

# Linguagem de Emojis

## Compiladores

João Luiz Miranda Cilli

Samuel Lima Braz

Tony ALbert Lima



























## Introdução

A Linguagem de Emojis (LE) é uma linguagem de programação baseada em emojis, inspirada na estrutura e sintaxe da linguagem C. O objetivo principal foi substituir os símbolos convencionais de C por emojis, criando assim uma nova forma de expressar lógica de programação. Este documento apresenta a definição da linguagem, incluindo seus símbolos, expressões regulares, autômatos finitos e regras gramaticais.

As regras utilizadas para definir a sintaxe da linguagem foram baseadas inteiramente em C, onde cada emoji foi selecionado para representar operadores lógicos, estruturas de controle, tipos de dados, entre outros, contendo singelas diferenças.

# Símbolos da Linguagem

A Linguagem utiliza uma combinação de emojis para representar diversos elementos da linguagem C, como:

- Tipos de dados
  - Caracteres ASCII (char): 
  - Inteiro (int): 
  - Real (float): 
  - Vazio (void): 
  - Bool (bool): 
- Comandos:
  - Atribuição (=): 
  - Entrada (scanf): 
  - Saída (printf): 
  - Condicional
    - if: 
    - else: 
  - Repetição
    - while: 
    - for: 
- Operadores
  - Relacionais
    - Igualdade (==) : 
    - Diferença (!=): 
    - Maior (>): 
    - Menor (<): 
    - Maior igual (>=): 
    - Menor igual (<=): 
  - Lógicos
    - Conjunção (&&): 
    - Disjunção (||): 
    - Negação (!): 
  - Aritméticos
    - Adição (+): 
    - Subtração (-): 
    - Multiplicação (\*): 
    - Divisão (/): 
    - Resto (mod)(%): 

- Símbolos especiais
  - Chaves {}: 🖐️🖐️
  - Parênteses (): 🤔🤔
  - Colchetes []: 🖐️🖐️
  - Virgula “,”: ❤️
  - Ponto e vírgula “;”: ❤️
  - Ponto final “.”: 💥
  - Aspas simples ('): 🙏
- Blocos de comandos
  - begin/end ({}): 🖐️🖐️
- Palavras reservadas
  - true: 👍
  - false: 🖐️
  - break: ❌
  - continue: ✅
  - return: 🔄
  - main: 🧑

## Conjuntos

Comando = {Atribuição, Condicional, Repetição, Função, Bloco, Continue, Break, Return}

Tipo = {abc, i, ♣️, 🥚}

Operador = {✖️, +, -, ÷, 🐛, 🍷, 🖐️, ==, !=, 🙌, 🤔, 🙌, 🤔, 🤔}
































Booleano = {👍, 🖐️}

# Expressões Regulares

Esta seção apresenta as expressões regulares que definem os padrões para a linguagem. As expressões regulares foram baseadas na linguagem C, com adaptações para otimizar o reconhecimento de padrões na nossa linguagem, facilitando a implementação dos autômatos.

Tabela 1 - Expressões Regulares

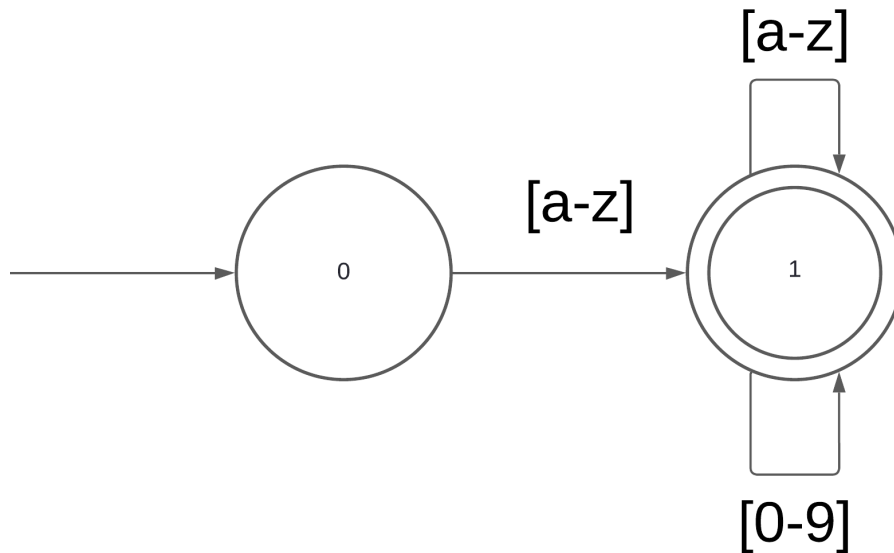
Regra	Expressão Regular	Descrição
Nome	[a-z]([0-9]    [a-z])*	Nomes válidos que começam com uma letra minúscula e podem ser seguidos por números ou letras minúsculas.
Número	[0-9]*(☀️[0-9]*)?	Números reais ou inteiros válidos, podendo incluir um ponto decimal.
Caracter	👉[ASCII]👈	Captura caracteres ASCII que estão entre aspas simples.
Sinal	!(+    -)?	Sinalização de uma variável, podendo ser precedida por sinais de adição ou subtração.
Break	✖️👉	Representa a instrução para escapar de um bloco de código.
Continue	✅👉	Representa a instrução para forçar o próximo passo em um loop ou iteração.
Operação	Valor Operador Valor	Operação entre dois valores, utilizando operadores específicos.
Valor	Sinal [Número   Carácter   Booleano   Operação   Parênteses   Função]	Variáveis sinalizadas, incluindo números, caracteres, booleanos, operações, parênteses e funções.
Parênteses	👉Valor👈	Indica a abertura e fechamento dos parênteses que envolvem um valor.
Declaração de Variável	[Tipo] Nome (≡ Valor)? (❤️ Nome (≡ Valor)?)*👉	Declarar uma variável com tipo específico e nome, podendo atribuir um valor inicial opcionalmente.

Atribuição	Nome  Valor 	Atribuição de um valor a uma variável
Bloco	 Comando* 	Indica o início e o fim de um bloco de comandos delimitado por chaves {}.
Condicional	  Valor  Comando	Blocos condicionais que contêm testes lógicos baseados em valores específicos.
While	  Valor  Comando	Laços while baseados em condições lógicas definidas por valores específicos.
For	  ((Atribuição    Declaração de Variável    Valor) (  (Atribuição    Declaração de Variável    Valor))* )?  ((Atribuição    Valor) (  (Atribuição    Valor))* )?  ((Atribuição    Valor) (  (Atribuição    Valor))* )?  Comando	laços for com opções detalhadas para inicialização, condição e incremento/decremento.
Retorno	 Valor 	Define como retornar um valor usando esta expressão regular.
Declaração de Função	Tipo Nome  (Tipo Nome (  Tipo Nome)* )?  Bloco	Declaração de uma função com tipo definido, nome e parâmetros opcionais seguidos pelo bloco contendo os comandos.
Main	    Bloco	Declaração da função principal
Função	Nome  (Valor (  Valor)* )? 	Valor retornado por uma função
Chamada de Função	Função 	A execução de uma função sem captura de retorno

# Autômatos Finitos

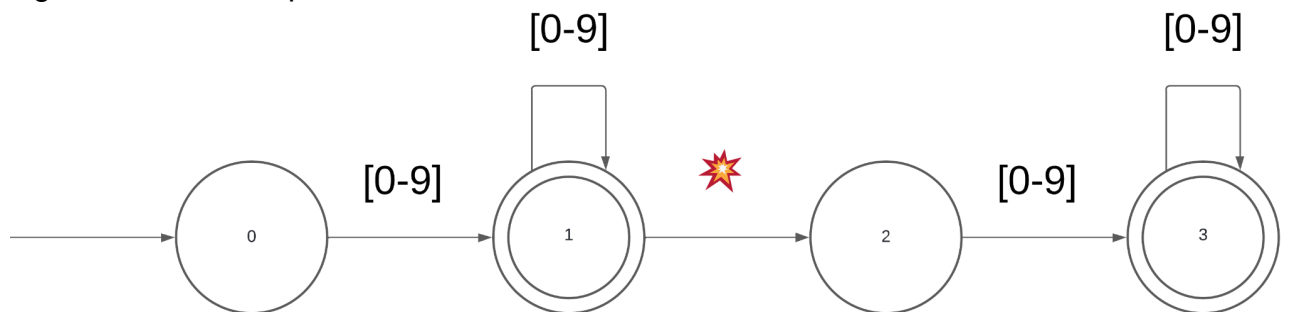
## Nome

Figura 1: Autômato para o token Nome



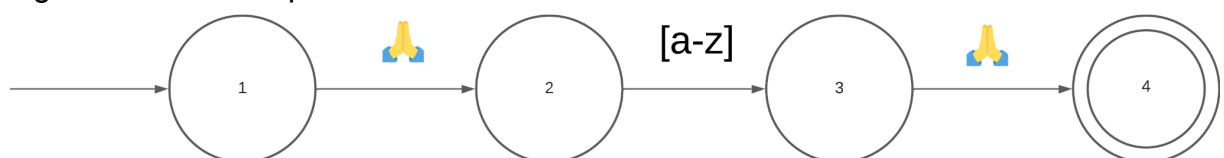
## Número

Figura 2: Autômato para o token Número



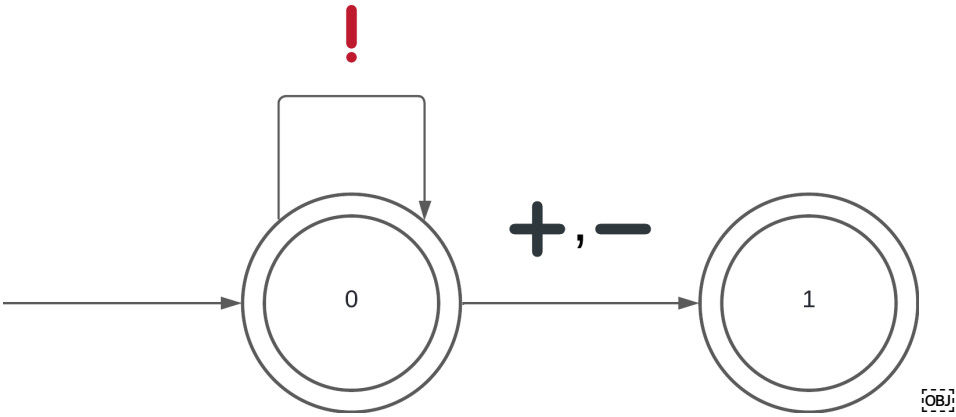
## Caracter

Figura 3: Autômato para o token Caractér



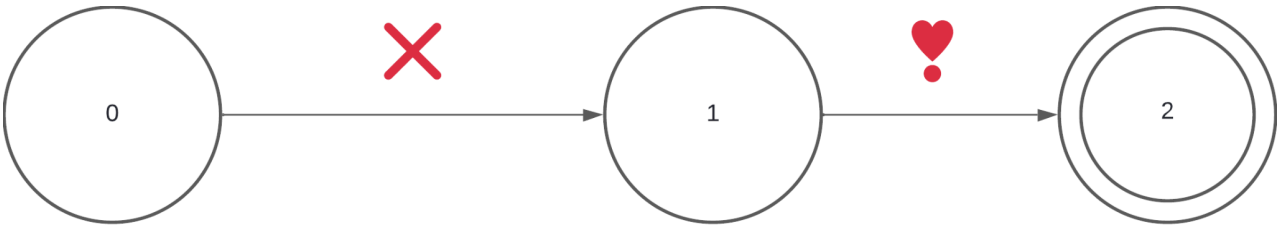
**Sinal**

Figura 4: Autômato para o token Sinal



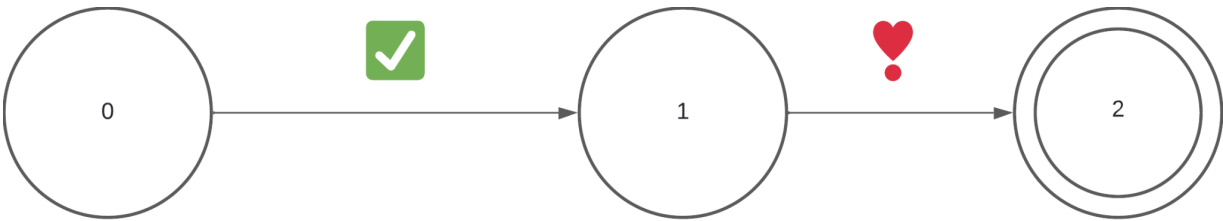
**Break**

Figura 5: Autômato para o token Break



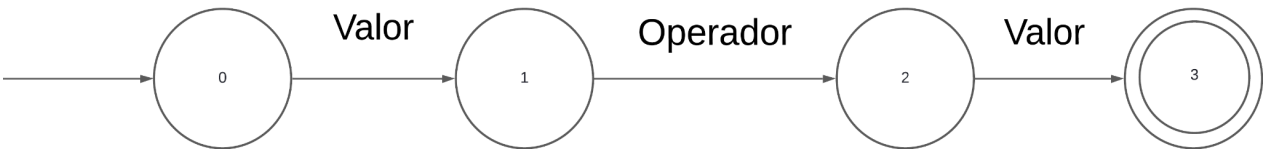
**Continue**

Figura 6: Autômato para o token Continue



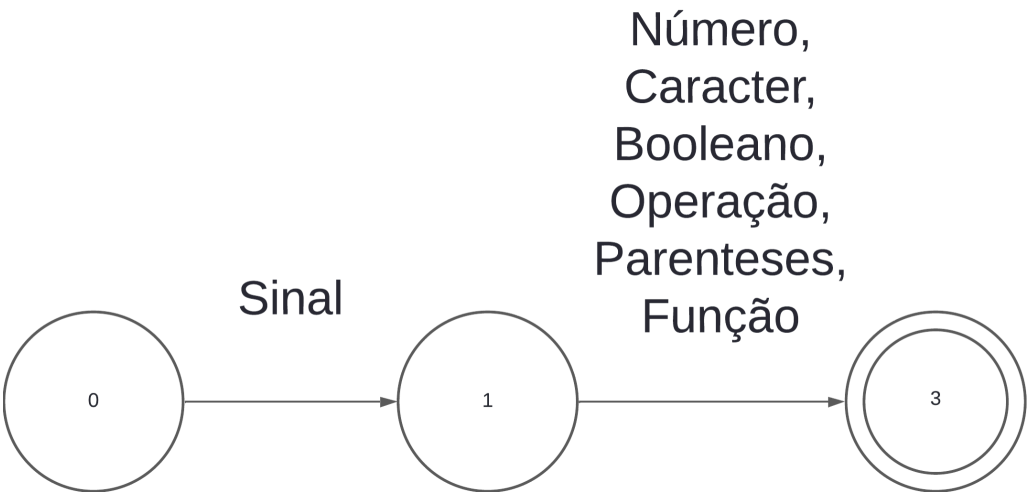
**Operação**

Figura 7: Autômato para o token Operação



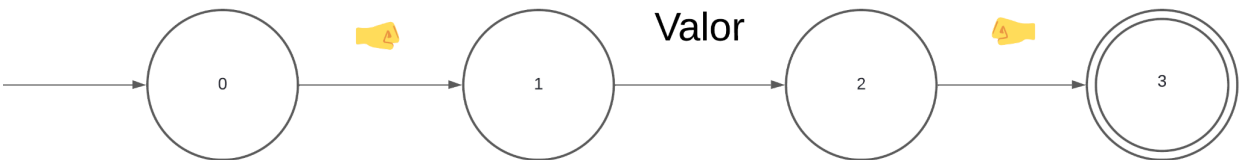
**Valor**

Figura 8: Autômato para o token Valor



**Parênteses**

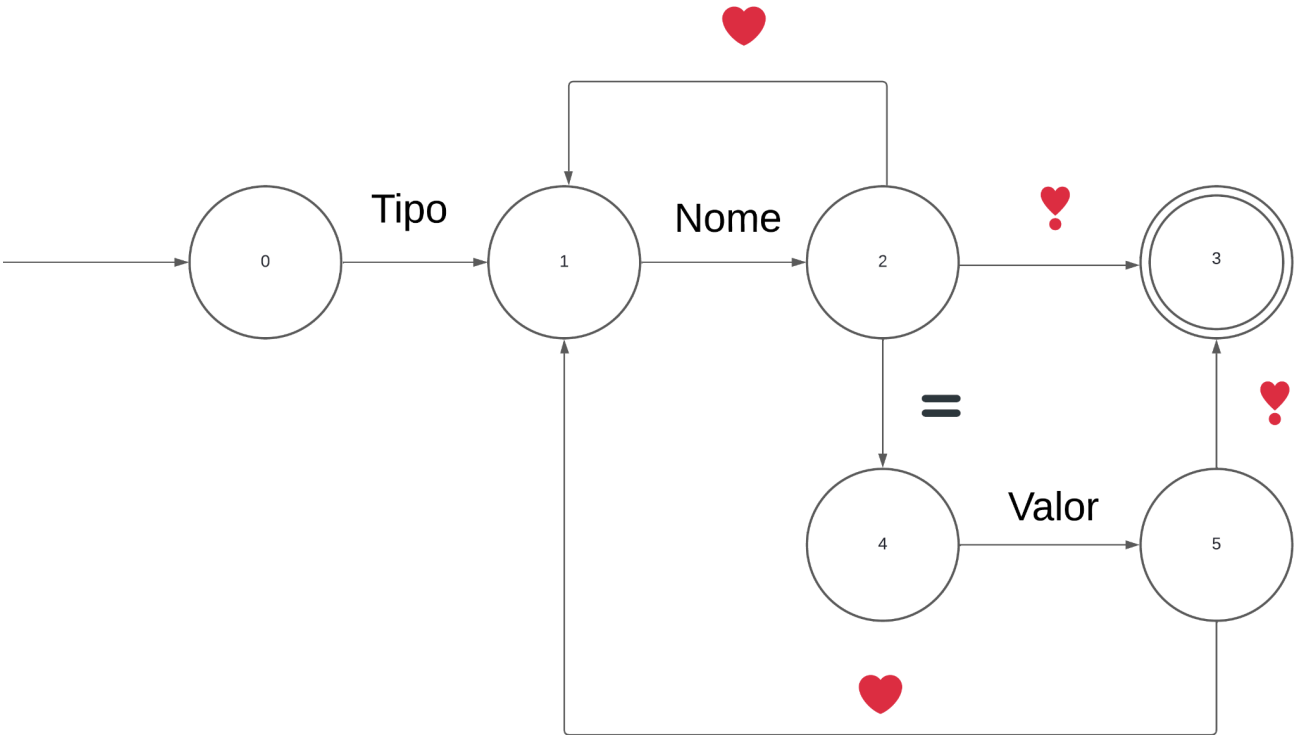
Figura 9: Autômato para o token Parênteses





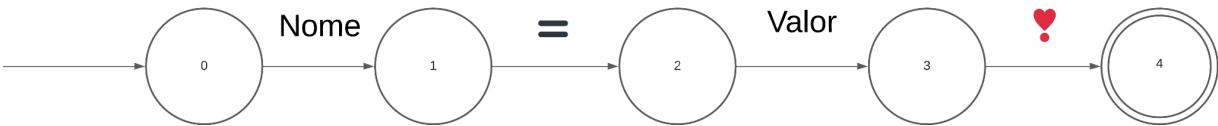
**Declaração de Variável**

Figura 10: Autômato para o token Declaração



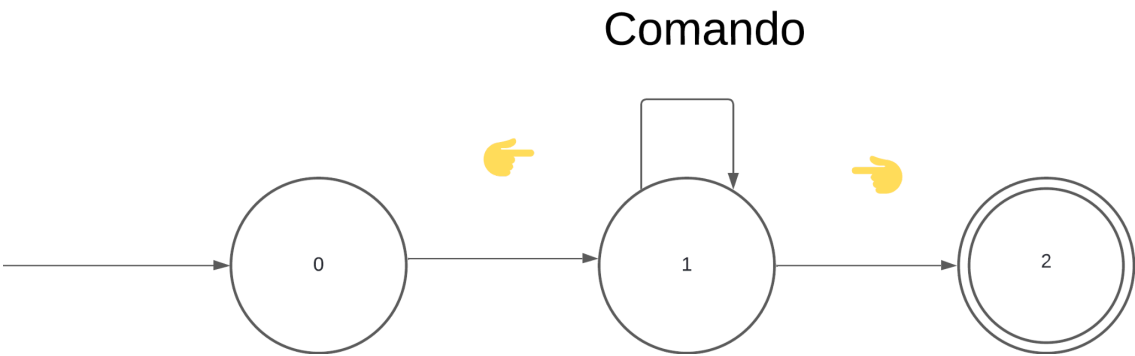
**Atribuição:**

Figura 11: Autômato para o token Atribuição



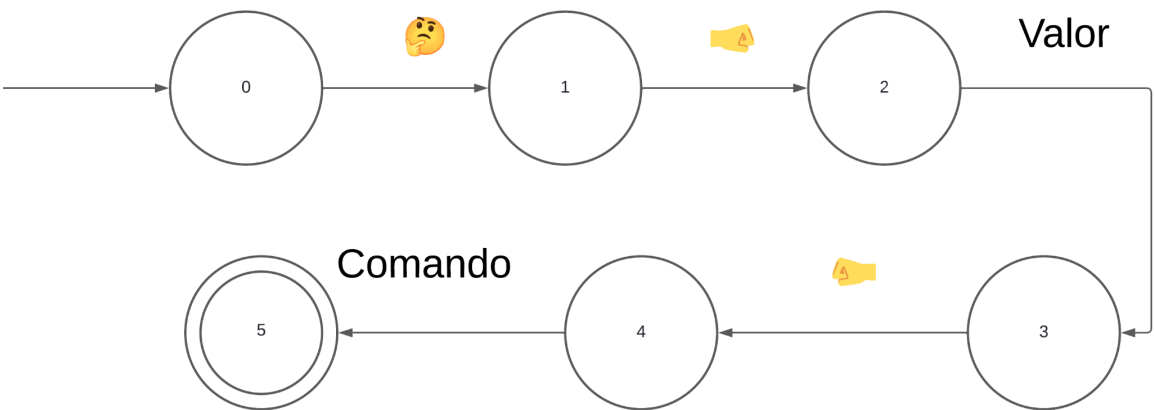
**Bloco:**

Figura 12: Autômato para o token Bloco



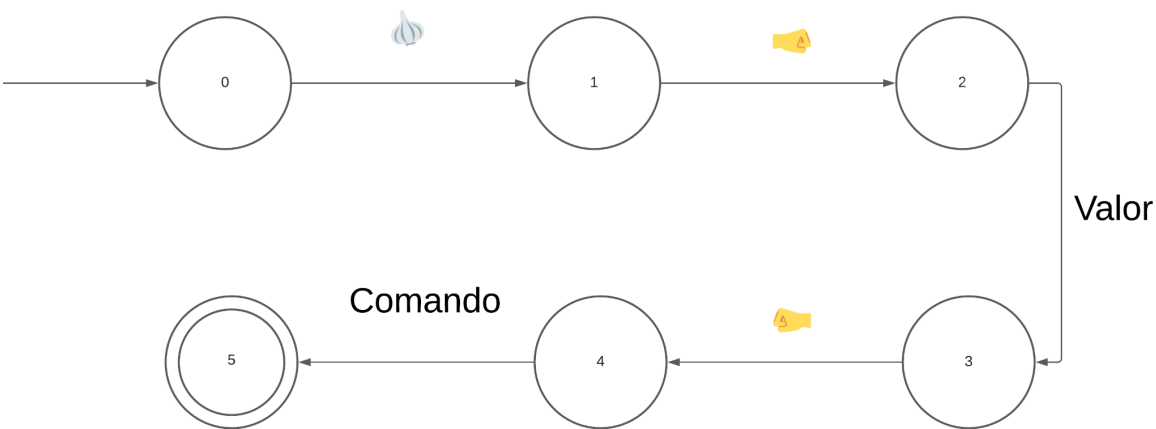
**Condicional:**

Figura 13: Autômato para o token Condicional



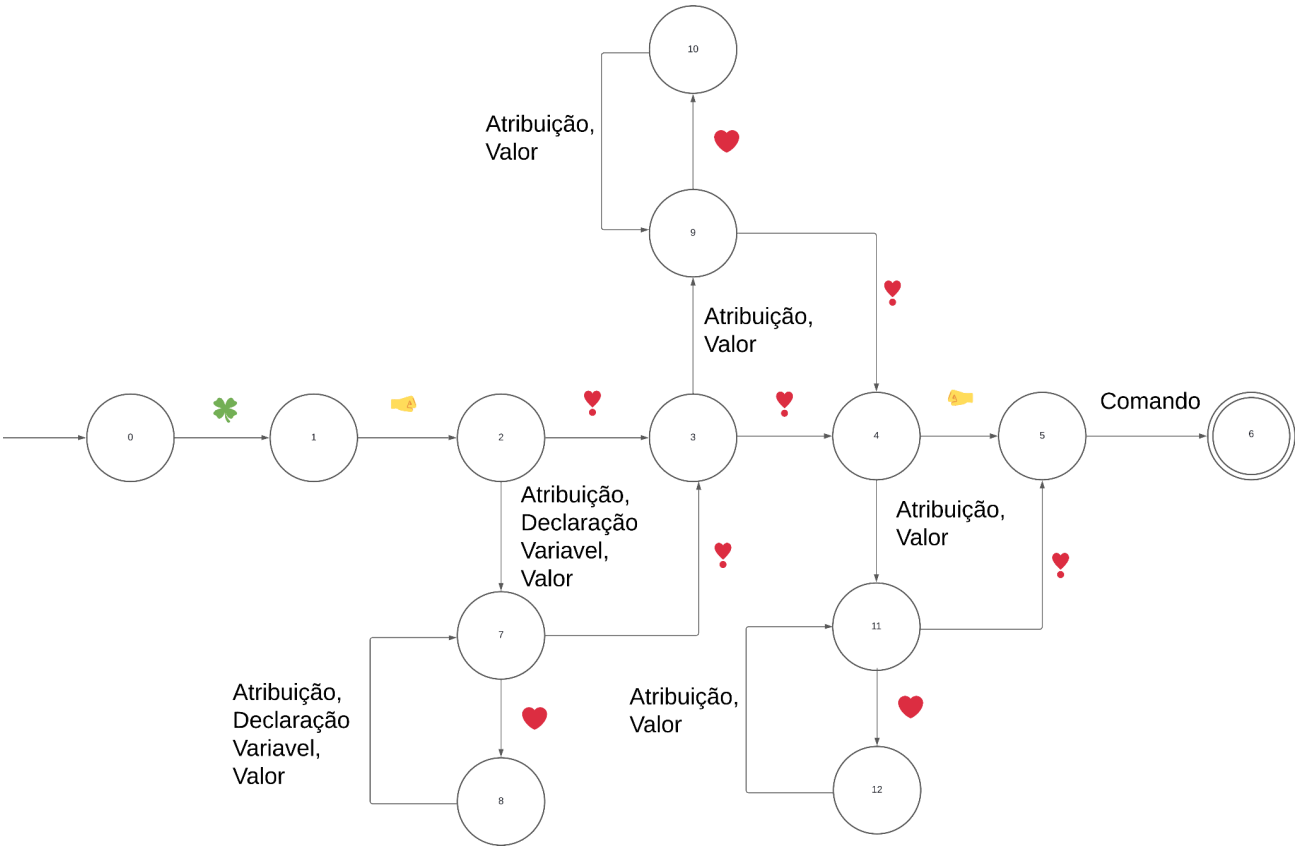
**While**

Figura 14: Autômato para o token While



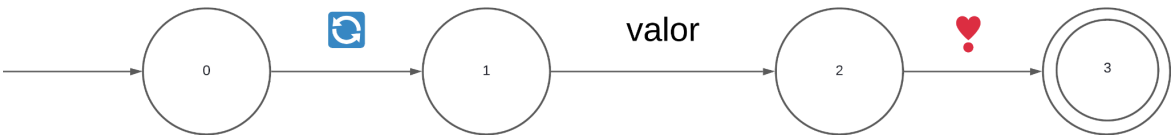
**For**

Figura 15: Autômato para o token For



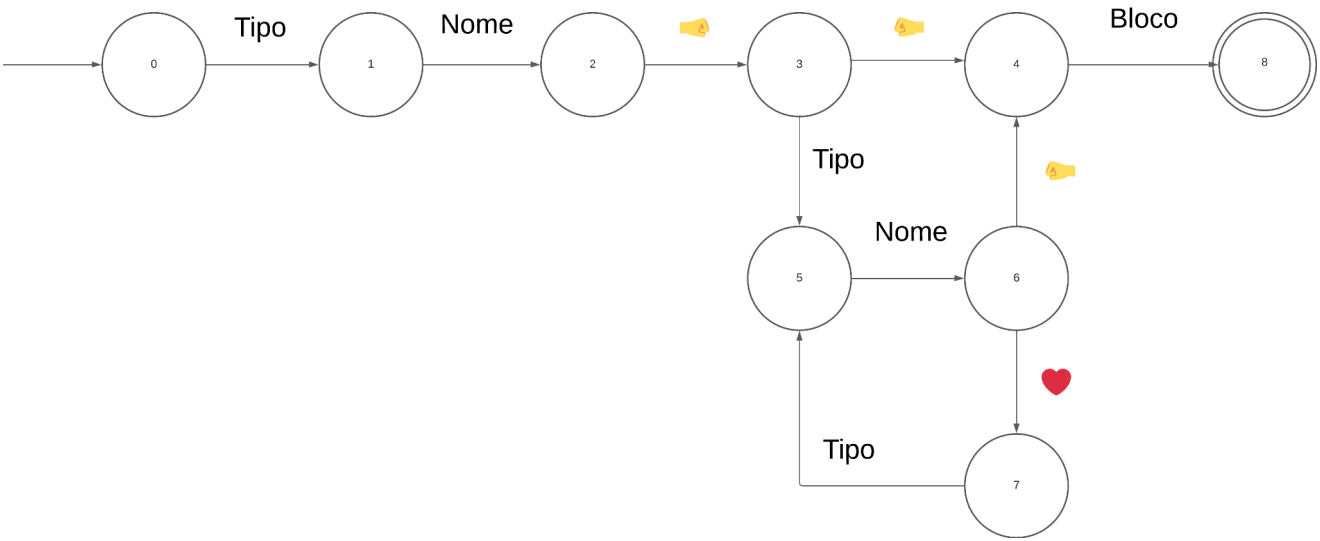
**Return**

Figura 16: Autômato para o token Return



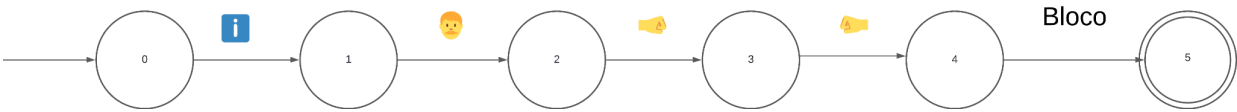
**Declaração de Função**

Figura 17: Autômato para a Declaração de Função



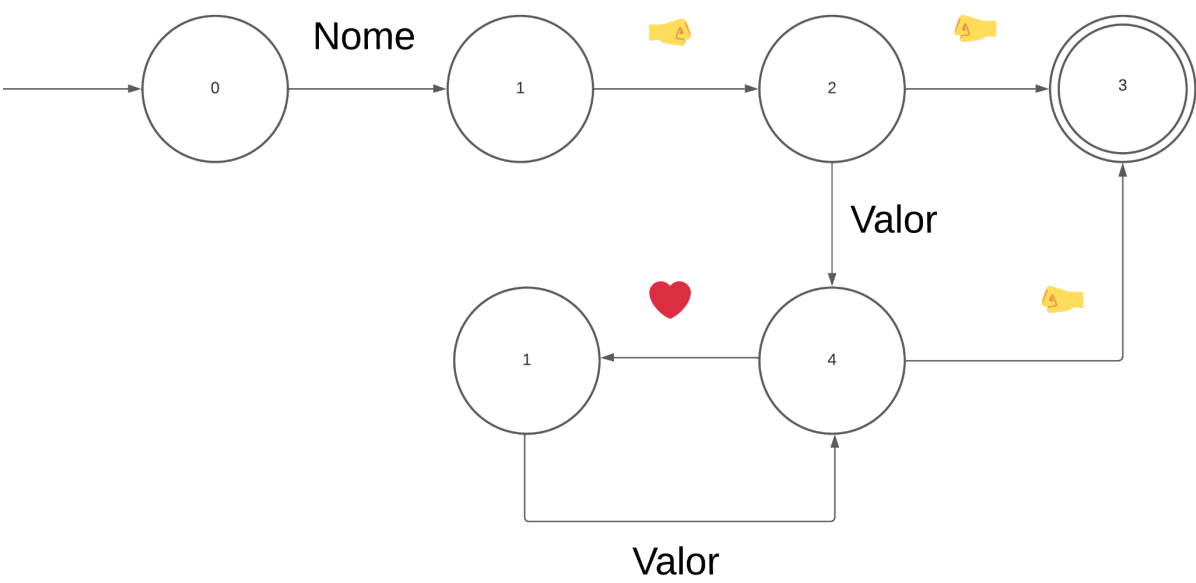
**Main**

Figura 18: Autômato para a Main



**Função**

Figura 19: Autômato para Função



**Chamada de Função**

Figura 20: Autômato para Chamada de Função

