Nome: Gustavo Antonio Martini

Documentação da linguagem de programação #Monolith

**Introdução:**

#Monolith é uma linguagem de programação simples e excêntrica. Ela é baseada em um conjunto de regras simples que são fáceis de aprender e usar. A linguagem é também diferenciada em seu uso de variáveis com nomes somente em maiúsculas, blocos definidos com abre e fecha chaves e construções entre underlines.

**Características:**

#Monolith possui as seguintes características:

- **Variáveis:** Variáveis em #Monolith devem ter nomes somente em maiúsculas. Elas podem ser declaradas em qualquer lugar no código, mas a atribuição deve ser feita em um próximo comando.
- **Tipos:** #Monolith suporta três tipos de dados: inteiro (_Integer_), ponto flutuante (_Float_) e caractere (_Char_).
- **Operadores:** #Monolith suporta os seguintes operadores:
  - **Aritméticos:** adição (+), subtração (-), multiplicação (*), divisão (/).
  - **Lógicos:** igualdade (::), e (&&), ou (||), negação (!), maior (>>), menor (<<).

**Construções:**

#Monolith suporta as seguintes construções:

- **_repeater_:** repete um bloco de código enquanto uma condição for verdadeira.
  - _repeater_ <valor> <operador> <valor> $> <... linhas no laço ...> $<;
- **_for_:** repete um bloco de código um número especificado de vezes.
  - _for_ <início>, <final>, <nova_variável_de_controle> $> <... linhas no laço ...> $<;
- **_if_:** executa um bloco de código se uma condição for verdadeira.
  - _if_ <valor> <operador> <valor> $> <... linhas após codição ...> $<;
- **_print_:** imprime uma variável ou uma string no console.
  - _print_ <variável> || <String>;
- **_reader_:** lê uma entrada do usuário e atribui o valor a uma variável.
  - _reader_ <variável>;

- **_new_**: identifica como atribuição de uma nova variável
    - _new_ <tipo> <nome-variável>;

**Sintaxe:**

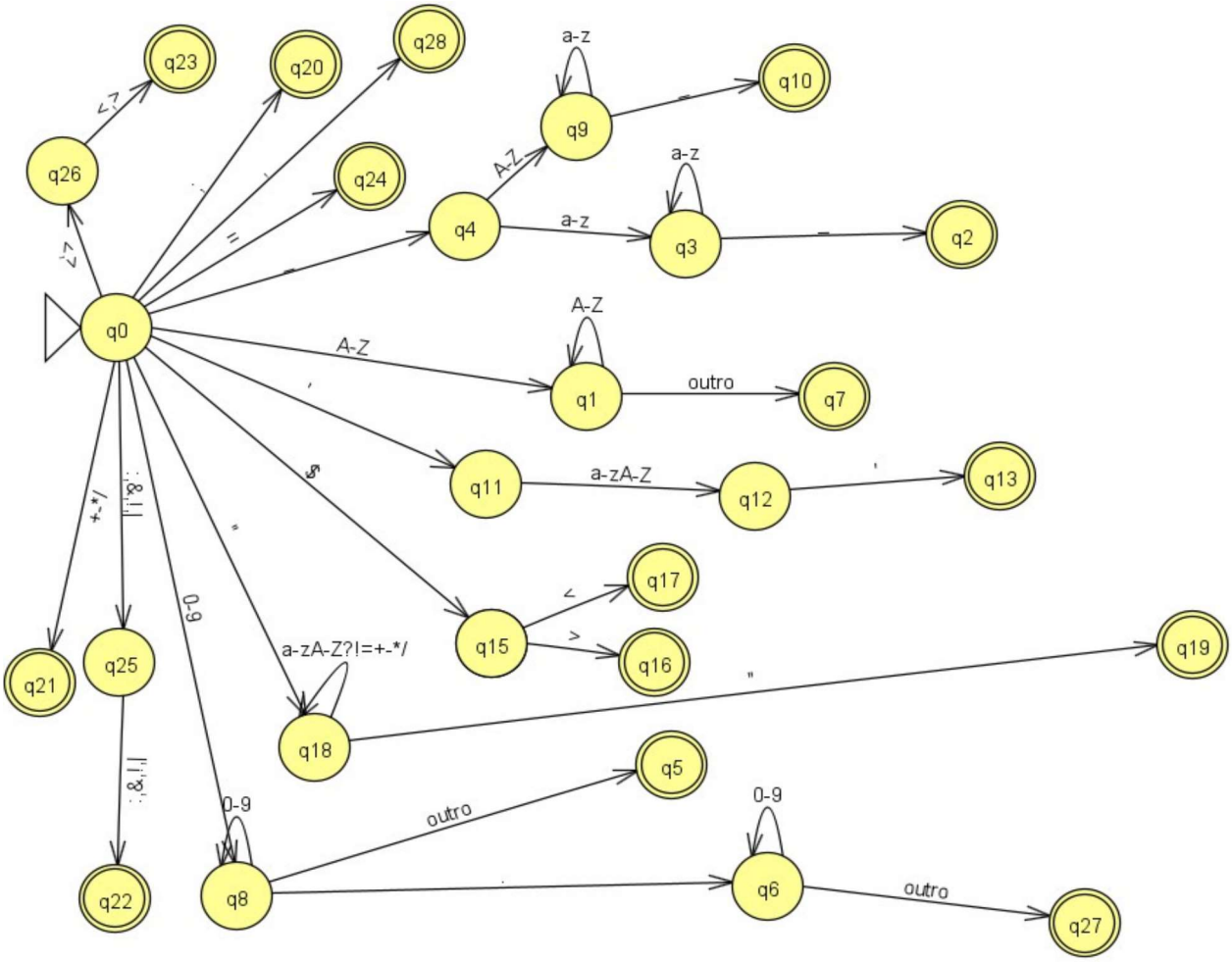
A sintaxe de #Monolith é baseada em um conjunto de regras simples:

- **Fim de linha:** O fim de linha é considerado um delimitador de instrução, o caractere que representa fim é “;”.
- **Blocos:** Blocos são definidos com a abertura e o fechamento de chaves ($> e $<).
- **Construções:** Construções são definidas entre underlines (_).
- **Comandos:** Comandos são instruções que executam uma ação específica.

**Expressões Regulares:**

| **Tipo** | **Alfabeto** | **Expressão** | **Representa** |
|---|---|---|---|
| Palavra-chave | key | _(a-z)+_ | repeater, if, for, new, print, reader |
| Identificador | id | (A-Z)+ | |
| Tipo | tp | _(A-Z)(a-z)*_ | int, ft, ch |
| Integer | int | (0-9)+ | |
| Float | ft | (0-9)+.(0-9)* | |
| Char | ch | '(a-zA-Z)' | |
| Final-Linha | ; | ; | |
| Abre Bloco | > | $> | |
| Fecha Bloco | < | $< | |
| String | str | "(a-zA-Z?!=+-*/)*" | |
| Aritméticos | ari | (+-/*) | |
| declaração | = | (=) | |
| Lógicos | log | (::,&&,||,!!) | |
| Relacional | rel | (<<,>>) | |

**Automato:**




**Gramática:**

G=([<programa>, <declaracoes>, <declaracao>, <declaracao-tipo>, <atribuicao>, <multi>, <expressao>, <termo>, <funcao>, <impressao>, <sera>, <logrel>], [new, tp, id, =, ch, ε, /, *, +, -, ;, int, ft, repeater, if, for, <, >, print, reader, str, log, rel], P, <programa>)

P = {

<programa> -> <declaracoes> ;

<declaracoes> -> <declaracao> ; <declaracoes> | ε

<declaracao> -> <atribuicao> | <declaracao-tipo> | <funcao>

<declaracao-tipo> -> new tp id

<atribuicao> -> id = <multi> | id = ch

<multi> -> <termo> | <expressao> + <termo> | <expressao> - <termo> | <expressao> * <termo> | <expressao> / <termo> | <multi> * <termo> | <multi> / <termo>

<expressao> -> <termo> | <expressao> + <termo> | <expressao> - <termo>

<termo> -> id | int | ft

<funcao> -> repeater <sera> | if <sera> | for int , int , id > <declaracoes> < | print <impressao> | reader id | reader ch

<impressao> -> id | str

<sera> ->  <multi> log <logrel> > <declaracoes> < | <multi> rel <logrel> > <declaracoes> < |

<logrel> -> <multi> | <multi> log <multi> | <multi> rel <multi> | <multi> log <logrel> | <multi> rel <logrel>

}

**Exemplos:**

Aqui está um exemplo de código #Monolith:

```
_new_ _Integer_ TESTE;
_new_ _Integer_ VAR;
_new_ _Char_ CARACTERE;

VAR = 10 + 1;
TESTE = 5;
CARACTERE = 'c';

_print_ "Teste de linguagem";

_if_ 10 << VAR $>
    _print_ "Variável maior que dez";
$<;

_repeater_ 10 << VAR $>
    VAR = VAR - 10;
$<;

_for_ 1, 10, I $>
    TESTE = TESTE * 2;
$<;

_print_ TESTE;
```