Projeto da Disciplina de Linguagem de Programação: Conceitos e Paradigmas

Ana Luisa Estevam Dantas
Gdiael Souto Barros
Italo Bruno Brand Nardy
Tarsila Samille Santos da Silveira

19 de julho 2022

1 Introdução

- **Visão Geral:** A linguagem chamada Brasileira, tem o intuito de atender todo público alvo e por paradigma imperativo. É uma linguagem simples e leve implementada em Haskell.
- **Tipos de dados primitivos:** inteiro, decimal, texto, caracter, logico e matrix.
- **Alocação de variável:** Estática e dinâmica.
- **Tipagem:** Estática
- **Variáveis:** Escopo (Stack)
  Tipo: String
  Tempo de vida: A vinculação estática ocorre antes da execução e permanece inalterado durante a execução do programa
- **Variáveis:** Escopo (Heap)
  Tipo: String
  Tempo de vida: São alocadas e liberadas por instruções explícitas, tem efeito durante a execução do programa.

2 Design da implementação

2.1 BNF

```
<program> → principal { <main_stmt_list> }

<main_stmt_list> -> <function-declaration>
    | <stmt_list>
    | <comment>
    | <main_stmt_list>  <main_stmt_list>

<comment> -> \\  <words>

<words> -> <word> | <word> <words>

<function-call> -> <word> ( <parameter-list> );


<function-declaration>  ->  função <type>  <word> ( <parameter-list>  )  { <stmt_list> }

<parameter-list> -> <type>  <word>
    | <parameter-list> , <type>  <word>

<stmt_list> → <stmt>;
    | <stmt> ; <stmt_list>

<stmt> → <type> <word>  =  <exp>

<exp> -> <expression> | <bool> | <word>

<bool> -> "verdadeiro"
    | "falso"

<expression> → <number> + <number>
    | <number> - <number>
    | <number> / <number>
    | <number> * <number>
    | <number>
```

```
<type> → "inteiro"
    | "decimal"
    | "logico"
    | "texto"
    | "caracter"
    | "matrix"

<if_stmt> → se ( <logic_expr> ) {<stmt>}
    | se ( <logic_expr> ) {<stmt> } senao { <stmt> }
    | se ( <logic_expr> ) {<stmt> } senao <if_stmt>

<logic_expr>  → <var> == <expression>
    | <var> != <expression>
    | <var> > <expression>
    | <var> < <expression>
    | <var> >= <expression>
    | <var> >= <expression>

<iteration-statement> -> enquanto ( <logic_expr> ) <stmt>
                       | faça <stmt> enquanto ( <logic_expr> ) ;
                       | para ( <logic_expr> ; <logic_expr> ;<logic_expr> ) <stmt>

<number> -> <digit> | <non-zero-digit> <number>

<digit> -> "0" | <non-zero-digit>

<non-zero-digit> -> "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"

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