UNIVERSIDADE FEDERAL DE ITAJUBÁ

DOCUMENTAÇÃO DOTCODE

FERNANDO COSTA LASMAR - 2021024278
CARLOS HENRIQUE CRUZ XAVIER - 2021015751

ITAJUBÁ
2023

1. EXPRESSÕES REGULARES

# Tipos de dados
tipoVar = (“int” | “real” | “letra” | “palavra”)
palavra = (_) (letra)+

# Operadores relacionais
relacional = (“==” | ”>=” | ”<=” | ”<” | ”>” | ”!=”)

# Operadores lógicos
logico = (“and” | “or” | “not”)

# Operadores aritméticos
aritmetico = (“+” | “-” | “/” | “*” | “%”)

# Símbolos especiais
pontoV = “;”
virgula = “,”
abreP = “(”
fechaP = “)”
comentário = ”#”
abreAsp = “
fechaAsp = “

# Bloco de comando
abreC = “{”
fechaC = “}”
inicioP = “begin”
fimP = “end”

# Comandos de atribuição
sinalAtr = “|=”

atribuicao = atribuicao = (tipoVar)(nomeVar)(sinalAtr) (digito { ponto | [aritmetico {digito | nomeVar} ponto]} | letra+ ponto | nomeVar {ponto | [aritmetico {digito | nomeVar} ponto]} | palavra ponto)

# Comandos de entrada e saída
entrada = “read”
entrada = (nomeVar)(sinalAtr)(read)(abreP)(fechaP)(ponto)

saida = “write”
saida = (write)(abreP)(abreAsp)(letra | num | [abreC nomeVar fechaC])*(fechaAsp)(fechaP)(ponto)

# Comandos condicionais
condicional = “if”
desviocondicional = “else”

teste = (var)(relacional)(var)
condicional = (if)(abreP)(teste)(logico teste)*(fechaP)(abreC)(expressao+)(fechaC)

# Comandos de repetição
repeticao = “while” | “for”

enquanto = (while)(abreP)(teste)(virgula teste)*(fechaP)(abreC)(expressao*)(fechaC)
para = (for)(nomeVar)(range)(abreP)(num)(fechaP)(abreC)(expressao)(fechaC)

# Palavras reservadas
sinais = “+” | “-”
num = [0…9]
char = [a…z, A…z]
ponto = “.”
under = “_”
entrada = “read”
saida = “write”
alcance = “range”

2. TOKENS

DEFINIÇÃO	TOKEN
tipoVar	TIPOVAR
relacional	RELACIONAL
logico	LOGICO
aritmetico	ARITMETICO
pontoV	PONTOV
virgula	VIRGULA
abreP	ABREP
fechaP	FECHAP
comentários	COMENTARIOS

abreC	ABREC
fechaC	FECHAC
inicioP	INICIOP
fimP	FIMP
atribuição	ATRIBUICAO
entrada	READ
saída	WRITE
digito	DIGITO
abreAsp	ABREASP
fechaAsp	FECHAASP
sinalAtr	SINALATR
sinais	SINAIS
num	NUM
letra	LETRA
ponto	PONTO
para	FOR
enquanto	WHILE
condicional	IF
desviocondicional	ELSE
expressao	EXPRESSAO
nomeVar	NOMEVAR
string	PALAVRA
teste	TESTE
char	CHAR
comentarios	HASH

3. AUTÔMATOS

 # inicioPrograma = begin abreC expressao* fechaC end
Regra para definir o início e o fim do programa.

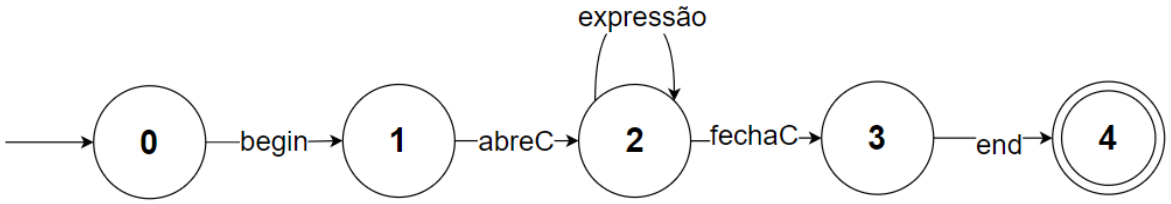

Figura 1 - INICIO / FIM PROGRAMA

 # nomeVar = letra (num | letra)*

A expressão regular acima representa o padrão de nomeação de uma variável, exigindo pelo menos um letra inicial e permitindo repetição de letras e/ou números a partir dessa posição.

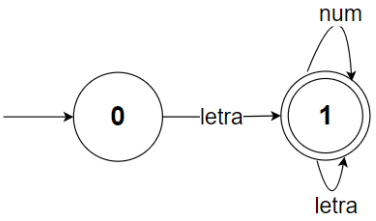

Figura 2 - TOKEN NOMEVAR

 # digito = (sinal)? num+ [ponto(num)+]?

Essa expressão regular é flexível para corresponder a diferentes formas de números, sendo eles inteiros, decimais e com ou sem sinal.

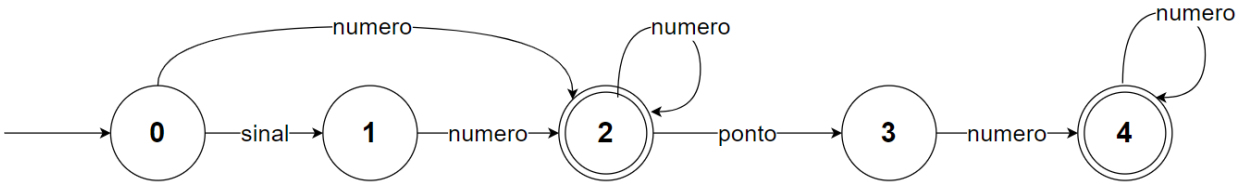

Figura 3 - TOKEN DIGITO

 # atribuicao = (tipoVar)(nomeVar)(sinalAtr)
(digito { ponto | [aritmetico {digito | nomeVar} ponto]} |
letra+ ponto |
nomeVar {ponto | [aritmetico {digito | nomeVar} ponto]} |
palavra ponto)

Essa expressão é flexível para corresponder a diferentes formas de atribuições, requerendo, primariamente, a especificação de uma variável destinatária para a qual será atribuído um valor. O valor atribuído pode assumir a forma de um dígito, uma letra, uma string, uma palavra, outra variável já definida, e também uma expressão aritmética entre duas variáveis ou/e dois dígitos.

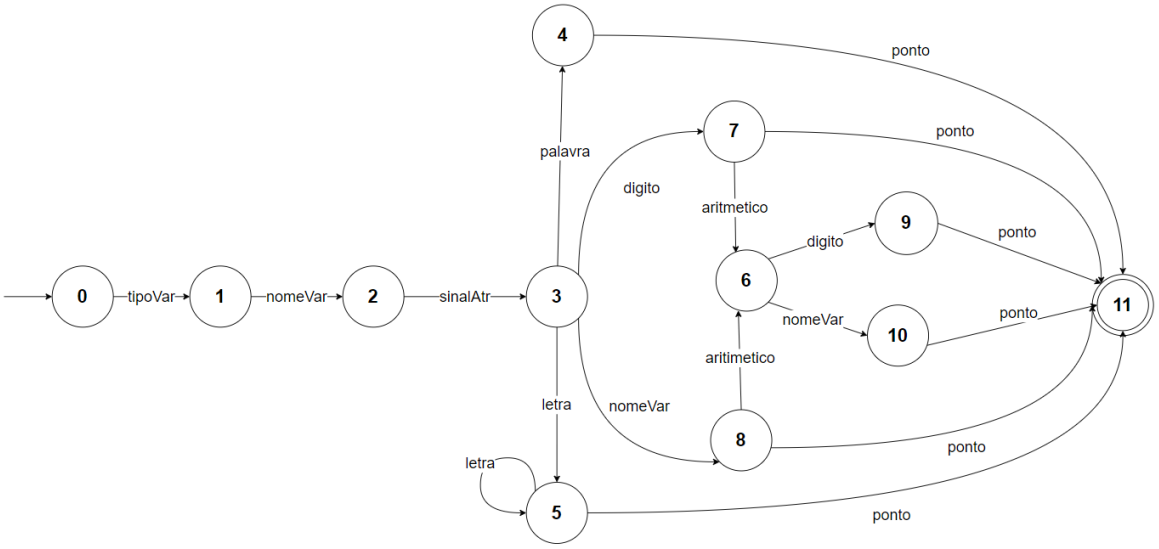

Figura 4 - TOKEN ATRIBUIÇÃO

Exemplo: int compiladores |= 34 + 22 ; int compiladores |= 22 - a1

 # entrada = (nomeVar)(sinalAtr)(read)(abreP)(fechaP)(ponto)

Essa expressão permite a leitura externa de dados, como por exemplo, do teclado.

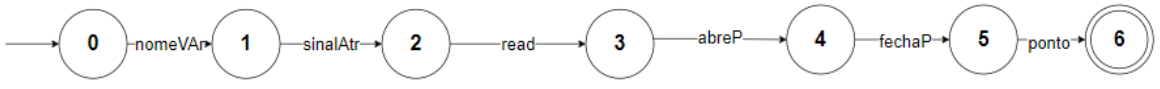

Figura 5 - TOKEN READ

Exemplo: compiladores |= read().

 # saida = (write)(abreP)(abreAsp)(letra | num | [abreC nomeVar fechaC])* (fechaAsp)(fechaP)(ponto)

Essa expressão permite a escrita de dados em um terminal.

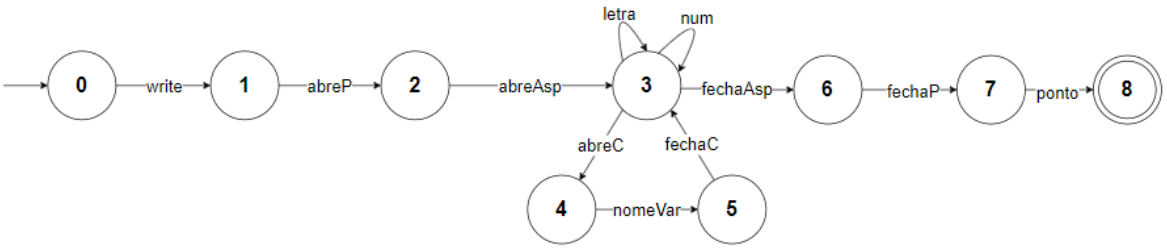

Figura 6 - TOKEN WRITE

 # teste = (abreP)(nomeVar)(relacional)(nomeVar)(fechaP)

A expressão regular acima descreve um teste relacional qualquer.

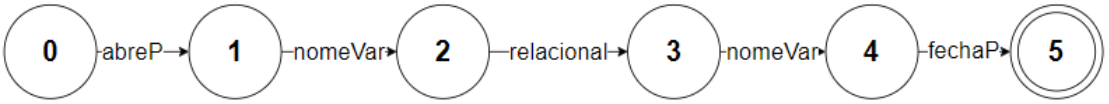

Figura 7 - TOKEN TESTE

 # condicional = (if)(abreP)(teste)(logico teste)*(fechaP)(abreC)(expressao+) (fechaC)

A expressão regular descreve a estrutura básica dos comandos condicionais "if", "else if" e "else". É necessário iniciar com um comando "if" e é possível construir múltiplas condições, permitindo a inclusão de vários testes relacionais através de operadores lógicos dentro dos parênteses.

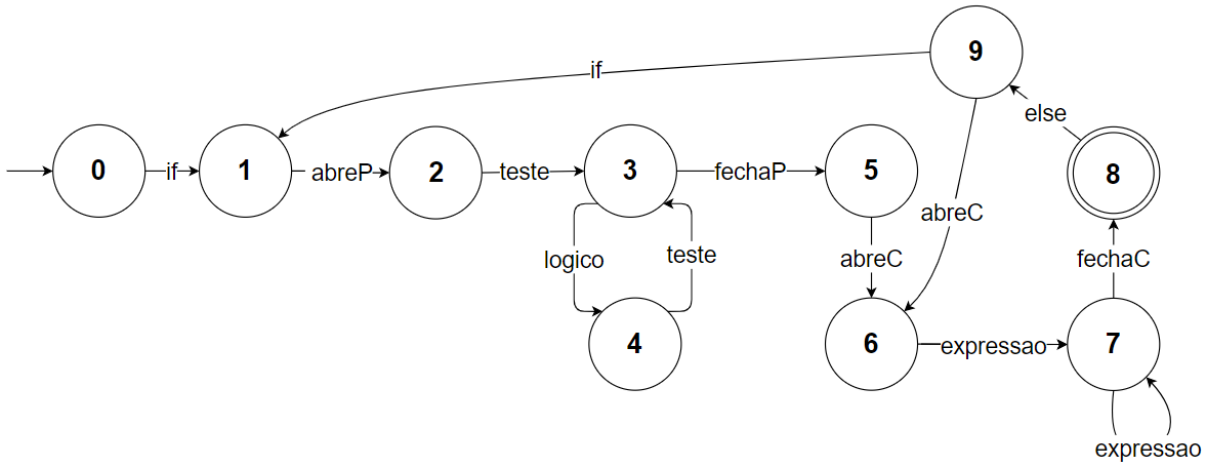

Figura 8 - TOKEN IF / ELSE

 # repeticao = (while)(abreP)(teste)(logico teste)*(fechaP)(abreC)(expressao*)(fechaC)

A expressão regular descreve a estrutura básica de um comando ***while***. É necessário iniciar com um comando "while" e é possível construir múltiplos testes relacionais através de operadores lógicos dentro dos parênteses. Vale destacar que não é obrigatória a declaração de uma expressão dentro do comando construído.

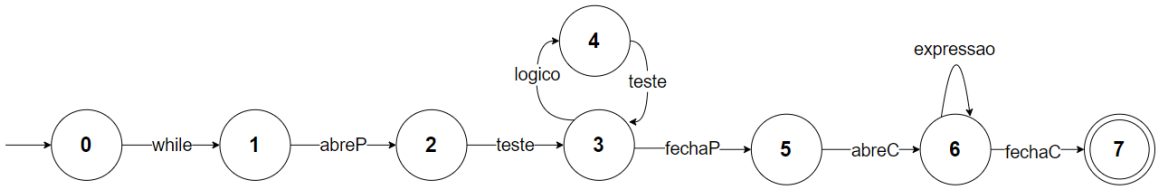

Figura 9 - TOKEN WHILE

 # repeticao = (for)(nomeVar)(range)(abreP)(num)(fechaP)(abreC)(expressao)(fechaC)

A expressão regular descreve a estrutura básica de um comando ***for*** especificando a variável que será sujeita a incremento / decremento, juntamente com a estipulação do limite numérico que serve como critério de término para a execução do comando.

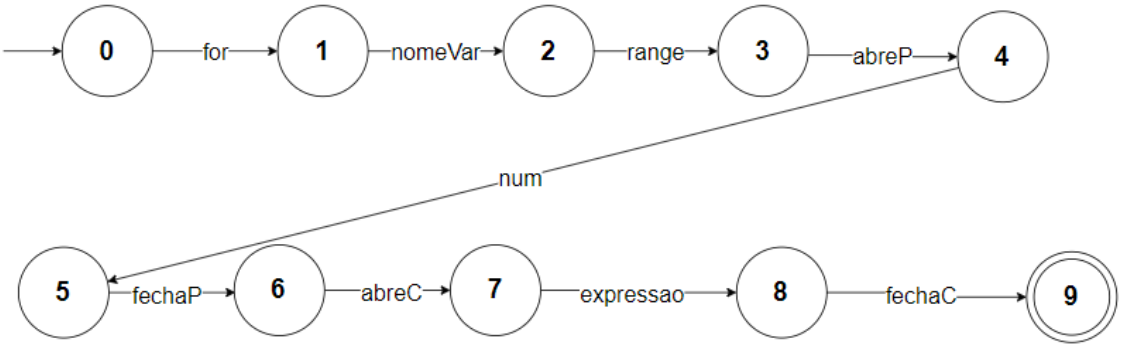

Figura 10 - TOKEN FOR

 # expressao = (condicional | repeticao | atribuicao | entrada | saida)+

O termo ‘expressão’ encapsula as diversas formas de construção dentro da linguagem, abrangendo condicionais, estruturas de repetição, atribuições, operações de entrada e saída. A estrutura como um todo pode ocorrer mais de uma vez, sendo obrigatória a ocorrência de pelo menos uma destas uma vez.

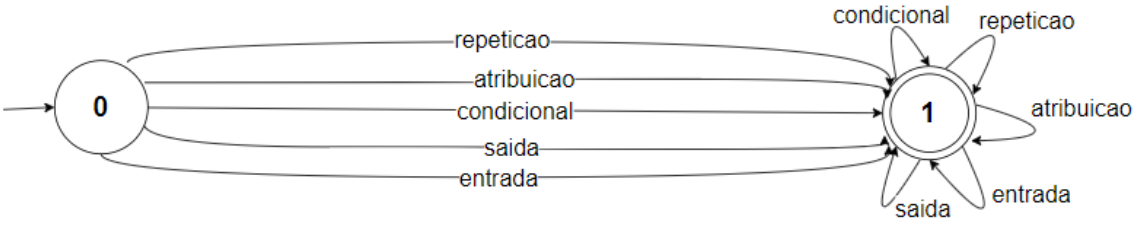

Figura 11 - TOKEN EXPRESSAO

 # sinalAtr = (|)(=)

O termo “sinalAtr” faz a atribuição de valores e expressões a variáveis usando o símbolo “|=”.

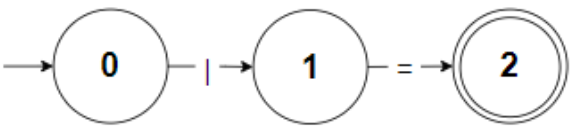

Figura 12 - TOKEN SINALATR

 # palavra = (under)(letra)+

Essa regra representa a definição do tipo palavra (String). No qual deve sempre começar por ‘*underline’* e ser seguido por pelo menos uma letra.

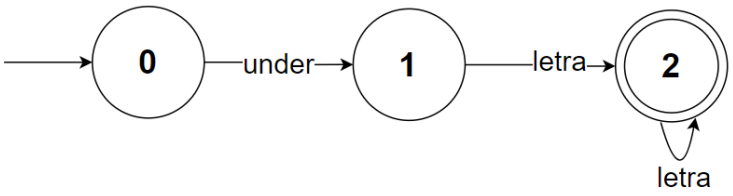

Figura 13 - TOKEN PALAVRA

 # comentário = (hash) letra* (ponto)

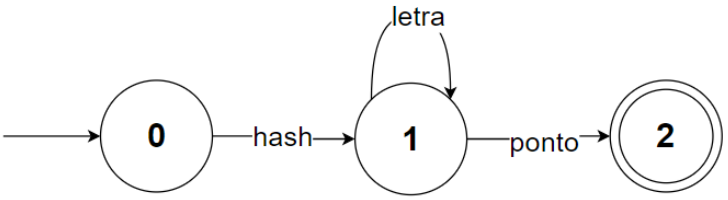

Figura 14 - TOKEN HASH

4. EXEMPLO DE CÓDIGO

```
1   int a1 |= 5.
2   int a2 |= a1 - 1.
3   real a3 |= 6.48.
4   palavra p1 |= _Carlos.
5   letra l1 |= "a".
6
7   for a1 range(10){
8       if(a1 > a2){
9           write("valor a1 = {a1}").
10      }
11  }
12
13  write("valor de 0 a 10 : ").
14  a2 |= read().
15
16  while((a2 < a3)and(a2 < a1)){
17      write("valor de a2 {a2}").
18  }
19
```

Figura 15 - EXEMPLO DE CODIGO