UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIÊNCIAS EXATAS E BIOLÓGICAS
DEPARTAMENTO DE COMPUTAÇÃO
PROF. JOSÉ ROMILDO MALAQUIAS

MARCOS PAULO FERREIRA RODRIGUES - 14.2.4341
TIAGO SARDI MENDONÇA - 17.2.5976

RESUMO
Desenvolvimento de uma mini linguagem de programação com documentação inicialmente voltada à análise léxica.

Ouro Preto, 2018

SUMÁRIO

1 A LINGUAGEM PROERD

O objetivo da mini linguagem de programação PROERD é direcionado ao aprendizado de programação para iniciantes brasileiros, especialmente crianças. Sua linguagem possui palavras reservadas na língua portuguesa.
Segue, abaixo descrito, a gramática livre de contexto da PROERD.

1.1 GRAMÁTICA LIVRE DE CONTEXTO

Program -> *Funs*
Funs -> *Fun*
Funs -> *Fun Funs*

Fun -> *TypeId* (*TypeIds*) = *Exp*

TypeId -> INTEIRO id
TypeId -> BINARIO id
TypeId -> TEXTO id
TypeId -> CARACTER id
TypeId -> FLUTUANTE id
TypeIds ->

Exp -> id
Exp -> id <- Exp
Exp -> Exp + Exp
Exp -> Exp - Exp
Exp -> Exp / Exp
Exp -> Exp * Exp
Exp -> Exp % Exp
Exp -> - Exp
Exp -> *Exp* = Exp
Exp -> *Exp* > Exp
Exp -> *Exp* >= Exp
Exp -> *Exp* <= Exp
Exp -> *Exp* < Exp
Exp -> *Exp* E Exp
Exp -> *Exp* OU Exp
Exp -> NAO Exp
Exp -> (Exp)

Exp -> id (Exp)
Exp -> SE Exp ENTAO Exp SENAO Exp

Exp -> ENQUANTO Exp FACA Exp
Exp -> PARA Exp ATE Exp FACA Exp FIM

2 TABELAS

Abaixo estão todas as tabelas de especificação da nossa linguagem

TABELA DE CARACTERES ESPECIAIS (TAB_E)					
!	@	#	$	%	&
*	(	)	_	=	+
[	]	{	}	?	/
;	:	\	|	-	,
.					

2.1 TAB_E

TABELA DE TOKENS (TAB_T)		
Token	**Lexema**	**Descrição**
T_SE	SE	Palavra reservada
T_NAO	NAO	Palavra reservada
T_ENTAO	ENTAO	Palavra reservada
T_SENAO	SENAO	Palavra reservada
T_FIM	FIM	Palavra reservada
T_ATT	<-	Atribuição
T_PARA	PARA	Palavra reservada
T_ATE	ATE	Palavra reservada
T_FACA	FACA	Palavra reservada’
T_ENQUANTO	ENQUANTO	Palavra reservada
T_SOMA	+	Operador aritmético
T_SUBTRAIR	-	Operador aritmético

T_MULT	*	Operador aritmético
T_DIV	/	Operador aritmético
T_RESTO	%	Operador aritmético
T_COMPARACAO	=	Operador booleano
T_TENTE	TENTE	Palavra reservada
T_CAPTURE	CAPTURE	Palavra reservada
T_PONTO	.	Ponto flutuante
T_PAR_ABERTO	(	Palavra reservada
T_PAR_FECHADO	)	Palavra reservada
T_FUNCAO	FUNCAO	Palavra reservada
T_E	E	Palavra reservada
T_OU	OU	Palavra reservada
T_MAIOR	>	Operador relacional
T_MAIORIGUAL	>=	Operador relacional
T_MENOR	<	Operador relacional
T_MENORIGUAL	<=	Operador relacional

2.2 TAB_T

TABELA DE PRECEDÊNCIA DOS TOKENS (TAB_PT)			
T_MULT	T_DIV	T_RESTO	T_PAR_ABERTO
T_PAR_FECHADO	T_SUBTRAIR	T_SOMA	T_COMPARACAO
T_FUNCAO	T_SE	T_NAO	T_ENTAO
T_SENAO	T_FIM	T_PARA	T_ATE
T_FACA	T_ATT	T_ENQUANTO	T_TENTE
T_CAPTURE	T_PONTO	T_E	T_OU
>	>=	<	<=

2.3 TAB_PT

TABELA DE BIBLIOTECAS NATIVAS (TAB_BN)			
Token	**Lexema**	**Descrição**	**Funções**
T_IO	EntradaSaida	Manipula entrada e saída de dados	Escreva, Leia
T_MATEMATICA	Matematica	Manipula operações matemáticas	Quadrado, Raiz, Aleartorio

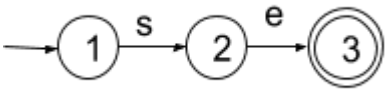
2.4 TAB_BN

3 TOKENS

Abaixo, segue descrição dos tokens utilizados.

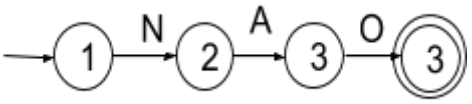
3.1 T_SE

Uso: Condição
Descrição: Usado para realizar condição na linguagem.
Sintaxe: SE <bool> ENTAO <exp> FIM
Autômato Finito:



3.2 T_NAO

Uso: Negação
Descrição: Usado para negar um valor binário..
Sintaxe: NAO <bool>
Autômato Finito:



3.3 T_SENAO

Uso: Condição
Descrição: Usado para executar uma sequencia de expressões a partir de uma condição

Sintaxe: SE <bool> ENTAO <exp> **SENAO** <exp> FIM

3.3 T_E

Uso: Condição
Descrição: Usado para concatenar condições
Sintaxe: <bool> **E** <bool>

3.4 T_OU

Uso: Condição
Descrição: Usado para concatenar condições
Sintaxe: <bool> **OU** <bool>

3.5 T_MAIOR

Uso: Comparação numérica
Descrição: Usado para comparar valores numéricos
Sintaxe: <exp> **>** <exp>

3.6 T_MAIORIGUAL

Uso: Comparação numérica
Descrição: Usado para comparar valores numéricos
Sintaxe: <exp> **>=** <exp>

3.7 T_MENOR

Uso: Comparação numérica
Descrição: Usado para comparar valores numéricos
Sintaxe: <exp> **<** <exp>

3.8 T_MENORIGUAL

Uso: Comparação numérica
Descrição: Usado para comparar valores numéricos
Sintaxe: <exp> **<=** <exp>

3.9 T_ENTAO

Uso: Condição
Descrição: Inicio de um bloco condicional
Sintaxe: SE <bool> **ENTAO** <exp> SENAO <exp> FIM

3.10 T_FIM

Uso: Condicional
Descrição: Usado para finalizar uma condição, laço ou função
Sintaxe: SE <bool> ENTAO <exp> SENAO <exp> **FIM**

3.11 T_ATT

Uso: Atribuição
Descrição: Usado para atribuir valor à variavel
Sintaxe: <var> **<-** <valor>

3.12 T_PARA

Uso: Laço de repetição
Descrição: Usado para iniciar uma estrutura de repetição
Sintaxe: PARA <var> <- <value> ATE <value> FACA <exp> FIM

3.13 T_ATE

Uso: Laço de repetição
Descrição: Usado para indicar o fim de uma estrutura de repetição
Sintaxe: PARA <var> <- <value> **ATE** <value> FACA <exp> FIM

3.14 T_FACA

Uso: Laço de repetição
Descrição: Usado para indicar o inicio da repetição
Sintaxe: PARA <var> <- <value> ATE <value> **FACA** <exp> FIM

3.15 T_ENQUANTO

Uso: Laço de repetição
Descrição: Usado para iniciar uma estrutura de repetição
Sintaxe: ENQUANTO <bool> FACA <exp> FIM

3.16 T_SOMA

Uso: Operador aritmético
Descrição: Usado para realizar soma entre números inteiros e flutuantes
Sintaxe: <var> **+** <var>

3.17 T_SUBTRAIR

Uso: Operador aritmético

Descrição: Usado para realizar subtração entre números inteiros e flutuantes
Sintaxe: <var> **-** <var>

3.18 T_MULT

Uso: Operador aritmético
Descrição: Usado para realizar multiplicação entre números inteiros e flutuantes
Sintaxe: <var> **-** <var>

3.19 T_DIV

Uso: Operador aritmético
Descrição: Usado para realizar divisão entre números inteiros e flutuantes
Sintaxe: <var> **/** <var>

3.20 T_RESTO

Uso: Operador aritmético
Descrição: Usado para obter resto da divisão entre números inteiros e flutuantes
Sintaxe: <var> **%** <var>

3.21 T_COMPARACAO

Uso: Comparador booleano
Descrição: Usado para realizar comparação entre valores
Sintaxe: <var> **=** <var>

3.22 T_TENTE

Uso: Palavra reservada
Descrição: Usado para iniciar um bloco de tentativa
Sintaxe: TENTE <exp> CAPTURE <exception> FIM

3.23 T_CAPTURE

Uso: Palavra reservada
Descrição: Usado para capturar uma exceção em um bloco de tentativa
Sintaxe: TENTE <exp> **CAPTURE** <exception> FIM

3.24 T_PONTO

Uso: Ponto flutuante
Descrição: Usado para delimitar casas decimais em pontos fluantes
Sintaxe: <inteiro>**.**<inteiro>

3.25 T_FUNCAO

Uso: Escopo
Descrição: Usado para iniciar um escopo
Sintaxe: FUNCAO <nome> () <exp> FIM

3.26 T_PAR_ABERTO

Uso: Escopo
Descrição: Usado para iniciar escopo
Sintaxe: FUNCAO <nome> **(**) <exp> FIM

3.27 T_BAR_FECHADO

Uso: Escopo
Descrição: Usado para finalizar escopo
Sintaxe: FUNCAO <nome> (**)** <exp> FIM

4 TIPO DE DADOS

A mini linguagem de programação PROERD é estaticamente tipada e possui os seguintes tipos primitivos e especiais:

Token	Padrão	Lexema
T_INT	65, 2311, 0, -433	INTEIRO
T_STRING	aa, aH, Ag, FGTS, F_d>,F3	TEXTO
T_CHAR	c,D,+,@,	CARACTER
T_FLUTUANTE	43.0	FLUTUANTE
T_BOOLEAN	verdadeiro,falso	BINARIO

4.1 Tipos primitivos de dados

Devido a escolha da linguagem ser estaticamente tipada, é perceptível o ganho em eficiência computacional, pois não será preciso verificar o tipo da variável em tempo de execução, entretanto perde em flexibilidade. E por fim, ganha em legibilidade, pois fica fácil saber o tipo da variável em seu tempo de vida.

4.2 T_ID

Descrição

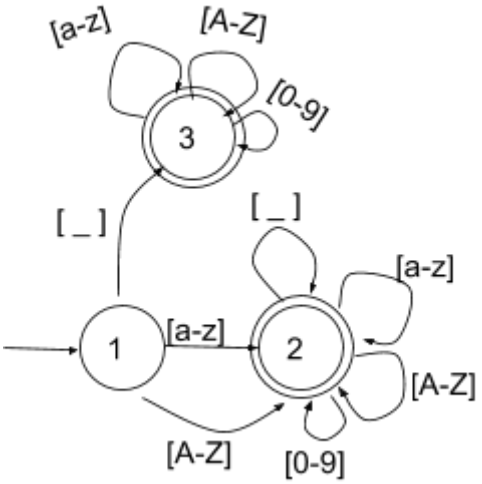
Identificador

Restrições: O primeiro caractere deve ser limitado pela expressão regular [a-z A-Z _]

Expressão Regular

[[a-z A-Z]+ | (_)] [A-z a-z 0-9 _]*

Autômato Finito



4.3 T_INT

Descrição

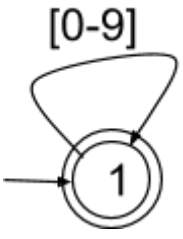
Representação

INTEIRO

Expressão Regular

[0-9]+

Autômato Finito



4.4 T_FLUTUANTE

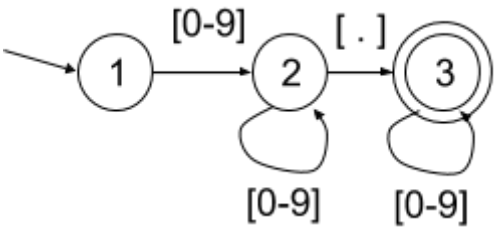
Representação

FLUTUANTE

Expressão Regular

[0-9]+.[0-9]+

Automato Finito



4.5 T_CHAR

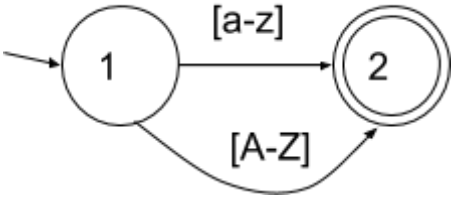
Descrição

Caractere do alfabeto

Expressão Regular

[a-z | A-Z]

Autômato Finito



4.5 T_STRING

Descrição

Cadeia de caracteres

Expressão Regular

[a-z A-Z 0-9 TAB_E]

5 BIBLIOTECAS NATIVAS

A tabela TAB_BN mostra a relação de bibliotecas nativas na linguagem de programação. Todas as bibliotecas e suas funções possuem carregamento implícito obrigatório.

6 EXEMPLO DE CÓDIGO FONTE

Para a criação de um programa, usaremos a seguinte sintaxe:

```
PROGRAMA
   PARA INTEIRO contador <- 1 ATE 10 FACA
      HelloWorld (contador)
   FIM
FIM

FUNCAO HelloWorld(INTEIRO numero)
   SE numero % 2 = 0 ENTAO
      ESCREVA (“Olá, mundo par “ + numero )
   SENAO
      ESCREVA (“Olá, mundo impar “ + numero)
   FIM
FIM
```

7 TRABALHOS FUTUROS

Como possíveis trabalhos futuros, leva-se em consideração de discussão os seguintes tópicos:

- Adicionar comentários utilizando o lexema #
- Criar as funções ESCREVA e LEIA
- Definir como será o escape da função ESCREVA
- Codificação BCD para os números decimais
- Polimorfismo da PROERD
- Codificação UTF-8
- Criação de Vetores (Lista de baixo nível)
- Definir a coerção
- Desenvolver esquema de sobrecarga de operações para permitir entrada de operadores pré-fixados, infixados e pós fixados.
- Amarração