**Análise semântica e tabela de símbolos da Tynecode**

**Aluno: Dericson Pablo**

1) O que é a tabela de símbolos? Para que ela é usada?

2) O que é uma ação semântica em uma gramática?

3) Descreva as ações ligadas a cada produção da gramática do arquivo Tynecode1.atg

import java.util.\*;

COMPILER TinyCode

class IntVar {                       // variavel inteira

   public int lBound, uBound, iValue;

   public IntVar(int lb, int ub, int v) {

      lBound = lb;

      uBound = ub;

      iValue = v;

   }

}

class BoolVar {           // variavel logica

   public boolean bValue;

   public BoolVar(boolean v) {

      bValue = v;

   }

}

//tabela de variaveis logicas

Dictionary lbooleans = new java.util.Hashtable();

//tabela de variaveis inteiras

Dictionary lints = new java.util.Hashtable();

//lista de comandos

Vector lprogram = new Vector();

//tabela de rotulos

Dictionary llabels = new java.util.Hashtable();

int statementNo = 0; //contador do programa

java.lang.String cmd ; //comando atual

java.lang.String varName;  //variavel sendo processada

java.lang.String progName; //programa sendo processado

CHARACTERS

    letter = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz".

    digit = "0123456789".

TOKENS

    Identifier = letter {letter | digit}.

    Number = ['-'] digit {digit}.

    true = "true".

    false = "false".

COMMENTS FROM "//" TO '\n'

IGNORE '\r' + '\n' + '\t'

PRODUCTIONS

    TinyCode = "program" Identifier (. progName = t.val; .)

               "." ProgramBody.

    ProgramBody = ["var" DeclareVar { DeclareVar }] "begin" { Statement } "end." (. lprogram.add("end"); .)

                  .

    DeclareVar = Identifier (. varName = t.val; .)

                 ":" DeclareVarBody.

    DeclareVarBody = "integer"

                     "(" Number            (. int lbound = (Integer.valueOf(t.val)).intValue(); .)

                     ".." Number           (. int ubound = (Integer.valueOf(t.val)).intValue(); .)

                     ")" ":=" Number       (. int cvalue = (Integer.valueOf(t.val)).intValue(); .)

                     ";"                   (. lints.put(varName,new IntVar(lbound,ubound,cvalue)); .)

                 |

                     "boolean" ":=" (true | false) (. boolean bvalue = Boolean.parseBoolean(t.val); .)

                       ";".

    Statement = [Identifier                (. llabels.put(t.val,statementNo); .)

                ":"] StatementBody.

    StatementBody = Assignment | Branching | Goto.

    Goto = "goto" Identifier               (. lprogram.add("goto " + t.val);

                                            statementNo++; .)

           ";".

    Assignment = "let" LValue              (. cmd = "asgn " + t.val + " "; .)

                 ":=" (RValue              (. cmd = cmd + t.val; .)

                 [OP                       (. cmd = cmd + " " + t.val; .)

                 RValue                    (. cmd = cmd + " " + t.val; .)

                 ]

             |

                 "not"                     (. cmd = cmd + "not"; .)

                 RValue                    (. cmd = cmd + t.val; .)

                 ) ";"                    (. lprogram.add(cmd); statementNo++; .)

                 .

    LValue = Identifier.

    RValue = Identifier | Number | true | false.

    OP = '+' | '-' | "and" | "or".

    Branching = "if"                       (. cmd = "if "; .)

                 CompareExpr

                 "goto"                    (. cmd = cmd + "goto"; .)

                 Identifier                (. cmd = cmd + t.val; .)

                 ";"

                 ["else" "goto"            (. cmd = cmd + "elsegoto"; .)

                 Identifier                (. cmd = cmd + t.val; .)

                 ";"]                     (. lprogram.add(cmd); statementNo++; .)

                 .

    CompareExpr = RValue                   (. cmd = cmd + t.val; .)

                  CompareOp                (. cmd = cmd + " " + t.val + " "; .)

                  RValue                   (. lprogram.add(cmd); statementNo++; .)

                  .

    CompareOp = '=' | "<>" | '<' | "<=" | '>' | ">=".

END TinyCode.

3) Crie uma função listAllVars() no arquivo atg. Esta ação deve listar todas as variáveis declaradas no programa. Insira a seguinte ação em uma produção adequada (. listAllVars(); .) .

4) Gere o scanner e o parser do arquivo atg acima e integre-os.

5) Compile o programa CountSum.in

RESPOSTAS:

1. **Tabela de símbolos** é uma estrutura de dados, geralmente uma árvore ou tabela de hash, utilizada em compiladores para o armazenamento de informações de identificadores, tais como constantes, funções, variáveis e tipos de dados. É utilizada em quase todas as fases de compilação, como a varredura, a análise sintática, a análise semântica, otimização e geração de código. Em cada fase ela pode ser utilizada como base para comparações ou mesmo atualizada com novos identificadores durante a saída de cada fase.