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 ::= [0 - 9]

Type ::= PIC9 | PICBOOL

BoolValue ::= TRUE | FALSE

OpRelational ::= '<=' | '>=' | '<' | '>' | '=' | '<>'

OpAdd ::= '+' | '-'

OpMult ::= '/' | '*'

WordSeparators ::= '\n' | '\t' | ' '

Comentario ::= '#' [Letter | Digit | ' ']* '\n'

Token ::= VOID | Identifier | Number | OpRelacional | OpAdd | OpMult | . | (|) | ' | # | 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

Gramática Sintática

Code ::= [GlobalDataDiv]? ProgramDiv eot
GlobalDataDiv ::= GLOBALDATA DIVISION '.' [VarDeclaration]*
ProgramDiv ::= PROGRAM DIVISION '.' MainProc [Procedure | Function]*

VarDeclaration ::= [VarPIC9Declaration | VarPICBOOLDeclaration]
VarPIC9Declaration ::= PIC9 Identifier [VALUE Number]? '.'
VarPICBOOLDeclaration ::= PICBOOL Identifier [VALUE BoolValue]? '.'

MainProc ::= MAIN '.' [Command]* END
Parametro ::= Type Identifier

Function ::= Identifier [Type | VOID] [USING Parametro [‘,’ Parametro]*]? ‘.’ [Command]* END
FunctionCall ::= CALL [Type | VOID] Identifier [Using]? ‘.’
Procedure ::= Identifier [USING Parametro [‘,’ Parametro]*]? ‘.’ [Command]* END
ProcedureCall ::= CALL Identifier [Using]? ‘.’

Command ::= [VarDeclaration | If_Statement | Until | Assignment | Display | ReturnStatement | FunctionCall | ProcedureCall | BreakStatement | ContinueStatement | StopRun]

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

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

Display ::= DISPLAY Expression ‘.’
ReturnStatement ::= RETURN Expression ‘.’
BreakStatement ::= BREAK ‘.’
ContinueStatement ::= CONTINUE ‘.’
StopRun ::= STOP RUN ‘.’