

Tupiniquim

Especificação da linguagem

- Estrutura geral de um programa:
 - Possui escopo global. As variáveis podem ser declaradas no escopo global ou em funções, devendo as declarações serem nas primeiras linhas;
 - O início da execução ocorre na função principal. A função não necessita ter parâmetros ou retornos, mas isso deve estar especificado.

Sintaxe:

```
tipoDeRetorno principal( tipoDoParametro primeiro, ... );  
{  
    /*Instruções*/  
}
```

- As definições de funções são feitas em arquivos, tanto externos quanto os que contêm a função principal, apenas a função deve estar escrita antes da sua chamada;
- Para inclusão de funções de arquivos externos, deve-se incluir o arquivo através de **inclua, nas primeiras linhas do arquivo chamador**:

Sintaxe:

```
inclua( nomeDoArquivo.tpq );
```

- O ponto e vírgula (;) funciona como finalizador das instruções e deve ser empregado com essa finalidade.

- Lista de Palavras-reservadas

inteiro	listaInteiro	inclua	repitaAte	caso
decimal	listaDecimal	devolva	repitaEnquanto	seja
booleano	listaBooleano	se	constante	vazio
caractere	listaCaractere	senao	verdadeiro	falso
repita	vezes			

- Nomes
 - 50 caracteres, Sensível à caixa
 - Não pode ser uma palavra-reservada, porém pode conter uma.
 - [a-z] [a-zA-Z0-9_]*
- Tipos e estrutura de dados
 - Declaração explícita. Não se admite declarações múltiplas.
 - Dados primitivos
 - Listagem
 - **vazio**
 - **inteiro**
 - **decimal**
 - **caractere**
 - **booleano**
 - **listaInteiro**
 - **listaDecimal**
 - **listaCaractere**
 - **listaBooleano**
 - Sendo **vazio**, uma palavra para representar que a função não possui parâmetros ou retornos.
 - Declaração de dados primitivos
Exemplos: **inteiro** x1; **decimal** seno
 - Constantes Literais
 - **inteiro:** [0-9]⁺
 - **decimal:** [0-9]⁺. [0-9]⁺
 - **caractere:** . - [\]
 - **booleano:** **verdadeiro|falso**
 - Operações
 - Operador Inversor: booleano
 - Operadores Negativo: inteiro e decimal
 - Operadores Multiplicativos: inteiro e decimal
 - Operadores Aditivos: inteiro e decimal
 - Operadores Comparativos: todos os 4
 - Operadores Lógicos: booleano
 - Operador Atributivo: todos os 4
 - Cadeias de Caracteres
 - Declaração/Formato
Exemplos: **listaCaractere** texto = "A amizade é como BTS: começa sem muitas pretensões, mas vai atingir o mundo!";
 listaCaractere grupo = "Monsta X";

- Operação: ++(concatenador)
- Arranjos
 - Declaração
 - Com Inicialização:
 - **listaInteiro** numeros = { 2, 1, 1, 0, 1, 9, 9, 5 };
 - **listaDecimal** valores = { 0.1, 0.15, 0.2, 0.25 };
 - **listaBooleano** bits = { **falso**, **verdadeiro**, **falso** };
 - Com definição da tamanho:
 - **listaInteiro** data[3];
 - Referência a elementos ocorre usando "[inteiro]", onde o primeiro elemento é o 0, e o último, (tamanho-1)
Exemplos: numeros[2]; bit[0];
 - Armazenamento sequencial na memória
- Equivalência de tipos
 - Não admite coerções ou cast
- Admite constantes nomeadas utilizando as funções do pré-compilador
- Atribuição e expressões
 - Atribuição
 - Através do operador =
 - Expressões Aritméticas, Relacionais e Lógicas
 - Operadores
 - Operador Inversor: **!**
 - Inverte o valor lógico de uma expressão
 - Operador Negativo: **-**
 - Inverte o sinal de um número
 - Operadores Multiplicativos: ***, /, %**
 - *****: Retorna a multiplicação de dois números
 - **/**: Retorna a divisão inteira de dois inteiros ou decimal de 2 decimais
 - **%**: Retorna o resto da divisão inteira de dois inteiros
 - Operadores Aditivos: **+, -**
 - **+**: Retorna a adição de dois números
 - **-**: Retorna a subtração de dois números
 - Operadores Comparativos: **<, >, <=, >=, ==, !=**
 - Operadores Lógicos: **&, &&, |, ||**
 - Operadores Atributivos: **=, +=, -=, *=, /=**
 - Precedência

Na figura abaixo, vemos os operadores suportados na linguagem, onde os operadores nas linhas superiores serão executados primeiro do que os inferiores.

() []	e-d
! - (unário)	d-e
* / %	e-d
+ -	e-d
< <= >= >	e-d
== !=	e-d
&	e-d
	e-d
&&	e-d
	e-d
= += -= *= /=	d-e

- Associatividade: Na tabela acima, os operadores associativos a direita estão em vermelho. Enquanto, os à esquerda estão em azul.
 - O tipo das operações é definido operador a operador.
 - Avaliação em curto-circuito para os operadores lógicos e os multiplicativos
- Sintaxe e exemplo de estruturas de controle
 - Comandos de seleção: **se, senao**

Sintaxe:

```
se ( /*Condição*/ )
{
    /*Instruções*/
}senao
{
    /*Instruções*/
}
```

OU

```
se ( /*Condição*/ )
{
    /*Instruções*/
}
```

- Comandos de Iteração
 - Controle por contador: **repita** (inteiro) **vezes**

Sintaxe:

```
repita 6 vezes
{
    /*Instruções*/
}
```

inteiro contador

```
repita contador vezes
{
    /*Instruções*/
}
```

■ Controle lógico: **repitaEnquanto** e **repitaAte**

● **repitaEnquanto**

Sintaxe:

```
repitaEnquanto( /*Condição*/ )
{
    /*Instruções*/
}
```

● **repitaAte**

Sintaxe:

```
repitaAte( /*Atribuição*/; /*Condição*/; /*Incremento*/ )
{
    /*Instruções*/
}
```

- Desvios Incondicionais: Nenhum

● Subprogramas

- Sintaxe:

```
tipoDeRetorno nomeDaFunção( tipo variavelDeEntrada, ... )
{
}
```

Exemplo

```
inteiro soma( inteiro a, inteiro b )
{
    /*Instruções*/
    devolva 0;
}
```

- Aceita a chamada de funções de outros arquivos utilizando bibliotecas.

- Aceita a passagem dinâmica e estática de parâmetros
- Não aceita programas como parâmetros.

Especificação dos Tokens

Categoria	Lexema	
Verd	verdadeiro	
Falso	falso	
ConstInt	[0-9] ⁺	
ConstReal	[0-9] ⁺ \.[0-9] ⁺	
ConstCarac	'[-\]'	
ConstLisCa	"[-\]" [*]	
Comentario	/*ConstLisCa*/	
Principal	principal	
Id	[a-z] ([a-z][A-Z][0-9][_]) [*]	
Se	se	
Senão	senao	
RepAte	repitaAte	
RepEnq	repitaEnquanto	
Ou	[-]	
E	[&-&&]	
OpU	[-][!]	
OpM	[*][/][%]	
OpA	[+][-]	
OpAtr	[+=][-=][*=][/=]	
OpC	< > <= >=	
OpI	== !=	
Atr	=	

APar	(
FPar)	
Ponto	.	
PV	;	
ACol	[
FCol]	
AChaves	{	
FChaves	}	
Virgula	,	
Inteiro	inteiro	
Decimal	decimal	
Booleano	booleano	
Caractere	caractere	
ListaInt	listaInteiro	
ListaDec	listaDecimal	
ListaBool	listaBooleano	
ListaCarac	listaCaractere	
Inclua	inclua	
Devolva	devolva	
Constante	constante	
Vazio	vazio	

