UNIVERSIDADE FEDERAL DE ALAGOAS
INSTITUTO DE COMPUTAÇÃO
COMP263 - Compiladores

LINGUAGEM DUMA: GRAMÁTICA E ANALISADOR SINTÁTICO

Eduarda Tatiane Caetano Chagas
Marcos Gleysson Silva do Nascimento

Maceió-AL
2017

Sumário

1 Gramática da Linguagem **3**

1.1 Especificação do analisador sintático 3

1.2 Gramática 3

1.3 Gramática LL(1) 5

2 Analisador Sintático **8**

2.1 Organização do projeto 8

2.2 Classes do pacote *grammar* 8

2.2.1 Symbol.java 8

2.2.2 Terminal.java 9

2.2.3 NonTerminalName.java 10

2.2.4 NonTerminal.java 11

2.2.5 Derivation.java 11

2.2.6 Grammar.java 13

2.3 Classes do pacote *syntacticAnalyzer* 16

2.3.1 PredictiveTable.java 16

2.3.2 PredictiveAnalyzer.java 19

2.3.3 SyntacticAnalyzer.java 22

2.4 Saída do Analisador para os códigos do trabalho anterior 23

2.4.1 HelloWorld.duma 23

2.4.2 Fibonacci.duma 24

2.4.3 ShellSort.duma 34

CAPÍTULO 1

GRAMÁTICA DA LINGUAGEM

1.1 Especificação do analisador sintático

Foi implementado um Analisador Sintático do tipo descendente **Preditivo Tabular**. Portanto, nas seções seguintes encontram-se, respectivamente, a gramática da linguagem Duma e a gramática na especificação LL(1), com as restrições necessárias, adequada para o tipo do analisador escolhido.

1.2 Gramática

PROGRAM = (1) ‘prDuma’ ‘id’ BEGIN

BEGIN = (2) FUNCTIONS INITIUM

FUNCTIONS = (3) RETURNTYPE ‘id’ PARAMS SCOPE FUNCTIONS | (4) ε

INITIUM = (5) ‘prInitials’ ‘tdInanis’ ‘prInitium’ ‘paramBegin’ ‘paramEnd’ SCOPE

PARAMS = (6) ‘paramBegin’ | (7) ‘paramEnd’ | (8) LISTPARAMS ‘paramEnd’

LISTPARAMS = (9) TYPE NAME ‘sepVirg’ LISTPARAMS | (10) TYPE NAME | (11) ‘prMatrix’ TYPE NAME | (12) ‘prMatrix’ TYPE NAME ‘sepVirg’ LISTPARAMS

RETURNTYPE = (13) TYPE | (14) ‘prMatrix’ TYPE

TYPE = (15) ‘tdInanis’ | (16) ‘tdInt’ | (17) ‘tdReal’ | (18) ‘tdLit’ | (19) ‘tdBool’ | (20) ‘tdSermo’

CONSTANT = (21) ‘cteNumInt’ | (22) ‘cteNumReal’ | (23) ‘cteLit’ | (24) ‘cteSermo’ | (25) ‘cteBool’

NAME = (26) ‘id’ ‘vetBegin’ Eb ‘vetEnd’ | (27) ‘id’

SCOPE = (28) ‘escBegin’ COMMANDS ‘escEnd’ ‘termCmd’

COMMANDS = (29) CMD ‘termCmd’ COMMANDS | (30) ε

CMD = (31) DECLARATION | (32) ‘id’ ATTRIBUTION | (33) ‘id’ FUNCCALL | (34) WRITE | (35) READ | (36) IFELSE | (37) WHILE | (38) DOWHILE | (39) FOR | (40) RETURN

DECLARATION = (41) TYPE NAME | (42) ‘prMatrix’ TYPE NAME

ATTRIBUTION = (43) ‘vetBegin’ Eb ‘vetEnd’ ‘instAtrib’ VALUE

VALUE = (44) ARRAY | (45) Eb

ARRAY = (46) ‘vetBegin’ ELEMENTS ‘vetEnd’ | (47) ‘vetBegin’ ‘vetEnd’

ELEMENTS = (48) CONSTANT ‘sepVirg’ ELEMENTS | (49) CONSTANT

FUNCCALL = (50) ‘paramBegin’ LISTPARAMSCALL ‘paramEnd’

LISTPARAMSCALL = (51) ITEMPARAM ‘sepVirg’ LISTPARAMSCALL | (52) ITEMPARAM

ITEMPARAM = (53) CONSTANT | (54) NAME

WRITE = (55) ‘prScribo’ ‘paramBegin’ MESSAGE ‘paramEnd’

MESSAGE = (56) ‘cteSermo’ | (57) ‘cteSermo’ ‘opCon’ MESSAGE | (58) NAME | (59) NAME ‘opCon’ MESSAGE

READ = (60) ‘prLectio’ ‘paramBegin’ NAME ‘paramEnd’

IFELSE = (61) IF ELSEIF ELSE

IF = (62) ‘selSi’ ‘paramBegin’ A ‘paramEnd’ ‘escBegin’ COMMANDS ‘escEnd’

ELSEIF = (63) ‘selSialiud’ ‘paramBegin’ Eb ‘paramEnd’ ‘escBegin’ COMMANDS ‘escEnd’ ELSEIF | (64) ε

ELSE = (65) ‘selAliud’ ‘escBegin’ COMMANDS ‘escEnd’ | (66) ε

WHILE = (67) ‘repDum’ ‘paramBegin’ A ‘paramEnd’ ‘escBegin’ COMMANDS ‘escEnd’

DOWHILE = (68) ‘repFacite’ ‘escBegin’ COMMANDS ‘escEnd’ ‘repDum’ ‘paramBegin’ Eb ‘paramEnd’

FOR = (69) ‘repQuia’ ‘id’ ‘repIn’ ‘repSpatium’ ‘paramBegin’ ITEMPARAM ‘sepVirg’ ITEMPARAM ‘sepVirg’ ITEMPARAM ‘paramEnd’ ‘escBegin’ COMMANDS ‘escEnd’

RETURN = (70)‘prReditus’ CONSTANT | (71) ‘prReditus’ NAME

Eb = (72) Eb ‘opLogOr’ Tb | (73) Tb

Tb = (74) Tb ‘opLogAnd’ Fb | (75) Fb

Fb = (76) Fb ‘opRel2’ Erel2 | (77) Erel2

Erel2 = (78) Erel2 ‘opRel1’ Erel1 | (79) Erel1

Erel1 = (80) Erel1 ‘opAritAd’ Ea | (81) Ea

Ea = (82) Ea ‘opAritMul’ Ta | (83) Ta

Ta = (84) ‘opAritUn’ Ta | (85) Fa

Ta = (86) ‘opLogNeg’ Ta | (87) Fa

Fa = (88) ‘paramBegin’ Eb ‘paramEnd’

Fa = (89) CONSTANT

Fa = (90) ’id’

Fa = (91) ‘id’ ‘vetBegin’ A ‘vetEnd’

1.3 Gramática LL(1)

PROGRAM = (1) ‘prDuma’ ‘id’ BEGIN

BEGIN = (2) FUNCTIONS INITIUM

FUNCTIONS = (3) RETURNTYPE ‘id’ PARAMS SCOPE FUNCTIONS | (4) ε

INITIUM = (5) ‘prInitials’ ‘tdInanis’ ‘prInitium’ ‘paramBegin’ ‘paramEnd’ SCOPE

PARAMS = (6) ‘paramBegin’ PARAMSFAT

PARAMSFAT = (7) ‘paramEnd’ | (8) LISTPARAMS ‘paramEnd’

LISTPARAMS = (9) TYPE NAME LISTPARAMSFAT | (10) ‘prMatrix’ TYPE NAME LISTPARAMSFAT

LISTPARAMSFAT = (11) ‘sepVirg’ LISTPARAMS | (12) ε

RETURNTYPE = (13) TYPE | (14) ‘prMatrix’ TYPE

TYPE = (15) ‘tdInanis’ | (16) ‘tdInt’ | (17) ‘tdReal’ | (18) ‘tdLit’ | (19) ‘tdBool’ | (20) ‘tdSermo’

CONSTANT = (21) ‘cteNumInt’ | (22) ‘cteNumReal’ | (23) ‘cteLit’ | (24) ‘cteSermo’ | (25) ‘cteBool’

NAME = (26) ‘id’ NAMEFAT

NAMEFAT = (27) ‘vetBegin’ Eb ‘vetEnd’ | (28) ε

SCOPE = (29) ‘escBegin’ COMMANDS ‘escEnd’ ‘termCmd’

COMMANDS = (30) CMD ‘termCmd’ COMMANDS | (31) ε

CMD = (32) DECLARATION | (33) ‘id’ CMDFAT | (34) WRITE | (35) READ | (36) IFELSE | (37) WHILE | (38) DOWHILE | (39) FOR | (40) RETURN

DECLARATION = (41) TYPE NAME | (42) ‘prMatrix’ TYPE NAME

CMDFAT = (43) ATTRIBUTION | (44) FUNCCALL

ATTRIBUTION = (45) NAMEFAT ‘instAtrib’ VALUE

VALUE = (46) ARRAY | (47) Eb

ARRAY = (48) ‘vetBegin’ ARRAYFAT

ARRAYFAT = (49) ELEMENTS ‘vetEnd’ | (50) ‘vetEnd’

ELEMENTS = (51) CONSTANT ELEMENTSFAT

ELEMENTSFAT = (52) ‘sepVirg’ ELEMENTS | (53) ε

FUNCCALL = (54) ‘paramBegin’ LISTPARAMSCALL ‘paramEnd’

LISTPARAMSCALL = (55) ITEMPARAM LISTPARAMSCALLFAT

LISTPARAMSCALLFAT = (56) ‘sepVirg’ LISTPARAMSCALL | (57) ε

ITEMPARAM = (58) CONSTANT | (59) NAME

WRITE = (60) ‘prScribo’ ‘paramBegin’ MESSAGE ‘paramEnd’

MESSAGE = (61) ‘cteSermo’ MESSAGEFAT | (62) NAME MESSAGEFAT

MESSAGEFAT = (63) ‘opCon’ MESSAGE | (64) ε

READ = (65) ‘prLectio’ ‘paramBegin’ NAME ‘paramEnd’

IFELSE = (66) IF ELSEIF ELSE

IF = (67) ‘selSi’ ‘paramBegin’ Eb ‘paramEnd’ SCOPE

ELSEIF = (68) ‘selSialiud’ ‘paramBegin’ Eb ‘paramEnd’ ‘escBegin’ COMMANDS ‘escEnd’ ELSEIF | (69) ε

ELSE = (70) ‘selAliud’ ‘escBegin’ COMMANDS ‘escEnd’| (71) ε

WHILE = (72) ‘repDum’ ‘paramBegin’ Eb ‘paramEnd’ ‘escBegin’ COMMANDS ‘escEnd’

DOWHILE = (73) ‘repFacite’ ‘escBegin’ COMMANDS ‘escEnd’ ‘repDum’ ‘paramBegin’ Eb ‘paramEnd’

FOR = (74) ‘repQuia’ ‘id’ ‘repIn’ ‘repSpatium’ ‘paramBegin’ ITEMPARAM ‘sepVirg’ ITEMPARAM ‘sepVirg’ ITEMPARAM ‘paramEnd’ ‘escBegin’ COMMANDS ‘escEnd’

RETURN = (75) ‘prReditus’ RETURNFAT

RETURNFAT = (76) CONSTANT | (77) NAME

Eb = (78) Tb Ebr

Ebr = (79) ’opLogOr’ Tb Ebr | (80) ε

Tb = (81) Fb Tbr

Tbr = (82) ’opLogAnd’ Fb Tbr | (83) ε

Fb = (84) Erel2 Fbr

Fbr = (85) ’opRel2’ Erel2 Fbr | (86) ε

Erel2 = (87) Erel1 Erel2r

Erel2r = (88) ’opRel1’ Erel1 Erel2r | (89) ε

Erel1 = (90) Ea Erel1r

Erel1r =(91) ’opAritAd’ Ea Erel1r | (92) ε

Ea = (93) Ta Ear

Ear = (94) ’opAritMul’ Ta Ear | (95) ε

Ta = (96) ‘opAritUn’ Ta | (97) ‘opLogNeg’ Ta | (98) Fa

Fa = (99) ‘paramBegin’ Eb ‘paramEnd’

Fa = (100) CONSTANT

Fa = (101) ’id’ Far

Far = (102) ‘vetBegin’ Eb ‘vetEnd’ | (103) ε

CAPÍTULO 2

ANALISADOR SINTÁTICO

2.1 Organização do projeto

A implementação do Analisador Sintático, contida em um pacote Java chamado *syntacticAnalyzer*, foi organizada em 9 classes, em que 3 delas estão presentes na raíz do pacote e as demais classes referentes a implementação da Gramática estão presentes no pacote *grammar* dentro do pacote *syntacticAnalyzer*.

Nas seções abaixo encontram-se os códigos de cada uma das classes implementadas.

2.2 Classes do pacote *grammar*

2.2.1 Symbol.java

```
package syntacticAnalyzer.grammar;

public class Symbol {
        private Boolean isTerminal;
        private Integer value;

        public Symbol(Boolean isTerminal, Integer value) {
                this.isTerminal = isTerminal;
                this.value = value;
        }

        public boolean isTerminal() {
                return isTerminal;
        }

        public void setTerminal(Boolean isTerminal) {
                this.isTerminal = isTerminal;
        }

        public int getValue() {
                return value;
```

```
        }

        public void setValue(Integer value) {
                this.value = value;
        }
}
```

2.2.2 Terminal.java

```
package syntacticAnalyzer.grammar;

import lexicalAnalyzer.Token;
import lexicalAnalyzer.TokenCategory;

public class Terminal extends Symbol {

        private TokenCategory category;
        private Token token;

        public Terminal(TokenCategory category) {
                super(true, category.getCategoryValue());
                this.category = category;
                token = null;
        }

        public Terminal(TokenCategory category, String value) {
                super(true, category.getCategoryValue());
                this.category = category;
                token = new Token();

        }

        public Terminal(Token token) {
                super(true, token.getCategory().getCategoryValue());
                this.token = token;
                this.category = token.getCategory();
        }

        public TokenCategory getCategory() {
                return category;
        }

        public String getTerminalValue() {
                return token.getValue();
```

```
        }

        public void setTerminalValue(String value) {
                token.setValue(value);
        }

        public Token getTerminalToken() {
                return token;
        }

}
```

2.2.3 NonTerminalName.java

```
package syntacticAnalyzer.grammar;

public enum NonTerminalName {


        PROGRAM(1), BEGIN(2), INITIUM(3), SCOPE(4), FUNCTIONS(5),
        RETURNTYPE(6), PARAMS(7), PARAMSFAT(8), LISTPARAMS(9),
        TYPE(10), NAME(11), LISTPARAMSFAT(12), CONSTANT(13),
        NAMEFAT(14), COMMANDS(15), CMD(16), DECLARATION(17),
        CMDFAT(18), WRITE(19), READ(20), IFELSE(21), WHILE(22),
        DOWHILE(23), FOR(24), RETURN(25), ATTRIBUTION(26),
        FUNCCALL(27),VALUE(28), ARRAY(29), Eb(30), ARRAYFAT(31),
        ELEMENTS(32), ELEMENTSFAT(33), LISTPARAMSCALL(34),
        LISTPARAMSCALLFAT(35),ITEMPARAM(36), MESSAGE(37),
        MESSAGEFAT(38), IF(39), ELSEIF(40),ELSE(41),
        RETURNFAT(42), Ebr(43), Tb(44), Tbr(45), Fb(46),
        Fbr(47), Erel1(48), Erel1r(49), Erel2(50), Erel2r(51),
        Ea(52) , Ear(53), Ta(54), Fa(55), Far(56);

        private int nonTerminalValue;

        private NonTerminalName(int value) {
                this.nonTerminalValue = value;
        }

        public int getNonTerminalValue() {
                return nonTerminalValue;
        }
}
```

2.2.4 NonTerminal.java

```
package syntacticAnalyzer.grammar;

import syntacticAnalyzer.grammar.NonTerminalName;

public class NonTerminal extends Symbol {

        private NonTerminalName name;
        private String tipo;
        private int index;

        public NonTerminal(NonTerminalName name) {
                super(false, name.getNonTerminalValue());
                this.name = name;
        }

        public NonTerminalName getName() {
                return name;
        }

        public int getIndex() {
                return index;
        }

        public void setIndex(int index) {
                this.index = index;
        }

        public String getTipo() {
                return tipo;
        }

        public void setTipo(String tipo) {
                this.tipo = tipo;
        }
}
```

2.2.5 Derivation.java

Esta classe é responsável por fornecer a estrutura de métodos *arraylist* necessária para armazenar os símbolos presentes nas derivações da gramática da linguagem Duma.

O código da classe segue um mesmo padrão e se mostra bastante repetitivo, sendo assim, foi

posto aqui um resumo do mesmo.

```
package syntacticAnalyzer.grammar;

import java.util.ArrayList;
import syntacticAnalyzer.grammar.Symbol;

public class Derivation {

        private ArrayList<Symbol> symbolsList;

        public ArrayList<Symbol> getSymbolsList() {
                return symbolsList;
        }

        public Derivation() {
                symbolsList = new ArrayList<Symbol>();
        }

        public void clearDerivationList() {
                symbolsList.clear();
        }

        public void addSymbol(Symbol symb1) {
                symbolsList.add(symb1);
        }

        public void addDerivationSymbols(Symbol symb1,Symbol symb2){
                addSymbol(symb1);
                symbolsList.add(symb2);

        }

        public void addDerivationSymbols(Symbol symb1, Symbol symb2,
        Symbol symb3){
                addDerivationSymbols(symb1, symb2);
                symbolsList.add(symb3);
        }

        public void addDerivationSymbols(Symbol symb1,
    Symbol symb2, Symbol symb3, Symbol symb4) {
                addDerivationSymbols(symb1, symb2, symb3);
                symbolsList.add(symb4);
        }
```

```
        // ...

        public boolean isEmpty() {
                return symbolsList.isEmpty();
        }
}
```

2.2.6 Grammar.java

Esta classe é responsável por armazenar as derivações presentes na gramática da linguagem Duma, inserindo cada símbolo (Terminal e/ou Não-Terminal) de cada derivação em um *arraylist*. Essa inserção se deu por meio do uso de um método fornecido pela classe *Derivation.java*.

O código da classe segue um mesmo padrão e se mostra bastante repetitivo, sendo assim, foi posto aqui um resumo do mesmo.

```
package syntacticAnalyzer.grammar;

import java.util.ArrayList;
import lexicalAnalyzer.TokenCategory;
import syntacticAnalyzer.grammar.NonTerminal;
import syntacticAnalyzer.grammar.NonTerminalName;
import syntacticAnalyzer.grammar.Derivation;
import syntacticAnalyzer.grammar.Grammar;

public class Grammar {

        private static Grammar grammarSingleton;
        private ArrayList<Derivation> grammarMap;
        private Derivation derivationAux;

        private Grammar() {

                grammarMap = new ArrayList<Derivation>();
                loadGrammar();

        }

        public static Grammar getInstance() {
                if (grammarSingleton == null) {
                        grammarSingleton = new Grammar();
                }
```

```
        return grammarSingleton;
    }

    public ArrayList<Derivation> getGrammarMap() {
        return grammarMap;
    }

    private void loadGrammar() {

        // (1) 'prDuma' 'id' BEGIN
        derivationAux = new Derivation();
        derivationAux.addDerivationSymbols(new
        Terminal(TokenCategory.PRDUMA),new
        Terminal(TokenCategory.ID),new
        NonTerminal(NonTerminalName.BEGIN));
        grammarAddDerivation(derivationAux);

        // (2) FUNCTIONS INITIUM
        derivationAux.addDerivationSymbols(new
        NonTerminal(NonTerminalName.FUNCTIONS),
        new NonTerminal(NonTerminalName.INITIUM));
        grammarAddDerivation(derivationAux);

        // (3) RETURNTYPE 'id' PARAMS SCOPE FUNCTIONS
        derivationAux.addDerivationSymbols(new
        NonTerminal(NonTerminalName.RETURNTYPE), new
        Terminal(TokenCategory.ID),
        new NonTerminal(NonTerminalName.PARAMS), new
        NonTerminal(NonTerminalName.SCOPE),
        new NonTerminal(NonTerminalName.FUNCTIONS));
        grammarAddDerivation(derivationAux);

        // (4) epsilon
        grammarAddDerivation(null);

        // ...

        // (48) 'vetBegin' ARRAYFAT
        derivationAux.addDerivationSymbols(new
        Terminal(TokenCategory.VETBEGIN),
        new NonTerminal(NonTerminalName.ARRAYFAT));
        grammarAddDerivation(derivationAux);
```

```
// (49) ELEMENTS 'vetEnd'
derivationAux.addDerivationSymbols(new
NonTerminal(NonTerminalName.ELEMENTS),
new Terminal(TokenCategory.VETEND));
grammarAddDerivation(derivationAux);

// (50) 'vetEnd'
derivationAux.addSymbol(new Terminal(TokenCategory.VETEND));
grammarAddDerivation(derivationAux);

// (51) CONSTANT ELEMENTSFAT
derivationAux.addDerivationSymbols(new
NonTerminal(NonTerminalName.CONSTANT),
new NonTerminal(NonTerminalName.ELEMENTSFAT));
grammarAddDerivation(derivationAux);

// ...

// (99) 'paramBegin' Eb 'paramEnd'
derivationAux.addDerivationSymbols(new
Terminal(TokenCategory.PARAMBEGIN), new
NonTerminal(NonTerminalName.Eb),
new Terminal(TokenCategory.PARAMEND));
grammarAddDerivation(derivationAux);

// (100) CONSTANT
derivationAux.addSymbol(new NonTerminal(NonTerminalName.CONSTANT));
grammarAddDerivation(derivationAux);

// (101) 'id' Far
derivationAux.addDerivationSymbols(new
Terminal(TokenCategory.ID), new
NonTerminal(NonTerminalName.Far));
grammarAddDerivation(derivationAux);

// (102) 'vetBegin' Eb 'vetEnd'
derivationAux.addDerivationSymbols(new
Terminal(TokenCategory.VETBEGIN), new
NonTerminal(NonTerminalName.Eb),
        new Terminal(TokenCategory.VETEND));
grammarAddDerivation(derivationAux);

// (103) epsilon
```

```
                grammarAddDerivation(null);

        }

        private void grammarAddDerivation(Derivation derivation) {
                grammarMap.add(derivation);
                if (derivation != null) {
                        derivationAux = new Derivation();
                }
        }
}
```

2.3 Classes do pacote *syntacticAnalyzer*

2.3.1 PredictiveTable.java

```
package syntacticAnalyzer;

import java.util.HashMap;
import lexicalAnalyzer.TokenCategory;
import syntacticAnalyzer.grammar.NonTerminalName;

public class PredictiveTable {

        private HashMap<NonTerminalName, HashMap<TokenCategory,
    Integer>> predectiveTableMap;
        private HashMap<TokenCategory, Integer> terminaisMap;

        public PredictiveTable() {
                terminaisMap = new HashMap<TokenCategory, Integer>();
                predectiveTableMap = new HashMap<NonTerminalName,
                HashMap<TokenCategory, Integer>>();
                loadPredectiveTableMap();
        }

        public Integer getDerivationNumber(NonTerminalName
        nonTerminal, TokenCategory terminal) {
                return
        predectiveTableMap.get(nonTerminal).get(terminal);
        }

        // Tabela preditiva
```

```
    private void loadPredectiveTableMap() {

            terminaisMap.put(TokenCategory.PRDUMA, 0);
            predectiveTableMap.put(NonTerminalName.PROGRAM, terminaisMap);
            terminaisMap = new HashMap<TokenCategory, Integer>();

            terminaisMap.put(TokenCategory.TDINANIS, 1);
            terminaisMap.put(TokenCategory.TDINT, 1);
            terminaisMap.put(TokenCategory.TDREAL, 1);
            terminaisMap.put(TokenCategory.TDLIT, 1);
            terminaisMap.put(TokenCategory.TDBOOL, 1);
            terminaisMap.put(TokenCategory.TDSERMO, 1);
            terminaisMap.put(TokenCategory.PRINITIALS, 1);
            predectiveTableMap.put(NonTerminalName.BEGIN, terminaisMap);
            terminaisMap = new HashMap<TokenCategory, Integer>();

            terminaisMap.put(TokenCategory.TDINANIS, 2);
            terminaisMap.put(TokenCategory.TDINT, 2);
            terminaisMap.put(TokenCategory.TDREAL, 2);
            terminaisMap.put(TokenCategory.TDLIT, 2);
            terminaisMap.put(TokenCategory.TDBOOL, 2);
            terminaisMap.put(TokenCategory.TDSERMO, 2);
            terminaisMap.put(TokenCategory.PRINITIALS, 3);
            predectiveTableMap.put(NonTerminalName.FUNCTIONS, terminaisMap);
            terminaisMap = new HashMap<TokenCategory, Integer>();

            terminaisMap.put(TokenCategory.PRINITIALS, 4);
            predectiveTableMap.put(NonTerminalName.INITIUM, terminaisMap);
            terminaisMap = new HashMap<TokenCategory, Integer>();

            terminaisMap.put(TokenCategory.PARAMBEGIN, 5);
            predectiveTableMap.put(NonTerminalName.PARAMS, terminaisMap);
            terminaisMap = new HashMap<TokenCategory, Integer>();

            terminaisMap.put(TokenCategory.PARAMEND, 6);
            terminaisMap.put(TokenCategory.TDINANIS, 7);
            terminaisMap.put(TokenCategory.TDINT, 7);
            terminaisMap.put(TokenCategory.TDREAL, 7);
            terminaisMap.put(TokenCategory.TDLIT, 7);
            terminaisMap.put(TokenCategory.TDBOOL, 7);
            terminaisMap.put(TokenCategory.TDSERMO, 7);
            terminaisMap.put(TokenCategory.PRMATRIX, 7);
            predectiveTableMap.put(NonTerminalName.PARAMSFAT, terminaisMap);
```

```
            terminaisMap = new HashMap<TokenCategory, Integer>();

            terminaisMap.put(TokenCategory.TDINANIS, 8);
            terminaisMap.put(TokenCategory.TDINT, 8);
            terminaisMap.put(TokenCategory.TDREAL, 8);
            terminaisMap.put(TokenCategory.TDLIT, 8);
            terminaisMap.put(TokenCategory.TDBOOL, 8);
            terminaisMap.put(TokenCategory.TDSERMO, 8);
            terminaisMap.put(TokenCategory.PRMATRIX, 9);
            predectiveTableMap.put(NonTerminalName.LISTPARAMS, terminaisMap);
            terminaisMap = new HashMap<TokenCategory, Integer>();

            terminaisMap.put(TokenCategory.SEPVIRG, 10);
            terminaisMap.put(TokenCategory.PARAMEND, 11);
            predectiveTableMap.put(NonTerminalName.LISTPARAMSFAT, terminaisMap);
            terminaisMap = new HashMap<TokenCategory, Integer>();


        // ...

        terminaisMap.put(TokenCategory.OPARITUN, 95);
            terminaisMap.put(TokenCategory.OPLOGNEG, 96);
            terminaisMap.put(TokenCategory.PARAMBEGIN, 97);
            terminaisMap.put(TokenCategory.CTENUMINT, 97);
            terminaisMap.put(TokenCategory.CTENUMREAL, 97);
            terminaisMap.put(TokenCategory.CTEBOOL, 97);
            terminaisMap.put(TokenCategory.CTESERMO, 97);
            terminaisMap.put(TokenCategory.CTELIT, 97);
            terminaisMap.put(TokenCategory.ID, 97);
            predectiveTableMap.put(NonTerminalName.Ta, terminaisMap);
            terminaisMap = new HashMap<TokenCategory, Integer>();

            terminaisMap.put(TokenCategory.PARAMBEGIN, 98);
            terminaisMap.put(TokenCategory.CTENUMINT, 99);
            terminaisMap.put(TokenCategory.CTENUMREAL, 99);
            terminaisMap.put(TokenCategory.CTEBOOL, 99);
            terminaisMap.put(TokenCategory.CTESERMO, 99);
            terminaisMap.put(TokenCategory.CTELIT, 99);
            terminaisMap.put(TokenCategory.ID, 100);
            predectiveTableMap.put(NonTerminalName.Fa, terminaisMap);
            terminaisMap = new HashMap<TokenCategory, Integer>();

            terminaisMap.put(TokenCategory.VETBEGIN, 101);
```

```
                terminaisMap.put(TokenCategory.OPARITMUL, 102);
                terminaisMap.put(TokenCategory.OPARITAD, 102);
                terminaisMap.put(TokenCategory.OPREL1, 102);
                terminaisMap.put(TokenCategory.OPREL2, 102);
                terminaisMap.put(TokenCategory.OPLOGAND, 102);
                terminaisMap.put(TokenCategory.OPLOGOR, 102);
                terminaisMap.put(TokenCategory.PARAMEND, 102);
                terminaisMap.put(TokenCategory.TERMCMD, 102);
                terminaisMap.put(TokenCategory.VETEND, 102);
                predectiveTableMap.put(NonTerminalName.Far, terminaisMap);

        }
}
```

2.3.2 PredictiveAnalyzer.java

```
package syntacticAnalyzer;

import java.util.Stack;

import lexicalAnalyzer.Token;
import lexicalAnalyzer.LexicalAnalyzer;
import syntacticAnalyzer.SyntaticAnalyzer;
import syntacticAnalyzer.grammar.NonTerminal;
import syntacticAnalyzer.grammar.NonTerminalName;
import syntacticAnalyzer.grammar.Terminal;
import syntacticAnalyzer.PredictiveTable;
import syntacticAnalyzer.grammar.Derivation;
import syntacticAnalyzer.grammar.Grammar;
import syntacticAnalyzer.grammar.Symbol;

public class PredictiveAnalyzer {

        private Grammar grammar;
        private PredictiveTable predictiveTable;
        private LexicalAnalyzer lexicalAnalyzer;

        private Stack<Symbol> stack;
        private Derivation derivation;

        public PredictiveAnalyzer(Grammar grammar, PredictiveTable
        predictiveTable, LexicalAnalyzer lexicalAnalyzer) {
```

```
        this.grammar = grammar;
        this.predictiveTable = predictiveTable;
        this.lexicalAnalyzer = lexicalAnalyzer;

        stack = new Stack<Symbol>();
        derivation = new Derivation();
    }

    public void predictiveAnalyze() {

        Symbol topSymbol;
        Token token = new Token();
        Terminal terminal;
        NonTerminal topNonTerminal;
        Integer derivationNumber;
        Stack<Integer> prodCount = new Stack<Integer>();
        int leftCount = 0;
        int rightCount = 1;
        int rightCountAux = 0;

        if (lexicalAnalyzer.hasMoreTokens()) {

            token = lexicalAnalyzer.nextToken();

            terminal = new Terminal(token);
            stack.push(new NonTerminal(NonTerminalName.PROGRAM));
            prodCount.push(1);

            while (!stack.isEmpty()) {

                topSymbol = stack.peek();

                if (topSymbol.isTerminal()) {

                    if (topSymbol.getValue()==
                            terminal.getValue()) {
                        stack.pop();
                        listaToken(terminal);
                        if (lexicalAnalyzer.hasMoreTokens()){
                        token=lexicalAnalyzer.nextToken();
                        terminal=new Terminal(token);
                        }
```

```
                } else {
                        SyntaticAnalyzer.printError(token);
                        System.exit(1);
                }

} else {

        topNonTerminal = (NonTerminal) topSymbol;

        derivationNumber = null;

        derivationNumber = predictiveTable.getDerivationNumber(
                topNonTerminal.getName(),terminal.getCategory());

        if (derivationNumber != null) {
                leftCount = prodCount.pop();
                rightCountAux = rightCount;

                derivation = grammar.getGrammarMap().get(derivationNumber);
                if (derivation != null) {
                        System.out.print(topNonTerminal.getName() + "(" +
                        leftCount + ")" + " = ");
                        stack.pop();
                        // TO REMOVE
                        Symbol symb;
                        Terminal term;
                        NonTerminal nonTerm;

                        for(int i=derivation.getSymbolsList().size()-1;i>=0;i--){

                                symb = derivation.getSymbolsList().get(i);
                                if (symb.isTerminal()) {
                                        term = (Terminal) symb;
                                } else {
                                        nonTerm = (NonTerminal) symb;
                                }

                                stack.push(symb);
                        }

                        for(int i=0;i<derivation.getSymbolsList().size();i++){
                                symb = derivation.getSymbolsList().get(i);
                                if (symb.isTerminal()) {
```

```
                                        term = (Terminal) symb;
                                        System.out.print("'" +
                    term.getCategory().toString().toLowerCase() + "'" + " ");
                                } else {
                                        nonTerm = (NonTerminal) symb;
                                        System.out.print(nonTerm.getName() + "(" +
                                        ++rightCount + ")" + " ");
                                }
                        }
                        System.out.println();
                } else {
                        System.out.println(topNonTerminal.getName() + "(" +
                                                leftCount + ")" + " = epsilon");
                        stack.pop();
                }

                if (rightCount > rightCountAux) {
                        int aux = rightCount;
                        while (aux > rightCountAux) {
                                prodCount.push(aux--);
                        }
                }

                } else {
                        SyntaticAnalyzer.printError(terminal.getTerminalToken());
                        System.exit(1);
                }
        }
       }
      }
}

        private void listaToken(Terminal terminal) {

                System.out.println(" " + terminal.getCategory().name().toLowerCase()
                + " = " + "[" + terminal.getValue() + ", "+ "'" +
                terminal.getTerminalToken().getValue()+ "'" + ", " +
                terminal.getTerminalToken().getLine()+ ", " +
                terminal.getTerminalToken().getColumn() + "]");
        }
}
```

2.3.3 SyntacticAnalyzer.java

```
package syntacticAnalyzer;

import lexicalAnalyzer.LexicalAnalyzer;
import lexicalAnalyzer.Token;
import syntacticAnalyzer.PredictiveAnalyzer;
import syntacticAnalyzer.PredictiveTable;
import syntacticAnalyzer.grammar.Grammar;

public class SyntaticAnalyzer {

        private Grammar grammar;
        private PredictiveAnalyzer predictiveAnalyzer;
        private PredictiveTable predictiveTable;

        public SyntaticAnalyzer(LexicalAnalyzer lexicalAnalyzer) {
                grammar = Grammar.getInstance();
                predictiveTable = new PredictiveTable();
                predictiveAnalyzer = new PredictiveAnalyzer(grammar, predictiveTable,
                lexicalAnalyzer);
        }

        public void analyze() {
                predictiveAnalyzer.predictiveAnalyze();
        }

        public static void printError(Token token) {
                System.err.println("Erro no token " + token.getCategory() + "(" +
                token.getValue() + ")," + " na linha " + token.getLine() +
                " e coluna " + token.getColumn() + ".");
        }

}
```

2.4 Saída do Analisador para os códigos do trabalho anterior

2.4.1 HelloWorld.duma

O resultado para este exemplo foi o seguinte:

```
PROGRAM(1) = 'prduma' 'id' BEGIN(2)
        prduma = [1, 'duma', 0, 0]
        id = [10, 'hello_word', 0, 5]
BEGIN(2) = FUNCTIONS(3) INITIUM(4)
```

```
FUNCTIONS(3) = epsilon
INITIUM(4) = 'prinitials' 'tdinanis' 'prinitium' 'parambegin' 'paramend' SCOPE(5)
        prinitials = [3, 'initials', 2, 0]
        tdinanis = [11, 'inanis', 2, 9]
        prinitium = [2, 'initium', 2, 16]
        parambegin = [20, '(', 2, 24]
        paramend = [21, ')', 2, 25]
SCOPE(5) = 'escbegin' COMMANDS(6) 'escend' 'termcmd'
        escbegin = [18, '{', 2, 27]
COMMANDS(6) = CMD(7) 'termcmd' COMMANDS(8)
CMD(7) = WRITE(9)
WRITE(9) = 'prscribo' 'parambegin' MESSAGE(10) 'paramend'
        prscribo = [6, 'scribo', 4, 1]
        parambegin = [20, '(', 4, 7]
MESSAGE(10) = 'ctesermo' MESSAGEFAT(11)
        ctesermo = [28, '"Alo mundo"', 4, 8]
MESSAGEFAT(11) = epsilon
        paramend = [21, ')', 4, 19]
        termcmd = [24, ';', 4, 20]
COMMANDS(8) = epsilon
        escend = [19, '}', 6, 0]
        termcmd = [24, ';', 6, 1]
```

2.4.2 Fibonacci.duma

```
PROGRAM(1) = 'prduma' 'id' BEGIN(2)
        prduma = [1, 'duma', 0, 0]
        id = [10, 'fibonacci', 0, 5]
BEGIN(2) = FUNCTIONS(3) INITIUM(4)
FUNCTIONS(3) = RETURNTYPE(5) 'id' PARAMS(6) SCOPE(7) FUNCTIONS(8)
RETURNTYPE(5) = TYPE(9)
TYPE(9) = 'tdinanis'
        tdinanis = [11, 'inanis', 2, 0]
        id = [10, 'fib', 2, 7]
PARAMS(6) = 'parambegin' PARAMSFAT(10)
        parambegin = [20, '(', 2, 10]
PARAMSFAT(10) = LISTPARAMS(11) 'paramend'
LISTPARAMS(11) = TYPE(12) NAME(13) LISTPARAMSFAT(14)
TYPE(12) = 'tdint'
        tdint = [12, 'integer', 2, 11]
NAME(13) = 'id' NAMEFAT(15)
        id = [10, 'limit', 2, 19]
```

```
NAMEFAT(15) = epsilon
LISTPARAMSFAT(14) = epsilon
        paramend = [21, ')', 2, 24]
SCOPE(7) = 'escbegin' COMMANDS(16) 'escend' 'termcmd'
        escbegin = [18, '{', 2, 25]
COMMANDS(16) = CMD(17) 'termcmd' COMMANDS(18)
CMD(17) = DECLARATION(19)
DECLARATION(19) = TYPE(20) NAME(21)
TYPE(20) = 'tdint'
        tdint = [12, 'integer', 4, 1]
NAME(21) = 'id' NAMEFAT(22)
        id = [10, 'count', 4, 9]
NAMEFAT(22) = epsilon
        termcmd = [24, ';', 4, 14]
COMMANDS(18) = CMD(23) 'termcmd' COMMANDS(24)
CMD(23) = DECLARATION(25)
DECLARATION(25) = TYPE(26) NAME(27)
TYPE(26) = 'tdint'
        tdint = [12, 'integer', 5, 1]
NAME(27) = 'id' NAMEFAT(28)
        id = [10, 'fib1', 5, 9]
NAMEFAT(28) = epsilon
        termcmd = [24, ';', 5, 13]
COMMANDS(24) = CMD(29) 'termcmd' COMMANDS(30)
CMD(29) = DECLARATION(31)
DECLARATION(31) = TYPE(32) NAME(33)
TYPE(32) = 'tdint'
        tdint = [12, 'integer', 6, 1]
NAME(33) = 'id' NAMEFAT(34)
        id = [10, 'fib2', 6, 9]
NAMEFAT(34) = epsilon
        termcmd = [24, ';', 6, 13]
COMMANDS(30) = CMD(35) 'termcmd' COMMANDS(36)
CMD(35) = DECLARATION(37)
DECLARATION(37) = TYPE(38) NAME(39)
TYPE(38) = 'tdint'
        tdint = [12, 'integer', 7, 1]
NAME(39) = 'id' NAMEFAT(40)
        id = [10, 'fib3', 7, 9]
NAMEFAT(40) = epsilon
        termcmd = [24, ';', 7, 13]
COMMANDS(36) = CMD(41) 'termcmd' COMMANDS(42)
CMD(41) = 'id' CMDFAT(43)
```

```
        id = [10, 'fib1', 9, 1]
CMDFAT(43) = ATTRIBUTION(44)
ATTRIBUTION(44) = NAMEFAT(45) 'instatrib' VALUE(46)
NAMEFAT(45) = epsilon
        instatrib = [50, '=', 9, 6]
VALUE(46) = Eb(47)
Eb(47) = Tb(48) Ebr(49)
Tb(48) = Fb(50) Tbr(51)
Fb(50) = Erel2(52) Fbr(53)
Erel2(52) = Erel1(54) Erel2r(55)
Erel1(54) = Ea(56) Erel1r(57)
Ea(56) = Ta(58) Ear(59)
Ta(58) = Fa(60)
Fa(60) = CONSTANT(61)
CONSTANT(61) = 'ctenumint'
        ctenumint = [25, '1', 9, 8]
Ear(59) = epsilon
Erel1r(57) = epsilon
Erel2r(55) = epsilon
Fbr(53) = epsilon
Tbr(51) = epsilon
Ebr(49) = epsilon
        termcmd = [24, ';', 9, 9]
COMMANDS(42) = CMD(62) 'termcmd' COMMANDS(63)
CMD(62) = 'id' CMDFAT(64)
        id = [10, 'fib2', 10, 1]
CMDFAT(64) = ATTRIBUTION(65)
ATTRIBUTION(65) = NAMEFAT(66) 'instatrib' VALUE(67)
NAMEFAT(66) = epsilon
        instatrib = [50, '=', 10, 6]
VALUE(67) = Eb(68)
Eb(68) = Tb(69) Ebr(70)
Tb(69) = Fb(71) Tbr(72)
Fb(71) = Erel2(73) Fbr(74)
Erel2(73) = Erel1(75) Erel2r(76)
Erel1(75) = Ea(77) Erel1r(78)
Ea(77) = Ta(79) Ear(80)
Ta(79) = Fa(81)
Fa(81) = CONSTANT(82)
CONSTANT(82) = 'ctenumint'
        ctenumint = [25, '1', 10, 8]
Ear(80) = epsilon
Erel1r(78) = epsilon
```

```
Erel2r(76) = epsilon
Fbr(74) = epsilon
Tbr(72) = epsilon
Ebr(70) = epsilon
        termcmd = [24, ';', 10, 9]
COMMANDS(63) = CMD(83) 'termcmd' COMMANDS(84)
CMD(83) = IFELSE(85)
IFELSE(85) = IF(86) ELSEIF(87) ELSE(88)
IF(86) = 'selsi' 'parambegin' Eb(89) 'paramend' 'escbegin' COMMANDS(90) 'escend'
        selsi = [30, 'si', 12, 1]
        parambegin = [20, '(', 12, 4]
Eb(89) = Tb(91) Ebr(92)
Tb(91) = Fb(93) Tbr(94)
Fb(93) = Erel2(95) Fbr(96)
Erel2(95) = Erel1(97) Erel2r(98)
Erel1(97) = Ea(99) Erel1r(100)
Ea(99) = Ta(101) Ear(102)
Ta(101) = Fa(103)
Fa(103) = 'id' Far(104)
        id = [10, 'limit', 12, 5]
Far(104) = epsilon
Ear(102) = epsilon
Erel1r(100) = epsilon
Erel2r(98) = epsilon
Fbr(96) = 'oprel2' Erel2(105) Fbr(106)
        oprel2 = [45, '==', 12, 11]
Erel2(105) = Erel1(107) Erel2r(108)
Erel1(107) = Ea(109) Erel1r(110)
Ea(109) = Ta(111) Ear(112)
Ta(111) = Fa(113)
Fa(113) = CONSTANT(114)
CONSTANT(114) = 'ctenumint'
        ctenumint = [25, '0', 12, 14]
Ear(112) = epsilon
Erel1r(110) = epsilon
Erel2r(108) = epsilon
Fbr(106) = epsilon
Tbr(94) = epsilon
Ebr(92) = epsilon
        paramend = [21, ')', 12, 15]
        escbegin = [18, '{', 12, 17]
COMMANDS(90) = CMD(115) 'termcmd' COMMANDS(116)
CMD(115) = WRITE(117)
```

```
WRITE(117) = 'prscribo' 'parambegin' MESSAGE(118) 'paramend'
        prscribo = [6, 'scribo', 13, 2]
        parambegin = [20, '(', 13, 8]
MESSAGE(118) = 'ctesermo' MESSAGEFAT(119)
        ctesermo = [28, '"0"', 13, 9]
MESSAGEFAT(119) = epsilon
        paramend = [21, ')', 13, 12]
        termcmd = [24, ';', 13, 13]
COMMANDS(116) = epsilon
        escend = [19, '}', 14, 1]
ELSEIF(87) = epsilon
ELSE(88) = epsilon
        termcmd = [24, ';', 14, 2]
COMMANDS(84) = CMD(120) 'termcmd' COMMANDS(121)
CMD(120) = WHILE(122)
WHILE(122) = 'repdum' 'parambegin' Eb(123) 'paramend' 'escbegin' COMMANDS(124)
'escend'
        repdum = [34, 'dum', 16, 1]
        parambegin = [20, '(', 16, 5]
Eb(123) = Tb(125) Ebr(126)
Tb(125) = Fb(127) Tbr(128)
Fb(127) = Erel2(129) Fbr(130)
Erel2(129) = Erel1(131) Erel2r(132)
Erel1(131) = Ea(133) Erel1r(134)
Ea(133) = Ta(135) Ear(136)
Ta(135) = Fa(137)
Fa(137) = 'id' Far(138)
        id = [10, 'count', 16, 6]
Far(138) = epsilon
Ear(136) = epsilon
Erel1r(134) = epsilon
Erel2r(132) = 'oprel1' Erel1(139) Erel2r(140)
        oprel1 = [44, '<', 16, 12]
Erel1(139) = Ea(141) Erel1r(142)
Ea(141) = Ta(143) Ear(144)
Ta(143) = Fa(145)
Fa(145) = 'id' Far(146)
        id = [10, 'limit', 16, 14]
Far(146) = epsilon
Ear(144) = epsilon
Erel1r(142) = epsilon
Erel2r(140) = epsilon
Fbr(130) = epsilon
```

```
Tbr(128) = epsilon
Ebr(126) = epsilon
        paramend = [21, ')', 16, 19]
        escbegin = [18, '{', 16, 21]
COMMANDS(124) = CMD(147) 'termcmd' COMMANDS(148)
CMD(147) = IFELSE(149)
IFELSE(149) = IF(150) ELSEIF(151) ELSE(152)
IF(150) = 'selsi' 'parambegin' Eb(153) 'paramend' 'escbegin' COMMANDS(154) 'escend'
        selsi = [30, 'si', 17, 2]
        parambegin = [20, '(', 17, 5]
Eb(153) = Tb(155) Ebr(156)
Tb(155) = Fb(157) Tbr(158)
Fb(157) = Erel2(159) Fbr(160)
Erel2(159) = Erel1(161) Erel2r(162)
Erel1(161) = Ea(163) Erel1r(164)
Ea(163) = Ta(165) Ear(166)
Ta(165) = Fa(167)
Fa(167) = 'id' Far(168)
        id = [10, 'count', 17, 6]
Far(168) = epsilon
Ear(166) = epsilon
Erel1r(164) = epsilon
Erel2r(162) = 'oprel1' Erel1(169) Erel2r(170)
        oprel1 = [44, '<', 17, 12]
Erel1(169) = Ea(171) Erel1r(172)
Ea(171) = Ta(173) Ear(174)
Ta(173) = Fa(175)
Fa(175) = CONSTANT(176)
CONSTANT(176) = 'ctenumint'
        ctenumint = [25, '2', 17, 14]
Ear(174) = epsilon
Erel1r(172) = epsilon
Erel2r(170) = epsilon
Fbr(160) = epsilon
Tbr(158) = epsilon
Ebr(156) = epsilon
        paramend = [21, ')', 17, 15]
        escbegin = [18, '{', 17, 17]
COMMANDS(154) = CMD(177) 'termcmd' COMMANDS(178)
CMD(177) = WRITE(179)
WRITE(179) = 'prscribo' 'parambegin' MESSAGE(180) 'paramend'
        prscribo = [6, 'scribo', 18, 3]
        parambegin = [20, '(', 18, 10]
```

```
MESSAGE(180) = 'ctesermo' MESSAGEFAT(181)
        ctesermo = [28, '"1 "', 18, 11]
MESSAGEFAT(181) = epsilon
        paramend = [21, ')', 18, 15]
        termcmd = [24, ';', 18, 16]
COMMANDS(178) = epsilon
        escend = [19, '}', 19, 2]
ELSEIF(151) = epsilon
ELSE(152) = 'selaliud' 'escbegin' COMMANDS(182) 'escend'
        selaliud = [31, 'aliud', 19, 4]
        escbegin = [18, '{', 19, 10]
COMMANDS(182) = CMD(183) 'termcmd' COMMANDS(184)
CMD(183) = 'id' CMDFAT(185)
        id = [10, 'fib3', 20, 3]
CMDFAT(185) = ATTRIBUTION(186)
ATTRIBUTION(186) = NAMEFAT(187) 'instatrib' VALUE(188)
NAMEFAT(187) = epsilon
        instatrib = [50, '=', 20, 8]
VALUE(188) = Eb(189)
Eb(189) = Tb(190) Ebr(191)
Tb(190) = Fb(192) Tbr(193)
Fb(192) = Erel2(194) Fbr(195)
Erel2(194) = Erel1(196) Erel2r(197)
Erel1(196) = Ea(198) Erel1r(199)
Ea(198) = Ta(200) Ear(201)
Ta(200) = Fa(202)
Fa(202) = 'id' Far(203)
        id = [10, 'fib1', 20, 10]
Far(203) = epsilon
Ear(201) = epsilon
Erel1r(199) = 'oparitad' Ea(204) Erel1r(205)
        oparitad = [38, '+', 20, 15]
Ea(204) = Ta(206) Ear(207)
Ta(206) = Fa(208)
Fa(208) = 'id' Far(209)
        id = [10, 'fib2', 20, 17]
Far(209) = epsilon
Ear(207) = epsilon
Erel1r(205) = epsilon
Erel2r(197) = epsilon
Fbr(195) = epsilon
Tbr(193) = epsilon
Ebr(191) = epsilon
```

```
        termcmd = [24, ';', 20, 21]
COMMANDS(184) = CMD(210) 'termcmd' COMMANDS(211)
CMD(210) = 'id' CMDFAT(212)
        id = [10, 'fib1', 21, 3]
CMDFAT(212) = ATTRIBUTION(213)
ATTRIBUTION(213) = NAMEFAT(214) 'instatrib' VALUE(215)
NAMEFAT(214) = epsilon
        instatrib = [50, '=', 21, 8]
VALUE(215) = Eb(216)
Eb(216) = Tb(217) Ebr(218)
Tb(217) = Fb(219) Tbr(220)
Fb(219) = Erel2(221) Fbr(222)
Erel2(221) = Erel1(223) Erel2r(224)
Erel1(223) = Ea(225) Erel1r(226)
Ea(225) = Ta(227) Ear(228)
Ta(227) = Fa(229)
Fa(229) = 'id' Far(230)
        id = [10, 'fib2', 21, 10]
Far(230) = epsilon
Ear(228) = epsilon
Erel1r(226) = epsilon
Erel2r(224) = epsilon
Fbr(222) = epsilon
Tbr(220) = epsilon
Ebr(218) = epsilon
        termcmd = [24, ';', 21, 14]
COMMANDS(211) = CMD(231) 'termcmd' COMMANDS(232)
CMD(231) = 'id' CMDFAT(233)
        id = [10, 'fib2', 22, 3]
CMDFAT(233) = ATTRIBUTION(234)
ATTRIBUTION(234) = NAMEFAT(235) 'instatrib' VALUE(236)
NAMEFAT(235) = epsilon
        instatrib = [50, '=', 22, 8]
VALUE(236) = Eb(237)
Eb(237) = Tb(238) Ebr(239)
Tb(238) = Fb(240) Tbr(241)
Fb(240) = Erel2(242) Fbr(243)
Erel2(242) = Erel1(244) Erel2r(245)
Erel1(244) = Ea(246) Erel1r(247)
Ea(246) = Ta(248) Ear(249)
Ta(248) = Fa(250)
Fa(250) = 'id' Far(251)
        id = [10, 'fib3', 22, 10]
```

```
Far(251) = epsilon
Ear(249) = epsilon
Erel1r(247) = epsilon
Erel2r(245) = epsilon
Fbr(243) = epsilon
Tbr(241) = epsilon
Ebr(239) = epsilon
        termcmd = [24, ';', 22, 14]
COMMANDS(232) = CMD(252) 'termcmd' COMMANDS(253)
CMD(252) = WRITE(254)
WRITE(254) = 'prscribo' 'parambegin' MESSAGE(255) 'paramend'
        prscribo = [6, 'scribo', 23, 3]
        parambegin = [20, '(', 23, 10]
MESSAGE(255) = NAME(256) MESSAGEFAT(257)
NAME(256) = 'id' NAMEFAT(258)
        id = [10, 'fib3', 23, 11]
NAMEFAT(258) = epsilon
MESSAGEFAT(257) = 'opcon' MESSAGE(259)
        opcon = [46, '.', 23, 16]
MESSAGE(259) = 'ctesermo' MESSAGEFAT(260)
        ctesermo = [28, '" "', 23, 18]
MESSAGEFAT(260) = epsilon
        paramend = [21, ')', 23, 21]
        termcmd = [24, ';', 23, 22]
COMMANDS(253) = epsilon
        escend = [19, '}', 24, 2]
        termcmd = [24, ';', 24, 3]
COMMANDS(148) = CMD(261) 'termcmd' COMMANDS(262)
CMD(261) = 'id' CMDFAT(263)
        id = [10, 'count', 26, 2]
CMDFAT(263) = ATTRIBUTION(264)
ATTRIBUTION(264) = NAMEFAT(265) 'instatrib' VALUE(266)
NAMEFAT(265) = epsilon
        instatrib = [50, '=', 26, 8]
VALUE(266) = Eb(267)
Eb(267) = Tb(268) Ebr(269)
Tb(268) = Fb(270) Tbr(271)
Fb(270) = Erel2(272) Fbr(273)
Erel2(272) = Erel1(274) Erel2r(275)
Erel1(274) = Ea(276) Erel1r(277)
Ea(276) = Ta(278) Ear(279)
Ta(278) = Fa(280)
Fa(280) = 'id' Far(281)
```

```
        id = [10, 'count', 26, 10]
Far(281) = epsilon
Ear(279) = epsilon
Erel1r(277) = 'oparitad' Ea(282) Erel1r(283)
        oparitad = [38, '+', 26, 16]
Ea(282) = Ta(284) Ear(285)
Ta(284) = Fa(286)
Fa(286) = CONSTANT(287)
CONSTANT(287) = 'ctenumint'
        ctenumint = [25, '1', 26, 17]
Ear(285) = epsilon
Erel1r(283) = epsilon
Erel2r(275) = epsilon
Fbr(273) = epsilon
Tbr(271) = epsilon
Ebr(269) = epsilon
        termcmd = [24, ';', 26, 18]
COMMANDS(262) = epsilon
        escend = [19, '}', 27, 1]
        termcmd = [24, ';', 27, 2]
COMMANDS(121) = epsilon
        escend = [19, '}', 28, 0]
        termcmd = [24, ';', 28, 1]
FUNCTIONS(8) = epsilon
INITIUM(4) = 'prinitials' 'tdinanis' 'prinitium' 'parambegin' 'paramend' SCOPE(288)
        prinitials = [3, 'initials', 30, 0]
        tdinanis = [11, 'inanis', 30, 9]
        prinitium = [2, 'initium', 30, 16]
        parambegin = [20, '(', 30, 23]
        paramend = [21, ')', 30, 24]
SCOPE(288) = 'escbegin' COMMANDS(289) 'escend' 'termcmd'
        escbegin = [18, '{', 30, 26]
COMMANDS(289) = CMD(290) 'termcmd' COMMANDS(291)
CMD(290) = DECLARATION(292)
DECLARATION(292) = TYPE(293) NAME(294)
TYPE(293) = 'tdint'
        tdint = [12, 'integer', 31, 1]
NAME(294) = 'id' NAMEFAT(295)
        id = [10, 'limit', 31, 9]
NAMEFAT(295) = epsilon
        termcmd = [24, ';', 31, 14]
COMMANDS(291) = CMD(296) 'termcmd' COMMANDS(297)
CMD(296) = READ(298)
```

```
READ(298) = 'prlectio' 'parambegin' NAME(299) 'paramend'
        prlectio = [8, 'lectio', 32, 1]
        parambegin = [20, '(', 32, 8]
NAME(299) = 'id' NAMEFAT(300)
        id = [10, 'limit', 32, 9]
NAMEFAT(300) = epsilon
        paramend = [21, ')', 32, 14]
        termcmd = [24, ';', 32, 15]
COMMANDS(297) = CMD(301) 'termcmd' COMMANDS(302)
CMD(301) = 'id' CMDFAT(303)
        id = [10, 'fib', 33, 1]
CMDFAT(303) = FUNCCALL(304)
FUNCCALL(304) = 'parambegin' LISTPARAMSCALL(305) 'paramend'
        parambegin = [20, '(', 33, 4]
LISTPARAMSCALL(305) = ITEMPARAM(306) LISTPARAMSCALLFAT(307)
ITEMPARAM(306) = NAME(308)
NAME(308) = 'id' NAMEFAT(309)
        id = [10, 'limit', 33, 5]
NAMEFAT(309) = epsilon
LISTPARAMSCALLFAT(307) = epsilon
        paramend = [21, ')', 33, 10]
        termcmd = [24, ';', 33, 11]
COMMANDS(302) = epsilon
        escend = [19, '}', 34, 0]
        termcmd = [24, ';', 34, 1]
```

2.4.3 ShellSort.duma

```
PROGRAM(1) = 'prduma' 'id' BEGIN(2)
        prduma = [1, 'duma', 0, 0]
        id = [10, 'shell_sort', 0, 5]
BEGIN(2) = FUNCTIONS(3) INITIUM(4)
FUNCTIONS(3) = RETURNTYPE(5) 'id' PARAMS(6) SCOPE(7) FUNCTIONS(8)
RETURNTYPE(5) = TYPE(9)
TYPE(9) = 'tdinanis'
        tdinanis = [11, 'inanis', 2, 0]
        id = [10, 'shell', 2, 7]
PARAMS(6) = 'parambegin' PARAMSFAT(10)
        parambegin = [20, '(', 2, 12]
PARAMSFAT(10) = LISTPARAMS(11) 'paramend'
LISTPARAMS(11) = TYPE(12) NAME(13) LISTPARAMSFAT(14)
TYPE(12) = 'tdint'
        tdint = [12, 'integer', 2, 13]
```

```
NAME(13) = 'id' NAMEFAT(15)
        id = [10, 'tamanho', 2, 21]
NAMEFAT(15) = epsilon
LISTPARAMSFAT(14) = 'sepvirg' LISTPARAMS(16)
        sepvirg = [23, ',', 2, 28]
LISTPARAMS(16) = 'prmatrix' TYPE(17) NAME(18) LISTPARAMSFAT(19)
        prmatrix = [17, 'matrix', 2, 30]
TYPE(17) = 'tdint'
        tdint = [12, 'integer', 2, 37]
NAME(18) = 'id' NAMEFAT(20)
        id = [10, 'numeros', 2, 45]
NAMEFAT(20) = epsilon
LISTPARAMSFAT(19) = epsilon
        paramend = [21, ')', 2, 52]
SCOPE(7) = 'escbegin' COMMANDS(21) 'escend' 'termcmd'
        escbegin = [18, '{', 2, 53]
COMMANDS(21) = CMD(22) 'termcmd' COMMANDS(23)
CMD(22) = DECLARATION(24)
DECLARATION(24) = TYPE(25) NAME(26)
TYPE(25) = 'tdint'
        tdint = [12, 'integer', 4, 1]
NAME(26) = 'id' NAMEFAT(27)
        id = [10, 'gap', 4, 9]
NAMEFAT(27) = epsilon
        termcmd = [24, ';', 4, 12]
COMMANDS(23) = CMD(28) 'termcmd' COMMANDS(29)
CMD(28) = 'id' CMDFAT(30)
        id = [10, 'gap', 6, 4]
CMDFAT(30) = ATTRIBUTION(31)
ATTRIBUTION(31) = NAMEFAT(32) 'instatrib' VALUE(33)
NAMEFAT(32) = epsilon
        instatrib = [50, '=', 6, 8]
VALUE(33) = Eb(34)
Eb(34) = Tb(35) Ebr(36)
Tb(35) = Fb(37) Tbr(38)
Fb(37) = Erel2(39) Fbr(40)
Erel2(39) = Erel1(41) Erel2r(42)
Erel1(41) = Ea(43) Erel1r(44)
Ea(43) = Ta(45) Ear(46)
Ta(45) = Fa(47)
Fa(47) = CONSTANT(48)
CONSTANT(48) = 'ctenumint'
        ctenumint = [25, '1', 6, 10]
```

```
Ear(46) = epsilon
Erel1r(44) = epsilon
Erel2r(42) = epsilon
Fbr(40) = epsilon
Tbr(38) = epsilon
Ebr(36) = epsilon
        termcmd = [24, ';', 6, 11]
COMMANDS(29) = CMD(49) 'termcmd' COMMANDS(50)
CMD(49) = DOWHILE(51)
DOWHILE(51) = 'repfacite' 'escbegin' COMMANDS(52) 'escend' 'repdum' 'parambegin'
Eb(53) 'paramend'
        repfacite = [35, 'facite', 8, 1]
        escbegin = [18, '{', 8, 7]
COMMANDS(52) = CMD(54) 'termcmd' COMMANDS(55)
CMD(54) = 'id' CMDFAT(56)
        id = [10, 'gap', 9, 2]
CMDFAT(56) = ATTRIBUTION(57)
ATTRIBUTION(57) = NAMEFAT(58) 'instatrib' VALUE(59)
NAMEFAT(58) = epsilon
        instatrib = [50, '=', 9, 6]
VALUE(59) = Eb(60)
Eb(60) = Tb(61) Ebr(62)
Tb(61) = Fb(63) Tbr(64)
Fb(63) = Erel2(65) Fbr(66)
Erel2(65) = Erel1(67) Erel2r(68)
Erel1(67) = Ea(69) Erel1r(70)
Ea(69) = Ta(71) Ear(72)
Ta(71) = Fa(73)
Fa(73) = CONSTANT(74)
CONSTANT(74) = 'ctenumint'
        ctenumint = [25, '3', 9, 8]
Ear(72) = 'oparitmul' Ta(75) Ear(76)
        oparitmul = [39, '*', 9, 9]
Ta(75) = Fa(77)
Fa(77) = 'id' Far(78)
        id = [10, 'gap', 9, 10]
Far(78) = epsilon
Ear(76) = epsilon
Erel1r(70) = 'oparitad' Ea(79) Erel1r(80)
        oparitad = [38, '+', 9, 13]
Ea(79) = Ta(81) Ear(82)
Ta(81) = Fa(83)
Fa(83) = CONSTANT(84)
```

```
CONSTANT(84) = 'ctenumint'
        ctenumint = [25, '1', 9, 14]
Ear(82) = epsilon
Erel1r(80) = epsilon
Erel2r(68) = epsilon
Fbr(66) = epsilon
Tbr(64) = epsilon
Ebr(62) = epsilon
        termcmd = [24, ';', 9, 15]
COMMANDS(55) = epsilon
        escend = [19, '}', 10, 1]
        repdum = [34, 'dum', 10, 2]
        parambegin = [20, '(', 10, 5]
Eb(53) = Tb(85) Ebr(86)
Tb(85) = Fb(87) Tbr(88)
Fb(87) = Erel2(89) Fbr(90)
Erel2(89) = Erel1(91) Erel2r(92)
Erel1(91) = Ea(93) Erel1r(94)
Ea(93) = Ta(95) Ear(96)
Ta(95) = Fa(97)
Fa(97) = 'id' Far(98)
        id = [10, 'gap', 10, 6]
Far(98) = epsilon
Ear(96) = epsilon
Erel1r(94) = epsilon
Erel2r(92) = 'oprel1' Erel1(99) Erel2r(100)
        oprel1 = [44, '<', 10, 10]
Erel1(99) = Ea(101) Erel1r(102)
Ea(101) = Ta(103) Ear(104)
Ta(103) = Fa(105)
Fa(105) = 'id' Far(106)
        id = [10, 'tamanho', 10, 12]
Far(106) = epsilon
Ear(104) = epsilon
Erel1r(102) = epsilon
Erel2r(100) = epsilon
Fbr(90) = epsilon
Tbr(88) = epsilon
Ebr(86) = epsilon
        paramend = [21, ')', 10, 19]
        termcmd = [24, ';', 10, 20]
COMMANDS(50) = CMD(107) 'termcmd' COMMANDS(108)
CMD(107) = DOWHILE(109)
```

```
DOWHILE(109) = 'repfacite' 'escbegin' COMMANDS(110) 'escend' 'repdum' 'parambegin'
Eb(111) 'paramend'
        repfacite = [35, 'facite', 12, 1]
        escbegin = [18, '{', 12, 7]
COMMANDS(110) = CMD(112) 'termcmd' COMMANDS(113)
CMD(112) = 'id' CMDFAT(114)
        id = [10, 'gap', 13, 2]
CMDFAT(114) = ATTRIBUTION(115)
ATTRIBUTION(115) = NAMEFAT(116) 'instatrib' VALUE(117)
NAMEFAT(116) = epsilon
        instatrib = [50, '=', 13, 6]
VALUE(117) = Eb(118)
Eb(118) = Tb(119) Ebr(120)
Tb(119) = Fb(121) Tbr(122)
Fb(121) = Erel2(123) Fbr(124)
Erel2(123) = Erel1(125) Erel2r(126)
Erel1(125) = Ea(127) Erel1r(128)
Ea(127) = Ta(129) Ear(130)
Ta(129) = Fa(131)
Fa(131) = 'id' Far(132)
        id = [10, 'gap', 13, 8]
Far(132) = epsilon
Ear(130) = 'oparitmul' Ta(133) Ear(134)
        oparitmul = [39, '/', 13, 12]
Ta(133) = Fa(135)
Fa(135) = CONSTANT(136)
CONSTANT(136) = 'ctenumint'
        ctenumint = [25, '3', 13, 14]
Ear(134) = epsilon
Erel1r(128) = epsilon
Erel2r(126) = epsilon
Fbr(124) = epsilon
Tbr(122) = epsilon
Ebr(120) = epsilon
        termcmd = [24, ';', 13, 15]
COMMANDS(113) = CMD(137) 'termcmd' COMMANDS(138)
CMD(137) = FOR(139)
FOR(139) = 'repquia' 'id' 'repin' 'repspatium' 'parambegin' ITEMPARAM(140) 'sepvirg'
ITEMPARAM(141) 'sepvirg' ITEMPARAM(142) 'paramend' 'escbegin' COMMANDS(143) 'escend'
        repquia = [33, 'quia', 14, 2]
        id = [10, 'a', 14, 7]
        repin = [37, 'in', 14, 9]
        repspatium = [36, 'spatium', 14, 12]
```

```
        parambegin = [20, '(', 14, 19]
ITEMPARAM(140) = NAME(144)
NAME(144) = 'id' NAMEFAT(145)
        id = [10, 'gap', 14, 20]
NAMEFAT(145) = epsilon
        sepvirg = [23, ',', 14, 23]
ITEMPARAM(141) = NAME(146)
NAME(146) = 'id' NAMEFAT(147)
        id = [10, 'tamanho', 14, 24]
NAMEFAT(147) = epsilon
        sepvirg = [23, ',', 14, 31]
ITEMPARAM(142) = CONSTANT(148)
CONSTANT(148) = 'ctenumint'
        ctenumint = [25, '1', 14, 32]
        paramend = [21, ')', 14, 33]
        escbegin = [18, '{', 14, 34]
COMMANDS(143) = CMD(149) 'termcmd' COMMANDS(150)
CMD(149) = 'id' CMDFAT(151)
        id = [10, 'value', 15, 6]
CMDFAT(151) = ATTRIBUTION(152)
ATTRIBUTION(152) = NAMEFAT(153) 'instatrib' VALUE(154)
NAMEFAT(153) = epsilon
        instatrib = [50, '=', 15, 12]
VALUE(154) = Eb(155)
Eb(155) = Tb(156) Ebr(157)
Tb(156) = Fb(158) Tbr(159)
Fb(158) = Erel2(160) Fbr(161)
Erel2(160) = Erel1(162) Erel2r(163)
Erel1(162) = Ea(164) Erel1r(165)
Ea(164) = Ta(166) Ear(167)
Ta(166) = Fa(168)
Fa(168) = 'id' Far(169)
        id = [10, 'a', 15, 14]
Far(169) = 'vetbegin' Eb(170) 'vetend'
        vetbegin = [47, '[', 15, 15]
Eb(170) = Tb(171) Ebr(172)
Tb(171) = Fb(173) Tbr(174)
Fb(173) = Erel2(175) Fbr(176)
Erel2(175) = Erel1(177) Erel2r(178)
Erel1(177) = Ea(179) Erel1r(180)
Ea(179) = Ta(181) Ear(182)
Ta(181) = Fa(183)
Fa(183) = 'id' Far(184)
```

```
        id = [10, 'i', 15, 16]
Far(184) = epsilon
Ear(182) = epsilon
Erel1r(180) = epsilon
Erel2r(178) = epsilon
Fbr(176) = epsilon
Tbr(174) = epsilon
Ebr(172) = epsilon
        vetend = [48, ']', 15, 17]
Ear(167) = epsilon
Erel1r(165) = epsilon
Erel2r(163) = epsilon
Fbr(161) = epsilon
Tbr(159) = epsilon
Ebr(157) = epsilon
        termcmd = [24, ';', 15, 18]
COMMANDS(150) = CMD(185) 'termcmd' COMMANDS(186)
CMD(185) = 'id' CMDFAT(187)
        id = [10, 'j', 16, 6]
CMDFAT(187) = ATTRIBUTION(188)
ATTRIBUTION(188) = NAMEFAT(189) 'instatrib' VALUE(190)
NAMEFAT(189) = epsilon
        instatrib = [50, '=', 16, 8]
VALUE(190) = Eb(191)
Eb(191) = Tb(192) Ebr(193)
Tb(192) = Fb(194) Tbr(195)
Fb(194) = Erel2(196) Fbr(197)
Erel2(196) = Erel1(198) Erel2r(199)
Erel1(198) = Ea(200) Erel1r(201)
Ea(200) = Ta(202) Ear(203)
Ta(202) = Fa(204)
Fa(204) = 'id' Far(205)
        id = [10, 'i', 16, 10]
Far(205) = epsilon
Ear(203) = epsilon
Erel1r(201) = 'oparitad' Ea(206) Erel1r(207)
        oparitad = [38, '-', 16, 12]
Ea(206) = Ta(208) Ear(209)
Ta(208) = Fa(210)
Fa(210) = 'id' Far(211)
        id = [10, 'gap', 16, 14]
Far(211) = epsilon
Ear(209) = epsilon
```

```
Erel1r(207) = epsilon
Erel2r(199) = epsilon
Fbr(197) = epsilon
Tbr(195) = epsilon
Ebr(193) = epsilon
        termcmd = [24, ';', 16, 17]
COMMANDS(186) = CMD(212) 'termcmd' COMMANDS(213)
CMD(212) = WHILE(214)
WHILE(214) = 'repdum' 'parambegin' Eb(215) 'paramend' 'escbegin' COMMANDS(216)
'escend'
        repdum = [34, 'dum', 18, 6]
        parambegin = [20, '(', 18, 9]
Eb(215) = Tb(217) Ebr(218)
Tb(217) = Fb(219) Tbr(220)
Fb(219) = Erel2(221) Fbr(222)
Erel2(221) = Erel1(223) Erel2r(224)
Erel1(223) = Ea(225) Erel1r(226)
Ea(225) = Ta(227) Ear(228)
Ta(227) = Fa(229)
Fa(229) = 'id' Far(230)
        id = [10, 'j', 18, 10]
Far(230) = epsilon
Ear(228) = epsilon
Erel1r(226) = epsilon
Erel2r(224) = 'oprel1' Erel1(231) Erel2r(232)
        oprel1 = [44, '>=', 18, 12]
Erel1(231) = Ea(233) Erel1r(234)
Ea(233) = Ta(235) Ear(236)
Ta(235) = Fa(237)
Fa(237) = CONSTANT(238)
CONSTANT(238) = 'ctenumint'
        ctenumint = [25, '0', 18, 15]
Ear(236) = epsilon
Erel1r(234) = epsilon
Erel2r(232) = epsilon
Fbr(222) = epsilon
Tbr(220) = 'oplogand' Fb(239) Tbr(240)
        oplogand = [40, '&&', 18, 17]
Fb(239) = Erel2(241) Fbr(242)
Erel2(241) = Erel1(243) Erel2r(244)
Erel1(243) = Ea(245) Erel1r(246)
Ea(245) = Ta(247) Ear(248)
Ta(247) = Fa(249)
```

```
Fa(249) = 'id' Far(250)
        id = [10, 'value', 18, 20]
Far(250) = epsilon
Ear(248) = epsilon
Erel1r(246) = epsilon
Erel2r(244) = 'oprel1' Erel1(251) Erel2r(252)
        oprel1 = [44, '<', 18, 26]
Erel1(251) = Ea(253) Erel1r(254)
Ea(253) = Ta(255) Ear(256)
Ta(255) = Fa(257)
Fa(257) = 'id' Far(258)
        id = [10, 'a', 18, 28]
Far(258) = 'vetbegin' Eb(259) 'vetend'
        vetbegin = [47, '[', 18, 29]
Eb(259) = Tb(260) Ebr(261)
Tb(260) = Fb(262) Tbr(263)
Fb(262) = Erel2(264) Fbr(265)
Erel2(264) = Erel1(266) Erel2r(267)
Erel1(266) = Ea(268) Erel1r(269)
Ea(268) = Ta(270) Ear(271)
Ta(270) = Fa(272)
Fa(272) = 'id' Far(273)
        id = [10, 'j', 18, 30]
Far(273) = epsilon
Ear(271) = epsilon
Erel1r(269) = epsilon
Erel2r(267) = epsilon
Fbr(265) = epsilon
Tbr(263) = epsilon
Ebr(261) = epsilon
        vetend = [48, ']', 18, 31]
Ear(256) = epsilon
Erel1r(254) = epsilon
Erel2r(252) = epsilon
Fbr(242) = epsilon
Tbr(240) = epsilon
Ebr(218) = epsilon
        paramend = [21, ')', 18, 32]
        escbegin = [18, '{', 18, 34]
COMMANDS(216) = CMD(274) 'termcmd' COMMANDS(275)
CMD(274) = 'id' CMDFAT(276)
        id = [10, 'a', 19, 7]
CMDFAT(276) = ATTRIBUTION(277)
```

```
ATTRIBUTION(277) = NAMEFAT(278) 'instatrib' VALUE(279)
NAMEFAT(278) = 'vetbegin' Eb(280) 'vetend'
        vetbegin = [47, '[', 19, 8]
Eb(280) = Tb(281) Ebr(282)
Tb(281) = Fb(283) Tbr(284)
Fb(283) = Erel2(285) Fbr(286)
Erel2(285) = Erel1(287) Erel2r(288)
Erel1(287) = Ea(289) Erel1r(290)
Ea(289) = Ta(291) Ear(292)
Ta(291) = Fa(293)
Fa(293) = 'id' Far(294)
        id = [10, 'j', 19, 9]
Far(294) = epsilon
Ear(292) = epsilon
Erel1r(290) = 'oparitad' Ea(295) Erel1r(296)
        oparitad = [38, '+', 19, 11]
Ea(295) = Ta(297) Ear(298)
Ta(297) = Fa(299)
Fa(299) = 'id' Far(300)
        id = [10, 'gap', 19, 13]
Far(300) = epsilon
Ear(298) = epsilon
Erel1r(296) = epsilon
Erel2r(288) = epsilon
Fbr(286) = epsilon
Tbr(284) = epsilon
Ebr(282) = epsilon
        vetend = [48, ']', 19, 16]
        instatrib = [50, '=', 19, 18]
VALUE(279) = Eb(301)
Eb(301) = Tb(302) Ebr(303)
Tb(302) = Fb(304) Tbr(305)
Fb(304) = Erel2(306) Fbr(307)
Erel2(306) = Erel1(308) Erel2r(309)
Erel1(308) = Ea(310) Erel1r(311)
Ea(310) = Ta(312) Ear(313)
Ta(312) = Fa(314)
Fa(314) = 'id' Far(315)
        id = [10, 'a', 19, 20]
Far(315) = 'vetbegin' Eb(316) 'vetend'
        vetbegin = [47, '[', 19, 21]
Eb(316) = Tb(317) Ebr(318)
Tb(317) = Fb(319) Tbr(320)
```

```
Fb(319) = Erel2(321) Fbr(322)
Erel2(321) = Erel1(323) Erel2r(324)
Erel1(323) = Ea(325) Erel1r(326)
Ea(325) = Ta(327) Ear(328)
Ta(327) = Fa(329)
Fa(329) = 'id' Far(330)
        id = [10, 'j', 19, 22]
Far(330) = epsilon
Ear(328) = epsilon
Erel1r(326) = epsilon
Erel2r(324) = epsilon
Fbr(322) = epsilon
Tbr(320) = epsilon
Ebr(318) = epsilon
        vetend = [48, ']', 19, 23]
Ear(313) = epsilon
Erel1r(311) = epsilon
Erel2r(309) = epsilon
Fbr(307) = epsilon
Tbr(305) = epsilon
Ebr(303) = epsilon
        termcmd = [24, ';', 19, 24]
COMMANDS(275) = CMD(331) 'termcmd' COMMANDS(332)
CMD(331) = 'id' CMDFAT(333)
        id = [10, 'j', 20, 7]
CMDFAT(333) = ATTRIBUTION(334)
ATTRIBUTION(334) = NAMEFAT(335) 'instatrib' VALUE(336)
NAMEFAT(335) = epsilon
        instatrib = [50, '=', 20, 9]
VALUE(336) = Eb(337)
Eb(337) = Tb(338) Ebr(339)
Tb(338) = Fb(340) Tbr(341)
Fb(340) = Erel2(342) Fbr(343)
Erel2(342) = Erel1(344) Erel2r(345)
Erel1(344) = Ea(346) Erel1r(347)
Ea(346) = Ta(348) Ear(349)
Ta(348) = Fa(350)
Fa(350) = 'id' Far(351)
        id = [10, 'j', 20, 11]
Far(351) = epsilon
Ear(349) = epsilon
Erel1r(347) = 'oparitad' Ea(352) Erel1r(353)
        oparitad = [38, '-', 20, 13]
```

```
Ea(352) = Ta(354) Ear(355)
Ta(354) = Fa(356)
Fa(356) = 'id' Far(357)
        id = [10, 'gap', 20, 15]
Far(357) = epsilon
Ear(355) = epsilon
Erel1r(353) = epsilon
Erel2r(345) = epsilon
Fbr(343) = epsilon
Tbr(341) = epsilon
Ebr(339) = epsilon
        termcmd = [24, ';', 20, 18]
COMMANDS(332) = epsilon
        escend = [19, '}', 21, 6]
        termcmd = [24, ';', 21, 7]
COMMANDS(213) = CMD(358) 'termcmd' COMMANDS(359)
CMD(358) = WRITE(360)
WRITE(360) = 'prscribo' 'parambegin' MESSAGE(361) 'paramend'
        prscribo = [6, 'scribo', 23, 6]
        parambegin = [20, '(', 23, 13]
MESSAGE(361) = 'ctesermo' MESSAGEFAT(362)
        ctesermo = [28, '"chegou aqui!"', 23, 14]
MESSAGEFAT(362) = epsilon
        paramend = [21, ')', 23, 28]
        termcmd = [24, ';', 23, 29]
COMMANDS(359) = CMD(363) 'termcmd' COMMANDS(364)
CMD(363) = 'id' CMDFAT(365)
        id = [10, 'a', 24, 6]
CMDFAT(365) = ATTRIBUTION(366)
ATTRIBUTION(366) = NAMEFAT(367) 'instatrib' VALUE(368)
NAMEFAT(367) = 'vetbegin' Eb(369) 'vetend'
        vetbegin = [47, '[', 24, 7]
Eb(369) = Tb(370) Ebr(371)
Tb(370) = Fb(372) Tbr(373)
Fb(372) = Erel2(374) Fbr(375)
Erel2(374) = Erel1(376) Erel2r(377)
Erel1(376) = Ea(378) Erel1r(379)
Ea(378) = Ta(380) Ear(381)
Ta(380) = Fa(382)
Fa(382) = 'id' Far(383)
        id = [10, 'j', 24, 8]
Far(383) = epsilon
Ear(381) = epsilon
```

```
Erel1r(379) = 'oparitad' Ea(384) Erel1r(385)
        oparitad = [38, '+', 24, 10]
Ea(384) = Ta(386) Ear(387)
Ta(386) = Fa(388)
Fa(388) = 'id' Far(389)
        id = [10, 'gap', 24, 12]
Far(389) = epsilon
Ear(387) = epsilon
Erel1r(385) = epsilon
Erel2r(377) = epsilon
Fbr(375) = epsilon
Tbr(373) = epsilon
Ebr(371) = epsilon
        vetend = [48, ']', 24, 15]
        instatrib = [50, '=', 24, 17]
VALUE(368) = Eb(390)
Eb(390) = Tb(391) Ebr(392)
Tb(391) = Fb(393) Tbr(394)
Fb(393) = Erel2(395) Fbr(396)
Erel2(395) = Erel1(397) Erel2r(398)
Erel1(397) = Ea(399) Erel1r(400)
Ea(399) = Ta(401) Ear(402)
Ta(401) = Fa(403)
Fa(403) = 'id' Far(404)
        id = [10, 'value', 24, 19]
Far(404) = epsilon
Ear(402) = epsilon
Erel1r(400) = epsilon
Erel2r(398) = epsilon
Fbr(396) = epsilon
Tbr(394) = epsilon
Ebr(392) = epsilon
        termcmd = [24, ';', 24, 24]
COMMANDS(364) = epsilon
        escend = [19, '}', 25, 4]
        termcmd = [24, ';', 25, 5]
COMMANDS(138) = epsilon
        escend = [19, '}', 26, 1]
        repdum = [34, 'dum', 26, 2]
        parambegin = [20, '(', 26, 5]
Eb(111) = Tb(405) Ebr(406)
Tb(405) = Fb(407) Tbr(408)
Fb(407) = Erel2(409) Fbr(410)
```

```
Erel2(409) = Erel1(411) Erel2r(412)
Erel1(411) = Ea(413) Erel1r(414)
Ea(413) = Ta(415) Ear(416)
Ta(415) = Fa(417)
Fa(417) = 'id' Far(418)
        id = [10, 'gap', 26, 6]
Far(418) = epsilon
Ear(416) = epsilon
Erel1r(414) = epsilon
Erel2r(412) = 'oprel1' Erel1(419) Erel2r(420)
        oprel1 = [44, '>', 26, 10]
Erel1(419) = Ea(421) Erel1r(422)
Ea(421) = Ta(423) Ear(424)
Ta(423) = Fa(425)
Fa(425) = CONSTANT(426)
CONSTANT(426) = 'ctenumint'
        ctenumint = [25, '1', 26, 12]
Ear(424) = epsilon
Erel1r(422) = epsilon
Erel2r(420) = epsilon
Fbr(410) = epsilon
Tbr(408) = epsilon
Ebr(406) = epsilon
        paramend = [21, ')', 26, 13]
        termcmd = [24, ';', 26, 14]
COMMANDS(108) = epsilon
        escend = [19, '}', 28, 0]
        termcmd = [24, ';', 28, 1]
FUNCTIONS(8) = epsilon
INITIUM(4) = 'prinitials' 'tdinanis' 'prinitium' 'parambegin' 'paramend' SCOPE(427)
        prinitials = [3, 'initials', 30, 0]
        tdinanis = [11, 'inanis', 30, 9]
        prinitium = [2, 'initium', 30, 16]
        parambegin = [20, '(', 30, 23]
        paramend = [21, ')', 30, 24]
SCOPE(427) = 'escbegin' COMMANDS(428) 'escend' 'termcmd'
        escbegin = [18, '{', 30, 25]
COMMANDS(428) = CMD(429) 'termcmd' COMMANDS(430)
CMD(429) = WRITE(431)
WRITE(431) = 'prscribo' 'parambegin' MESSAGE(432) 'paramend'
        prscribo = [6, 'scribo', 32, 4]
        parambegin = [20, '(', 32, 10]
MESSAGE(432) = 'ctesermo' MESSAGEFAT(433)
```

```
        ctesermo=[28,'"Digite a quantidade de numeros a serem ordenados: "',32,11]
MESSAGEFAT(433) = epsilon
        paramend = [21, ')', 32, 63]
        termcmd = [24, ';', 32, 64]
COMMANDS(430) = CMD(434) 'termcmd' COMMANDS(435)
CMD(434) = READ(436)
READ(436) = 'prlectio' 'parambegin' NAME(437) 'paramend'
        prlectio = [8, 'lectio', 33, 4]
        parambegin = [20, '(', 33, 10]
NAME(437) = 'id' NAMEFAT(438)
        id = [10, 'n', 33, 11]
NAMEFAT(438) = epsilon
        paramend = [21, ')', 33, 12]
        termcmd = [24, ';', 33, 13]
COMMANDS(435) = CMD(439) 'termcmd' COMMANDS(440)
CMD(439) = DECLARATION(441)
DECLARATION(441) = 'prmatrix' TYPE(442) NAME(443)
        prmatrix = [17, 'matrix', 35, 4]
TYPE(442) = 'tdint'
        tdint = [12, 'integer', 35, 11]
NAME(443) = 'id' NAMEFAT(444)
        id = [10, 'vetor', 35, 19]
NAMEFAT(444) = 'vetbegin' Eb(445) 'vetend'
        vetbegin = [47, '[', 35, 24]
Eb(445) = Tb(446) Ebr(447)
Tb(446) = Fb(448) Tbr(449)
Fb(448) = Erel2(450) Fbr(451)
Erel2(450) = Erel1(452) Erel2r(453)
Erel1(452) = Ea(454) Erel1r(455)
Ea(454) = Ta(456) Ear(457)
Ta(456) = Fa(458)
Fa(458) = 'id' Far(459)
        id = [10, 'n', 35, 25]
Far(459) = epsilon
Ear(457) = epsilon
Erel1r(455) = epsilon
Erel2r(453) = epsilon
Fbr(451) = epsilon
Tbr(449) = epsilon
Ebr(447) = epsilon
        vetend = [48, ']', 35, 26]
        termcmd = [24, ';', 35, 27]
COMMANDS(440) = CMD(460) 'termcmd' COMMANDS(461)
```

```
CMD(460) = WRITE(462)
WRITE(462) = 'prscribo' 'parambegin' MESSAGE(463) 'paramend'
        prscribo = [6, 'scribo', 37, 4]
        parambegin = [20, '(', 37, 10]
MESSAGE(463) = 'ctesermo' MESSAGEFAT(464)
        ctesermo = [28, '"Digite os numeros:"', 37, 11]
MESSAGEFAT(464) = epsilon
        paramend = [21, ')', 37, 31]
        termcmd = [24, ';', 37, 32]
COMMANDS(461) = CMD(465) 'termcmd' COMMANDS(466)
CMD(465) = FOR(467)
FOR(467) = 'repquia' 'id' 'repin' 'repspatium' 'parambegin' ITEMPARAM(468) 'sepvirg'
ITEMPARAM(469) 'sepvirg' ITEMPARAM(470) 'paramend' 'escbegin' COMMANDS(471) 'escend'
        repquia = [33, 'quia', 39, 1]
        id = [10, 'a', 39, 6]
        repin = [37, 'in', 39, 8]
        repspatium = [36, 'spatium', 39, 11]
        parambegin = [20, '(', 39, 18]
ITEMPARAM(468) = CONSTANT(472)
CONSTANT(472) = 'ctenumint'
        ctenumint = [25, '1', 39, 19]
        sepvirg = [23, ',', 39, 20]
ITEMPARAM(469) = NAME(473)
NAME(473) = 'id' NAMEFAT(474)
        id = [10, 'n', 39, 21]
NAMEFAT(474) = epsilon
        sepvirg = [23, ',', 39, 22]
ITEMPARAM(470) = CONSTANT(475)
CONSTANT(475) = 'ctenumint'
        ctenumint = [25, '1', 39, 23]
        paramend = [21, ')', 39, 24]
        escbegin = [18, '{', 39, 25]
COMMANDS(471) = CMD(476) 'termcmd' COMMANDS(477)
CMD(476) = READ(478)
READ(478) = 'prlectio' 'parambegin' NAME(479) 'paramend'
        prlectio = [8, 'lectio', 40, 5]
        parambegin = [20, '(', 40, 11]
NAME(479) = 'id' NAMEFAT(480)
        id = [10, 'vetor', 40, 12]
NAMEFAT(480) = 'vetbegin' Eb(481) 'vetend'
        vetbegin = [47, '[', 40, 17]
Eb(481) = Tb(482) Ebr(483)
Tb(482) = Fb(484) Tbr(485)
```

```
Fb(484) = Erel2(486) Fbr(487)
Erel2(486) = Erel1(488) Erel2r(489)
Erel1(488) = Ea(490) Erel1r(491)
Ea(490) = Ta(492) Ear(493)
Ta(492) = Fa(494)
Fa(494) = 'id' Far(495)
        id = [10, 'a', 40, 18]
Far(495) = epsilon
Ear(493) = epsilon
Erel1r(491) = epsilon
Erel2r(489) = epsilon
Fbr(487) = epsilon
Tbr(485) = epsilon
Ebr(483) = epsilon
        vetend = [48, ']', 40, 19]
        paramend = [21, ')', 40, 20]
        termcmd = [24, ';', 40, 21]
COMMANDS(477) = epsilon
        escend = [19, '}', 41, 4]
        termcmd = [24, ';', 41, 5]
COMMANDS(466) = CMD(496) 'termcmd' COMMANDS(497)
CMD(496) = 'id' CMDFAT(498)
        id = [10, 'shell', 43, 4]
CMDFAT(498) = FUNCCALL(499)
FUNCCALL(499) = 'parambegin' LISTPARAMSCALL(500) 'paramend'
        parambegin = [20, '(', 43, 9]
LISTPARAMSCALL(500) = ITEMPARAM(501) LISTPARAMSCALLFAT(502)
ITEMPARAM(501) = NAME(503)
NAME(503) = 'id' NAMEFAT(504)
        id = [10, 'n', 43, 10]
NAMEFAT(504) = epsilon
LISTPARAMSCALLFAT(502) = 'sepvirg' LISTPARAMSCALL(505)
        sepvirg = [23, ',', 43, 11]
LISTPARAMSCALL(505) = ITEMPARAM(506) LISTPARAMSCALLFAT(507)
ITEMPARAM(506) = NAME(508)
NAME(508) = 'id' NAMEFAT(509)
        id = [10, 'vetor', 43, 12]
NAMEFAT(509) = epsilon
LISTPARAMSCALLFAT(507) = epsilon
        paramend = [21, ')', 43, 17]
        termcmd = [24, ';', 43, 18]
COMMANDS(497) = CMD(510) 'termcmd' COMMANDS(511)
CMD(510) = WRITE(512)
```

```
WRITE(512) = 'prscribo' 'parambegin' MESSAGE(513) 'paramend'
        prscribo = [6, 'scribo', 45, 4]
        parambegin = [20, '(', 45, 10]
MESSAGE(513) = 'ctesermo' MESSAGEFAT(514)
        ctesermo = [28, '"Numeros ordenados: "', 45, 11]
MESSAGEFAT(514) = epsilon
        paramend = [21, ')', 45, 32]
        termcmd = [24, ';', 45, 33]
COMMANDS(511) = CMD(515) 'termcmd' COMMANDS(516)
CMD(515) = FOR(517)
FOR(517) = 'repquia' 'id' 'repin' 'repspatium' 'parambegin' ITEMPARAM(518) 'sepvirg'
ITEMPARAM(519) 'sepvirg' ITEMPARAM(520) 'paramend' 'escbegin' COMMANDS(521) 'escend'
        repquia = [33, 'quia', 47, 1]
        id = [10, 'a', 47, 6]
        repin = [37, 'in', 47, 8]
        repspatium = [36, 'spatium', 47, 11]
        parambegin = [20, '(', 47, 18]
ITEMPARAM(518) = CONSTANT(522)
CONSTANT(522) = 'ctenumint'
        ctenumint = [25, '1', 47, 19]
        sepvirg = [23, ',', 47, 20]
ITEMPARAM(519) = NAME(523)
NAME(523) = 'id' NAMEFAT(524)
        id = [10, 'n', 47, 21]
NAMEFAT(524) = epsilon
        sepvirg = [23, ',', 47, 22]
ITEMPARAM(520) = CONSTANT(525)
CONSTANT(525) = 'ctenumint'
        ctenumint = [25, '1', 47, 23]
        paramend = [21, ')', 47, 24]
        escbegin = [18, '{', 47, 25]
COMMANDS(521) = CMD(526) 'termcmd' COMMANDS(527)
CMD(526) = WRITE(528)
WRITE(528) = 'prscribo' 'parambegin' MESSAGE(529) 'paramend'
        prscribo = [6, 'scribo', 48, 5]
        parambegin = [20, '(', 48, 11]
MESSAGE(529) = NAME(530) MESSAGEFAT(531)
NAME(530) = 'id' NAMEFAT(532)
        id = [10, 'vetor', 48, 12]
NAMEFAT(532) = 'vetbegin' Eb(533) 'vetend'
        vetbegin = [47, '[', 48, 17]
Eb(533) = Tb(534) Ebr(535)
Tb(534) = Fb(536) Tbr(537)
```

```
Fb(536) = Erel2(538) Fbr(539)
Erel2(538) = Erel1(540) Erel2r(541)
Erel1(540) = Ea(542) Erel1r(543)
Ea(542) = Ta(544) Ear(545)
Ta(544) = Fa(546)
Fa(546) = 'id' Far(547)
        id = [10, 'a', 48, 18]
Far(547) = epsilon
Ear(545) = epsilon
Erel1r(543) = epsilon
Erel2r(541) = epsilon
Fbr(539) = epsilon
Tbr(537) = epsilon
Ebr(535) = epsilon
        vetend = [48, ']', 48, 19]
MESSAGEFAT(531) = 'opcon' MESSAGE(548)
        opcon = [46, '.', 48, 21]
MESSAGE(548) = 'ctesermo' MESSAGEFAT(549)
        ctesermo = [28, '" "', 48, 23]
MESSAGEFAT(549) = epsilon
        paramend = [21, ')', 48, 26]
        termcmd = [24, ';', 48, 27]
COMMANDS(527) = epsilon
        escend = [19, '}', 49, 1]
        termcmd = [24, ';', 49, 2]
COMMANDS(516) = epsilon
        escend = [19, '}', 51, 0]
        termcmd = [24, ';', 51, 1]
```