|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Universidade Federal da Bahia**  Instituto de Matemática e Estatistica  Departamento de Ciência da Computação | | | | |
|  | Disciplina: LFA | Docente: Marlo Souza | 07/09/17 | Ciência da Computação |
| Discente: **Jorge Ueiner, Mateus Barbosa e Otávio Augusto** | | | |

**PARSER PARA SIMPLE PASCAL**

# eXPRESSÕES REGULARES

As expressões regulares que definem as palavras reservadas da linguagem são apresentadas a seguir:

* Palavras de escopo: (*program* + *var);*
* Identificadores: *(a\*b\*c\*d\*e\*f\*g\*h\*i\*j\*k\*l\*m\*n\*o\*p\*q\*r\*s\*t\*u\*v\*x\*w\*y\*z\* A\*B\*C\*D\*E\*F\*G\*H\*I\*J\*K\*L\*M\*N\*O\*P\*Q\*R\*S\*T\*U\*V\*X\*W\*Y\*Z)+.(a\*b\*c\*d \*e\*f\*g\*h\*i\*j\*k\*l\*m\*n\*o\*p\*q\*r\*s\*t\*u\*v\*x\*w\*y\*z\* A\*B\*C\*D\*E\*F\*G\*H\*I\*J\*K\*L\* M\*N\*O\*P\*Q\*R\*S\*T\*U\*V\*X\*W\*Y\*Z\*1\*2\*3\*4\*5\*6\*7\*8\*9\*0\*\_\*)\**;
* Delimitadores: (*begin + end);*
* Operadores de lógicos: ( not + and +or + \\* +div + mod + = + <> + < + > + <= + => + \( + \) ); Obs: “\” representa barra de escape.
* Operadores de atribuição: :=;
* Operadores de fluxo: (if + then + else + while + do);

# GRAMÁTICA PROPOSTA

A gramática proposta para a linguagem Simple Pascal é dada por G = (V, T, S, P), onde:

* V = { ***S***, ***A***, ***B***, ***C***, ***D***, ***E***, ***F***, ***V***, ***R***, ***P***, ***N***, ***M*** };
* T = { λ, *a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, x , w, y, z, A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, X, W, Y, Z, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, \_*, =, <, >,:, .,\*, , ; };
* S;
* P = { ***S*** -> program ***A*** ; ***V*** ***C*** .

***A*** -> a***B*** **|** A***B |*** b***B |*** B***B*** *|*c***B* |** C***B* |** d***B* |** D***B* |** e***B* |** E***B* |** f***B* |** F***B* |** g***B* |** h***B* |** H***B* |** i***B* |** I***B* |** j***B* |** J***B* |** k***B* |** K***B* |** l***B* |** L***B* |** m***B* |** M***B* |** n***B*** **|** N***B* |** o***B*** **|** O***B* |**p***B* |**P***B* |**q***B* |**Q***B* |** r***B* |** R***B* |** s***B*** **|** S***B* |** t***B*** **|** T***B*** **|** u***B* |** U***B* |** v***B*** **|** V***B*** x***B* |**X***B* |** w***B* |**W***B* |**y***B* |**Y***B* |**z***B* |**Z***B***

***B*** -> a***B*** **|** A***B |*** b***B |*** B***B*** *|*c***B* |** C***B* |** d***B* |** D***B* |** e***B* |** E***B* |** f***B* |** F***B* |** g***B* |** h***B* |** H***B* |** i***B* |** I***B* |** j***B* |** J***B* |** k***B* |** K***B* |** l***B* |** L***B* |** m***B* |** M***B* |** n***B*** **|** N***B* |** o***B*** **|** O***B* |**p***B* |**P***B* |**q***B* |**Q***B* |** r***B* |** R***B* |** s***B*** **|** S***B* |** t***B*** **|** T***B*** **|** u***B* |** U***B* |** v***B*** **|** V***B*** x***B* |**X***B* |** w***B* |**W***B* |**y***B* |**Y***B* |**z***B* |**Z***B* |** 0***B* |**1***B* |** 2***B* |** 3***B* |**4***B* |**5***B* |** 6***B* |** 7***B*** **|** 8***B* |** 9***B* |** \_***B* |** λ

***N*** -> 0***M* |**1***M* |** 2***M* |** 3***M* |**4***M* |**5***M* |** 6***M* |** 7***M*** **|** 8***M* |** 9***M***

***M*** -> 0***M* |**1***M* |** 2***M* |** 3***M* |**4***M* |**5***M* |** 6***M* |** 7***M*** **|** 8***M* |** 9***M* |**λ

***V*** -> var ***D***

***D*** -> ***AE*** : int; **|** ***AE*** :bool; **| *AE*** : real;**| *D* |** λ

***E*** ->, ***AE* |** λ

***C*** -> begin ***F*** end

***F*** -> ***A*** := ***A* | *A*** := ***N* | *A*** := ***P* | *A*** := ***R* | *A*** := ***R* |** if(***P***) then ***C* |** if(***P***) then ***C*** else ***C* |** while(***P***) do ***C* |** λ

***P*** -> ***A*** <> ***A* | *A*** <= ***A* | *A*** => ***A* | *A*** > ***A* |** True **|** False **| *A*** < ***A* | *A*** and ***A* | *A*** or ***A* |** not ***A* | *A*** <> ***N* | *A*** <= ***N* | *A*** => ***N* | *A*** > ***N* | *A*** < ***N* | *N*** <> ***A* | *N*** <= ***A* | *N*** => ***A* | *N*** > ***A* | *N*** < ***A* | *A*** and ***P* | *A*** or ***P* | *A*** not ***P* | *P*** or ***P* | *P*** and ***P* | *P*** or ***A* | *P*** and ***A* | *R*** <> ***R* | *R*** <= ***R* | *R*** => ***R* | *R*** > ***R* | *R*** < ***R* | *R*** <> ***A* | *R*** <= ***A* | *R*** => ***A* | *R*** > ***A* | *R*** < ***A* | *A*** <> ***R* | *A*** <= ***R* | *A*** => ***R* | *A*** > ***R* | *A*** < ***R* | *R*** <> ***N* | *R*** <= ***N* | *R*** => ***N* | *R*** > ***N* | *R*** < ***N* | *N*** <> ***R* | *N*** <= ***R* | *N*** => ***R* | *N*** > ***R* | *N*** < ***R***

***R*** -> ***N*** mod ***N* | *N*** div ***N* | *N* \* *N* | *A*** mod ***A* | *A*** div ***A* | *A* \* *A* | *R*** mod ***R* | *R*** div ***R* | *R* \* *R* | *A*** mod ***N* | *A*** div ***N* | *A* \* *N* | *N*** mod ***A* | *N*** div ***A* | *N* \* *A* | *R* \* *A* | *R*** mod ***A* | *R*** div ***A* | *A* \* *R* | *A*** mod ***R* | *A*** div ***R* | *R* \* *N* | *R*** mod ***N* | *R*** div ***N* | *N* \* *R* | *N*** mod ***R* | *N*** div ***R***

# ANALISADOR SINTÁTICO

Para gera o analisador sintático, inicialmente foi criado um arquivo g.l para definir o analisador léxico (definição de tokens). Posteriormente, foi criado o arquivo g.y com as definições das regras da gramatica na sintaxe do bison definidas no tópico **2**.

Os comandos utilizados para gerar o analisador sintático (parser) são apresentados a seguir:

* C:\Users\Mateus Barbosa\Desktop\trabalho-lfat>bison -d g.y
* C:\Users\Mateus Barbosa\Desktop\trabalho-lfat>flex g.l
* C:\Users\Mateus Barbosa\Desktop\trabalho-lfat>gcc -o g.exe g.tab.c lex.yy.c

Com isso foram gerados os arquivos y.tab.g, y.tab.c, lex.yy.c e por fim foi gerado g.exe. Em seguida, o programa foi executado para as entradas especificadas no tópico 4.

# Discussão das entradas

1. Caso de Teste:

program test;

var num: int;

begin

num := 23;

end.

Ao executar o programa com essa entrada o parser não identificou nenhum erro.

1. Caso de Teste:

program test1;

var num : bool;

begin

num := True;

if(num then

Ao executar o programa com essa entrada o parser identificou erro de sintaxe e exibiu uma mensagem de erro indicando a linha com o if.

1. Caso de Teste:

program test2

begin

Ao executar o programa com essa entrada o parser identificou erro de sintaxe devido a falta do ponto virgula e exibiu uma mensagem de erro indicando a linha com o begin.