(Parte I) – Funções em C#

Modularização - Técnica de programação que se caracteriza pela divisão de um programa em **subprogramas**:

- **Funções e procedimentos (Métodos em POO)**

- Função

Na linguagem C#, definiremos função como uma rotina - método - bloco de instruções com função bem definida, com objetos próprios (variáveis locais, constantes,…), que realiza uma tarefa específica e retorna um valor (numérico, literal, lógico, endereço, etc.)

- Procedimento

Uma função que **não retorna nenhum valor** (retorno nulo) é chamada, em algumas linguagens, de procedimento, em C#, **é uma função com retorno do tipo void**

- Métodos

Abordaremos mais a Frente no contexto de POO as funções como métodos da classe".

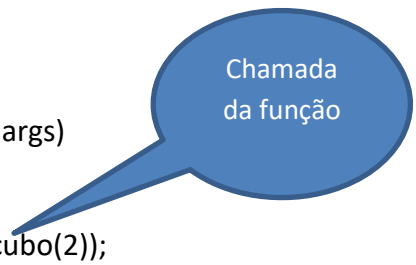
Sintaxe :

```
static tipo_de_retorno nome_função (declaração de parâmetros)
{     declarações locais

      bloco de comandos (instruções…)
      return valor_de_retorno;
}
```

- Exemplo – Função para calcular o cubo de um Número

```
public static int cubo(int L)
{
        return L * L * L;
 }
 static void Main(string[] args)
{
      Console.WriteLine(cubo(2));
// após execução a função devolve o valor de retorno que será utilizado na  instrução da chamada
}
```



Exemplo – **Função para calcular o fatorial de um Número**

```
public static int Fatorial(int n)
{
        int fat = 1;
        int i = n;
        while (i > 0)
        {
            fat = fat * i;
            i--;
        }

        return fat;
}
```

```
static void Main(string[] args)
{
  int x,resultado;

   x=Convert.ToInt32(Console.Readline())
   y=Convert.ToInt32(Console.Readline())
   resultado = Fatorial(x) + Fatorial(y);
   ....
}
```

Exemplo – **Função que retorna o maior entre dois números**

```
static int maximo(int a, int b)
{
        if(a>b)
                return a;
        else
                return b;
}
```

Passagem de Parâmetros por Valor e Referência

- Passagem de Parâmetros por Valor

- Nesse tipo de passagem de parâmetro uma cópia do valor é efetuada
- Uma alteração em seus valores, nas respetivas sub-rotinas (funções), não acarreta alteração nas variáveis externas a eles associadas, pois ocupam endereços de memória diferentes daquelas.

Exemplo:

```
static int somaUM(int a, int b)
{
        a++;
        b++;
        return a + b;
 }

static void Main(string[] args)
{
        int x = 3, y = 4;
        int resultado;

        resultado = somaUM(x, y);
        ...
 }
```

- Passagem de Parâmetros por Referência

- Os parâmetros por referência não ocupam um novo espaço de memória, mas apenas criam um "apelido" para a variável associada, e por isso são utilizados quando dentro da função queremos alterar os valores de origem.
- Este tipo de parâmetro pode ser utilizado também para retornar valores de uma sub-rotina.
- A linguagem C# indica os parâmetros por referência com o termo ref ("referência") antes da identificação dos tipos de dados

(Ver exemplo a seguir)

– **Exemplo – Passagem de parâmetros por referência**

```
static void Ordena(ref double a, ref double b)
{
        double x;

         if (a > b)
        {
             x = a;
             a = b;
             b = x;
        }

              return;
}

 static void Main(string[] args)
{
        double x = 5, y = 4;

        Ordena(ref x, ref y);  // na chamada x= 5 e y=4, após a execução     valores
                               //  foram trocados – "sem ref" isto não aconteceria

        Console.WriteLine(x);  //  mostra 4

        Console.WriteLine(y);  // mostra 5
    }
```