Linguagem Textual em Português

Guilherme Zaborowsky Graicer

Ideia

Uma LTP é linguagem de programação feita para pessoas relativamente leigas nesta área (e que falam português) conseguirem entender. Ela transforma o código em uma espécie de texto, que contém os caracteres especiais da língua portuguesa. Ela não diferencia letras maiúsculas e minúsculas.

Modo de Uso

Variáveis

Declaração:

dimensione a variável <nome> como um <tipo>

(os nomes podem ser qualquer um com letras e "_"s, e os tipos possíveis são Inteiro e Booleano)

Atribuição:

coloque {<valor>} na variável <nome>

Condicionais

se a condição {<condição>} for verdadeira, faça:

    <código>

fim do condicional

Loop (while)

enquanto a condição {<condição>} for verdadeira, faça:

<código>

fim do enquanto

Funções

Função sem retorno:

A função sem retorno <nome> recebe os argumentos(<argumentos>) e faz:

    <código>

fim da função

Função com retorno:

A função <nome>, que recebe os argumentos (<argumentos>) e retorna um <tipo do retorno>, faz:

    <código>

fim da função

Impressões na tela

imprima na tela {<expressão>}

Nota de uso

Todo programa nesta linguagem deve ter pelo menos uma **função sem retorno** chamada **Principal** , que será executada quando o programa for.

Exemplo de um código (Fibonacci)

A função fibonacci, que recebe os argumentos (n como um inteiro) e retorna um inteiro, faz:

```
    dimensione a variável flag como um booleano
    coloque {falso} na variável flag
    se a condição {n = 0} for verdadeira, faça:
        coloque {1} na variável fibonacci
        coloque {verdadeiro} na variável flag
    fim do condicional
    se a condição {n = 1} for verdadeira, faça:
        coloque {1} na variável fibonacci
        coloque {verdadeiro} na variável flag
    fim do condicional
```

*** ver fora do modo apresentação para ver o código todo

EBNF

Program = "A", "função", SubDec|FuncDec;

SubDec = “sem”, "retorno", “identifier”, "recebe", "os", "argumentos", “(“, { | (“identifier”, “como”, "um", Type)}, “)”, "e", "faz", ":" “\n”, { | (Statement, “\n”)}, “fim”, "da", “função”;

FuncDec = “identifier”, ",", "que", "recebe", "os", "argumentos", “(“, { | (“identifier”, “como”, "um", Type)}, “)”, “e”, "retorna", "um", Type, ",", "faz", ":", “\n”, { | (Statement, “\n”)}, “fim”, "da", “função”;

RelExpression = Expression, {“=” | ”>” | ”<”}, Expression;

Expression = Term, {(“+” | “-” | ”ou”),Term | ;

Term = Factor, {(“*” | ”/” | ”e”), Factor} | ;

Factor = “number” | {“boolean” | ”identifier”| ("resultado", "da", "função", "identifier", "com", "os", "argumentos", “(“, { | (RelExpression)}, “)”) | {(“+” | ”-” | ”not”), Factor} | (“(“, RelExpression, “)”)| "verdadeiro" | "falso" ;

Statement = |((“coloque”, "{", RelExpression, "}","na", "variável", "identifier") | (“imprima”, "na", "tela", "{", RelExpression, "}") | (“dimensione”, "a", "variável", “identifier”, “como”, "um", Type) | (“se”, "a", "condição", "{", RelExpression, "}", “for”, "verdadeira", ",", "faça", ":", “\n”, {| (Statement, “\n”), {| (“senão”, ",", "faça", ":", “\n”, {| (Statement, “\n”)}}, “fim”, “do”, "condicional") | (“chame”,"a", "função", “identifier”, "com", "os", "argumentos", “(“, {| {RelExpression, {| “,”}}) | ("enquanto", "a", "condição", "{", RelExpression, "}", “for”, "verdadeira", ",", "faça", ":", “\n”, |(Statement, “\n”), “fim”, “do”, "enquanto"));

Type = "inteiro"|"booleano"