LP2 – Aula 08
TEÓRICA

Profª Mª Denilce Veloso
- denilce.veloso@fatec.sp.gov.br
- denilce@gmail.com

C# .NET – Procedimentos e Funções

SubRotinas – C#

Uma subrotina é qualquer bloco de código agrupado com fins de reutilização e organização. As subrotinas são extremamente importantes em programação. Principalmente em programação orientada a objetos, as rotinas servem para encapsular instruções, e estas colocadas em métodos, acabam sendo reaproveitadas em outros programas ou telas. As subrotinas são divididas em dois tipos: procedimentos e funções

C# .NET – Procedimentos vs Funções

- Funções sempre retornam algum dado/valor ao código que a chamou (código executor), através do comando return.

- Procedimentos nunca retornam valor, serão sempre void.

Em C#, chamamos os procedimentos de funções com retorno void.

*** é possível passar parâmetros nas funções e procedimentos*

C# .NET - Funções

- **Sem retorno**

```
void Soma1 ()
        {
            Double resultado = 2 + 3;

            MessageBox.Show("O resultado é:" +
resultado.ToString());
        }
```

Chamada

Soma1();

C# .NET - Funções

- **Com retorno**

```
double Soma2()
        {
            Double resultado = 2 + 3;
            return resultado;
        }
```

Chamada

```
double x =  Soma2();
```

C# .NET – Funções

- **PASSAGEM DE PARÂMETROS**

Funções (ou método se estiverem dentro da classe) podem receber parâmetros de duas formas diferentes:

Por valor: Nesse caso, uma cópia do parâmetro é criada na pilha local do método, e mesmo que o parâmetro seja modificado pelo método, esta modificação não será visível fora dele.

Por referência: Nesse caso, a referência do parâmetro na memória é passada e qualquer modificação que este sofrer no método será visível externamente. Para passar parâmetros por referência usa-se a palavra reservada ref antes do tipo do parâmetro.

Quando não é especificado o tipo de passagem do parâmetro por default, a passagem é por valor e não requer nenhum modificador adicional para tal fim.

C# .NET - Funções

- **Sem retorno e com parâmetros**

```
 void Soma3(double x, double y) {
x=40;
            Double resultado = x + y;

            MessageBox.Show("O resultado
é:" + resultado.ToString());
        }
```

Chamada

```
Double a=2; Double b=3;
Soma3(a,b);
MessageBox.Show(a);
```

C# .NET - Procedimentos vs Funções

- **Com retorno e com parâmetros**

```
        double Soma4(double x, double y)
        {
X=50;
            Double resultado = x + y;
            return resultado;
        }
```

Chamada

```
double a=2;
double b=3;
double x =  Soma4(a,b);
MessageBox.Show(a);
```

C# .NET - Funções

- **Sem retorno e com parâmetro por referência**

```
void Soma5(ref double resultado, ref double a)
        {
             a=40;
            resultado = 2 + 3;
        }
```

Chamada

```
double r=0;
Double x=20;
Soma5(ref r, ref x);
MessageBox.Show(x);
```

C# .NET - Funções

- **Com retorno com parâmetro por referência**

```
Double Soma6(ref double resultado)
        {
            resultado = 2 + 3;
            return resultado;
        }
```

*** nesse caso a variável resultado e o return terão o mesmo valor*

Chamada

```
double r=0;
Double x = Soma6(ref r);
MessageBox.Show(r); // ou x
```

- **Parâmetros somente de saída**

O **parâmetro out** significa um **parâmetro** de referência. Assim, os **parâmetros de saída** são semelhantes aos **parâmetros** de referência, exceto pelo fato de que eles transferem dados para fora do método. A palavra-chave **out** força os argumentos a serem passados por referência.

If (double.TryParse(string, out retorno))

C# .NET - Procedimentos vs Funções

- **Sem retorno e com parâmetro de saída**

```
void Soma7(out double resultado)
        {
            resultado = 2 + 3;
        }
```

Chamada

```
double r=0;
Soma7(out r);
```

C# .NET - Funções

- **Com retorno e com parâmetro de saída**

```
Double Soma8(out double resultado)
        {
            resultado = 2 + 3;
            return resultado;
        }
```

*** nesse caso a variável resultado e o return terão o mesmo valor*

Chamada

```
double r=0;
double x = Soma8(out r);
```

C# .NET - Funções

- **Sem retorno, com parâmetro de saída e com parâmetro opcional**

```
void Soma9(out double retorno, double x,
          double y, double z=0) //funcao passando
                                    // parâmetro de saída
                                    //e parâmetro opcional
      {
          if (z > 0)
            retorno = (x + y + z) / 3;
          else
           retorno = (x + y) / 2;
        }
```

Chamada

```
double r=0;
Soma9(out r, 2, 3);
```

AVISO

>> Não esquecer de instalar o SQL SERVER para as próximas aulas

- Através da loja https://azureforeducation.microsoft.com/devtools

- Ou se quiser uma versão mais leve baixa aqui a versão Expressa

https://www.microsoft.com/pt-br/sql-server/sql-server-downloads

>> Por favor, avisar no grupo da classe.