'>'|'>='|'=='|'<>';`

PVIG : `',';`

PONTO : `'.';`

DPONTOS : `',';`

VIG : `',';`

ABPAR : `'(';`

FPAR : `')';`

ATRIB : `':=';`

RESERVADA : `'PROGRAM'|'BEGIN' | 'END' | 'WHILE' | 'DO' | 'READ' | 'VAR' | 'WRITE' | 'IF' | 'ELSE';`

```
WS : (' |\t|\n|\r')+ -> skip;  
COMENTARIO : '/' ~[\r\n]* -> skip;
```

Analizador Sintático

Foram feitos alguns ajustes, referentes aos erros propositais apresentados na gramática.

o expr foi convertido para um expr para operações aritméticas e operações booleanas, onde foram colocadas as regras gramaticais respectivas para a atualização dos operadores e tokens corretamente.

```
MeALg4 X  
C:\> antl4 > ME > MeALg4  
1 grammar MeAL;  
2 import Lex;  
3  
4  
5 prog: 'PROGRAM' IDENTIFIER PVIG decls cmdcomp PONTO | 'PROGRAM' IDENTIFIER PVIG cmdcomp PONTO;  
6 decls: 'VAR' listDecl;  
7 listDecl: declTip | declTip listDecl;  
8 declTip: listId DPONTOS tip PVIG;  
9 listId: IDENTIFIER | IDENTIFIER VIG listId;  
10 tip: 'STRING' | 'INTEGER' | 'BOOLEAN';  
11 cmdComp: 'BEGIN' listCmd 'END';  
12 listCmd: cmd PVIG | cmd PVIG listCmd;  
13 cmd: cmdIf | cmdWhile | cmdRead | cmdWrite | cmdAtrib | cmdAtrib | cmdComp;  
14 cmdIf: 'IF' expr 'THEN' cmd | 'IF' expr 'THEN' cmd 'ELSE' cmd;  
15 cmdWhile: 'WHILE' expr 'DO' cmd | 'WHILE' exprBool 'DO' cmd;  
16 cmdRead: 'READ' ABPAR listId FPAR;  
17 cmdWrite: 'WRITE' ABPAR listW FPAR;  
18 listW: elemW | elemW VIG listW;  
19 elemW: expr | CADEIA;  
20 cmdAtrib: IDENTIFIER ATRIB expr;  
21 expr: CTE OPREL expr | CTE OPAD expr | CTE | CTE OPMULT expr | ABPAR expr FPAR | IDENTIFIER OPREL expr | IDENTIFIER OPAD expr |  
22 IDENTIFIER OPMULT expr | IDENTIFIER OPLOG expr | IDENTIFIER ATRIB expr | OPNEG? IDENTIFIER | OPNEG? ABPAR exprBool FPAR | exprBool | IDENTIFIER | CADEIA;  
23 exprBool: IDENTIFIER | CTE | ABPAR exprBool FPAR | IDENTIFIER OPREL IDENTIFIER |  
24 CTE OPREL CTE | CTE OPREL IDENTIFIER | IDENTIFIER OPREL CTE | IDENTIFIER OPREL BOOLEAN | OPNEG? BOOLEAN;
```

Testes

O código que foi usado para testar a gramática, 2 comandos foram usados para fazer os testes.

```
grun MeAL prog -gui
```

```
grun MeAl prog -tokens
```

Entradas

Teste 01

1. **PROGRAM** TESTE01;
VAR x: **INTEGER**;
BEGIN
WHILE (x < y) **DO** y := **TRUE**;
END.

Teste 02

```
1. PROGRAM TESTE02;  
   VAR varTeste01: STRING;  
   BEGIN  
     WHILE (x < y) DO varTeste01 := AMADEU;  
   END.
```

Teste 03

```
1. PROGRAM TESTE03;  
   VAR x: BOOLEAN;  
   BEGIN  
     IF ( x == TRUE) THEN x := FALSE ELSE x := TRUE ;  
   END.  
  
2. PROGRAM TESTE03;  
   VAR x: BOOLEAN;  
   BEGIN  
     IF (TRUE) DO x := FALSE;  
   END.
```

- **-tokens**

Teste 01

```

[@0,0:6='PROGRAM',<'PROGRAM'>,1:0]
[@1,8:14='TESTE01',<IDENTIFIER>,1:8]
[@2,15:15=';',<'>',1:15]
[@3,20:22='VAR',<'VAR'>,3:0]
[@4,24:24='x',<IDENTIFIER>,3:4]
[@5,25:25=':',<':'>,3:5]
[@6,27:33='INTEGER',<'INTEGER'>,3:7]
[@7,34:34=';',<'>',3:14]
[@8,39:43='BEGIN',<'BEGIN'>,5:0]
[@9,48:52='WHILE',<'WHILE'>,7:0]
[@10,54:54='(',<'('>,7:6]
[@11,55:55='x',<IDENTIFIER>,7:7]
[@12,57:57='<',<OPREL>,7:9]
[@13,59:59='y',<IDENTIFIER>,7:11]
[@14,60:60=')',<')'>,7:12]
[@15,62:63='DO',<'DO'>,7:14]
[@16,65:65='y',<IDENTIFIER>,7:17]
[@17,67:68=':=' ,<':'='>,7:19]
[@18,70:73='TRUE',<IDENTIFIER>,7:22]
[@19,74:74=';',<'>',7:26]
[@20,79:81='END',<'END'>,9:0]
[@21,82:82='.',<'.'>,9:3]
[@22,85:84='<EOF>',<EOF>,10:0]
PS C:\antlr4\ME>

```

Teste 02

```

[@0,0:6='PROGRAM',<'PROGRAM'>,1:0]
[@1,8:14='TESTE02',<IDENTIFIER>,1:8]
[@2,15:15=';',<'>,1:15]
[@3,18:20='VAR',<'VAR'>,2:0]
[@4,22:31='varTeste01',<IDENTIFIER>,2:4]
[@5,32:32=':',<':'>,2:14]
[@6,34:39='STRING',<'STRING'>,2:16]
[@7,40:40=';',<'>,2:22]
[@8,43:47='BEGIN',<'BEGIN'>,3:0]
[@9,50:54='WHILE',<'WHILE'>,4:0]
[@10,56:56='(',<'('>,4:6]
[@11,57:57='x',<IDENTIFIER>,4:7]
[@12,59:59='<',<OPREL>,4:9]
[@13,61:61='y',<IDENTIFIER>,4:11]
[@14,62:62=')',<')'>,4:12]
[@15,64:65='DO',<'DO'>,4:14]
[@16,67:76='varTeste01',<IDENTIFIER>,4:17]
[@17,78:79=':=' ,<':='>,4:28]
[@18,81:86='AMADEU',<IDENTIFIER>,4:31]
[@19,87:87=';',<'>,4:37]
[@20,90:92='END',<'END'>,5:0]
[@21,93:93='.',< '.'>,5:3]
[@22,96:95='<EOF>',<EOF>,6:0]
PS C:\antlr4\ME>

```

Teste 03

```

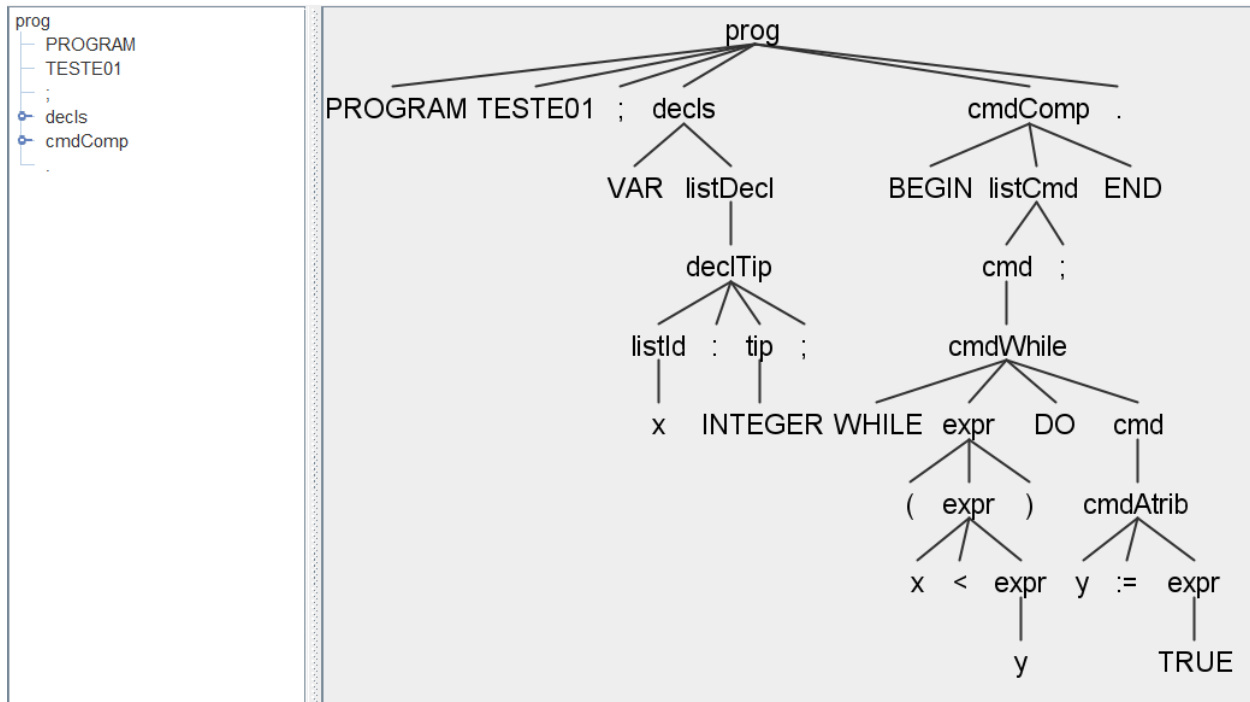
[@0,0:6='PROGRAM',<'PROGRAM'>,1:0]
[@1,8:14='TESTE03',<IDENTIFIER>,1:8]
[@2,15:15=';',<'>',1:15]
[@3,20:22='VAR',<'VAR'>,3:0]
[@4,24:24='x',<IDENTIFIER>,3:4]
[@5,25:25=':',<':'>,3:5]
[@6,27:33='BOOLEAN',<'BOOLEAN'>,3:7]
[@7,34:34=';',<'>',3:14]
[@8,39:43='BEGIN',<'BEGIN'>,5:0]
[@9,48:49='IF',<'IF'>,7:0]
[@10,51:51='(',<'('>,7:3]
[@11,53:53='x',<IDENTIFIER>,7:5]
[@12,55:56='==',<OPREL>,7:7]
[@13,58:61='TRUE',<IDENTIFIER>,7:10]
[@14,62:62=')',<')'>,7:14]
[@15,64:67='THEN',<'THEN'>,7:16]
[@16,69:69='x',<IDENTIFIER>,7:21]
[@17,71:72=':=' ,<':='>,7:23]
[@18,74:78='FALSE',<IDENTIFIER>,7:26]
[@19,80:83='ELSE',<'ELSE'>,7:32]
[@20,85:85='x',<IDENTIFIER>,7:37]
[@21,87:88=':=' ,<':='>,7:39]
[@22,90:93='TRUE',<IDENTIFIER>,7:42]
[@23,95:95=';',<'>,7:47]
[@24,100:102='END',<'END'>,9:0]
[@25,103:103='.',<'.'>,9:3]
[@26,106:105='<EOF>',<EOF>,10:0]

```

- -gui (Árvore)

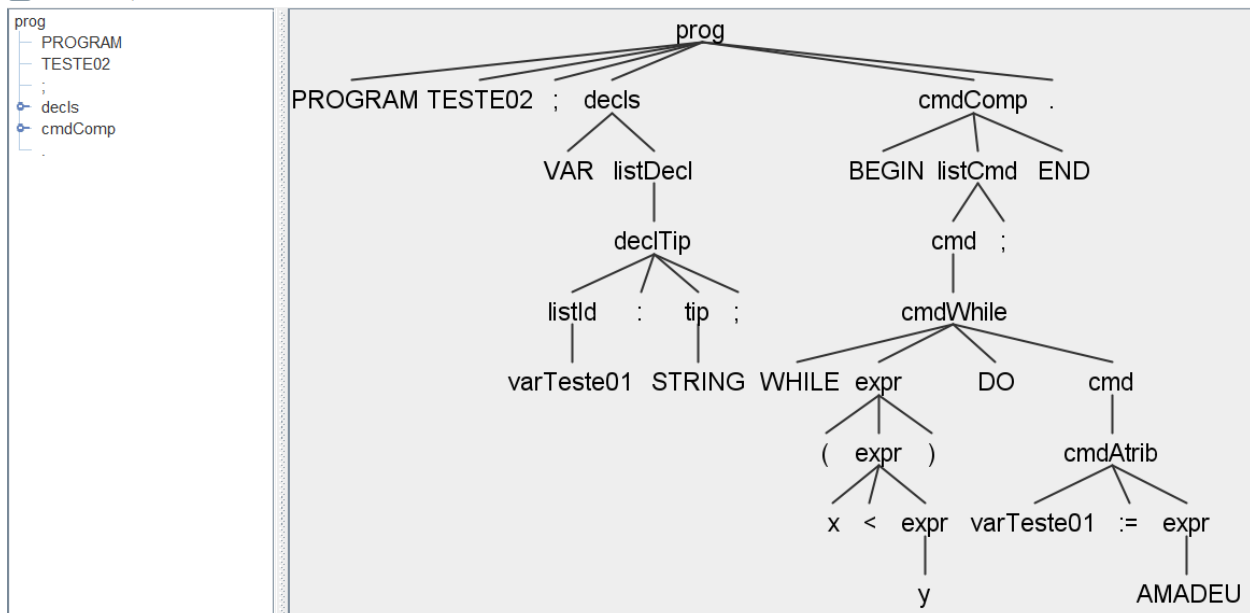
Teste 01

Parse Tree Inspector



Teste 02

Parse Tree Inspector



Teste 03

