Compiladores - 2014.2
Prof. Gustavo Carvalho
Projeto - Definição das Gramáticas - Entrega 1.1

Alunos: Arthur Gomes e Juvenal Bisneto
Linguagem: Cobol

Gramática Léxica

Identifier ::= Letter [Letter | Digit]*

Letter ::= [a-z] | [A-Z] | ‘_’

Number ::= [Digit]+ | [‘(‘ [‘-’ | ‘+’]? [Digit]+ ‘)’]

Digit ::= [0 - 9]

Type ::= PIC9 | PICBOOL

BoolValue ::= TRUE | FALSE

OpRelational ::= ‘<=’ | ‘>=’ | ‘<’ | ‘>’ | ‘=’ | ‘<>’

OpAdd ::= ‘+’ | ‘-’

OpMult ::= ‘/’ | ‘*’

WordSeparators ::= ‘\n’ | ‘\t’ | ‘ ’

Comment ::= ‘#’ [Letter | Digit | ‘ ‘ | OpAdd | ApMult | OpRelational]* ‘\n’

Token ::= Identifier | Number | OpRelacional | OpAdd | OpMult | . | (|) | Comment | IF | THEN | ELSE | END-IF | PERFORM | UNTIL | END-PERFORM | WordSeparators | VALUE | PROGRAM | GLOBALDATA | DIVISION | CALL | MAIN | USING | END | DISPLAY | ACCEPT | FROM | COMPUTE | STOP | RUN | RETURN | BREAK | CONTINUE | EOF

Gramática Sintática

Code ::= [GlobalDataDiv]? ProgramDiv
GlobalDataDiv ::= GLOBALDATA DIVISION ‘.’ [VarDeclaration]*
ProgramDiv ::= PROGRAM DIVISION ‘.’ [Function]* MainProc
VarDeclaration ::= [VarPIC9Declaration | VarPICBOOLDeclaration]
VarPIC9Declaration ::= PIC9 Identifier [VALUE Number]? ‘.’
VarPICBOOLDeclaration ::= PICBOOL Identifier [VALUE BoolValue]? ‘.’

MainProc ::= MAIN ‘.’ [Command]* END
Parameter::= Type Identifier

```
Function ::= Identifier [Type | VOID] [USING Parameter [‘,’ Parametro]*]? ‘.’ [VarDeclaration]*
[Command]* END
FunctionCall ::= CALL Identifier [USING Parameter [‘,’ Parameter ]*]? ‘.’

Command ::=  IfStatement | Until | Assignment | Display | FunctionCall | ProcedureCall |
BreakStatement | ContinueStatement | ReturnStatement

Expression ::=  BooleanExpression | [COMPUTE ‘(‘ ArithmeticExpression ‘)’]
BooleanExpression ::= [BooleanParcel OpRelacional BooleanParcel] | BoolValue
BooleanParcel ::= BoolValue | [COMPUTE ‘(‘ ArithmeticExpression ‘)’] | [‘(‘ BooleanExpression ’)’]
ArithmeticExpression ::= Term  [OpAdd ArithmeticExpression]?
Term ::= Factor [OpMult Term]?
Factor ::= Identifier | Number | [‘(‘ ArithmeticExpression ‘)’]

Assignment ::= ACCEPT Identifier FROM [Expression | FunctionCall] ‘.’
IfStatement ::= IF ‘(‘ BooleanExpression ‘)’ [Command]+ [THEN [Command]+]? END-IF
Until ::= PERFORM UNTIL ‘(‘ BooleanExpression ‘)’ [Command]+ END-PERFORM

Display ::= DISPLAY [Identifier | Expression] ‘.’
ReturnStatement ::= RETURN [Identifier | Expression] ‘.’
BreakStatement ::= BREAK ‘.’
ContinueStatement ::= CONTINUE ‘.’
```