

**Curso de Ciência de Computação**

**COMPILADORES**

**Prof. Tarcísio Lucas**

**Compilador Arretado**

**Análise Léxica – Parte 1**

Seu time terá a missão de desenvolver um Analisador Léxico para uma linguagem de programação inspirada em C em seus comandos e inspirada no vocabulário pernambucano em suas mensagens de erro.

**Expressões Regulares**

Segue a descrição dos tipos de tokens que devem ser considerados.

|  |  |
| --- | --- |
| **Inteiro** | Sequência de dígitos entre 0 e 9. Desconsiderar uso de sinal por simplicidade  Ex: 100, 123, 932000, 1. |
| **Real** | Sequência de dígitos, seguido de ponto, seguido de outra sequência de dígitos.  Desconsiderar uso de sinal por simplicidade  Ex: 100.25, 5.7, 132.85 |
| **Char** | Aspa simples, seguida de letra ou número, seguida de aspa simples.  Ex: ‘a’, ‘1’ |
| **Identificador** | Letra, seguido de letras ou números. Pode ser apenas uma letra. Apenas letras minúsculas e sem acento.  Ex: n, num, num1, n1 |
| **operador relacional** | <, >, <=, >=, ==, <> |
| **operador aritmético** | +, -, \*, /, % |
| **operador atribuição** | = |
| **caractere especial** | ), (, {, }, ; , , |
| **Palavras reservadas** | int, float, char, while, main, if, else |

**Expressão regular de cada token:**

Considere:

* digito -> [0-9]
* letra -> [a-zA-Z]

Descreva as expressões regulares para os tokens descritos na tabela anterior.

|  |  |
| --- | --- |
| **Tipo Token** | **Expressão regular** |
| **Inteiro** | digito+ |
| **Real** | (digito+ . digito+) |
| **Char** | ’(letra|digito)’ |
| **Identificador** | letra(letra|digito)\* |
| **operador relacional** | [<>=]=? |
| **operador aritmético** | [+-\*/%] |
| **operador atribuição** | [=] |
| **caractere especial** | [(){};,] |
| **palavras reservadas** | (int| float|char|while |main | if | else) |

Bom trabalho!