Ficha de Trabalho N.º3

Objectivos

Estudo dos Subprogramas: Procedimentos; Passagem de parâmetros; Funções.

Procedimentos e Funções em 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 do primeiro não devolver qualquer valor (tendo então o caracterizador *void*) enquanto que o segundo obriga à inclusão do tipo do dado a ser devolvido.

Procedimentos em C#

```
static 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> Pode ou não possuir parâmetros. No caso de possuir parâmetros, estes podem ser de duas espécies: por valor ou por referência.

Os parâmetros por referência devem ser precedidos de *ref*. A cada parâmetro deve ter associado o respectivo tipo de dados. Um parâmetro que seja utilizado para saída de valores deve der precedido de *out*.

A chamada a procedimento do programa chamante deve incluir os parâmetros (se o procedimento os possuir) e incluir a palavra reservada *ref*, se o parâmetro for a passar por referência. O mesmo tipo de considerações se aplicam a *out*.

<bloco> é o bloco do procedimento e rege-se pelas mesmas leis que o bloco do programa principal (declaração de constantes, tipos, variáveis, etc.).

Funções em C#

```
static <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.

A referir adicionalmente a inclusão de **return** que interrompe a execução do função chamada devolvendo o fluxo de execução ao programa (ou sub-programa) chamante para a instrução seguinte à chamada da função, devolvendo igualmente o valor calculado ou expressão <expressão ou valor>.

1 - Elabore o algoritmo de um procedimento que determine e apresente no monitor os *n* primeiros múltiplos de um número inteiro *m* (*n* e *m* devem ser parâmetros de entrada do procedimento). Implemente o algoritmo em linguagem C#, devendo construir um programa principal (main) que se encarrega das tarefas de I/O e chamar o procedimento criado.

2 - Faça o algoritmo de um procedimento que, sendo dado como parâmetro de entrada um número inteiro, calcule e apresente no monitor todos os seus múltiplos inferiores a 100. Implemente o algoritmo em linguagem C#, tendo em conta os requisitos (relativos à estrutura do programa) definidos em1.

3 - Elabore um procedimento que apresente no monitor todos os números pares entre dois números inteiros *n* e *m* (*n*<*m*). Os valores de *n* e *m* devem ser passados como parâmetros para o procedimento e no caso de (*m*<*n*) deve ser apresentada a mensagem "Valores Inválidos!".

4 - Analise o seguinte programa e verifique a diferença existente entre a passagem de parâmetros por valor e por referência. Inclua os comentários que achar apropriados para clarificar a diferença.

```
class ExemploPassParam
{
    static void Main(string[] args)
    {
        int a, b;
        a=1;
        b=2;
        Console.WriteLine("Passagem de parâmetros por valor");
        proc1(a, b);
        Console.WriteLine("a = {0}\t b = {1}\n",a,b);
        Console.WriteLine("Passagem de parâmetros por referência");
        proc2(ref a, ref b);
        Console.WriteLine("a = {0}\t b = {1}\n",a,b);
    }
    static void proc1(int x, int y)
    {
        Console.WriteLine("a = {0}\t b = {1}",x,y);
        x=100;
        y=200;
    }
    static void proc2(ref int x, ref int y)
    {
        Console.WriteLine("a = {0}\t b = {1}",x,y);
        x=100;
        y=200;
    }
}
```

5 - Implemente um procedimento que permita trocar o valor de duas variáveis, cujo valor foi especificado pelo utilizador. Teste-o num pequeno programa criado para o efeito.

6 - Escreva um procedimento que, sendo dados como parâmetros as medidas dos dois catetos de um triângulo rectângulo, calcule e mostre no monitor a medida da hipotenusa.

7 - Escreva uma nova versão do subprograma anterior que calcule a medida da hipotenusa, mas não a mostre no monitor. Em vez disso a medida da hipotenusa deve ser enviada para o exterior por forma a ser utilizada no exterior do subprograma.

8 - a) Se tivesse que devolver dois valores, a solução usada na questão anterior seria utilizável? Justifique.
b) Mostre qual a alternativa, supondo a devolução da hipotenusa e de um número aleatório gerado na função.

9 - Elabore um programa em C# que determine o quadrado de um número inteiro ***n***. O número ***n*** deve ser pedido ao utilizador através de uma função e o seu quadrado deve ser calculado através de outra função.

10 - Escreva um programa em C# que permita a conversão de temperaturas em graus Centígrados para graus Fahrenheit e vice versa. Para tal deverá escrever dois procedimentos, um para realizar a conversão para Celsius e outro para realizar a conversão para Fahreneit. Cada um destes procedimentos deverá receber como parâmetro o valor da temperatura a converter e mostrar no monitor a temperatura convertida ($C = 5(F - 32)/9$; $F = (9C/5)+32$).

11 - Use funções para efectuar as conversões de temperaturas do exercício anterior.

12 - Escreva um programa em C# que utilize uma função chamada multiplo que receba dois valores inteiros e devolva *true* ou *false* se um dos valores é múltiplo (ou não) do outro. O programa principal, deverá aceitar dois números e dizer se os números introduzidos são ou não múltiplos um do outro.

13 - a) Implemente uma função chamada `factorial` que calcule *n!* (factorial de *n* em que *n* é um inteiro positivo). O valor de *n* deve ser um parâmetro da função e o valor calculado deve ser enviado para o exterior
b) Altere o programa anterior de forma a mostrar o factorial dos números de 1 a 10.

14 - Escreva um programa em C# que inclua uma função chamada `potencia` que permita calcular x^n (x real, n inteiro positivo).

15 - Considere a função $f(x)$. Escreva um programa que utilize funções C# para calcular o valor de $f(x)$ para um x e n dados pelo utilizador. Sugestão: utilize as funções `factorial` e `potencia` dos exercícios 13 e 14.

$$f(x) = \sum_{i=1}^{n} \frac{x^i}{i!}$$

16 - Faça o cálculo da função anterior usando uma única função por forma a que o resultado seja obtido mais eficientemente.