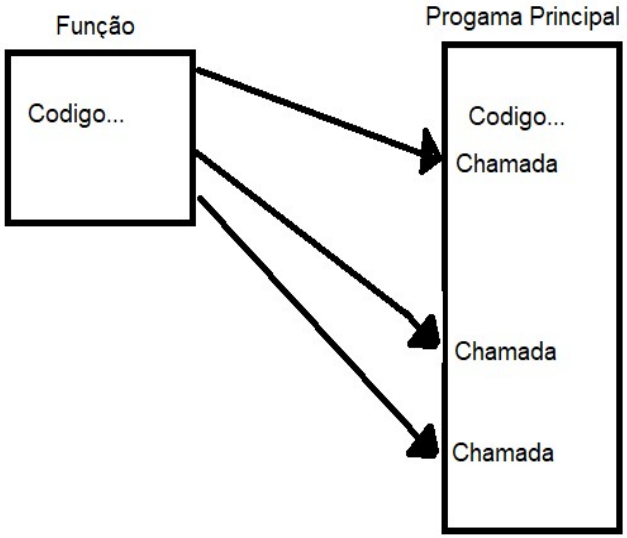
Funções

Funções

As funções são blocos de código separados do programa principal e que podem ser chamados quando necessário.



Função

Geralmente todas as linguagens de programação possibilita esse encapsulamento e isso é feito na forma de uma função ou de um procedimento (um procedimentos é análogo à uma função, entretanto nele não existe qualquer devolução).

A ideia básica de uma função, implementada em alguma linguagem de programação, é encapsular um código que poderá ser invocado/chamado por qualquer outro trecho do programa. Seu significado e uso são muito parecidos com o de funções matemáticas, ou seja, existe um nome, uma definição e posterior invocação à função.

Por que usar o conceito de função?

Assim, implementar códigos com objetivos específicos (como computar o cosseno de qualquer valor) apresentam três grandes vantagens:

- Facilita o desenvolvimento (desenvolvimento modular): implementa-se uma particular unidade, um trecho menor, concentrando-se nele, até que ele esteja funcionando com alto grau de confiabilidade;

- Organização: o código fica melhor organizado e portanto mais fácil de ser mantido;

- Reaproveitamento: sempre que precisar aplicar o código encapsulado em qualquer outro trecho de código (ou noutro código), pode-se utilizar aquele que já foi implementado e é confiável.

Utilizamos funções para obter:

- Clareza do código: separando pedaços de código da função main(), podemos entender mais facilmente o que cada parte do código faz. Além disso, para procurarmos por uma certa acão feita pelo programa, basta buscar a função correspondente. Isso torna muito mais fácil o ato de procurar por erros.
- Reutilização: muitas vezes queremos executar uma certa tarefa várias vezes ao longo do programa. Repetir todo o código para essa operação é muito trabalhoso, e torna mais difícil a manutenção do código: se acharmos um erro nesse código, teremos que corrigi-lo em todas as repetições do código. Chamar uma função diversas vezes contorna esses dois problemas.
- Independência: uma função é relativamente independente do código que a chamou. Uma função pode modificar variáveis globais ou ponteiros, mas limitando-se aos dados fornecidos pela chamada de função.

Definição (Portugol)

Do ponto de vista prático, a estrutura básica de uma função em uma linguagem de programação está representada abaixo, com a declaração da função e sua lista de parâmetros formais, seguido de sua invocação (quando providenciamos os parâmetro efetivos).

Declaração:
```
[eventual_tipo_de_retorno] nome_da_funcao (lista_parametros_formais)
    comando1
    ...
    comandoN
    devolva EXP //# devolve o valor em EXP para quem chamou a funcao
```

Uso:
```
var = nome_da_funcao(lista_parametros_efetivos)
```

Definindo uma função

Uma função pode necessitar de alguns dados para que possa realizar alguma ação baseada neles. Esses dados são chamados parâmetros da função. Além disso, a função pode retornar um certo valor, que é chamado valor de retorno. Os parâmetros (e seus tipos) devem ser especificados explicitamente, assim como o tipo do valor de retorno.

```
[tipo de retorno da função] nome_da_função (1º parâmetro, 2º parâmetro, …)
{
   //código
}
```

Função

```
Private Static Tipo_Retorno Nome_função(Parametros){
//códigos.
}
```

Tipo de Retorno.

No tipo de retorno indica que, caso sua função possua algum retorno, ela vai mandar alguma variável para o programa principal, deve-se colocar aqui o tipo que esta variável terá no programa. Caso não deseje que possua retorno para o programa principal preencher com o comando “Void”.

Nome da Função.

Aqui é o nome pelo qual a função será chamada, este nome pode ser qualquer coisa que não possua caracteres especiais e não seja uma palavra reservada no sistema.

Parâmetros.

Os parâmetros a entrada da função, são uma ou mais variáveis que desejamos que nosso programa principal mandasse para dentro da função, visto que a função é um mundo separado do programa principal, é através dos parâmetros que as funções possuem acesso as variáveis do programa principal. Caso não deseje utilizar parâmetros apenas abra e feche os parênteses.

Tipos de funções

O tipo de retorno, além dos tipos normais de variáveis (char, int, float, double e suas variações), pode ser o tipo especial void, que na verdade significa que não há valor de retorno.

Exemplos ➔

Exemplo função void

```
static void mold()
{
    Console.ForegroundColor = ConsoleColor.Red;
    Console.BackgroundColor = ConsoleColor.DarkBlue;
    int c=2;
    Console.SetCursorPosition(5, c);
    Console.WriteLine("                                    ");
    for (c=3; c <= 8; c++) {
        Console.SetCursorPosition(5, c);
        Console.WriteLine("                                    ");
    }
    Console.SetCursorPosition(5, c);
    Console.WriteLine("                                    ");
}
0 referências
static void Main(string[] args)
{//inicio
    mold();//chamando a função
    Console.ReadKey();
}//fim
```

Exemplo função int

```
static void mold()
{
    Console.ForegroundColor = ConsoleColor.Red;
    Console.BackgroundColor = ConsoleColor.DarkBlue;
    int c=2;
    Console.SetCursorPosition(5, c);
    Console.WriteLine("                                                  ");
    for (c=3; c <= 8; c++) {
        Console.SetCursorPosition(5, c);
        Console.WriteLine("                                                  ");
    }
    Console.SetCursorPosition(5, c);
    Console.WriteLine("                                                  ");
}
1 referência
static int subtracao(int n1, int n2) {
    int r = n1 - n2;
    Console.SetCursorPosition(7, 7);
    Console.WriteLine("Resultado: " + r);
    return 0;
}
```

```
static void Main(string[] args)
{//inicio
    Console.Title = "Função";
    mold();//chamando a função
    Console.ForegroundColor = ConsoleColor.Yellow;
    Console.SetCursorPosition(7, 4);
    Console.Write("Entre com o primeiro numero: ");
    int a = Convert.ToInt32(Console.ReadLine());
    Console.SetCursorPosition(7, 5);
    Console.Write("Entre com o segundo numero: ");
    int b = int.Parse(Console.ReadLine());
    subtracao(a, b);
    Console.ReadKey();
}//fim
```

Exemplo função com for(repetição)

```
static void mold()
{
    Console.ForegroundColor = ConsoleColor.Red;
    Console.BackgroundColor = ConsoleColor.DarkBlue;
    int c=2;
    Console.SetCursorPosition(5, c);
    Console.WriteLine("                                                  ");
    for (c=3; c <= 15; c++) {
        Console.SetCursorPosition(5, c);
        Console.WriteLine("                                                  ");
    }
    Console.SetCursorPosition(5, c);
    Console.WriteLine("                                                  ");
}
1 referência
static int tab(int n1, int n2) {
    for (int c = 0; c <= n2; c++) {
        int r = n1 * c;
        Console.SetCursorPosition(7, c+6);
        Console.WriteLine(n1 + " x " + c + " = " + r);
    }
    return 0;
}
```

```
static void Main(string[] args)
{//inicio
    Console.Title = "Função";
    mold();//chamando a função
    Console.ForegroundColor = ConsoleColor.Yellow;
    Console.SetCursorPosition(7, 4);
    Console.Write("Entre com o valor da tabuada  : ");
    int a = Convert.ToInt32(Console.ReadLine());
    Console.SetCursorPosition(7, 5);
    Console.Write("Tabuada até qual numero:  ");
    int b = int.Parse(Console.ReadLine());
    tab(a, b);
    Console.ReadKey();
}//fim
```

Exemplo função int

```
static void mold()
{
    Console.ForegroundColor = ConsoleColor.Red;
    Console.BackgroundColor = ConsoleColor.DarkBlue;
    int c=2;
    Console.SetCursorPosition(5, c);
    Console.WriteLine("                                                  ");
    for (c=3; c <= 8; c++) {
        Console.SetCursorPosition(5, c);
        Console.WriteLine("                                                  ");
    }
    Console.SetCursorPosition(5, c);
    Console.WriteLine("                                                  ");
}
1 referência
static int soma(int n1, int n2) {
    int r = n1 + n2;
    return r;
}
```

```
static void Main(string[] args)
{//inicio
    Console.Title = "Função";
    mold();//chamando a função
    Console.ForegroundColor = ConsoleColor.Yellow;
    Console.SetCursorPosition(7, 4);
    Console.Write("Entre com o primeiro numero: ");
    int a = Convert.ToInt32(Console.ReadLine());
    Console.SetCursorPosition(7, 5);
    Console.Write("Entre com o segundo numero: ");
    int b = int.Parse(Console.ReadLine());
    int resultado = soma(a, b);
    Console.SetCursorPosition(7, 7);
    Console.Write("Resultado: " + resultado);
    Console.ReadKey();
}//fim
```

Exemplo função double

```
static void mold()
{
    Console.ForegroundColor = ConsoleColor.Red;
    Console.BackgroundColor = ConsoleColor.DarkBlue;
    int c=2;
    Console.SetCursorPosition(5, c);
    Console.WriteLine("                                                  ");
    for (c=3; c <= 15; c++) {
        Console.SetCursorPosition(5, c);
        Console.WriteLine("                                                  ");
    }
    Console.SetCursorPosition(5, c);
    Console.WriteLine("                                                  ");
}
1 referência
static double tempM(double n1, double n2) {
    double r = (n1 + n2) / 2;
    return r;
}
```

```
static void Main(string[] args)
{//inicio
    Console.Title = "Função";
    mold();//chamando a função
    Console.ForegroundColor = ConsoleColor.Yellow;
    Console.SetCursorPosition(7, 4);
    Console.Write("Entre com a temperatura 1: ");
    double a = Convert.ToInt32(Console.ReadLine());
    Console.SetCursorPosition(7, 5);
    Console.Write("Entre com a temperatura 2: ");
    double b = double.Parse(Console.ReadLine());
    Console.WriteLine("Temperatura Média: "+tempM(a, b));
    Console.ReadKey();
}//fim
```

Exercícios

1 - Crie um algoritmo no qual o usuário poderá escolher qual operação matemática ele poderá executar, adição, subtração, multiplicação e divisão. Como regra você deverá implementar funções que retornam os resultados das operações matemáticas.

2 - Crie um programa no qual o usuário deverá informar três valores inteiros (segmentos de retas). O programa deverá informar se esses valores representam os lados de um triângulo.

Dica: Dados três segmentos de reta distintos, se a soma das medidas de dois deles é sempre maior que a medida do terceiro, então, eles podem formar um triângulo.