



Fundamentos de Programação

Sub-rotinas

Estruturação de programas

- O desenvolvimento de um programa é um processo dinâmico. Frequentemente, é necessário rever códigos para corrigir erros, fazer atualizações ou utilizar como base para resolução de outro problema.
- Por isso, é fundamental que um programa seja escrito de forma organizada, facilitando sua manutenção, reuso e/ou adaptação.

Sub-rotinas

- Uma sub-rotina é um trecho de código computacional que realiza uma tarefa bem definida. Exemplos:
 - *Calcular área de um círculo.*
 - *Converter graus Celsius para Fahrenheit.*
- Uma única sub-rotina pode ser executada várias vezes em um mesmo programa. Pode também ser executada por diferentes programas.

Sub-rotinas

- Você já utiliza sub-rotinas em seus programas:
 - **WriteLine** é uma sub-rotina da classe Console que realiza a tarefa de imprimir na tela do computador
 - **ReadLine** é uma sub-rotina que lê valores do teclado e retorna o valor lido para que o mesmo possa ser utilizado
 - **Sqrt** é uma sub-rotina da classe Math que calcula e devolve o valor da raiz quadrada de um número
 - **Pow** é uma sub-rotina da classe Math que calcula e retorna o valor da potência, dados base e expoente indicados

Sub-rotinas

- Repare que você não precisa saber como as sub-rotinas **WriteLine**, **ReadLine**, **Sqrt** e **Pow** foram implementadas, precisa saber apenas:
 - o que a sub-rotina faz, isto é, qual a tarefa executada por ela
 - como ela pode ser utilizada (sua sintaxe)

Vantagem do uso de sub-rotinas

- Uma mesma tarefa pode ser implementada uma única vez e ser utilizada **várias vezes** (por um ou mais programas):
 - erros precisam ser corrigidos em um único lugar
 - a reutilização da sub-rotina (chamada) pode ser feita de forma simples
 - o código fica legível, mais fácil de ser entendido e mais compacto
 - o uso de sub-rotinas possibilita a modularização do código de um programa, isto é, o desenvolvimento do código organizado em módulos funcionais

Programando sub-rotinas

- Existem três situações distintas no processo de programação envolvendo sub-rotinas:

1) Declaração: a declaração de uma sub-rotina especifica sua sintaxe.

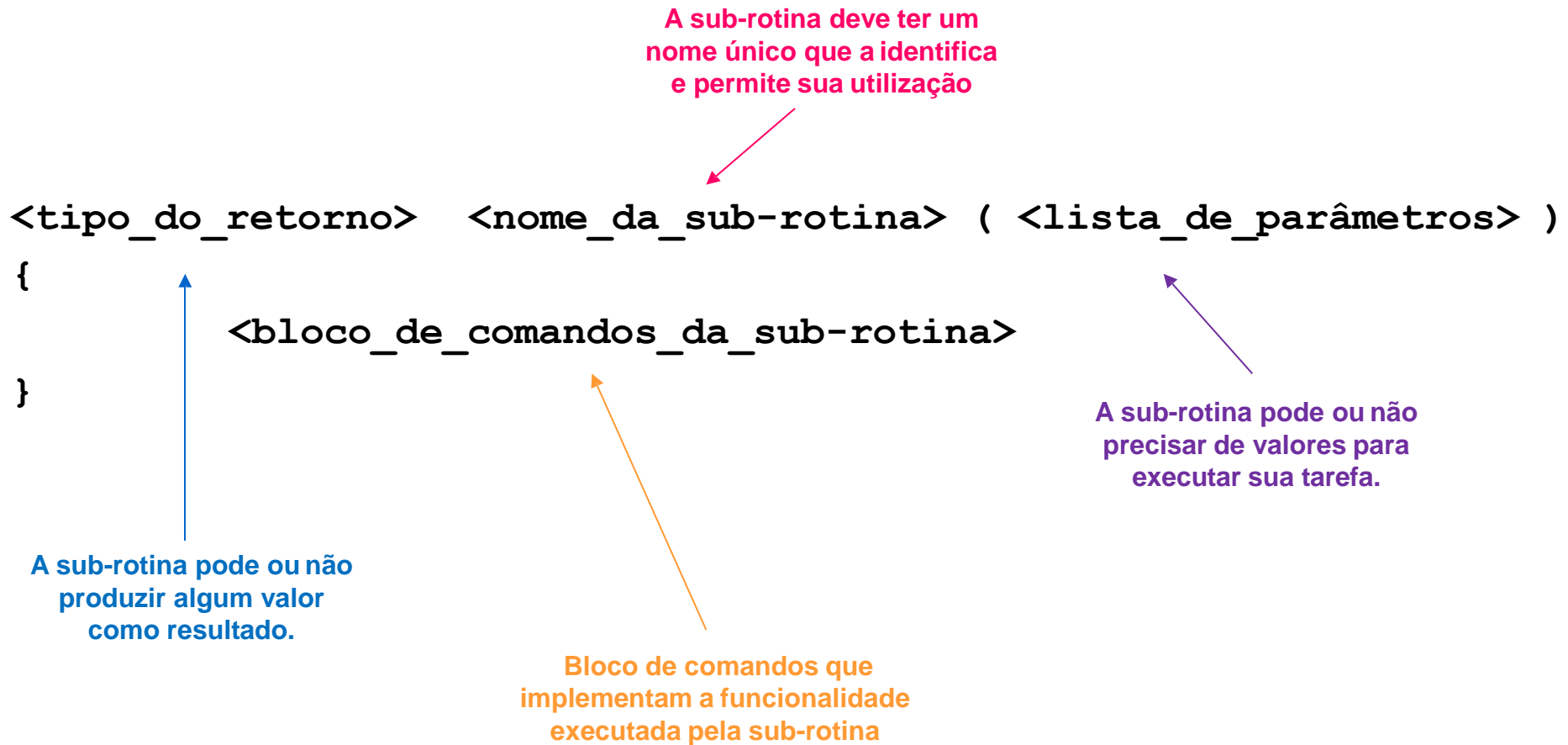
Permite que o usuário que não implementou a sub-rotina e quer utilizá-la conheça seu nome, os valores que necessita e o valor que será retornado.

Programando sub-rotinas

2) Definição: a definição de uma sub-rotina é a implementação da mesma. É uma extensão da declaração, dado que, além da sintaxe, contém o código que realiza a sua tarefa.

3) Chamada: a chamada à uma sub-rotina é a utilização da mesma dentro de alguma outra sub-rotina ou programa. Uma sub-rotina não funciona sozinha, precisa ser chamada cada vez que sua execução for necessária.

Definição de uma sub-rotina



Definição de uma sub-rotina

```
static double celsiusFahrenheit ( double tc )
{
    double tf;
    tf = 1.8 * tc + 32;
    return tf;
}

static void Main(string[] args)
{
    double cels;
    double fahr;
    Console.Write("Entre com temperatura em Celsius: ");
    cels = Convert.ToDouble(Console.ReadLine());
    fahr = celsiusFahrenheit(cels);
    Console.Write("Temperatura em Fahrenheit:");
    Console.Write(fahr);
}
```

Definição de uma sub-rotina

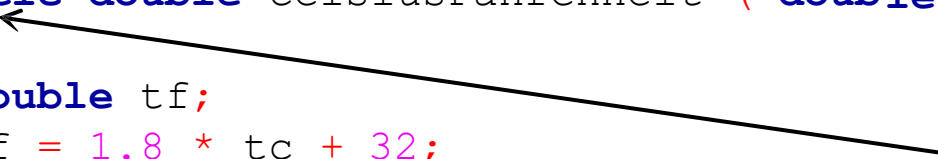
```
static double celsiusFahrenheit ( double tc )  
{  
    double tf;  
    tf = 1.8 * tc + 32;  
    return tf;  
}
```

```
tipoDeRetorno  nomeDaSubrotina ( parametros )  
{  
    bloco_de_comandos_da_sub-rotina  
}
```

```
static void Main(string[] args)  
{  
    double cels;  
    double fahr;  
    Console.Write("Entre com temperatura em Celsius: ");  
    cels = Convert.ToDouble(Console.ReadLine());  
    fahr = celsiusFahrenheit(cels);  
    Console.Write("Temperatura em Fahrenheit:");  
    Console.Write(fahr);  
}
```

Definição de uma sub-rotina

```
static double celsiusFahrenheit ( double  
{  
    double tf;  
    tf = 1.8 * tc + 32;  
    return tf;  
}
```



```
static void Main(string[] args)  
{  
    double cels;  
    double fahr;  
    Console.Write("Entre com temperatura em Celsius: ");  
    cels = Convert.ToDouble(Console.ReadLine());  
    fahr = celsiusFahrenheit(cels);  
    Console.Write("Temperatura em Fahrenheit:");  
    Console.Write(fahr);  
}
```

A sub-rotina pode ou não produzir um valor como resultado.

Quando um valor é produzido, seu tipo deve ser indicado.

Se não há retorno de valor, o tipo de retorno informado deve ser void.

Definição de uma sub-rotina

```
static double celsiusFahrenheit ( double
{
    double tf;
    tf = 1.8 * tc + 32;
    return tf;
}
```

O *nome* da sub-rotina:

- precisa ser único
- deve seguir as mesmas regras de identificadores de variáveis
- deve identificar a ação executada pela sub-rotina

```
static void Main(string[] args)
{
    double cels;
    double fahr;
    Console.Write("Entre com temperatura em Celsius: ");
    cels = Convert.ToDouble(Console.ReadLine());
    fahr = celsiusFahrenheit(cels);
    Console.Write("Temperatura em Fahrenheit:");
    Console.Write(fahr);
}
```

Definição de uma sub-rotina

```
static double celsiusFahrenheit ( double tc )  
{  
    double tf;  
    tf = 1.8 * tc + 32;  
    return tf;  
}
```

```
static void Main(string[] args)  
{  
    double cels;  
    double fahr;  
    Console.Write("Entre com temperatura em Celsius: ");  
    cels = Convert.ToDouble(Console.ReadLine());  
    fahr = celsiusFahrenheit(cels);  
    Console.Write("Temperatura em Fahrenheit:");  
    Console.Write(fahr);  
}
```

A *lista de parâmetros* deve indicar os valores que precisam ser fornecidos quando a sub-rotina for chamada. Para cada valor (parâmetro), devem ser especificados:

- tipo do parâmetro
- nome do parâmetro no bloco da sub-rotina

Definição de uma sub-rotina

```
static double celsiusFahrenheit ( double
{
    double tf;
    tf = 1.8 * tc + 32;
    return tf;
}
```

```
static void Main(string[] args)
{
    double cels;
    double fahr;
    Console.Write("Entre com temperatura em Celsius: ");
    cels = Convert.ToDouble(Console.ReadLine());
    fahr = celsiusFahrenheit(cels);
    Console.Write("Temperatura em Fahrenheit:");
    Console.Write(fahr);
}
```

O *bloco de comandos* deve ter, em seu início, a declaração das variáveis necessárias no código, sem incluir os identificadores da lista de parâmetros.

Só depois devem aparecer os comandos que implementam a tarefa a ser executada pela sub- rotina.

Definição de uma sub-rotina

```
static double celsiusFahrenheit ( double
{
    double tf;
    tf = 1.8 * tc + 32;
    return tf; ←
```

```
static void Main(string[] args)
{
    double cels;
    double fahr;
    Console.Write("Entre com temperatura em Celsius: ");
    cels = Convert.ToDouble(Console.ReadLine());
    fahr = celsiusFahrenheit(cels);
    Console.Write("Temperatura em Fahrenheit:");
    Console.Write(fahr);
}
```

O bloco de comandos deve terminar com um comando de retorno, seguido do valor resultante da tarefa executada pela sub-rotina.

Se a sub-rotina não produzir um valor, o comando de retorno pode ser omitido.

Definição de uma sub-rotina

```
static double celsiusFahrenheit ( double tc )  
{  
    double tf;  
    tf = 1.8 * tc + 32;  
    return tf;  
}
```

Toda sub-rotina precisa ser *definida* ou *declarada* antes de ser chamada.

```
static void Main(string[] args)  
{  
    double cels;  
    double fahr;  
    Console.Write("Entre com temperatura em Celsius: ");  
    cels = Convert.ToDouble(Console.ReadLine());  
    fahr = celsiusFahrenheit(cels);  
    Console.Write("Temperatura em Fahrenheit:");  
    Console.Write(fahr);  
}
```

Definição de uma sub-rotina

```
using System;

static void Main(string[] args)
{
    double cels;
    double fahr;
    Console.Write("Entre com temperatura em Celsius:");
    cels = Convert.ToDouble(Console.ReadLine());
    fahr = celsiusFahrenheit(cels);
    Console.Write("Temperatura em Fahrenheit:");
    Console.Write(fahr);
}

static double celsiusFahrenheit ( double tc )
{
    double tf;
    tf = 1.8 * tc + 32;

    return tf;
}
```

As sub-rotinas `Console.Write`, `Console.WriteLine` e `Console.ReadLine` podem ser utilizadas porque a biblioteca que contém sua declaração foi referenciada no início do arquivo, antes do programa.

Definição de uma sub-rotina

```
static void Main(string[] args)
{
    double cels;
    double fahr;
    Console.Write("Entre com temperatura em Celsius:");
    cels = Convert.ToDouble(Console.ReadLine());
    fahr = celsiusFahrenheit(cels);
    Console.Write("Temperatura em Fahrenheit:");
    Console.Write(fahr);
}
```

```
static double celsiusFahrenheit ( double tc )
{
    double tf;
    tf = 1.8 * tc + 32;
    return tf;
}
```

A chamada de uma sub-rotina deve incluir somente o seu nome e o valor de cada parâmetro.

O tipo de retorno e o tipo dos parâmetros não aparecem na chamada.

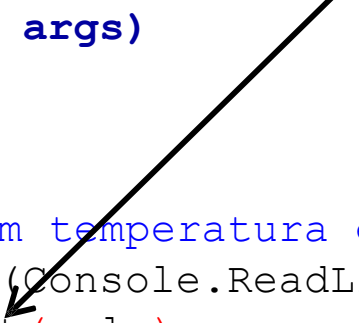
Definição de uma sub-rotina

```
nome_da_sub-rotina ( lista_com_um_valor_para_cada_parametro );
```

```
static void Main(string[] args)
{
    double cels;
    double fahr;
    Console.Write("Entre com temperatura em Celsius: ");
    cels = Convert.ToDouble(Console.ReadLine());
    fahr = celsiusFahrenheit(cels);
    Console.Write("Temperatura em Fahrenheit:");
    Console.Write(fahr);

}

static double celsiusFahrenheit ( double tc )
{
    double tf;
    tf = 1.8 * tc + 32;
    return tf;
}
```



Definição de uma sub-rotina

```
static void Main(string[] args)
{
    double cels;
    double fahr;
    Console.Write("Entre com temperatura em Celsius: ");
    cels = Convert.ToDouble(Console.ReadLine());
    fahr = celsiusFahrenheit(cels);
    Console.Write("Temperatura em Fahrenheit:");
    Console.Write(fahr);
}

static double celsiusFahrenheit ( double tc)
{
    double tf;
    tf = 1.8 * tc + 32;
    return tf;
}
```

No caso da sub-rotina `celsiusFahrenheit`, o único parâmetro, `tc`, tem tipo `double`. Então, um valor do tipo `double` precisa ser passado como parâmetro.

Na chamada, o valor armazenado pela variável `cels` é passado como parâmetro para a sub-rotina.

Execução de uma sub-rotina

```

1
2
3
4
5 static void Main(string[] args) {
6     double cels;
7 double fahr;
8
9     Console.Write("Temperatura em Celsius: ");
10    cels = Convert.ToDouble(Console.ReadLine());
11
12    fahr = celsiusFahrenheit(cels);
13
14    Console.Write("Temperatura em Fahrenheit:");
15    Console.Write(fahr);
16
17
18 }
19
20 static double celsiusFahrenheit(double tc) {
21     double tf;
22     tf = 1.8 * tc + 32;
23     return tf;
24 }

```

TESTE DE MESA

[illegible]

Execução de uma sub-rotina

```

1
2
3
4
5 static void Main(string[] args) {
6     double cels;
7 double fahr;    8
9     Console.Write("Temperatura em Celsius: ");
10 cels = Convert.ToDouble(Console.ReadLine());
11
12     fahr = celsiusFahrenheit(cels);
13
14     Console.Write("Temperatura em Fahrenheit:");
15 Console.Write(fahr);
16
17
18 }
19
20 static double celsiusFahrenheit(double tc) {
21     double tf;
22     tf = 1.8 * tc + 32;
23     return tf;
24 }

```

TESTE DE MESA

[illegible]

Execução de uma sub-rotina

```

1
2
3
4
5 static void Main(string[] args) {
6     double cels;
7 double fahr;    8
9     Console.Write("Temperatura Celsius: ");
10    cels = Convert.ToDouble(Console.ReadLine());
11
12    fahr = celsiusFahrenheit(cels);
13
14    Console.Write("Temperatura em Fahrenheit:");
15 Console.Write(fahr);
16
17
18 }
19
20 static double celsiusFahrenheit(double tc) {
21     double tf;
22     tf = 1.8 * tc + 32;
23     return tf;
24 }

```

TESTE DE MESA

[illegible]

Entre com temperatura em Celsius:

Execução de uma sub-rotina

TESTE DE MESA

```
1
2
3
4
5 static void Main(string[] args) {
6     double cels;
7     double fahr;
8
9     Console.Write("Temperatura em Celsius: ");
10    cels = Convert.ToDouble(Console.ReadLine());
11
12    fahr = celsiusFahrenheit(cels);
13
14    Console.Write("Temperatura em Fahrenheit:");
15    Console.Write(fahr);
16
17
18 }
19
20 static double celsiusFahrenheit(double tc) {
21     double tf;
22     tf = 1.8 * tc + 32;
23     return tf;
24 }
```

linha	cels	fahr	tc	tf
5	?	?		
10	20.0	?		

Entre com temperatura em Celsius: 20

Execução de uma sub-rotina

```

1
2
3
4
5 static void Main(string[] args) {
6     double cels;
7 double fahr; 8
9     Console.Write("Temperatura em Celsius: ");
10 cels = Convert.ToDouble(Console.ReadLine());
11
12     fahr = celsiusFahrenheit(cels);
13
14     Console.Write("Temperatura em Fahrenheit:");
15 Console.Write(fahr);
16
17
18 }
19
20 static double celsiusFahrenheit(double tc) {
21     double tf;
22     tf = 1.8 * tc + 32;
23     return tf;
24 }

```

TESTE DE MESA

[illegible]

Entre com temperatura em Celsius: 20

Execução de uma sub-rotina

```
1
2
3
4
5 static void Main(string[] args) {
6     double cels;
7 double fahr; 8
9     Console.Write("Temperatura em Celsius: ");
10 cels = Convert.ToDouble(Console.ReadLine());
11
12     fahr = celsiusFahrenheit(cels);
13
14     Console.Write("Temperatura em Fahrenheit:");
15 Console.Write(fahr);
16
17
18 }
19
20 static double celsiusFahrenheit(double tc) {
21     double tf;
22     tf = 1.8 * tc + 32;
23     return tf;
24 }
```

TESTE DE MESA

linha	cels	fahr	tc	tf
5	?	?		
10	20.0	?		
12	20.0	?		
20			20.0	?

Entre com temperatura em Celsius: 20

Execução de uma sub-rotina

```
1
2
3
4
5 static void Main(string[] args) {
6     double cels;
7 double fahr; 8
9     Console.Write("Temperatura em Celsius: ");
10 cels = Convert.ToDouble(Console.ReadLine());
11
12     fahr = celsiusFahrenheit(cels);
13
14     Console.Write("Temperatura em Fahrenheit:");
15 Console.Write(fahr);
16
17
18 }
19
20 static double celsiusFahrenheit(double tc) {
21     double tf;
22     tf = 1.8 * tc + 32;
23     return tf;
24 }
```

TESTE DE MESA

linha	cels	fahr	tc	tf
5	?	?		
10	20.0	?		
12	20.0	?		
20			20.0	?
22			20.0	68.0

Entre com temperatura em Celsius: 20

Execução de uma sub-rotina

```
1
2
3
4
5 static void Main(string[] args) {
6     double cels;
7 double fahr; 8
9     Console.Write("Temperatura em Celsius: ");
10 cels = Convert.ToDouble(Console.ReadLine());
11
12     fahr = celsiusFahrenheit(cels);
13
14     Console.Write("Temperatura em Fahrenheit:");
15 Console.Write(fahr);
16
17
18 }
19
20 static double celsiusFahrenheit(double tc) {
21     double tf;
22     tf = 1.8 * tc + 32;
23     return tf;
24 }
```

TESTE DE MESA

linha	cels	fahr	tc	tf
5	?	?		
10	20.0	?		
12	20.0	?		
20			20.0	?
22			20.0	68.0

Entre com temperatura em Celsius: 20

Execução de uma sub-rotina

```
1
2
3
4
5 static void Main(string[] args) {
6     double cels;
7     double fahr; 8
9     Console.Write("Temperatura em Celsius: ");
10    cels = Convert.ToDouble(Console.ReadLine());
11
12    fahr = celsiusFahrenheit(cels);
13
14    Console.Write("Temperatura em Fahrenheit:");
15    Console.Write(fahr);
16
17
18 }
19
20 static double celsiusFahrenheit(double tc) {
21     double tf;
22     tf = 1.8 * tc + 32;
23     return tf;
24 }
```

TESTE DE MESA

linha	cels	fahr	tc	tf
5	?	?		
10	20.0	?		
12	20.0	?		
20			20.0	?
22			20.0	68.0
12	20.0	?		

Entre com temperatura em Celsius: 20

Execução de uma sub-rotina

```
1
2
3
4
5 static void Main(string[] args) {
6     double cels;
7 double fahr; 8
9     Console.Write("Temperatura em Celsius: ");
10 cels = Convert.ToDouble(Console.ReadLine());
11
12 fahr = 68;
13
14 Console.Write("Temperatura em Fahrenheit:");
15 Console.Write(fahr);
16
17
18 }
19
20 static double celsiusFahrenheit(double tc) {
21     double tf;
22     tf = 1.8 * tc + 32;
23     return tf;
24 }
```

TESTE DE MESA

linha	cels	fahr	tc	tf
5	?	?		
10	20.0	?		
12	20.0	?		
20			20.0	?
22			20.0	68.0
12	20.0	?		

Entre com temperatura em Celsius: 20

Execução de uma sub-rotina

```
1
2
3
4
5 static void Main(string[] args) {
6     double cels;
7 double fahr; 8
9     Console.Write("Temperatura em Celsius: ");
10 cels = Convert.ToDouble(Console.ReadLine());
11
12 fahr = celsiusFahrenheit(cels);
13
14 Console.Write("Temperatura em Fahrenheit:");
15 Console.Write(fahr);
16
17
18 }
19
20 static double celsiusFahrenheit(double tc) {
21     double tf;
22     tf = 1.8 * tc + 32;
23     return tf;
24 }
```

TESTE DE MESA

linha	cels	fahr	tc	tf
5	?	?		
10	20.0	?		
12	20.0	?		
20			20.0	?
22			20.0	68.0
12	20.0	68.0		

Entre com temperatura em Celsius: 20

Execução de uma sub-rotina

```
1
2
3
4
5 static void Main(string[] args) {
6     double cels;
7 double fahr; 8
9     Console.Write("Temperatura em Celsius: ");
10 cels = Convert.ToDouble(Console.ReadLine());
11
12     fahr = celsiusFahrenheit(cels);
13
14 Console.Write("Temperatura em Fahrenheit:");
15 Console.Write(fahr);
16
17
18 }
19
20 static double celsiusFahrenheit(double tc) {
21     double tf;
22     tf = 1.8 * tc + 32;
23     return tf;
24 }
```

TESTE DE MESA

linha	cels	fahr	tc	tf
5	?	?		
10	20.0	?		
12	20.0	?		
20			20.0	?
22			20.0	68.0
12	20.0	68.0		

```
Entre com temperatura em Celsius: 20
Temperatura em Fahrenheit:
```

Execução de uma sub-rotina

```
1
2
3
4
5 static void Main(string[] args) {
6     double cels;
7 double fahr; 8
9     Console.Write("Temperatura em Celsius: ");
10 cels = Convert.ToDouble(Console.ReadLine());
11
12     fahr = celsiusFahrenheit(cels);
13
14     Console.Write("Temperatura em Fahrenheit:");
15     Console.Write(fahr);
16
17
18 }
19
20 static double celsiusFahrenheit(double tc) {
21     double tf;
22     tf = 1.8 * tc + 32;
23     return tf;
24 }
```

TESTE DE MESA

linha	cels	fahr	tc	tf
5	?	?		
10	20.0	?		
12	20.0	?		
20			20.0	?
22			20.0	68.0
12	20.0	68.0		

```
Entre com temperatura em Celsius: 20
Temperatura em Fahrenheit:
68.000000
```

Execução de uma sub-rotina

```
1
2
3
4
5 static void Main(string[] args) {
6     double cels;
7 double fahr; 8
9     Console.Write("Temperatura em Celsius: ");
10 cels = Convert.ToDouble(Console.ReadLine());
11
12     fahr = celsiusFahrenheit(cels);
13
14     Console.Write("Temperatura em Fahrenheit:");
15 Console.Write(fahr);
16
17
18 }
19
20 static double celsiusFahrenheit(double tc) {
21     double tf;
22     tf = 1.8 * tc + 32;
23     return tf;
24 }
```

TESTE DE MESA

linha	cels	fahr	tc	tf
5	?	?		
10	20.0	?		
12	20.0	?		
20			20.0	?
22			20.0	68.0
12	20.0	68.0		

```
Entre com temperatura em Celsius: 20
Temperatura em Fahrenheit:
68.000000
```

Execução de uma sub-rotina

```
1
2
3
4
5 static void Main(string[] args) {
6     double cels;
7 double fahr;
8
9     Console.Write("Temperatura em Celsius: ");
10 cels = Convert.ToDouble(Console.ReadLine());
11
12     fahr = celsiusFahrenheit(cels);
13
14     Console.Write("Temperatura em Fahrenheit:");
15 Console.Write(fahr);
16
17
18 }
19
20 static double celsiusFahrenheit(double tc) {
21
22     double tf;
23     tf = 1.8 * tc + 32;
24     return tf;
25 }
```

TESTE DE MESA

linha	cels	fahr	tc	tf
5	?	?		
10	20.0	?		
12	20.0	?		
20			20.0	?
22			20.0	68.0
12	20.0	68.0		

```
Entre com temperatura em Celsius: 20
Temperatura em Fahrenheit:
68.000000
```

Tipos de sub-rotina

- ***Procedimento:*** sub-rotina que não retorna valor para o programa ou sub-rotina que a chamou.
- Um procedimento normalmente executa uma ação que não gera dados.

Exemplo de procedimento

```
1 static void imprimeFormatoHora(int qtdMinutos)
2 {
3     int hora, min;
4     hora = qtdMinutos / 60;
5     min  = qtdMinutos % 60;
6 Console.Write("{0}:{1}", hora, min); 7
7 }
8
9 static void Main(string[] args)
10 {
11     int minutos;
12     Console.Write("Minutos para converter em hora:");
13     minutos = Convert.ToInt32(Console.ReadLine());
14     Console.Write("{0} minutos equivale a ", minutos);
15     imprimeFormatoHora(minutos);
16 Console.ReadKey();
17 }
```

Exemplo de procedimento

```
1 static void imprimeFormatoHora(int qtdMinutos)
2 {
3     int hora, min;
4     hora = qtdMinutos / 60;
5     min = qtdMinutos % 60;
6     Console.WriteLine("{0}:{1}", hora, min);
7 }
8
9 static void Main(string[] args)
10 {
11     int minutos;
12     Console.WriteLine("Minutos para converter em hora:");
13     minutos = Convert.ToInt32(Console.ReadLine());
14     Console.WriteLine("{0} minutos equivale a ", minutos);
15     imprimeFormatoHora(minutos);
16     Console.ReadKey();
17 }
```

Tipo de retorno

Como não há retorno de valor algum, o tipo indicado é **void**.

Exemplo de procedimento

```
1 static void imprimeFormatoHora(int qtdMinutos)
2 {
3     int hora, min;
4     hora = qtdMinutos / 60;
5     min = qtdMinutos % 60;
6     Console.WriteLine("{0}:{1}", hora, min);
7 return; 8
9 }
10 static void Main(string[] args)
11 {
12     int minutos;
13     Console.WriteLine("Minutos para converter em hora:");
14     minutos = Convert.ToInt32(Console.ReadLine());
15     Console.WriteLine("{0} minutos equivale a ", minutos);
16     imprimeFormatoHora(minutos);
17     Console.ReadKey();
18 }
```

Comando de retorno

Como não há retorno de valor algum, o comando **return** aparece **sozinho** (apenas seguido do ponto e vírgula) **ou é omitido**.

Exemplo de procedimento

```
1 static void imprimeFormatoHora(int qtdMinutos)
2 {
3     int hora, min;
4     hora = qtdMinutos / 60;
5     min = qtdMinutos % 60;
6     Console.WriteLine("{0}:{1}", hora, min);
7     return;
8 Console.WriteLine("Esta linha nunca sera executada! "); 9
9 }
10
11 static void Main(string[] args)
12 {
13     int minutos;
14     Console.WriteLine("Minutos para converter em hora:");
15     minutos = Convert.ToInt32(Console.ReadLine());
16     Console.WriteLine("{0} minutos equivale a ", minutos);
17     imprimeFormatoHora(minutos);
18     Console.ReadKey();
19 }
```

Comando de retorno

Cuidado ao usar o comando **return** no meio de uma sub-rotina, pois, mesmo que haja comandos depois, a sub-rotina é encerrada assim que o comando **return** for executado.

Exemplo de procedimento

```
1 static void imprimeFormatoHora(int qtdMinutos)
2 {
3     int hora, min;
4     hora = qtdMinutos / 60;
5     min = qtdMinutos % 60;
6 Console.WriteLine("{0}:{1}", hora, min); 7
7 }
8
9 static void Main(string[] args)
10 {
11     int minutos;
12     Console.WriteLine("Minutos para converter em hora:");
13     minutos = Convert.ToInt32(Console.ReadLine());
14     Console.WriteLine("{0} minutos equivale a ", minutos);
15     imprimeFormatoHora(minutos);
16 Console.ReadKey();
17 }
```

Chamada

A chamada a um procedimento normalmente aparece sozinha em uma linha de comando.

Execução de um procedimento

```

1 static void imprimeFormatoHora(int qtdMinutos)
2 {
3     int hora, min;
4     hora = qtdMinutos / 60;
5     min = qtdMinutos % 60;
6 Console.WriteLine("{0}:{1}", hora, min);
7 }
8
9 static void Main(string[] args)
10 {
11     int minutos;
12     Console.WriteLine("Minutos para converter em hora:");
13     minutos = Convert.ToInt32(Console.ReadLine());
14     Console.WriteLine("{0} minutos equivale a ", minutos);
15     imprimeFormatoHora(minutos);
16 Console.ReadKey();
17 }

```

TESTE DE MESA

[illegible]

Execução de um procedimento

TESTE DE MESA

```
1 static void imprimeFormatoHora(int qtdMinutos)
2 {
3     int hora, min;
4     hora = qtdMinutos / 60;
5     min = qtdMinutos % 60;
6 Console.WriteLine("{0}:{1}", hora, min);
7 }
8
9 static void Main(string[] args)
10 {
11     int minutos;
12     Console.WriteLine("Minutos para converter em hora:");
13     minutos = Convert.ToInt32(Console.ReadLine());
14     Console.WriteLine("{0} minutos equivale a ", minutos);
15     imprimeFormatoHora(minutos);
16 Console.ReadKey();
17 }
```

[illegible]

Execução de um procedimento

```
1 static void imprimeFormatoHora(int qtdMinutos)
2 {
3     int hora, min;
4     hora = qtdMinutos / 60;
5     min = qtdMinutos % 60;
6 Console.WriteLine("{0}:{1}", hora, min);
7 }
8
9 static void Main(string[] args)
10 {
11     int minutos;
12     Console.WriteLine("Minutos para converter em hora:");
13     minutos = Convert.ToInt32(Console.ReadLine());
14     Console.WriteLine("{0} minutos equivale a ", minutos);
15     imprimeFormatoHora(minutos);
16 Console.ReadKey();
17 }
```

Entre com os minutos:

TESTE DE MESA

[illegible]

Execução de um procedimento

```

1  static void imprimeFormatoHora(int qtdMinutos)
2  {
3      int hora, min;
4      hora = qtdMinutos / 60;
5      min = qtdMinutos % 60;
6      Console.WriteLine("{0}:{1}", hora, min);
7  }
8
9  static void Main(string[] args)
10 {
11     int minutos;
12     Console.WriteLine("Minutos para converter em hora:");
13     minutos = Convert.ToInt32(Console.ReadLine());
14     Console.WriteLine("{0} minutos equivale a ", minutos);
15     imprimeFormatoHora(minutos);
16     Console.ReadKey();
17 }

```

Entre com os minutos: 345

TESTE DE MESA

[illegible]

Execução de um procedimento

```

1  static void imprimeFormatoHora(int qtdMinutos)
2  {
3      int hora, min;
4      hora = qtdMinutos / 60;
5      min = qtdMinutos % 60;
6      Console.WriteLine("{0}:{1}", hora, min);
7  }
8
9  static void Main(string[] args)
10 {
11     int minutos;
12     Console.WriteLine("Minutos para converter em hora:");
13     minutos = Convert.ToInt32(Console.ReadLine());
14     Console.WriteLine("{0} minutos equivale a ", minutos);
15     imprimeFormatoHora(minutos);
16     Console.ReadKey();
17 }

```

Entre com os minutos: 345
345 minutos equivale a

TESTE DE MESA

[illegible]

Execução de um procedimento

```
1 static void imprimeFormatoHora(int qtdMinutos)
2 {
3     int hora, min;
4     hora = qtdMinutos / 60;
5     min = qtdMinutos % 60;
6 Console.WriteLine("{0}:{1}", hora, min);
7 }
8
9 static void Main(string[] args)
10 {
11     int minutos;
12     Console.WriteLine("Minutos para converter em hora:");
13     minutos = Convert.ToInt32(Console.ReadLine());
14     Console.WriteLine("{0} minutos equivale a ", minutos);
15     imprimeFormatoHora(minutos);
16 Console.ReadKey();
17 }
```

Entre com os minutos: 345
345 minutos equivale a

TESTE DE MESA

[illegible]

Execução de um procedimento

```

1  static void imprimeFormatoHora(int qtdMinutos)
2  {
3      int hora, min;
4      hora = qtdMinutos / 60;
5      min = qtdMinutos % 60;
6      Console.WriteLine("{0}:{1}", hora, min);
7  }
8
9  static void Main(string[] args)
10 {
11     int minutos;
12     Console.WriteLine("Minutos para converter em hora:");
13     minutos = Convert.ToInt32(Console.ReadLine());
14     Console.WriteLine("{0} minutos equivale a ", minutos);
15     imprimeFormatoHora(minutos);
16     Console.ReadKey();
17 }

```

Entre com os minutos: 345
345 minutos equivale a

TESTE DE MESA

[illegible]

Execução de um procedimento

```
1 static void imprimeFormatoHora(int qtdMinutos)
2 {
3     int hora, min;
4     hora = qtdMinutos / 60;
5     min = qtdMinutos % 60;
6 Console.WriteLine("{0}:{1}", hora, min);
7 }
8
9 static void Main(string[] args)
10 {
11     int minutos;
12     Console.WriteLine("Minutos para converter em hora:");
13     minutos = Convert.ToInt32(Console.ReadLine());
14     Console.WriteLine("{0} minutos equivale a ", minutos);
15     imprimeFormatoHora(minutos);
16 Console.ReadKey();
17 }
```

Entre com os minutos: 345
345 minutos equivale a

TESTE DE MESA

[illegible]

Execução de um procedimento

```
1 static void imprimeFormatoHora(int qtdMinutos)
2 {
3     int hora, min;
4     hora = qtdMinutos / 60;
5     min = qtdMinutos % 60;
6 Console.WriteLine("{0}:{1}", hora, min);
7 }
8
9 static void Main(string[] args)
10 {
11     int minutos;
12     Console.WriteLine("Minutos para converter em hora:");
13     minutos = Convert.ToInt32(Console.ReadLine());
14     Console.WriteLine("{0} minutos equivale a ", minutos);
15     imprimeFormatoHora(minutos);
16 Console.ReadKey();
17 }
```

Entre com os minutos: 345
345 minutos equivale a

TESTE DE MESA

[illegible]

Execução de um procedimento

```
1 static void imprimeFormatoHora(int qtdMinutos)
2 {
3     int hora, min;
4     hora = qtdMinutos / 60;
5     min = qtdMinutos % 60;
6 Console.WriteLine("{0}:{1}", hora, min);
7 }
8
9 static void Main(string[] args)
10 {
11     int minutos;
12     Console.WriteLine("Minutos para converter em hora:");
13     minutos = Convert.ToInt32(Console.ReadLine());
14     Console.WriteLine("{0} minutos equivale a ", minutos);
15     imprimeFormatoHora(minutos);
16 Console.ReadKey();
17 }
```

```
Entre com os minutos: 345
345 minutos equivale a 5:45
```

TESTE DE MESA

[illegible]

Execução de um procedimento

```
1 static void imprimeFormatoHora(int qtdMinutos)
2 {
3     int hora, min;
4     hora = qtdMinutos / 60;
5     min = qtdMinutos % 60;
6 Console.WriteLine("{0}:{1}", hora, min);
7 }
8
9 static void Main(string[] args)
10 {
11     int minutos;
12     Console.WriteLine("Minutos para converter em hora:");
13     minutos = Convert.ToInt32(Console.ReadLine());
14     Console.WriteLine("{0} minutos equivale a ", minutos);
15     imprimeFormatoHora(minutos);
16 Console.ReadKey();
17 }
```

```
Entre com os minutos: 345
345 minutos equivale a 5:45
```

TESTE DE MESA

[illegible]

Execução de um procedimento

```
1 static void imprimeFormatoHora(int qtdMinutos)
2 {
3     int hora, min;
4     hora = qtdMinutos / 60;
5     min = qtdMinutos % 60;
6 Console.WriteLine("{0}:{1}", hora, min);
7 }
8
9 static void Main(string[] args)
10 {
11     int minutos;
12     Console.WriteLine("Minutos para converter em hora:");
13     minutos = Convert.ToInt32(Console.ReadLine());
14     Console.WriteLine("{0} minutos equivale a ", minutos);
15     imprimeFormatoHora(minutos);
16 Console.ReadKey();
17 }
```

```
Entre com os minutos: 345
345 minutos equivale a 5:45
```

TESTE DE MESA

[illegible]

Tipos de sub-rotina

- ***Função:*** sub-rotina que retorna um único valor para o programa ou sub-rotina que a chamou.
- Funções são utilizadas para realizar uma operação e retornam alguma resposta relativa à operação realizada.

Exemplo de função

```
3 static int converteEmMinutos(int hora, int min)
4 {
5     int totalMinutos;
6     totalMinutos = hora * 60 + min;
7     return totalMinutos;
8 }
9
10 static void Main(string[] args)
11 {
12     int hora,min,qtdMinutos;
13     Console.Write("Horas e minutos:");
14     hora = Convert.ToInt32(Console.ReadLine());
15     min = Convert.ToInt32(Console.ReadLine());
16     qtdMinutos = converteEmMinutos(hora, min);
17     Console.Write("{0}:{1} = {2} minutos.",
18         hora, min, qtdMinutos);
19 }
```


Exemplo de função

```
3 static int converteEmMinutos(int hora, int min)
4 {
5     int totalMinutos;
6     totalMinutos = hora * 60 + min;
7     return totalMinutos;
8 }
9
10 static void Main(string[] args)
11 {
12     int hora,min,qtMinutos;
13     Console.Write("Horas e minutos:");
14     hora = Convert.ToInt32(Console.ReadLine());
15     min = Convert.ToInt32(Console.ReadLine());
16     qtMinutos = converteEmMinutos(hora, min);
17     Console.Write("{0}:{1} = {2} minutos.",
18         hora, min, qtMinutos);
19 }
```

Tipo de retorno

O tipo informado deve ser definido com base no dado produzido pela função. Neste caso, o valor que a função vai retornar deve ser um número inteiro.

Exemplo de função

```
3 static int converteEmMinutos(int hora, int min)
4 {
5     int totalMinutos;
6     totalMinutos = hora * 60 + min;
7     return totalMinutos;
8 }
9
10 static void Main(string[] args)
11 {
12     int hora,min,qtddMinutos;
13     Console.Write("Horas e minutos:");
14     hora = Convert.ToInt32(Console.ReadLine());
15     min = Convert.ToInt32(Console.ReadLine());
16     qtddMinutos = converteEmMinutos(hora, min);
17     Console.Write("{0}:{1} = {2} minutos.",
18         hora, min, qtddMinutos);
19 }
```

Comando de retorno

O valor que será informado no retorno da função precisa ser indicado junto ao comando **return**.

Neste caso, este valor corresponde ao conteúdo da variável `totalMinutos`.

Exemplo de função

```
3 static int converteEmMinutos(int hora, int min)
4 {
5     int totalMinutos;
6     totalMinutos = hora * 60 + min;
7     return totalMinutos;
8 }
9
10 static void Main(string[] args)
11 {
12     int hora,min,qtddMinutos;
13     Console.Write("Horas e minutos:");
14     hora = Convert.ToInt32(Console.ReadLine());
15     min = Convert.ToInt32(Console.ReadLine());
16     qtddMinutos = converteEmMinutos(hora, min);
17     Console.Write("{0}:{1} = {2} minutos.",
18         hora, min, qtddMinutos);
19 }
```

Chamada

Como a função, ao ser executada, retorna um valor para a sub-rotina que a chamou, este valor normalmente é armazenado em uma variável de mesmo tipo (recomendável para quem está começando a programar).

Exemplo de função

```
3 static int converteEmMinutos(int hora, int min)
4 {
5     int totalMinutos;
6     totalMinutos = hora * 60 + min;
7     return totalMinutos;
8 }
9
10 static void Main(string[] args)
11 {
12     int hora,min,qtdMinutos;
13     Console.Write("Horas e minutos:");
14     hora = Convert.ToInt32(Console.ReadLine());
15     min = Convert.ToInt32(Console.ReadLine());
16     qtdMinutos = converteEmMinutos(hora, min);
17     Console.Write("{0}:{1} = {2} minutos.",
18         hora, min, qtdMinutos);
19 }
```

Chamada

Outras opções são imprimir o valor de retorno, utilizar o valor em uma expressão, ou utilizá-lo em qualquer outra situação onde um valor do mesmo tipo é esperado.

Execução de uma função

```

3 static int converteEmMinutos(int hora, int min)
4 {
5     int totalMinutos;
6     totalMinutos = hora * 60 + min;
7 return totalMinutos; 8
9 }
10 static void Main(string[] args)
11 {
12     int hora,min,qtdMinutos;
13     Console.WriteLine("Horas e minutos:");
14     hora = Convert.ToInt32(Console.ReadLine());
15     min = Convert.ToInt32(Console.ReadLine());
16     qtdMinutos = converteEmMinutos(hora, min);
17     Console.WriteLine("{0}:{1} = {2} minutos.",
18 hora, min, qtdMinutos); 19 }

```

TESTE DE MESA

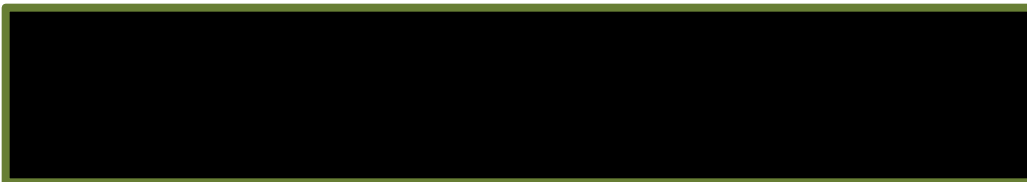
[illegible]

Execução de uma função

```
3 static int converteEmMinutos(int hora, int min)
4 {
5     int totalMinutos;
6     totalMinutos = hora * 60 + min;
7 return totalMinutos;
8 }
9
10 static void Main(string[] args)
11 {
12     int hora,min, qtdMinutos;
13     Console.WriteLine("Horas e minutos:");
14     hora = Convert.ToInt32(Console.ReadLine());
15     min = Convert.ToInt32(Console.ReadLine());
16     qtdMinutos = converteEmMinutos(hora, min);
17     Console.WriteLine("{0}:{1} = {2} minutos.",
18 hora, min, qtdMinutos);
19 }
```

TESTE DE MESA

linha	hora	min	hora	min	totalMinutos
10	?	?			



Execução de uma função

```
3 static int converteEmMinutos(int hora, int min)
4 {
5     int totalMinutos;
6     totalMinutos = hora * 60 + min;
7 return totalMinutos;
8 }
9
10 static void Main(string[] args)
11 {
12     int hora,min,qtdMinutos;
13     Console.WriteLine("Horas e minutos:");
14     hora = Convert.ToInt32(Console.ReadLine());
15     min = Convert.ToInt32(Console.ReadLine());
16     qtdMinutos = converteEmMinutos(hora, min);
17     Console.WriteLine("{0}:{1} = {2} minutos.",
18 hora, min, qtdMinutos);
19 }
```

TESTE DE MESA

linha	hora	min	hora	min	totalMinutos
10	?	?			

Entre com horas e minutos:

Execução de uma função

```

3 static int converteEmMinutos(int hora, int min)
4 {
5     int totalMinutos;
6     totalMinutos = hora * 60 + min;
7 return totalMinutos; 8
8 }
9
10 static void Main(string[] args)
11 {
12     int hora,min,qtddMinutos;
13     Console.Write("Horas e minutos:");
14     hora = Convert.ToInt32(Console.ReadLine());
15     min = Convert.ToInt32(Console.ReadLine());
16     qtddMinutos = converteEmMinutos(hora, min);
17     Console.Write("{0}:{1} = {2} minutos.",
18 hora, min, qtddMinutos); 19 }

```

Entre com horas e minutos: 5

TESTE DE MESA

[illegible]

Execução de uma função

```

3 static int converteEmMinutos(int hora, int min)
4 {
5     int totalMinutos;
6     totalMinutos = hora * 60 + min;
7 return totalMinutos; 8
9 }
10 static void Main(string[] args)
11 {
12     int hora,min,qtdMinutos;
13     Console.Write("Horas e minutos:");
14     hora = Convert.ToInt32(Console.ReadLine());
15     min = Convert.ToInt32(Console.ReadLine());
16     qtdMinutos = converteEmMinutos(hora, min);
17     Console.Write("{0}:{1} = {2} minutos.",
18 hora, min, qtdMinutos); 19 }

```

```
Entre com horas e minutos: 5
45
```

TESTE DE MESA

[illegible]

Execução de uma função

```
3 static int converteEmMinutos(int hora, int min)
4 {
5     int totalMinutos;
6     totalMinutos = hora * 60 + min;
7 return totalMinutos;
8 }
9
10 static void Main(string[] args)
11 {
12     int hora,min,qtdMinutos;
13     Console.Write("Horas e minutos:");
14     hora = Convert.ToInt32(Console.ReadLine());
15     min = Convert.ToInt32(Console.ReadLine());
16     qtdMinutos = converteEmMinutos(hora, min);
17     Console.Write("{0}:{1} = {2} minutos.",
18 hora, min, qtdMinutos);
19 }
```

TESTE DE MESA

linha	hora	min	hora	min	totalMinutos
10	?	?			
14	5	45			

```
Entre com horas e minutos: 5
45
```

Execução de uma função

```

3 static int converteEmMinutos(int hora, int min)
4 {
5     int totalMinutos;
6     totalMinutos = hora * 60 + min;
7 return totalMinutos; 8
}
9
10 static void Main(string[] args)
11 {
12     int hora,min,qtdMinutos;
13     Console.WriteLine("Horas e minutos:");
14     hora = Convert.ToInt32(Console.ReadLine());
15     min = Convert.ToInt32(Console.ReadLine());
16     qtdMinutos = converteEmMinutos(hora, min);
17     Console.WriteLine("{0}:{1} = {2} minutos.",
18 hora, min, qtdMinutos); 19 }

```

```
Entre com horas e minutos: 5
45
```

TESTE DE MESA

[illegible]

Execução de uma função

```
3 static int converteEmMinutos(int hora, int min)
4 {
5     int totalMinutos;
6     totalMinutos = hora * 60 + min;
7     return totalMinutos;
8 }
9
10 static void Main(string[] args)
11 {
12     int hora,min,qtddMinutos;
13     Console.WriteLine("Horas e minutos:");
14     hora = Convert.ToInt32(Console.ReadLine());
15     min = Convert.ToInt32(Console.ReadLine());
16     qtddMinutos = converteEmMinutos(hora, min);
17     Console.WriteLine("{0}:{1} = {2} minutos.",
18 hora, min, qtddMinutos); 19 }
```

TESTE DE MESA

linha	hora	min	hora	min	totalMinutos
10	?	?			
14	5	45			
3			5	45	?
6			5	45	345

```
Entre com horas e minutos: 5
45
```

Execução de uma função

```
3 static int converteEmMinutos(int hora, int min)
4 {
5     int totalMinutos;
6     totalMinutos = hora * 60 + min;
7     return totalMinutos;
8 }
9
10 static void Main(string[] args)
11 {
12     int hora,min,qtddMinutos;
13     Console.WriteLine("Horas e minutos:");
14     hora = Convert.ToInt32(Console.ReadLine());
15     min = Convert.ToInt32(Console.ReadLine());
16     qtddMinutos = converteEmMinutos(hora, min);
17     Console.WriteLine("{0}:{1} = {2} minutos.",
18 hora, min, qtddMinutos); 19 }
```

TESTE DE MESA

linha	hora	min	hora	min	totalMinutos
10	?	?			
14	5	45			
3			5	45	?
6			5	45	345

```
Entre com horas e minutos: 5
45
```

Execução de uma função

```

3 static int converteEmMinutos(int hora, int min)
4 {
5     int totalMinutos;
6     totalMinutos = hora * 60 + min;
7 return totalMinutos; 8
9 }
10 static void Main(string[] args)
11 {
12     int hora,min,qtdMinutos;
13     Console.Write("Horas e minutos:");
14     hora = Convert.ToInt32(Console.ReadLine());
15     min = Convert.ToInt32(Console.ReadLine());
16     qtdMinutos = converteEmMinutos(hora, min);
17     Console.Write("{0}:{1} = {2} minutos.",
18 hora, min, qtdMinutos); 19 }

```

```
Entre com horas e minutos: 5
45
```

TESTE DE MESA

[illegible]

Execução de uma função

```

3 static int converteEmMinutos(int hora, int min)
4 {
5     int totalMinutos;
6     totalMinutos = hora * 60 + min;
7 return totalMinutos; 8
8 }
9
10 static void Main(string[] args)
11 {
12     int hora,min,qtdMinutos;
13     Console.Write("Horas e minutos:");
14     hora = Convert.ToInt32(Console.ReadLine());
15     min = Convert.ToInt32(Console.ReadLine());
16     qtdMinutos = converteEmMinutos(hora, min);
17     Console.Write("{0}:{1} = {2} minutos.",
18         hora, min, qtdMinutos);
19 }

```

```
Entre com horas e minutos: 5
45
5:45 = 345 minutos.
```

TESTE DE MESA

[illegible]

Execução de uma função

```

3 static int converteEmMinutos(int hora, int min)
4 {
5     int totalMinutos;
6     totalMinutos = hora * 60 + min;
7 return totalMinutos; 8
}
9
10 static void Main(string[] args)
11 {
12     int hora,min,qtdMinutos;
13     Console.WriteLine("Horas e minutos:");
14     hora = Convert.ToInt32(Console.ReadLine());
15     min = Convert.ToInt32(Console.ReadLine());
16     qtdMinutos = converteEmMinutos(hora, min);
17     Console.WriteLine("{0}:{1} = {2} minutos.",
18 hora, min, qtdMinutos); 19 }

```

```
Entre com horas e minutos: 5
45
5:45 = 345 minutos.
```

TESTE DE MESA

[illegible]

Projetando uma sub-rotina

- Durante a criação de um programa, quando é identificada uma tarefa que pode ser codificada como uma sub-rotina, é necessário primeiramente analisar a sua **SEMÂNTICA** para que então seja possível definir a sua **SINTAXE**.

Projetando uma sub-rotina

SEMÂNTICA: refletindo sobre a tarefa a ser executada pela sub-rotina, é necessário definir:

- Quais dados de entrada são necessários?
- A tarefa deve produzir um resultado como retorno?

Projetando uma sub-rotina

SINTAXE: após definir os dados de entrada e o resultado da sub-rotina, é necessário especificar o tipo de cada um dos parâmetros e o tipo de retorno.

- Se a sub-rotina não tem parâmetro, ainda assim os parênteses são necessários (mesmo sem conteúdo algum).
- Se não há retorno de valor, o tipo é **void**.

Projetando uma sub-rotina

- Depois de definir a declaração (ou protótipo) da sub-rotina, o procedimento utilizado para codificar seu bloco de comandos é idêntico ao que você já utilizava para criar programas:
 - identificar as variáveis adicionais necessárias;
 - declarar tais variáveis;
 - fazer refinamentos ou utilizar o método direto para implementar a tarefa necessária.
- Ao final, deve-se usar o comando de retorno.

Exercícios

- 1) Faça um procedimento que receba três valores reais e imprima a média aritmética desses valores.
- 2) Considerando a fórmula para o cálculo da distância entre dois pontos (x_1, y_1) e (x_2, y_2) :
 - a) Escreva uma função que receba como parâmetros as coordenadas de dois pontos e retorne a distância entre eles.
 - b) Escreva um programa C# (função principal) que capture do teclado as coordenadas dos 3 vértices de um triângulo, calcule e imprima o perímetro deste triângulo. Este programa deve utilizar a função do item anterior.
- 3) Escreva um procedimento que recebe dois números inteiros e imprime, a soma, o produto, a diferença, o quociente e o resto entre esses dois números.

Variáveis e escopo

- Escopo: contexto que define a visibilidade e acessibilidade das variáveis em diferentes partes do programa.
- Toda variável tem um escopo. O escopo da variável equivale às linhas de código onde a variável pode ser acessada, lida e/ou modificada.

Variáveis e escopo

- São denominadas variáveis locais:
 - as variáveis declaradas na sub-rotina;
 - todos os parâmetros recebidos pela sub-rotina.
- O escopo de uma variável local corresponde apenas ao bloco de comandos de sua sub-rotina.
- Dentro de uma sub-rotina não se tem acesso a variáveis declaradas em outra sub-rotina.

Variáveis e escopo

- A reserva de memória para uma variável local é condicionada à execução da sub-rotina:
 - A reserva é refeita cada vez que a sub-rotina é executada (podem ser reservados endereços distintos a cada execução).
 - Quando a execução da sub-rotina termina, os espaços de memória reservados (e as respectivas variáveis associadas) são liberados para outros usos e já não podem ser acessados.

Variáveis e escopo

Os nomes das variáveis locais coincidentemente são iguais mas elas são completamente independentes.

O endereço de memória da variável `raio` da função `main` é diferente do endereço de memória do parâmetro `raio` da sub-rotina `volume_cilindro`.

```
static double volume_cilindro(double raio, double altura)
{
    double volume = 3.14159 * raio * raio * altura;
    return volume;
}

static void Main(string[] args)
{
    double raio, altura, volume;
    Console.Write("Entre com o valor do raio: ");
    raio = Convert.ToDouble(Console.ReadLine());
    Console.Write("Entre com o valor da altura: ");
    altura = Convert.ToDouble(Console.ReadLine());

    volume = volume_cilindro(raio, altura);

    Console.Write("Volume do cilindro = ");
    Console.Write(volume);
}
```

Variáveis e escopo

A modificação do nome dos parâmetros e variáveis da função não implica em qualquer modificação no programa.

```
static double volume_cilindro(double r, double h)
{
    double v = 3.14159 * r * r * h;
    return v;
}

static void Main(string[] args)
{
    double raio, altura, volume;
    Console.Write("Entre com o valor do raio: ");
    raio = Convert.ToDouble(Console.ReadLine());
    Console.Write("Entre com o valor da altura: ");
    altura = Convert.ToDouble(Console.ReadLine());
    volume = volume_cilindro(raio, altura);
    Console.Write("Volume do cilindro = ");
    Console.Write(volume);
}
```

Variáveis e escopo

Neste exemplo, quando a função é chamada, o valor das variáveis `raio` e `altura` declaradas na função `main` são utilizados para inicializar os parâmetros `raio` e `altura` da função `volume_cilindro`.

```
static double volume_cilindro(double raio, double altura)
{
    double volume = 3.14159 * raio * raio * altura;
    return volume;
}

static void Main(string[] args)
{
    double raio, altura, volume;
    Console.Write("Entre com o valor do raio: ");
    raio = Convert.ToDouble(Console.ReadLine());
    Console.Write("Entre com o valor da altura: ");
    altura = Convert.ToDouble(Console.ReadLine());

    volume = volume_cilindro(raio, altura);

    Console.Write("Volume do cilindro = ");
    Console.Write(volume);
}
```

Chamadas e parâmetros

- Quando uma sub-rotina é chamada, é necessário indicar um valor de entrada para cada um de seus parâmetros.
- Este valor pode ser obtido diretamente de uma constante, de uma variável, de uma expressão ou, até mesmo, do valor de retorno de outra função.

```
volume = volume_cilindro( 4.0, 5.0 );  
volume = volume_cilindro( raio, 10.0 / 2 );  
volume = volume_cilindro( diametro / 2, altura );  
volume = volume_cilindro( raio, Math.Sqrt( 9 ) + Math.Sqrt( raio ) );
```

Chamadas e parâmetros

- Quando uma única variável indica o valor de um parâmetro em uma chamada, apenas o valor desta é repassado para a sub-rotina e utilizado na inicialização do parâmetro.

```
static double volume_cilindro (double r, double a)
{
    double v = 3.14159 * r * r * a;
    return v;
}
static void Main(string[] args)
{
    double raio=4.0, altura=5.0, volume;
    volume = volume_cilindro(raio, altura);
    Console.WriteLine("Volume do cilindro: %f", volume);
}
```

The diagram illustrates the parameter passing process. In the `Main` function, the variables `raio` and `altura` are initialized with the values `4.0` and `5.0` respectively. Red arrows show these values being passed to the `volume_cilindro` function, where they are assigned to the parameters `r` and `a`.

Chamadas e parâmetros

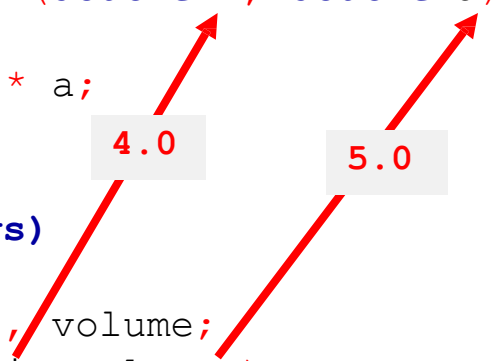
- Não há relação entre as variáveis **raio** e **r**.
- Se o valor de **r** for modificado na sub-rotina, a variável **raio** permanece inalterada.

```
static double volume_cilindro (double r, double a)
{
    double v = 3.14159 * r * r * a;
    return v;
}
static void Main(string[] args)
{
    double raio=4.0, altura=5.0, volume;
    volume = volume_cilindro(raio, altura);
    Console.WriteLine("Volume do cilindro: %f", volume);
}
```

Chamadas e parâmetros

- Esta independência entre variável e parâmetro, indica a ocorrência de ***passagem de parâmetro por valor***.

```
static double volume_cilindro (double r, double a)
{
    double v = 3.14159 * r * r * a;
    return v;
}
static void Main(string[] args)
{
    double raio=4.0, altura=5.0, volume;
    volume = volume_cilindro(raio, altura);
    Console.WriteLine("Volume do cilindro: %f", volume);
}
```



Chamadas e parâmetros

- Existe outro tipo de relação entre variável e parâmetro denominado ***passagem de parâmetro por referência***.
- Neste caso, o parâmetro da sub-rotina recebe um endereço de uma variável (ao invés de um valor de determinado tipo).

Chamadas e parâmetros

- Se houver alteração no local indicado pelo parâmetro que recebeu um endereço de variável, a respectiva variável será modificada, mesmo sem ter sido declarada na sub-rotina.

Chamadas e parâmetros

- A ***passagem de parâmetro por referência*** não será detalhada nesta disciplina.

Exercícios

- 4) Faça o teste de mesa dos programas gerados no exercício anterior.
- 5) Faça o teste de mesa do programa abaixo:

```
static int calculo (int p, int q)
{
    p = p * 10;
    q = q + 10;
    return (p + q) ;
}

static void Main(string[] args)
{
    int x = 2, y = 5;
    Console.WriteLine("{0} {1} {2}", x, y, calculo(x, y));
}
```

Subrotinas

Aula de Exercícios

Revisão

- A forma geral de uma subrotina é:

```
tipoDeDados nomeDaSubrotina ( listaParametros )  
{  
    corpo da subrotina;  
    return xxx;  
}
```

Revisão: Tipos de subrotinas

- Procedimentos

```
void ImprimeSoma (int a, int b)
{
    Console.WriteLine (a+b) ;
}
```

- Funções

```
int Soma (int a, int b)
{
    return (a+b) ;
}
```

Implementação de subrotinas em C#

```
int Func1(int a)
{
    ...
    return val;
}

static void Main(string[] args)
{
    ...
    x=Func1(10);
    ...
}
```

```
static void Main(string[] args)
{
    ...
    x=Func1(10);
    ...
}

int Func1(int a)
{
    ...
    return val;
}
```

Exemplo completo - Procedimento

Escreva um programa que leia as três notas de um aluno. Calcule a média desse aluno e mostre, ao final, a média calculada.

O cálculo da média e a impressões devem ser feitos por um procedimento.

```
static void calculaMedia(double nota1, double nota2, double nota3)
{
    double media;
    media = (nota1 + nota2 + nota3) / 3;
    Console.WriteLine("Media = {0:N2}", media);
}

static void Main(string[] args)
{
    double n1, n2, n3;
    Console.WriteLine("Digite as 3 notas: ");
    n1 = Convert.ToDouble(Console.ReadLine());
    n2 = Convert.ToDouble(Console.ReadLine());
    n3 = Convert.ToDouble(Console.ReadLine());
    calculaMedia (n1, n2, n3);
}
```


Exemplo completo - Função

Escreva um programa que calcule e imprima o quadrado de um número.
O cálculo deve ser feito por uma função.

```
static int quadrado (int x)
{
    return (x * x);
}

static int Main (string args[])
{
    int numero, res;
    Console.Write("Digite um numero inteiro: ");
    numero = Convert.ToInt32(Console.ReadLine());

    res = quadrado(numero);
    Console.Write("O quadrado de {0} é {1}.\n", numero, res);
}
```

Exercícios

- 1) Faça um procedimento que receba por parâmetro o tempo de duração de um experimento expresso em segundos e imprima na tela esse mesmo tempo em horas, minutos e segundos.
- 2) Faça uma função que receba por parâmetro o raio de uma esfera e calcule o seu volume:
$$v = (4 * \pi * R^3) / 3.$$
- 3) Faça uma função que receba a idade de uma pessoa em anos, meses e dias e retorne essa idade expressa em dias.

Exercícios

4) Faça uma função que receba dois números reais, a e b, e retorne o valor de $(A^2 + B^2)^{1/2}$.

Observação: para calcular a raiz quadrada será preciso utilizar a função `Math.Sqrt()`.

5) Considere as equações de movimento para calcular a posição (s) e velocidade (v) de uma partícula em um determinado instante t, dado sua aceleração a, posição inicial s0 e velocidade inicial v0, de acordo com as fórmulas:

$$s = s0 + v0*t + (a * t * t)/2$$

$$v = v0 + a*t$$

Escreva um programa C# completo que capture os valores de s0, v0, a e t, fornecidos pelo usuário via teclado, e calcule e imprima os valores de s e v.

Exercícios

6) Considerando o critério de aprovação de uma disciplina que determina que um aluno está aprovado se a média ponderada de suas três provas for maior ou igual a 5.0, onde a média é dada pela fórmula:

$$\text{Média} = (P1 + P2 + 2.0 * P3) / 4.0$$

- (a) Escreva uma função que receba como parâmetros as notas das duas primeiras provas de um aluno (P1 e P2) e retorne a nota mínima que o aluno precisa tirar na terceira prova para que seja aprovado.
- (b) Escreva um programa em C# completo que leia do teclado as duas primeiras notas de um aluno, calcule e imprima a nota mínima que o aluno precisa tirar na P3 para que seja aprovado. Este programa deve fazer uso da função do item anterior.