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**
	repitaContador			

- Nomes
    - 50 caracteres, Sensível à caixa
    - Não pode ser uma palavra-reservada, porém pode conter uma.
    - [a-z] [a-zA-Z0-9_]*

- Entrada e Saída de Dados
    - Entrada
      inteiro novo = recebaNoTeclado();

    - Saída
      escrevaNaTela(novo);
      escrevaNaTela(“%d”, novo);

- 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**: $[0\text{-}9]^+$
            - **decimal**: $[0\text{-}9]^+.[0\text{-}9]^+$
            - **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”;

  - Operações Exclusivas
    - Concatenação ocorre pela função **concatene**
      Sintaxe:

      listaDeCaractere resultado = **concatene**(listaDeCaractere primeiro, listaDeCaractere segundo);

- 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 a palavra-reservada **constante**

- Atribuição e expressões

  - Atribuição
    - Através do operador =

  - Expressões Aritméticas, Relacionais e Lógicas

Sabendo que nesse subtópico, onde é mencionado o termo “número”, leia-se como ou tipo inteiro ou decimal.

- Operadores
  - Operador Inversor: **!**

        - Inverte o valor lógico de uma expressão ou variável booleana

    - 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: **<,>,<=,>=,==,!=**
        - **<:** Retorna verdadeiro se a expressão a esquerda for menor que a direita
        - **>:** Retorna verdadeiro se a expressão a esquerda for maior que a direita
        - **<=:** Retorna verdadeiro se a expressão a esquerda for menor ou igual que a direita
        - **>=:** Retorna verdadeiro se a expressão a esquerda for maior ou igual que a direita
        - **==:** Retorna verdadeiro se a expressão a esquerda for igual que a direita
        - **!=:** Retorna verdadeiro se a expressão a esquerda for diferente que a direita

    - Operadores Lógicos: **&,&&,|,||**
        - **& e &&:** Ambos funcionam como o operador lógico E, tendo como diferença a precedência.
        - **| e ||:** Ambos funcionam como o operador lógico OU, tendo como diferença a precedência.

    - Operadores Atributivos: **+=,-=,*=,/=**
        - Funcionam como o operador do primeiro caractere, tendo a diferença, salvar o resultado da operação no primeiro operador.

- Precedência

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

Operadores	Precedência	Associatividade
() []	Nível 1	Da direita para esquerda
! -(unário)	Nível 2	Da esquerda para direita
* / %	Nível 3	Da direita para esquerda
+ -	Nível 4	Da direita para esquerda
< > <= >=	Nível 5	Não associativos
== !=	Nível 6	Não associativos
& |	Nível 7	Da direita para esquerda
&& ||	Nível 8	Da direita para esquerda
= += -= *= /=	Nível 9	Da esquerda para direita

        - Associatividade: Na tabela acima, a coluna mais à direita descreve a associatividade de cada conjunto de operadores.
        - 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*/
}
```

- **caso** (variavel) **seja** (constanteTipo)

Sendo variavel, uma variável dos tipos: inteiro, decimal, caractere, booleano; e constanteTipo, uma constante correspondente ao tipo de variavel.

```
Sintaxe:
caso ( /*variavel*/ )
{
        seja (constante1Tipo)
        {
                /*Instruções*/
        }
        .
        .
        .
        seja (constanteNTipo)
        {
                /*Instruções*/
        }
}
```

- Comandos de Iteração
  - Controle por contador:
    repitaContador(variavel;inicio;fim;tamanhoDoPasso)

```
Sintaxe:
inteiro n;

repitaContador( n; 1; 100; 2  )
{
        /*Instruções*/
}

inteiro contador;

repitaContador (contador; n; n*n; 1)
{
```

```
        /*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.
  - 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-zA-Z0-9_]*	
Se	se	
Senão	senao	
RepAte	repitaAte	
RepEnq	repitaEnquanto	
Ou	|{1,2}	
E	&{1,2}	
OpU	[-!]	
OpM	[*/%]	
OpA	[+-]	
OpAtr	[+-*/]=	
OpC	[<>]={0,1}	
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	
RepCont	repitaContador	
Caso	caso	
Seja	seja	