|  |  |
| --- | --- |
| http://www.brandsoftheworld.com/sites/default/files/styles/logo-thumbnail/public/112011/untitled-1_60.png | FACULDADE DOS GUARARAPES  **DEPARTAMENTO DE ENGENHARIA E TECNOLOGIA**  **BACHARELADO EM CIÊNCIAS DA COMPUTAÇÃO** |

**COMPILADORES**

**ANALISADOR LÉXICO E SINTÁTICO**

**COM JAVACC**

Elaborador por:

Bruno Antonio do Nascimento – Mat. 201501486

Felipe Blanco da Guarda – 201602200

Marcelino de Arruda Castor – 201600466

Jaboatão dos Guararapes

2017

|  |  |
| --- | --- |
| **FACULDADE DOS GUARARAPES** | **Grupo**  Bruno Antonio do Nascimento – Mat. 201501486  Felipe Blanco da Guarda – 201602200  Marcelino de Arruda Castor – 201600466 |

**COMPILADORES**

**ANALISADOR LÉXICO E SINTÁTICO**

**COM JAVACC**

Atividade escrita para o Curso de Bacharel em Ciência Computação da Faculdade dos Guararapes valendo ponto extra na U2 para a disciplina de Compiladores

Professor: Klaus Ribeiro Cavalcanti

Jaboatão dos Guararapes

2017

Sumário

Sumário 3

gRAMáTICA 3

parser 4

gRAMáTICA

Programa ::= Declaracoes

Declaracoes ::= (Variavel | Record)\*

Record ::= 'record' Identificador (('is' (Variavel)+ 'endrecord' ';' )

| ('('((String | Inteiro | Real)(',')?)+ ')')?)

Variavel ::= Identificador ':' TArg ';'

TArg ::= (VArray | Record | Tipo)

VArray ::= 'array' Inteiro '..' Inteiro (('of' Tipo)

| ('of record' Identificador))

Tipo ::= 'string' | 'integer' | 'real'

Identificador ::= Letra (Letra | Digito)\*

String ::= "\"" (~["\"","\n","\r"])\* "\""

Inteiro ::= Digito+

Real ::= Inteiro('.'Inteiro)?

Letra ::= [a-z] | [A-Z]

Digito ::= [0-9]

parser

options

{

STATIC = false;

}

PARSER\_BEGIN(Parser)

package javacc;

public class Parser

{

public static void main(String [] args) throws ParseException,

TokenMgrError

{

Parser parser;

if (args.length == 0)

{

System.out.println("Parser: Reading from standard input... ");

parser = new Parser(System.in);

}

else if (args.length == 1)

{

System.out.println("Parser: Reading from file " + args [0] + "...");

try

{

parser = new Parser(new java.io.FileInputStream(args [0]));

}

catch (java.io.FileNotFoundException e)

{

System.out.println("Parser: File " + args [0] + " not found.");

return;

}

}

else

{

System.out.println("Parser: Usage is one of:");

System.out.println(" java Parser < inputfile");

System.out.println("OR");

System.out.println(" java Scanner inputfile");

return;

}

try

{

parser.Start();

System.out.println("Parser: Program parsed successfully.");

}

catch (ParseException e)

{

System.out.println("Parser: Encountered errors on parsing.");

e.printStackTrace();

}

}

}

PARSER\_END(Parser)

SKIP :

{

" "

| "\n"

| "\r"

| "\r\n"

}

//PALAVRAS RESERVADAS

TOKEN : { < TSTRING : "string" > }

TOKEN : { < TINTEIRO : "integer" > }

TOKEN : { < TREAL : "real" > }

TOKEN : { < ARRAY : "array" > }

TOKEN : { < RECORD : "record" > }

TOKEN : { < IS : "is" > }

TOKEN : { < ENDRECORD : "endrecord" >}

TOKEN : { < OF : "of" > }

TOKEN : { < OFRECORD : "of record" > }

//TIPOS DE DADOS

TOKEN : { < STRING : "\"" (~["\n","\r"])\* "\"" > }

TOKEN : { < INTEIRO : (< DIGITO >)+ > }

TOKEN : { < REAL : <INTEIRO >(< PONTO > < INTEIRO >)? > }

TOKEN : { < IDENTIFICADOR : < LETRA > (< LETRA > | < DIGITO >)\* > }

// CARACTERES

TOKEN : { < LETRA : ["a"-"z","A"-"Z"] > }

TOKEN : { < DIGITO : ["0"-"9"] > }

// SIMBOLOS

TOKEN : { < DOISPONTOS : ":" > }

TOKEN : { < PONTOEVIRGULA : ";" > }

TOKEN : { < INTERVALO : ".." > }

TOKEN : { < VIRGULA : "," > }

TOKEN : { < PARENTEESQ : "(" > }

TOKEN : { < PARENTEDIR : ")" > }

TOKEN : { < PONTO : "." > }

void Start() :

{}

{

Declaracoes()

< EOF >

}

void Declaracoes() :

{}

{

(Variavel() | Record())\*

}

void Record() :

{}

{

< RECORD > <IDENTIFICADOR >(( < IS > (Variavel() )+ < ENDRECORD > < PONTOEVIRGULA > )

| ( < PARENTEESQ >((< STRING > | < INTEIRO > | < REAL >) (< VIRGULA >)? )+ < PARENTEDIR > )? )

}

void Variavel() :

{}

{

< IDENTIFICADOR > < DOISPONTOS > TArg() < PONTOEVIRGULA >

}

void TArg() :

{}

{

VArray() | Record() | Tipo()

}

void VArray():

{}

{

<ARRAY > <INTEIRO > <INTERVALO > <INTEIRO > ((< OF > Tipo())

| (< OFRECORD > <IDENTIFICADOR>))

}

void Tipo() :

{}

{

< TSTRING > | < TINTEIRO > | < TREAL >

}