

# Web Games

Prof. Luciano Alves

*Functions e passagem de  
parâmetros*

### Fundamentos sobre programa

Na programação, um algoritmo é um conjunto de instruções que são executados sequencialmente. Vejamos um exemplo abaixo:

```
var soma: int;  
var numero1 : int;  
var numero2 : int;
```

```
numero1 = 10;  
numero2 = 3;  
soma = numero1 + numero2;
```

```
trace("Soma = " + soma);
```

### **Fundamentos sobre programa**

Agora imagine um bloco de comandos que possua aproximadamente 10 linhas de código para realizar uma determinada operação. E que essa operação precise que seja realizada em 5 pontos diferentes do meu programa ? O código do meu programa seria **GRANDE**.

Agora imagine um bloco de código com 50 linhas de comandos para realizar uma operação e que esse bloco precisasse repetir em 20 pontos diferentes do código do seu jogo ? **CARAMBA!!! Meu código vai ficar UMA BAGUNÇA.**

### Utilizando uma function

Calma, NÃO SE DESESPERE.


Para situações como essa , existe uma estrutura que podemos utilizar para facilitar a nossa vida , na qual chamamos de **function**.

Uma **function** (ou **função** no bom português) é uma estrutura na qual definimos um bloco de comandos, e a execução do mesmo só ocorrerá no ponto no qual a **function** for chamada.

### Utilizando uma function

*Sintaxe :*

```
function <nome da função> ( ) : <tipo retorno> {  
    <comando1>;  
    <comando2>;  
    :  
    <comandon>  
}
```



Aqui definimos o TIPO DE DADOS que a função vai retornar. Veja na página seguinte os tipos de dados que ela pode retornar.

### Function : Tipos de dados de retorno

Veja abaixo os tipos de dados mais comuns que uma **function** pode retornar.

**int** : Retorna números inteiros de 32 bits (positivos e negativos), cujo intervalo é 2.147.483.648 a 2.147.483.647.

**uint** : Retorna números inteiros de 32 bits (somente), cujo intervalo é 0 a 4.294.967.295.

**Number** : Retorna números inteiros e de ponto flutuante (positivos e negativos) de 64 bits.

### Function : Tipos de dados de retorno

Veja abaixo os tipos de dados mais comuns que uma **function** pode retornar.

**String** : Retorna valores alfanuméricos (incluindo números, letras e símbolos) no formato Unicode (caracteres de 16 bits cada). Seu tamanho é variável.

**Boolean** : Retorna valores lógicos como **true** (Verdadeiro) e **false** (Falso).

**void**: O **void** indica que uma **function** NÃO VAI RETORNAR VALOR NENHUM. Simplesmente a **function** vai executar as instruções sem retornar nada.

### Utilizando uma function

*Exemplo : Com a janela de código (Actions) aberta, digite :*

```
function DigaOi ( ) : void {  
  
    trace("Oooooiii...");  
    trace("Tudo bem ?");  
  
}
```

```
DigaOi();  
trace("Fim do programa");
```



## Utilizando uma function

### Como funciona a sua execução ? (Siga os passos)

```
function DigaOi ( ) : void {
```

```
    trace("Oooooiii...");  
    trace("Tudo bem ?");
```

```
}
```

1º Definimos a função e o bloco de comandos

2º Na linha indicada pela SETA VERMELHA, ocorre a chamada para a função *DigaOi* . Neste momento ocorre um desvio para o código dentro da função, onde será executado seu código indicado pelo QUADRADO ROXO.

```
DigaOi();  
trace("Fim do programa");
```

3º Depois da execução da função, O PROGRAMA SEGUE SUA EXECUÇÃO NORMAL, partindo desta SETA.

### Utilizando uma function (Retornando valores)

Para que uma function retorne um valor, precisamos utilizar uma palavra chave chama **return** dentro da função. Veja Um exemplo abaixo:

```
function Soma ( ) : int {
```

Aqui definimos seu tipo de retorno com *int*.

```
    var num= 2;
```

```
    var num2 = 2;
```

```
    return (num + num2) ;
```

Aqui o return é chamado, como parâmetro passamos num + num2 (ambos com valor 2). Logo, aqui ela vai retornar 4.

```
}
```

```
var minhaSoma;
```

```
minhaSoma = Soma();
```

Aqui nesta linha, a variável “minhaSoma” vai receber o valor retornado pela função *Soma*.

### Utilizando uma function (Trabalhando com parâmetros)

*O que são parâmetros ? São um conjunto de valores que passamos para uma função para que elas sejam utilizadas dentro dela como parte do processamento. Veja a sintaxe abaixo da declaração de uma função.*

```
function <nome da função>( <variável1 > : <tipo>,..., <variável n>  
: <tipo> ) : <tipo> {  
  
}
```

**Utilizando uma function (Trabalhando com parâmetros)**

**Exemplo:**

```
function Soma(numero : int , numero 2 : int ) : <tipo> {  
  
    return (numero + numero2);  
  
}  
  
trace(Soma(12,3));  
  
trace(Soma(4,5));
```