



Disciplina: LINGUAGENS, AUTÔMATOS E COMPUTAÇÃO

Unidade de Aprendizagem: UA2 | LINGUAGENS LIVRES DE CONTEXTO



Importante!

O grupo deve listar os nomes de TODOS os participantes. Caso o nome de algum participante do grupo não seja listado, esse estudante não receberá nota.

Grupo de Trabalho

Estudante 1: Bruno Barbosa Andrade

Estudante 2: Bernardo Zamin

Estudante 3: Luca Temes D'andrea

Estudante 4: Otávio fogaça prates da cunha

UA2 | Avaliação de Aprendizagem

Proposta | Construir, de forma cooperativa, uma especificação de gramática livre do contexto para uma linguagem de programação cuja descrição da estrutura foi descrita no módulo de aprendizagem. O sistema a ser utilizado para a descrição da gramática será o ANTLR (disponível gratuitamente em <https://www.antlr.org/>). E os testes de funcionamento da gramática deverão ser realizados através da versão on-line do sistema, nomeado ANTLR Lab (disponível gratuitamente em <http://lab.antlr.org/>).

Registre neste espaço a produção do grupo de trabalho ▼

input:

```
progr test
begin
clear variable;
move 20 to variable;
if(variable)
then
begin
decr variable;
move variable to another;
end
else
incr variable;
while(another) do decr another;
end
```

```
1  progr teste
2  begin
3  clear x;
4  move 100 to x;
5  if(x)
6  then
7  begin
8  decr x;
9  decr variable;
10 move variable to another;
11 end
12 else
13 while(another) do decr another;
14 end
```



com o código:

grammar BareBones;

program: 'progr' V commandblock*EOF;

commandblock: 'begin'(allcommands)*'end';

allcommands: (incrFun|clearFun
|moveFun|decrFun);'ifFun|whileFun;

clearFun:'clear'V;
moveFun:'move'(V|N)'to'V;
decrFun:'decr'V;
incrFun:'incr'V;

ifFun: 'if'ifblock'then'
(commandblock|allcommands)'else'
(commandblock|allcommands);
ifblock:('V');

whileFun:'while'whileblock'do'
(commandblock|allcommands);
whileblock:('V');

V: [a-z]+ ('_' [a-z]+)*;
N: ([1-9][0-9]|'0');
WS: [\t\r\n]+ -> skip;

```
1  grammar BareBones;
2
3  program: 'progr' V commandblock*EOF;
4
5  commandblock: 'begin'(allcommands)*'end';
6
7  allcommands: (incrFun|clearFun
8  |moveFun|decrFun);'ifFun|whileFun;
9
10 clearFun:'clear'V;
11 moveFun:'move'(V|N)'to'V;
12 decrFun:'decr'V;
13 incrFun:'incr'V;
14
15 ifFun: 'if'ifblock'then'
16 (commandblock|allcommands)'else'
17 (commandblock|allcommands);
18 ifblock:('V');
19
20 whileFun:'while'whileblock'do'
21 (commandblock|allcommands);
22 whileblock:('V');
23
24 V: [a-z]+ ('_' [a-z]+)*;
25 N: ([1-9][0-9]|'0');
26 WS: [ \t\r\n]+ -> skip;
```



com o resultado final

Lexer Parser | Sample

```
1 grammar BareBones;
2
3 program: 'progr' V commandblock* EOF;
4
5 commandblock: 'begin' (allcommands)* 'end';
6
7 allcommands: (incrFun|clearFun
8 |moveFun|decrFun);' |ifFun|whileFun;
9
10
11 clearFun: 'clear' V;
12 moveFun: 'move' (V|N) 'to' V;
13 decrFun: 'decr' V;
14 incrFun: 'incr' V;
15
16 ifFun: 'if' ifblock 'then'
17 (commandblock|allcommands) 'else'
18 (commandblock|allcommands);
19 ifblock: '(' V ')';
20
21 whileFun: 'while' whileblock 'do'
22 (commandblock|allcommands);
23 whileblock: '(' V ')';
24
25 V: [a-z] | '(' | ')' | [a-z]*;
26 N: ([1-9] | [0-9]) '0';
27 WS: [ \t\r\n]* -> skip;
```

Input | sample expr

```
1 begin
2 decr x;
3 move x to y;
4 end
5 else
6 incr x;
7 while(y) do decr y;
8 end
```

Start rule

program

Run

Show profiler

Parser console

A:7 extraneous input '0' expecting 'to'

Tree Hierarchy