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 | '.' | ',' | '(' | ')' | IF | THEN |
ELSE | ENDIF | PERFORM | UNTIL | ENDPERFORM | VALUE | PROGRAM |
GLOBALDATA | DIVISION | VOID | CALL | MAIN | USING | ENDMAIN | ENDFUNCTION |
DISPLAY | ACCEPT | FROM | COMPUTE | 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 '.' [VarDeclaration]* [Command]* ENDMAIN
Function ::= Identifier [Type | VOID] [USING Type Identifier [',' Type Identifier]*]? '.'
    [VarDeclaration]* [Command]* ENDFUNCTION
FunctionCall ::= CALL Identifier [USING Identifier [',' Identifier]*]? '.'

Command ::= IfStatement | Until | Accept | Display | FunctionCall | BreakStatement |
    ContinueStatement | ReturnStatement

Expression ::= BooleanExpression | ArithmeticExpression
BooleanExpression ::= ['(' [Expression | Identifier] OpRelational [Expression | Identifier] ')'] |
    BoolValue
ArithmeticExpression ::= COMPUTE '(' ArithmeticParcel ')' | Number
ArithmeticParcel ::= ArithmeticTerm [OpAdd ArithmeticParcel]?
ArithmeticTerm ::= ArithmeticFactor [OpMult ArithmeticTerm]?
ArithmeticFactor ::= Identifier | Number | ['(' ArithmeticParcel ')']

Accept ::= ACCEPT Identifier FROM [Expression | FunctionCall | Identifier] '.'
IfStatement ::= IF BooleanExpression THEN [Command]+ [ELSE [Command]+]? ENDIF
Until ::= PERFORM UNTIL BooleanExpression '.' [Command]+ ENDPERFORM

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