Entrega 2.1
Compiladores - 2013.1 - 15/05/2013
Linguagem: BCPL
Grupo: Clodomir Santana, Sabrina Andrade

Gramática léxica BCPL

Identifier ::= Letter [Letter | Digit]*

Letter ::= [a-z] | [A-Z] | '_'

Type ::= INT | BOOL

Number ::= [Digit]+

Digit ::= [0 – 9]

BoolValue ::= TRUE | FALSE

Op_Relacional ::= '<=' | '>=' | '<' | '>'

Op_Comparativo ::= '==' | '~='

Op_Add ::= '+' | '-'

Op_Mult ::= '/' | '*'

ws ::= '\n' | '\t' | ' ' | '#'

Comentario ::= '#' [Identifier]* ['\n' | eot]

Token ::= Identifier | Number | Op_Relacional | Op_Aritmetico | { | } | ; | = | (|) | IF | DO | WHILE | TEST | THEN | ELSE | RETURN | BREAK | # | CONTINUE | LET | BE | GLOBAL | TRUE | FALSE | eot | INT | BOOL | writef | VOID | [|] | PROC | FUNC

Gramática sintática BCPL

Program ::= [Command]* [Procedure]* [Function]*

Procedure ::= PROC Type Identifier ‘(‘ [Parametro (‘,’ Parametro)*]? ‘)’ ‘:=’ ‘{‘ [Command]+ ‘}’

Function ::= FUNC (Type | VOID) Identifier ‘(‘ [Parametro (‘,’ Parametro)*]? ‘)’ BE ‘{‘ [Command]+ ‘}’

Expression ::= [BooleanExpression | ArithmeticExpression]

BooleanExpression ::= ComparacaoNumeros | ComparacaoBooleana

ValorBooleano ::= TRUE | FALSE

ComparacaoNumeros ::= ‘(‘ ArithmeticExpression ‘)’ Op_Relacional ‘(‘ ArithmeticExpression ‘)’

ComparacaoBooleana ::= [‘[‘ArithmeticExpression | ValorBooleano] Op_Comparativo [ValorBooleano | ArithmeticExpression ‘]’]) | ValorBooleano

ArithmeticExpression ::= Term [Op_Add ArithmeticExpression]?

Term ::= Factor [Op_Mult Term]?

Factor ::= Identifier | Number | [‘(‘ ArithmeticExpression ‘)’]

ExpressaoUnaria ::= Op_Add Number

Command ::= [VarDeclaration | If_Statement | Test_Statement | While | AssignmentCommand | Writef | ReturnCommand | Function_Call | Break_Statement | Continue_Statement]

AssignmentCommand ::= Identifier ‘:=’ [Expression | Function_Call] ‘;’

If_Statement ::= IF ‘(‘ BooleanExpression ‘)’ DO ‘{‘ [Command]+ ‘}’

Test_Statement ::= TEST ‘(‘ BooleanExpression ‘)’ THEN ‘{‘ [Command]+ ‘}’ ELSE ‘{‘ [Command]+ ‘}’

While ::= WHILE ’(‘ BooleanExpression ‘)’ DO ‘{‘ [Command]+ ‘}’

VarDeclaration ::= [GLOBAL | LET] [Type] [Identifier | ‘(‘ AssignmentCommand ‘)’]

ReturnCommand ::= RETURN Expression ‘;’

Function_Call ::= FUNC Identifier ‘(‘ [Expression [‘,’ Expression]*]? ‘)’

Procedure_Call ::= PROC Identifier (‘ [Expression [‘,’ Expression]*]?‘)’

Parametro ::= Type Identifier

Writef ::= writef ‘(‘ Expression ‘)’

Break_Statement ::= BREAK;

Continue_Statement ::= CONTINUE ;

Obs.: BCPL possui como separador de comandos a quebra de linha. Para facilitar a implementação, será considerado o ponto e vírgula (;) como separador da linguagem, na gramática.