FUNÇÃO

Funções são as estruturas que permitem ao usuário separar seus programas em blocos. Se não as tivéssemos, os programas teriam que ser curtos e de pequena complexidade. Para fazermos programas grandes e complexos temos de construí-los bloco a bloco.

Sintaxe de uma função:

```
static tipo_de_retorno nome_da_função (declaração_de_parâmetros)
{
        corpo_da_função
}
```

O tipo-de-retorno é o tipo de variável que a função vai retornar.

A declaração de parâmetros é uma lista com a seguinte forma geral: *tipo var1, tipo var2, ... , tipo varN*

O tipo deve ser especificado para cada uma das N variáveis de entrada. É na declaração de parâmetros que informamos ao compilador quais serão as entradas da função.

O corpo da função é onde as entradas são processadas, saídas são geradas ou outras coisas são feitas.

O Comando return

O comando **return** tem a seguinte forma geral: *return valor_de_retorno;*

Digamos que uma função está sendo executada. Quando se chega a uma declaração **return** a função é encerrada imediatamente e, se o valor de retorno é informado, a função retorna este valor. É importante lembrar que o valor de retorno fornecido tem que ser compatível com o tipo de retorno declarado para a função.

Exemplo 29: Cálculo do Dobro de um número

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            int a, r;
            Console.Write("Digite um numero: ");
            a = leia();
            dobro(a);
            Console.ReadKey();
        }

        static int leia()
        {
            return int.Parse(Console.ReadLine());
        }

        static void dobro(int n)
        {
            int d = n * 2;
            Console.WriteLine("Dobro de {0} = {1}", n, d);
        }

    }
}
```

Exemplo 30: Cálculo do Dobro de um número

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            int a, r;
            escreva();
            a = leia();
            r = dobro(a);
            exibe(a, r);
            Console.ReadKey();
        }

        static void escreva()
        {
            Console.Write("Digite um numero: ");
        }

        static int leia()
        {
            return int.Parse(Console.ReadLine());
        }

        static int dobro(int n)
        {
            return n * 2;
        }

        static void exibe(int num, int v)
        {
            Console.WriteLine ("Dobro de {0} = {1}", num, v);
        }
    }
}
```

Exemplo 31:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            int num;
            Console.Write("Entre com numero: ");
            num = int.Parse(Console.ReadLine());

            if (restoPorDois(num)==0)
                Console.WriteLine("\n\nO numero e par.\n");
            else
                Console.WriteLine("\n\nO numero e impar.\n");

            Console.ReadKey();
        }

        static int restoPorDois(int a)
        {
            return a % 2;
        }

    }
}
```

É importante notar que, como as funções retornam valores, podemos aproveitá-los para fazer atribuições, ou mesmo para que estes valores participem de expressões. Mas *não* podemos fazer:

```
func(a,b)=x;     /* Errado! */
```

Exemplo 32:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
             double num;
             Console.Write("Entre com um numero: ");
             num = double.Parse(Console.ReadLine());

             num = Quadrado (num);

             Console.WriteLine("\n\nO seu quadrado vale: {0}\n", num);
             Console.ReadKey();
        }

        static double Quadrado(double n)
        {
            return n * n;
        }

    }
}
```

O Tipo void

Em inglês, **void** quer dizer vazio.. Ele nos permite fazer funções que não retornam nada e funções que não têm parâmetros. Podemos agora escrever o protótipo de uma função que não retorna nada:

static void nome_da_função (declaração_de_parâmetros);

Neste tipo de função, não temos valor de retorno na declaração return. Aliás, neste caso, o comando return não é necessário na função.

Podemos, também, fazer funções que não têm parâmetros:

static tipo_de_retorno nome_da_função (void);

ou, ainda, que não tem parâmetros e não retornam nada:

void nome_da_função (void);

Um exemplo de funções que usam o tipo **void**:

Exemplo 33:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
             Mensagem();
              Console.WriteLine("\tAgora, este texto na função main\n");
              Mensagem();
            Console.ReadKey();
        }

        static void Mensagem()
        {
              Console.WriteLine("Este texto, está na função Mensagem.\n");
        }
    }
}
```

Se quisermos que a função retorne algo, devemos usar a declaração return. Se não quisermos, basta declarar a função como tendo tipo-de-retorno **void**. Devemos lembrar agora que a função **main()** é uma função e como tal devemos tratá-la. O compilador acha que a função **main()** deve retornar um inteiro. Isto pode ser interessante se quisermos que o sistema operacional receba um valor de retorno da função **main()**. Se o programa retornar zero, significa que ele terminou normalmente, e, se o programa retornar um valor diferente de zero, significa que o programa teve um término anormal.

Exercícios

1) Escreva um programa que leia dois números inteiros e crie uma função que deverá exibir qual será o maior número entre eles, verificando inclusive se são iguais.

2) Escreva um programa que leia um número e crie uma função que retornará o resultado do fatorial deste número informado pelo usuário.

3) Escreva um programa que leia duas notas e calcule a média informando inclusive se está aprovado ou reprovado. No programa deverá ter uma função chamada lerNota e outra função chamada calcularMedia que deverá retornar o cálculo para a função principal.

4) Escreva um programa que exiba o seguinte menu e crie as funções para realizar os cálculos:

   1 – Soma
   2 – Subtração
   3 – Divisão
   4 – Multiplicação
   5 – Resto da Divisão
   6 – Maior entre 2 números
   7 – Maior entre 3 números
   8 – Sair