

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.
 - 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;
 -
- Nomes
 - 50 caracteres, Sensível à caixa
 - `[a-z] ([a-z][A-Z][0-9][_])*`
- Tipos e estrutura de dados
 - Declaração explícita. Não se admite declarações múltiplas.
 - Dados primitivos
 - Declaração de dados primitivos
Exemplos: **inteiro** x1; **decimal** seno
 - Constantes Literais
 - **inteiro**: `[0-9]+`
 - **decimal**: `[0-9]+\.[0-9]+`
 - **caractere**: `.` - `[_]`
 - **booleano**: **verdadeiro**|**falso**
 - Operações
 - Operadores Unários: booleano
 - Operadores Multiplicativos: inteiro e decimal
 - Operadores Aditivos: inteiro e decimal
 - Operadores Comparativos: caractere, inteiro e decimal
 - Operador Igualitários: 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
 - **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** };
 - 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
 - =
 -
 - Expressões Aritméticas, Relacionais e Lógicas
 - Operadores
 - Operadores Unarios: !,-
 - Operadores Multiplicativos: *, /, %
 - Operadores Aditivos: +,-
 - Operadores Incrementadores: +=, -=, *=, /=
 - Operadores Comparativos: <,>,<=,>=,
 - Operadores Igualitários: ==,! =
 - Operadores Relativos: &,&&,,||
 - Operador Atributivo: =

■ Precedência

() []	e-d
! -	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
- 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: repitaAte

Sintaxe:

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

- Controle lógico: repitaEnquanto

Sintaxe:

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

- Desvios Incondicionais: Nenhum

- Subprogramas

- Sintaxe:

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

Exemplo

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

- Aceita a chamada de funções de outros arquivos utilizando bibliotecas.
- Aceita a passagem dinamica e estática de parametros
- Não aceita programas como parâmetros.

Especificação dos Tokens

1. e

Categoria	Lexema	
Verd	verdadeiro	
Falso	falso	
ConstInt	[0-9] ⁺	
ConstReal	[0-9] ⁺ \.[0-9] ⁺	
Id	[a-z] ([a-z][A-Z][0-9][<u> </u>]) [*]	
Se	se	
Senão	senao	

RepAte	repitaAte	
RepEnq	repitaEnquanto	
Ou	[] [][]	
E	[&] [&&]	
OpU	[-] [!]	
OpM	[*] [/] [%]	
OpA	[+] [-]	
OpInc	[+=] [-=] [*=] [/=]	
OpC	< > <= >=	
OpI	== !=	
Atr	=	
APar	(
FPar)	
Ponto	.	
PV	;	
ACol	[
FCol]	
AChaves	{	
FChaves	}	
Virgula	,	

2.

