FICHA 4

SUBPROGRAMAS

MODALIDADE:	Aprendizagem +	Escolha um item.	
CURSO:	Tecnologias de Programação de Sistemas de Informação		
UC:	Desenvolver programas complexos em linguagem de programação estruturada	CÓDIGO UC:	UC00607
FORMADOR/A:	Bruno Silva	DATA:	

OBJETIVOS

- Perceber a implementação de Procedimentos, Funções e Passagem de parâmetros com a linguagem C#

Não há uma diferenciação clara entre procedimentos e funções em C#. Na verdade, ambos são métodos que têm um nome e podem ter parâmetros, diferindo no facto que:

- ***Procedimentos***: este tipo de método não devolve qualquer valor (tendo então o caracterizador ***void***)

Procedimentos em C#

```
void <NomeProcedimento> (<ref,out> tipo_de_dado <var1>, ..., <ref/out>
tipo_de_dado <varn>)
{
      <bloco de programa>
}
```

  - Um procedimento é identificado com o nome <***NomeProcedimento***> seguido de parenteses ();
    - **Dentro dos parenteses, pode ou não ter parâmetros de entrada de valores** para utilizar no procedimento;
  - <***bloco de programa***> é o bloco do procedimento e rege-se pelas mesmas leis que o bloco do programa principal (onde colocamos o código que queremos).

- ***Funções***: obriga à inclusão do ***tipo do dado*** a ser devolvido (tendo então o caracterizador de um tipo de dados, que pode ser do tipo ***int***, ***string***, ***double***, ***float***, etc…), isto é, **retorna um determinado tipo de valores especificado pelo programador**.

Funções em C#

```
<tipo_de_dado_a_devolver> <NomeFunção> (<ref,out> tipo_de_dado <var1>,
..., <ref/out> tipo_de_dado <varn>)
{
      <bloco de programa>
      return <expressão ou valor> // (do tipo_de_dado_a_devolver)
}
```

- Uma função é identificada com o nome <***NomeFunção***>. Pode ou não possuir parâmetros.

- No entanto **é obrigatório ter um tipo a devolver** (a função devolve sempre um valor e esse valor é do tipo <***tipo_de_dado_a_devolver***>). No caso de possuir parâmetros, as considerações aplicadas aos procedimentos, aplicam-se aqui igualmente.

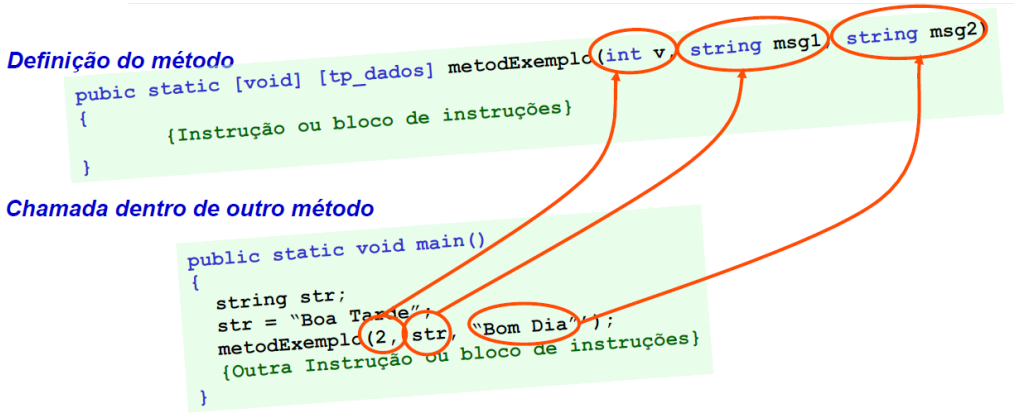
- <***bloco***> é o bloco do procedimento e rege-se pelas mesmas leis que o bloco do programa principal, tal como já referido a propósito do procedimento.

- **Na <u>última instrução</u> do bloco de instruções** incluir um retorno de dados (***<u>return</u>***) que interrompe a execução da função chamada, devolvendo o fluxo de execução ao programa (ou subprograma), devolvendo igualmente o valor ou expressão <***expressão ou valor***>.

Subprogramas - procedimentos e funções (passagem de dados por parâmetros ou referência)

Quando passamos parâmetros para um método, o C# vai adequar cada um desses aos parâmetros que se encontram descritos no método, respeitando a ordem pela qual os parâmetros são apresentados.

- O ***primeiro parâmetro declarado na definição do método <u>toma</u>*** o ***valor do primeiro argumento apresentado na chamada***, o ***segundo*** vai tomar o valor do ***segundo*** e ***assim sucessivamente***.



Quando utilizamos uma ***função*** ou ***procedimento*** podemos receber ou não valores de entrada. Mas dentro dos ***valores de entrada*** (***parâmetros),*** podemos ter ***2 tipos de parâmetros***:

- ***Por passagem por parâmetros (o mais usado)*** – estamos a trabalhar **com uma cópia dos dados que são passados para dentro de uma função ou procedimento**. Qualquer alteração estará confinada à cópia efetuada, e nunca ao argumento original.

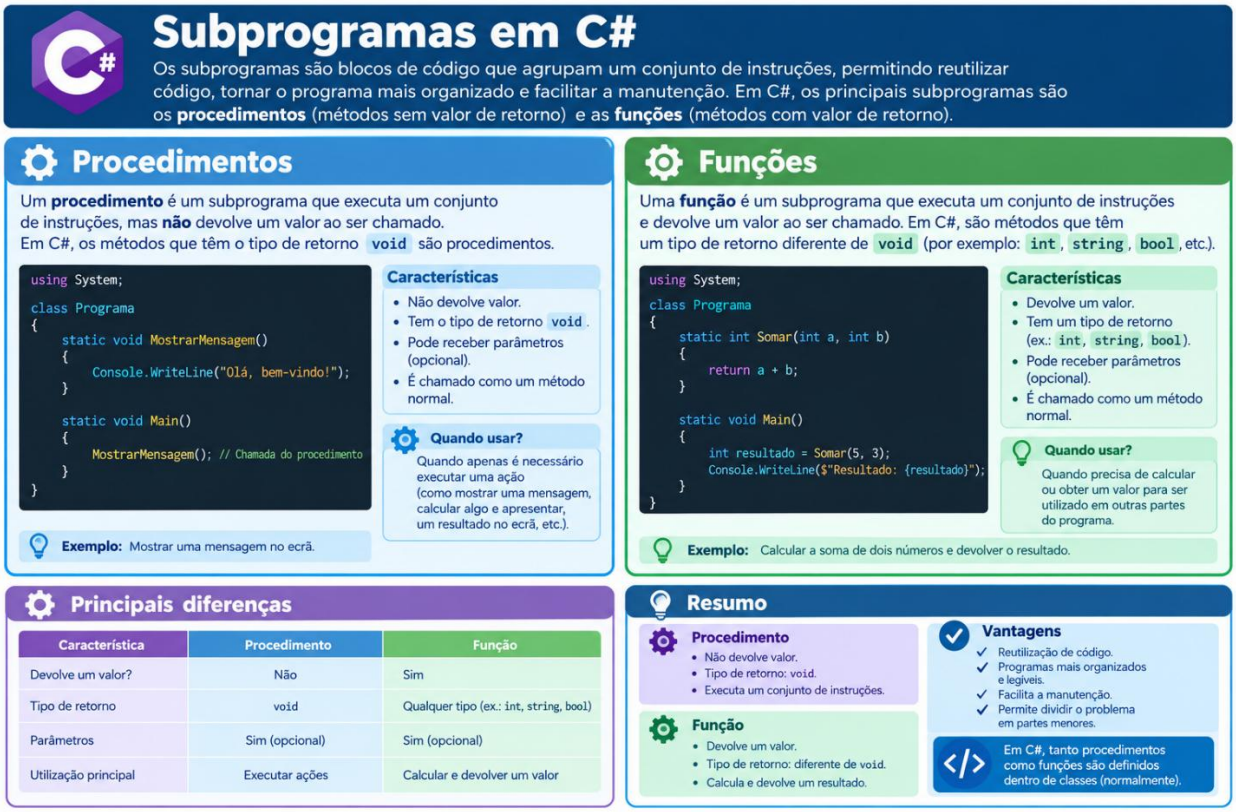
```
//Procedimento que vai receber 2 parametros de entrada
1 reference
static void multiplica_2x(int num1, int num2) {
    Console.WriteLine("Recebemos o num1 que tem {0} e o num2 que tem {1}", num1, num2);

    //Vamos alterar o valor da variavel num1
    num1 = 15;
    Console.WriteLine("Num1 tem o valor {0}", num1);
}

0 references
static void Main(string[] args) {
    int numero1 = 2, numero2 = 15;
    //vamos invocar o procedimento multiplica_2x
    //que vai passar dois parâmetros de entrada
    multiplica_2x(numero1, numero2);

    Console.WriteLine("Numero1 tem o valor {0}", numero1);
}
```

Como podemos observar, a alteração do valor de ***num1*** só teve efeito dentro subprograma, pois estamos a ***trabalhar com uma cópia dos valores originais*** que foram ***passados do programa principal***;



Exercícios

1) Elabore um **procedimento** com o nome "**Mostra_Nome**" que receba **1 parâmetro de entrada** chamado **nome** e dentro do procedimento, deve mostrar a mensagem "Olá, …" onde nas reticências deve estar o nome que recebeu no parâmetro de entrada.

No programa principal, deve **pedir** o nome do utilizador, **invocar** o procedimento e **passar** o valor do parâmetro de entrada.

2) Elabore um **procedimento** com o nome "**Soma_Numeros**" para receber 2 parâmetros de entrada (em que ***numero1*** e ***numero2*** devem ser **parâmetros de entrada do procedimento**).

Dentro do procedimento, deve realizar a soma de dois números e mostrar o resultado.

No programa principal, deve **pedir** os 2 números ao utilizador, **invocar** o procedimento e **passar** os valores dos parâmetros de entrada.

3) Elabore um **procedimento** com o nome "**Media_Notas**" para receber 3 parâmetros de entrada (em que ***nota1***, ***nota2*** e ***nota3*** devem ser **parâmetros de entrada do procedimento**).

Dentro do procedimento, deve realizar o cálculo da média de 3 notas e mostrar o resultado.

No programa principal, deve **pedir** as 3 notas ao utilizador, **invocar** o procedimento e **passar** os valores dos parâmetros de entrada.

4) Elabore uma **função** com o nome "**Converte_Temperatura**" para realizar a conversão da temperatura celsius para Fahrenheit (em que ***temp*** deve ser o **parâmetro de entrada da função**). Como estamos a trabalhar numa função, no final do código, tem de **retornar** o valor do resultado para o programa que invocou a função (**Dica:** o retorno deve ser do mesmo tipo de dados que declarou no início da função).

No programa principal, deve **pedir** a temperatura em celsius, **invocar** a função, **passar** o valor e **mostrar** o resultado **que foi devolvido da função.**

5) Elabore uma **função** com o nome "**Comparar_Numeros**" para receber dois números inteiros (em que ***num1 e num2*** devem ser os **parâmetros de entrada da função**) e verificar se algum dos números é maior que o outro **ou** igual.

Como estamos a trabalhar numa função, no final do código, tem de retornar o valor do resultado da comparação para o programa que invocou a função (**Dica:** o retorno deve ser do mesmo tipo de dados que declarou no início da função).

No programa principal, deve **pedir** os dados dos números, **invocar** a função, **passar** os valores e **mostrar** o resultado **que foi devolvido da função.**

6) Elabore uma **função** com o nome "**Devolve_Divisao**" para receber dois números inteiros (em que ***num1 e num2*** devem ser os **parâmetros de entrada da função**) e antes de fazer a divisão, deve verificar, se o 2º número é meor ou igual a 0, mostrando a mensagem "Não pode dividir por 0", senão, deve mostrar o resultado com a mensagem "Divisão: …".

Como estamos a trabalhar numa função, no final do código, tem de retornar o valor do resultado da operação para o programa que invocou a função (**Dica:** o retorno deve ser do mesmo tipo de dados que declarou no início da função).

No programa principal, deve **pedir** os dados dos números, **invocar** a função, **passar** os valores e **mostrar** o resultado **que foi devolvido da função.**