**LST PyC**

Linguagem Simples e Traduzida baseada em Python e C

LST PyC é uma linguagem estruturada que utiliza recursos intuitivos do Python e do C para proporcionar uma melhor experiência ao usuário. Considerando que as linguagens de programação exigem um certo conhecimento lógico, sintático e em inglês, muitos que lidam com a programação inicialmente sofrem com grandes dificuldades. Visando trazer uma experiência mais “newbie friendly”, a linguagem LST foi pensada.

Trazendo os comandos mais básicos, normalmente utilizados por iniciantes, a LST PyC pretende facilitar o contato inicial de não programadores (como estudantes de outras áreas que, por acaso do destino, precisam lidar com alguma disciplina relacionada à programação) com a programação.

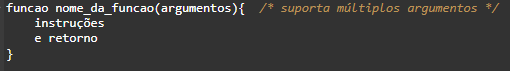
**ESPECIFICAÇÕES DA LINGUAGEM**

A LST PyC seguirá os modelos utilizados pelo C e pelo Python, proporcionando maior praticidade na utilização da mesma.

No C, as instruções são finalizadas com o uso do ;, obrigando o usuário a colocar o ; no final de cada linha.

A LST utilizará o reconhecimento de final de uma instrução a partir da quebra de linha (\n ou ENTER), reduzindo a quantidade de regras e exigências para a construção de um código simples ou complexo. Os blocos serão iniciados por { e finalizados por } e todas as instruções referentes a determinado comando deverão ser introduzidas nos seus respectivos blocos.

As funções serão definidas da seguinte forma:



As palavras reservadas e operadores da linguagem são definidas a seguir:

**PALAVRAS RESERVADAS**

| int | **natural** |
| --- | --- |
| float | **decimal** |
| scanf | **inserir** |
| printf | **mostrar** |
| return | **retornar** |
| void | **vazio** |
| for | **loop** |
| while | **quando** |
| do | **execute** |
| switch | **selecao** |
| case | **etapa** |
| break | **parar** |
| if | **condinicial** |
| If else | **condinter** |
| else | **condfinal** |
| def | **funcao** |

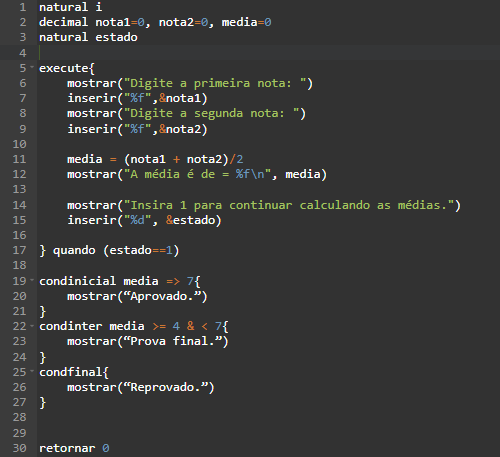
**OPERADORES RELACIONAIS**

| **>** | **Maior** |
| --- | --- |
| **>=** | **Maior ou igual** |
| **<** | **Menor** |
| **<=** | **Menor ou igual** |
| **==** | **Igual** |
| **!=** | **Diferente** |

**OPERADORES ARITMÉTICOS**

| **+** | **Operador de soma** |
| --- | --- |
| **-** | **Operador de subtração** |
| **/** | **Operador de divisão** |
| **\*** | **Operador de multiplicação** |
| **^** | **Operador de exponenciação** |
| **=** | **Operador de Atribuição** |

**Exemplo de aplicação:**



**Quanto aos estados do código:**

**Identificadores:**

(a..z)(A..Z|0..9|a..z)

**Números:**

(0..9)+

**Delimitadores:**

\n

**Operadores Relacionais:**

( > | >= | == | =< | < | != )

**Operador de Atribuição:**

=

**Caracteres Especiais:**

( [ | ] | \ | { | } | @ | ! | & | , | . | ; | : | ? | ‘ | “ | | | ( | ) | # | / )

**A linguagem contará com um estado para cada token:**

Palavras Reservadas, Identificadores, Números, Operadores Relacionais, Operadores de Atribuição e Caracteres Especiais.