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.
  - o repeater <valor> <operador> <valor> \$> <... linhas no laço ...> \$<;</p>
- for : repete um bloco de código um número especificado de vezes.
  - \_for\_ <início>, <final>, <nova\_varíavel\_de\_controle> \$> <... linhas no laço ...> \$<;</p>
- \_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.
  - o print <variavel> || <String>;
- reader\_: lê uma entrada do usuário e atribui o valor a uma variável.
  - o reader <variavel>;

## Sintaxe:

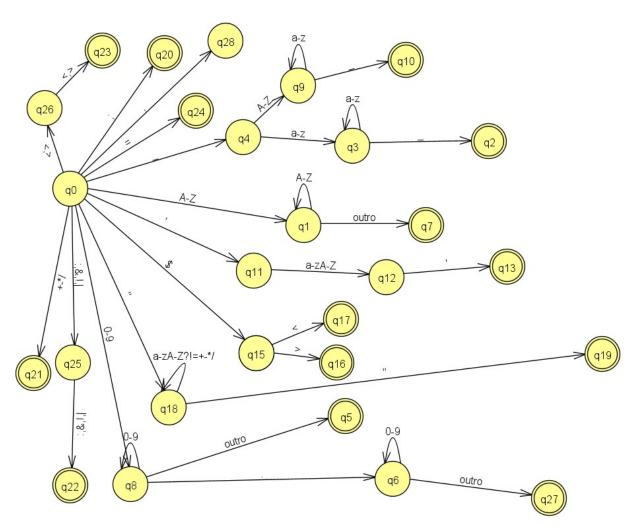
A sintaxe de #Monolith é baseada em um conjunto de regras simples:

- **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, \epsilon, /, *, +, -, ;, int, ft, repeater, if, for, <, >, print, reader, str, log, rel], P, <programa>)
```

```
<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 <termo> | 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 INT a;
a = 10;
PRINT a;
REPEATER a == 10 $>
       PRINT "a é igual a 10";
$<;
```

FOR 1 10 \$> PRINT i; i = i + 1;

IF a > 10 \$>

PRINT "a é maior que 10";

\$<;

\$<;