Funções

DCC 119 – Algoritmos

Detalhar ou simplificar?

Arrume seu quarto
Compre 5 pães na padaria
Ponha a mesa do café

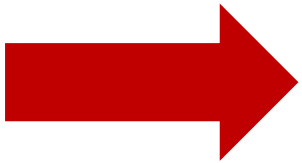
Organize suas roupas espalhadas
Arrume sua cama
Organize sua escrivaninha
Pegue o dinheiro na mesa
Vá até a padaria
Peça 5 pães de sal e pague por eles
Volte para casa
Retire os objetos que estão na mesa
Busque a toalha de mesa e estenda-a
....

Para cada peça de roupa espalhada pelo quarto:
- se estiver suja, coloque-a no cesto de roupas
- senão, dobre-a e guarde-a no guarda-roupas
Guarde o travesseiro
Dobre o cobertor e guarde-o
Estique o lençol
...

Detalhar ou simplificar?

Arrume seu quarto
Compre 5 pães na padaria
Ponha a mesa do café

O computador precisa de instruções muito simples e precisas.

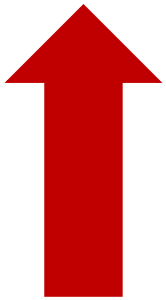


Organize suas roupas espalhadas
Arrume sua cama
Organize sua escrivaninha
Pegue o dinheiro na mesa
Vá até a padaria
Peça 5 pães de sal e pague por eles
Volte para casa
Retire os objetos que estão na mesa
Busque a toalha de mesa e estenda-a
....

Para cada peça de roupa espalhada pelo quarto:
- se estiver suja, coloque-a no cesto de roupas
- senão, dobre-a e guarde-a no guarda-roupas
Guarde o travesseiro
Dobre o cobertor e guarde-o
Estique o lençol
...

Detalhar ou simplificar?

Arrume seu quarto
Compre 5 pães na padaria
Ponha a mesa do café



Mas, para o programador, é mais fácil usar instruções que representam tarefas mais abrangentes...

Organize suas roupas espalhadas
Arrume sua cama
Organize sua escrivaninha
Pegue o dinheiro na mesa
Vá até a padaria
Peça 5 pães de sal e pague por eles
Volte para casa
Retire os objetos que estão na mesa
Busque a toalha de mesa e estenda-a
....

Para cada peça de roupa espalhada pelo quarto:
- se estiver suja, coloque-a no cesto de roupas
- senão, dobre-a e guarde-a no guarda-roupas
Guarde o travesseiro
Dobre o cobertor e guarde-o
Estique o lençol
...

Funções

Uma função é um trecho de código computacional que realiza uma **tarefa bem definida**.

Exemplos:

- *Calcular a área de um círculo dado seu raio;*
- *Converter graus Celsius para Fahrenheit;*
- *Calcular a nota final de um aluno dadas as notas obtidas nas avaliações;*
- *Imprimir as informações de um produto dado seu código;*
- *...*

Funções

- Por realizar uma tarefa bem definida a mesma função pode:
  - ser útil várias vezes em um mesmo programa.
  - ser útil para diferentes programas.
- A função pode receber dados para ser executada.
- A função pode produzir um resultado (por exemplo: função que calcula a área de um círculo dado seu raio).

Funções

Você já utiliza funções em seus programas:

- **`printf`** é uma função que realiza a tarefa de imprimir na tela do computador
- **`scanf`** é uma função que lê valores do teclado e armazena em variáveis indicadas entre parênteses
- **`sqrt`** é uma função que calcula e devolve o valor da raiz quadrada de um número
- **`pow`** é uma função que calcula e retorna o valor da potência, dados base e expoente indicados

Funções

Repare que você não precisa saber como as funções **`printf`**, **`scanf`**, **`sqrt`** e **`pow`** foram implementadas.

É necessário saber apenas:

- o que a função faz, isto é, qual a tarefa executada por ela (semântica);
- como ela pode ser utilizada (sintaxe).

Funções

Além de seu nome, a sintaxe da função pode (ou não) especificar:

- parâmetros – dados que a função precisa receber para executar;
- retorno – o tipo de resultado produzido pela função.

Criando e utilizando funções

Existem três situações distintas na criação e utilização de funções:

1) ***Declaração***

2) ***Definição***

3) ***Chamada***

O que seria...

1) ***Declaração:*** a declaração de uma função especifica sua sintaxe.

A declaração corresponde a um cabeçalho que apresenta tudo o que o usuário precisa saber para utilizar a função em seu programa.

```
 Comprar_paes
(deve ser informado: quantidade_de_paes)
```

O que seria...

2) ***Definição:*** a definição de uma função é a implementação da mesma. Além de conter o cabeçalho que especifica sua sintaxe (como na declaração), contém a sequência de instruções necessárias para realizar a tarefa.

```
Comprar_pães
(deve ser informado: quantidade_de_paes)
 1.Ir até a Padaria
 2.Perguntar o preço do pão
 3.Se dinheiro na carteira >= quantidade * preço
   3.1.Pegar os pães
   3.2.Pagar pelos pães e guardar o troco
 4.Voltar para casa
```

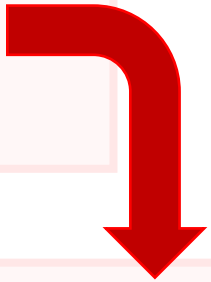
O que seria...

3) ***Chamada:*** a chamada a uma função é a utilização da mesma dentro de alguma outra função ou programa. Nesse caso, é necessário indicar o valor de cada informação necessária para executar a função.

```
Fazer_logo_que_acordar
 1. Arrumar_o_quarto
 2. Comprar_pães( quantidade = 5 )
 3. Por a mesa do café
```

Execução da função

```
Fazer_logo_que_acordar
 1. Arrumar_o_quarto
 2. Comprar_pães( quantidade = 5 )
 3. Por a mesa do café
```



```
Comprar_pães ( sendo que quantidade = 5 )
 1.Ir até a Padaria
 2.Perguntar o preço do pão
 3.Se dinheiro na carteira >= 5 * preço
   3.1.Pegar os 5 pães
   3.2.Pagar pelos 5 pães e guardar o troco
 4.Voltar para casa
```

Criando e utilizando funções

Para ilustrar a *declaração*, *definição* e *chamada* de funções, nos próximos slides teremos:

⇒ uma função que converte uma temperatura de graus Celsius para Fahrenheit, e

⇒ um programa que
  (1) lê uma temperatura em graus Celsius,
  (2) chama a função para convertê-la para Fahrenheit,
  (3) imprime a temperatura em Fahrenheit.

Criando e utilizando funções

```c
#include <stdio.h>

float celsius_fahrenheit ( float tc );

int main()
{
  float cels;
  float fahr;
  printf("Entre com temperatura em Celsius: ");
  scanf("%f", &cels);
  fahr = celsius_fahrenheit(cels);
  printf("Temperatura em Fahrenheit: ");
  printf("%f", fahr);
  return 0;
}

float celsius_fahrenheit ( float tc )
{
  float tf;
  tf = 1.8 * tc + 32;
  return tf;
}
```

Declaração

Chamada

Definição

Declaração de uma função

```c
#include <stdio.h>

float celsius_fahrenheit ( float tc );

int main()
{
  float cels;
  float fahr;
  printf("Entre com temperatura em Celsius: ");
  scanf("%f", &cels);
  fahr = celsius_fahrenheit(cels);
  printf("Temperatura em Fahrenheit: ");
  printf("%f", fahr);
  return 0;
}

float celsius_fahrenheit ( float tc )
{
  float tf;
  tf = 1.8 * tc + 32;
  return tf;
}
```

```c
tipo_de_retorno  nome_da_funcao ( lista_de_parametros );
```

A *declaração* de uma função fornece todas as informações necessárias para que o programador possa utilizá-la.

Declaração de uma função

```c
#include <stdio.h>

float celsius_fahrenheit ( float tc );

int main()
{
  float cels;
  float fahr;
  printf("Entre com temperatura em Celsius: ");
  scanf("%f", &cels);
  fahr = celsius_fahrenheit(cels);
  printf("Temperatura em Fahrenheit: ");
  printf("%f", fahr);
  return 0;
}

float celsius_fahrenheit ( float tc )
{
  float tf;
  tf = 1.8 * tc + 32;
  return tf;
}
```

Toda função precisa ser *definida* ou *declarada* antes de ser usada.

Declaração de uma função

```c
#include <stdio.h>

float celsius_fahrenheit ( float tc )
{
  float tf;
  tf = 1.8 * tc + 32;
  return tf;
}

int main()
{
  float cels;
  float fahr;
  printf("Entre com temperatura em Celsius:
  scanf("%f", &cels);
  fahr = celsius_fahrenheit(cels);
  printf("Temperatura em Fahrenheit: ");
  printf("%f", fahr);
  return 0;
}
```

Toda função precisa ser *definida* ou *declarada* antes de ser usada.

A *declaração* é desnecessária se a função for *definida* antes da *chamada*.

Definição de uma função

```c
#include <stdio.h>

float celsius_fahrenheit ( float tc )
{
  float tf;
  tf = 1.8 * tc + 32;
  return tf;
}

int main()
{
  float cels;
  float fahr;
  printf("Entre com temperatura em Celsius: ");
  scanf("%f", &cels);
  fahr = celsius_fahrenheit(cels);
  printf("Temperatura em Fahrenheit: ");
  printf("%f", fahr);
  return 0;
}
```

```c
tipoDeRetorno  nomeDaFuncao ( parametros )
{
   bloco_de_comandos_da_funcao
}
```

Definição de uma função

```c
#include <stdio.h>

float celsius_fahrenheit ( float tc )
{
  float tf;
  tf = 1.8 * tc + 32;
  return tf;
}

int main()
{
  float cels;
  float fahr;
  printf("Entre com temperatura em Celsius: ");
  scanf("%f", &cels);
  fahr = celsius_fahrenheit(cels);
  printf("Temperatura em Fahrenheit: ");
  printf("%f", fahr);
  return 0;
}
```

A função 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 função

```c
#include <stdio.h>

float celsius_fahrenheit ( float tc )
{
  float tf;
  tf = 1.8 * tc + 32;
  return tf;
}

int main()
{
  float cels;
  float fahr;
  printf("Entre com temperatura em Celsius: ");
  scanf("%f", &cels);
  fahr = celsius_fahrenheit(cels);
  printf("Temperatura em Fahrenheit: ");
  printf("%f", fahr);
  return 0;
}
```

O *nome* da função:

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

Definição de uma função

```c
#include <stdio.h>

float celsius_fahrenheit ( float tc )
{
  float tf;
  tf = 1.8 * tc + 32;
  return tf;
}

int main()
{
  float cels;
  float fahr;
  printf("Entre com temperatura em Celsius: ");
  scanf("%f", &cels);
  fahr = celsius_fahrenheit(cels);
  printf("Temperatura em Fahrenheit: ");
  printf("%f", fahr);
  return 0;
}
```

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

- **tipo do parâmetro**
- **nome do parâmetro no bloco da função**

Definição de uma função

```c
#include <stdio.h>

float celsius_fahrenheit ( float tc )
{
  float tf;
  tf = 1.8 * tc + 32;
  return tf;
}

int main()
{
  float cels;
  float fahr;
  printf("Entre com temperatura em Celsius: ");
  scanf("%f", &cels);
  fahr = celsius_fahrenheit(cels);
  printf("Temperatura em Fahrenheit: ");
  printf("%f", fahr);
  return 0;
}
```

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 função.

Definição de uma função

```c
#include <stdio.h>

float celsius_fahrenheit ( float tc )
{
  float tf;
  tf = 1.8 * tc + 32;
  return tf;
}

int main()
{
  float cels;
  float fahr;
  printf("Entre com temperatura em Celsius: ");
  scanf("%f", &cels);
  fahr = celsius_fahrenheit(cels);
  printf("Temperatura em Fahrenheit: ");
  printf("%f", fahr);
  return 0;
}
```

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

Se a função não produzir um valor, o comando de retorno pode ser omitido.

Chamada de uma função

```
nome_da_funcao ( lista_com_um_valor_para_cada_parametro );
```

```c
#include <stdio.h>

float celsius_fahrenheit ( float tc );

int main()
{
  float cels;
  float fahr;
  printf("Entre com temperatura em Celsius: ");
  scanf("%f", &cels);
  fahr = celsius_fahrenheit(cels);
  printf("Temperatura em Fahrenheit: ");
  printf("%f", fahr);
  return 0;
}

float celsius_fahrenheit ( float tc )
{
  float tf;
  tf = 1.8 * tc + 32;
  return tf;
}
```

Chamada de uma função

```c
#include <stdio.h>

float celsius_fahrenheit ( float tc );

int main()
{
  float cels;
  float fahr;
  printf("Entre com temperatura em Celsius: ");
  scanf("%f", &cels);
  fahr = celsius_fahrenheit(cels);
  printf("Temperatura em Fahrenheit: ");
  printf("%f", fahr);
  return 0;
}

float celsius_fahrenheit ( float tc )
{
  float tf;
  tf = 1.8 * tc + 32;
  return tf;
}
```

A *chamada* de uma função 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.

Chamada de uma função

```c
#include <stdio.h>

float celsius_fahrenheit ( float tc );

int main()
{
  float cels;
  float fahr;
  printf("Entre com temperatura em Celsius: ");
  scanf("%f", &cels);
  fahr = celsius_fahrenheit(cels);
  printf("Temperatura em Fahrenheit: ");
  printf("%f", fahr);
  return 0;
}

float celsius_fahrenheit ( float tc )
{
  float tf;
  tf = 1.8 * tc + 32;
  return tf;
}
```

No caso da função `celsius_fahrenheit`, o único parâmetro, `tc`, tem tipo `float`. Então, um valor do tipo `float` precisa ser passado como parâmetro.

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

Chamada de uma função

```c
#include <stdio.h>

float celsius_fahrenheit ( float tc );

int main()
{
  float cels;
  float fahr;
  printf("Entre com temperatura em Celsius: ");
  scanf("%f", &cels);
  fahr = celsius_fahrenheit(cels);
  printf("Temperatura em Fahrenheit: ");
  printf("%f", fahr);
  return 0;
}

float celsius_fahrenheit ( float tc )
{
  float tf;
  tf = 1.8 * tc + 32;
  return tf;
}
```

As funções `printf` e `scanf` podem ser utilizadas porque a biblioteca que contém sua declaração foi incluída no início do arquivo, antes do programa.

Projetando uma função

- Ao programar, quando é identificada uma tarefa específica que pode ser codificada como uma função, é necessário primeiramente analisar a sua ***SEMÂNTICA*** para que então seja possível definir a sua ***SINTAXE***.

Projetando uma função

SEMÂNTICA: refletindo sobre a tarefa a ser executada pela função, é necessário definir:

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

Projetando uma função

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

- Se a função 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 função

Depois de definir a declaração (ou protótipo) da função, você deve:

- identificar e declarar as variáveis adicionais necessárias;
- desenvolver uma sequência lógica de passos que realizam a tarefa atribuída à função;
- implementar estes passos traduzindo-os em instruções da linguagem C.

Ao final, deve-se usar o comando de retorno.

Exercício

1. Vamos refazer o programa que calcula área do círculo?

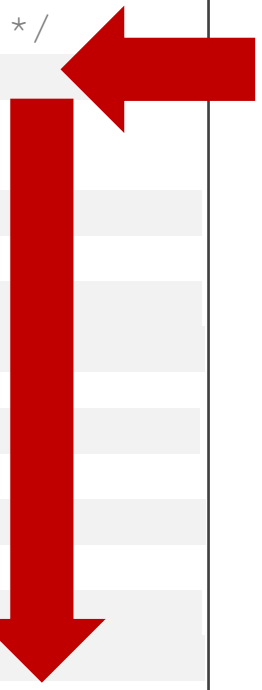
   a) Escreva uma função que recebe um número real, representando o raio do círculo, e que retorne um valor real representando a área.

   b) Faça um programa em C (função principal) que leia um valor real do teclado, chame a função da letra a) e imprima o resultado obtido.

Execução de uma função

Até agora, os programas continham apenas a função `main` e a execução das instruções era sequencial.

```
 1  #include <stdio.h>
 2
 3  /* Programa que calcula a área de um círculo */
 4  int main()
 5  {
 6    //Declara variáveis
 7    float raio, area;
 8    //Imprime informações para o usuário
 9    printf(" Programa que calcula area de ");
10    printf(" um circulo.\n Digite o raio: ");
11    //Lê dado de entrada
12    scanf("%f", &raio );
13    //Calcula area
14    area = 3.14159 * raio * raio;
15    //Imprime resultado
16    printf(" Area do circulo: %f", area);
17    return 0;
18  }
```



Sequência de execução: 4, 7, 9, 10, 12, 14, 16 e 17.

Execução de uma função

Com o uso de funções, a execução do programa passa a ter desvios...

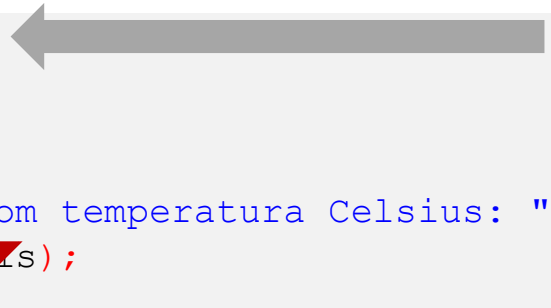
Execução de uma função

```
1 #include <stdio.h>
2
3 float celsius_fahrenheit ( float tc );
4
5 int main() {
6   float cels;
7   float fahr;
8
9   printf("Entre com temperatura Celsius: ");
10  scanf("%f", &cels);
11
12  fahr = celsius_fahrenheit(cels);
13
14  printf("Temperatura em Fahrenheit: ");
15  printf("%f", fahr);
16
17  return 0;
18 }
19
20 float celsius_fahrenheit ( float tc ) {
21   float tf;
22   tf = 1.8 * tc + 32;
23   return tf;
24 }
```

O início da execução é sempre na função `main`.

Execução de uma função

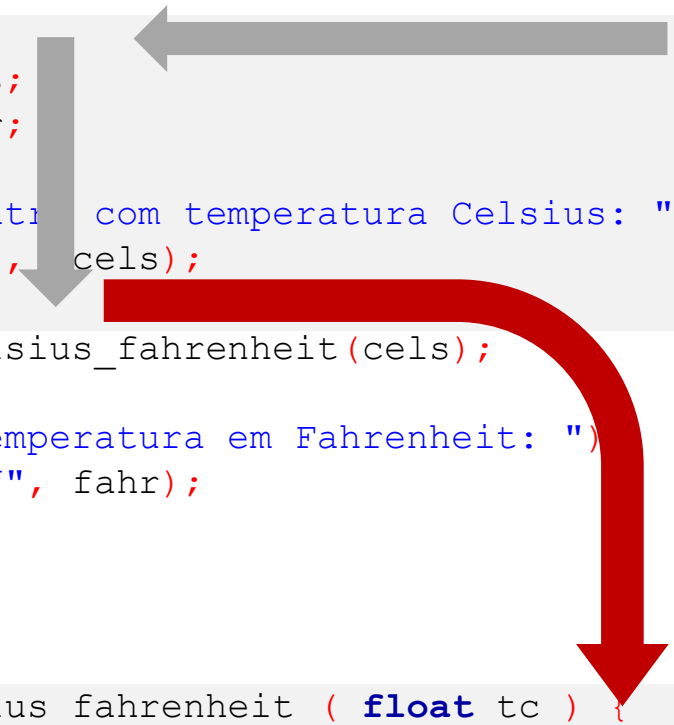
```
1  #include <stdio.h>
2
3  float celsius_fahrenheit ( float tc );
4
5  int main() {
6    float cels;
7    float fahr;
8
9    printf("Entr  om temperatura Celsius: ");
10   scanf("%f",  s);
11
12   fahr = celsius_fahrenheit(cels);
13
14   printf("Temperatura em Fahrenheit: ");
15   printf("%f", fahr);
16
17   return 0;
18 }
19
20 float celsius_fahrenheit ( float tc ) {
21    float tf;
22    tf = 1.8 * tc + 32;
23    return tf;
24 }
```



O fluxo de execução segue de forma sequencial até a chamada da função.

Execução de uma função

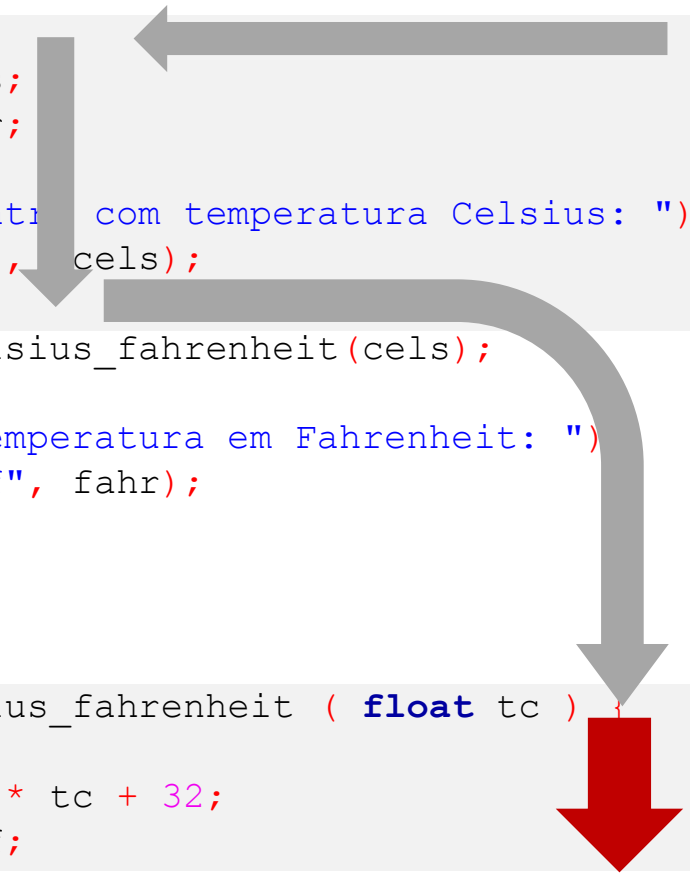
```
1  #include <stdio.h>
2
3  float celsius_fahrenheit ( float tc );
4
5  int main() {
6    float cels;
7    float fahr;
8
9    printf("Entr  com temperatura Celsius: ");
10   scanf("%f",  cels);
11
12   fahr = celsius_fahrenheit(cels);
13
14   printf("Temperatura em Fahrenheit: ")
15   printf("%f", fahr);
16
17   return 0;
18 }
19
20 float celsius_fahrenheit ( float tc ) {
21    float tf;
22    tf = 1.8 * tc + 32;
23    return tf;
24 }
```



Quando a função é chamada, suas instruções precisam ser executadas.
Por isso, o fluxo de execução passa para o início da definição da função.

Execução de uma função

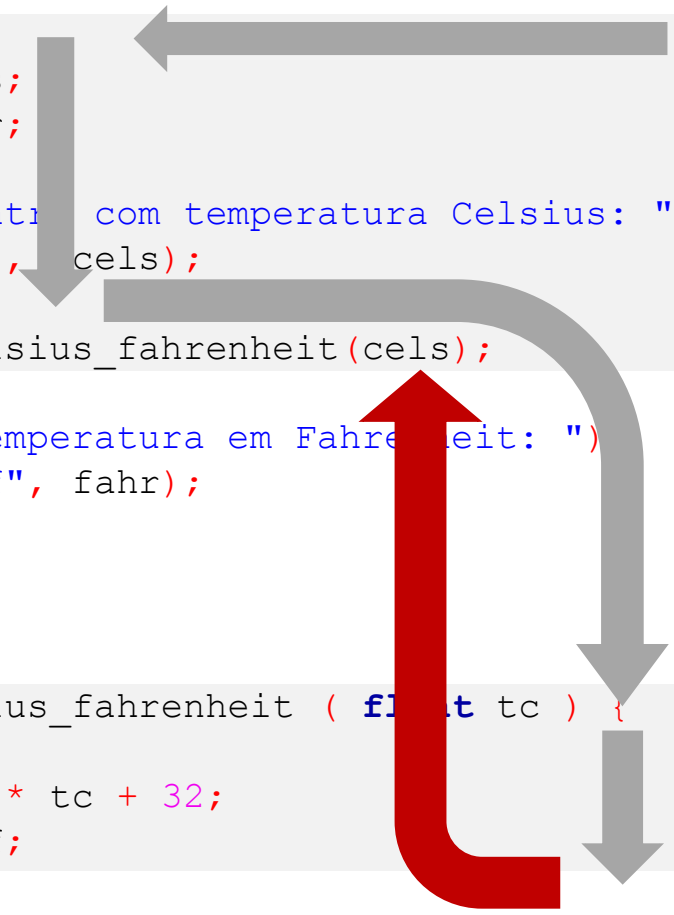
```
1  #include <stdio.h>
2
3  float celsius_fahrenheit ( float tc );
4
5  int main() {
6    float cels;
7    float fahr;
8
9    printf("Entr  com temperatura Celsius: ");
10   scanf("%f",  cels);
11
12   fahr = celsius_fahrenheit(cels);
13
14   printf("Temperatura em Fahrenheit: ")
15   printf("%f", fahr);
16
17   return 0;
18 }
19
20 float celsius_fahrenheit ( float tc ) {
21   float tf;
22   tf = 1.8 * tc + 32;
23   return tf;
24 }
```



No interior da função, o fluxo de execução continua sequencial.

Execução de uma função

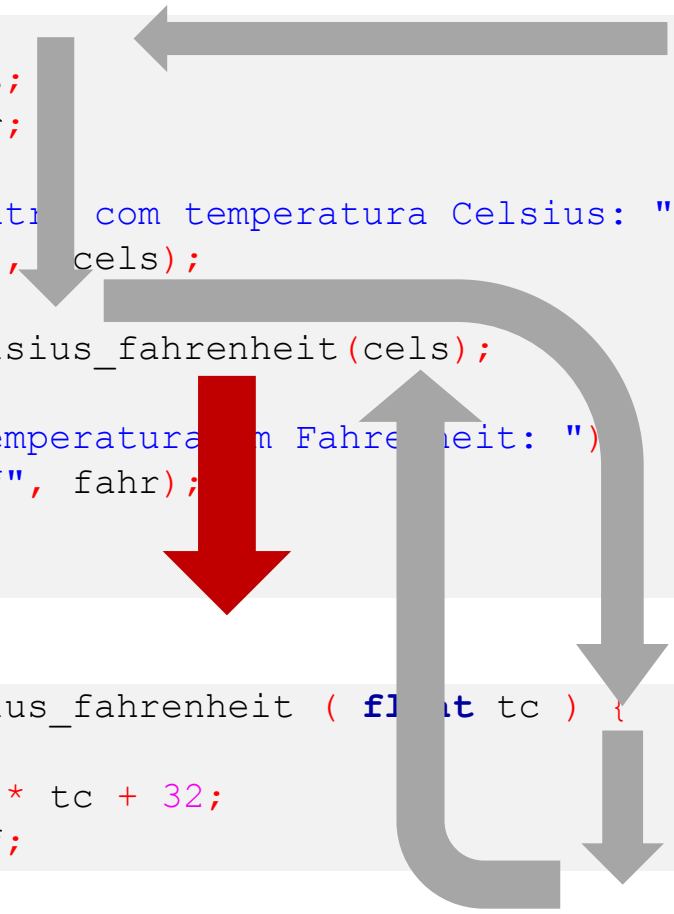
```
1  #include <stdio.h>
2
3  float celsius_fahrenheit ( float tc );
4
5  int main() {
6    float cels;
7    float fahr;
8
9    printf("Entr  com temperatura Celsius: ");
10   scanf("%f",  cels);
11
12   fahr = celsius_fahrenheit(cels);
13
14   printf("Temperatura em Fahre eit: ")
15   printf("%f", fahr);
16
17   return 0;
18 }
19
20 float celsius_fahrenheit ( fl t tc ) {
21   float tf;
22   tf = 1.8 * tc + 32;
23   return tf;
24 }
```



Terminada a execução da função, o fluxo de execução volta para a mesma instrução onde a função foi chamada.

Execução de uma função

```
1  #include <stdio.h>
2
3  float celsius_fahrenheit ( float tc );
4
5  int main() {
6    float cels;
7    float fahr;
8
9    printf("Entr  com temperatura Celsius: ");
10   scanf("%f",  cels);
11
12   fahr = celsius_fahrenheit(cels);
13
14   printf("Temperatura  m Fahre eit: ")
15   printf("%f", fahr);
16
17   return 0;
18 }
19
20 float celsius_fahrenheit ( fl t tc ) {
21   float tf;
22   tf = 1.8 * tc + 32;
23   return tf;
24 }
```



De volta à função, o fluxo de execução continua de forma sequencial até outra chamada ou até o fim da função main.

Teste de mesa

```
1  #include <stdio.h>
2
3  float celsius_fahrenheit ( float tc );
4
5  int main() {
6    float cels;
7    float fahr;
8
9    printf("Entre com temperatura Celsius: ");
10   scanf("%f", &cels);
11
12   fahr = celsius_fahrenheit(cels);
13
14   printf("Temperatura em Fahrenheit: ");
15   printf("%f", fahr);
16
17   return 0;
18 }
19
20 float celsius_fahrenheit ( float tc ) {
21    float tf;
22    tf = 1.8 * tc + 32;
23    return tf;
24 }
```

	main		funcao	
linha	cels	fahr	tc	tf

Teste de mesa

```
1  #include <stdio.h>
2
3  float celsius_fahrenheit ( float tc );
4
5  int main() {
6    float cels;
7    float fahr;
8
9    printf("Entre com temperatura Celsius: ");
10   scanf("%f", &cels);
11
12   fahr = celsius_fahrenheit(cels);
13
14   printf("Temperatura em Fahrenheit: ");
15   printf("%f", fahr);
16
17   return 0;
18 }
19
20 float celsius_fahrenheit ( float tc ) {
21   float tf;
22   tf = 1.8 * tc + 32;
23   return tf;
24 }
```

	main		funcao	
linha	cels	fahr	tc	tf
5	?	?		

Teste de mesa

```
1  #include <stdio.h>
2
3  float celsius_fahrenheit ( float tc );
4
5  int main() {
6    float cels;
7    float fahr;
8
9    printf("Entre com temperatura Celsius: ");
10   scanf("%f", &cels);
11
12   fahr = celsius_fahrenheit(cels);
13
14   printf("Temperatura em Fahrenheit: ");
15   printf("%f", fahr);
16
17   return 0;
18 }
19
20 float celsius_fahrenheit ( float tc ) {
21    float tf;
22    tf = 1.8 * tc + 32;
23    return tf;
24 }
```

	main		funcao	
linha	cels	fahr	tc	tf
5	?	?		

```
Entre com temperatura em Celsius:
```

Teste de mesa

```
1  #include <stdio.h>
2
3  float celsius_fahrenheit ( float tc );
4
5  int main() {
6    float cels;
7    float fahr;
8
9    printf("Entre com temperatura Celsius: ");
10   scanf("%f", &cels);
11
12   fahr = celsius_fahrenheit(cels);
13
14   printf("Temperatura em Fahrenheit: ");
15   printf("%f", fahr);
16
17   return 0;
18 }
19
20 float celsius_fahrenheit ( float tc ) {
21   float tf;
22   tf = 1.8 * tc + 32;
23   return tf;
24 }
```

	main		funcao	
linha	cels	fahr	tc	tf
5	?	?		
10	20.0	?		

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

Teste de mesa

```
1  #include <stdio.h>
2
3  float celsius_fahrenheit ( float tc );
4
5  int main() {
6    float cels;
7    float fahr;
8
9    printf("Entre com temperatura Celsius: ");
10   scanf("%f", &cels);
11
12   fahr = celsius_fahrenheit(cels);
13
14   printf("Temperatura em Fahrenheit: ");
15   printf("%f", fahr);
16
17   return 0;
18 }
19
20 float celsius_fahrenheit ( float tc ) {
21   float tf;
22   tf = 1.8 * tc + 32;
23   return tf;
24 }
```

	main		funcao	
linha	cels	fahr	tc	tf
5	?	?		
10	20.0	?		
12	20.0	?		

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

Teste de mesa

```
1  #include <stdio.h>
2
3  float celsius_fahrenheit ( float tc );
4
5  int main() {
6    float cels;
7    float fahr;
8
9    printf("Entre com temperatura Celsius: ");
10   scanf("%f", &cels);
11
12   fahr = celsius_fahrenheit(cels);
13
14   printf("Temperatura em Fahrenheit: ");
15   printf("%f", fahr);
16
17   return 0;
18 }
19
20 float celsius_fahrenheit ( float tc ) {
21    float tf;
22    tf = 1.8 * tc + 32;
23    return tf;
24 }
```

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

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

Teste de mesa

```
1  #include <stdio.h>
2
3  float celsius_fahrenheit ( float tc );
4
5  int main() {
6    float cels;
7    float fahr;
8
9    printf("Entre com temperatura Celsius: ");
10   scanf("%f", &cels);
11
12   fahr = celsius_fahrenheit(cels);
13
14   printf("Temperatura em Fahrenheit: ");
15   printf("%f", fahr);
16
17   return 0;
18 }
19
20 float celsius_fahrenheit ( float tc ) {
21   float tf;
22   tf = 1.8 * tc + 32;
23   return tf;
24 }
```

	main		funcao	
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
```

Teste de mesa

```
1  #include <stdio.h>
2
3  float celsius_fahrenheit ( float tc );
4
5  int main() {
6    float cels;
7    float fahr;
8
9    printf("Entre com temperatura Celsius: ");
10   scanf("%f", &cels);
11
12   fahr = celsius_fahrenheit(cels);
13
14   printf("Temperatura em Fahrenheit: ");
15   printf("%f", fahr);
16
17   return 0;
18 }
19
20 float celsius_fahrenheit ( float tc ) {
21    float tf;
22    tf = 1.8 * tc + 32;
23    return tf;
24 }
```

	main		funcao	
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
```

Teste de mesa

```
1  #include <stdio.h>
2
3  float celsius_fahrenheit ( float tc );
4
5  int main() {
6    float cels;
7    float fahr;
8
9    printf("Entre com temperatura Celsius: ");
10   scanf("%f", &cels);
11
12   fahr = celsius_fahrenheit(cels);
13
14   printf("Temperatura em Fahrenheit: ");
15   printf("%f", fahr);
16
17   return 0;
18 }
19
20 float celsius_fahrenheit ( float tc ) {
21   float tf;
22   tf = 1.8 * tc + 32;
23   return tf;
24 }
```

	main		funcao	
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
```

Teste de mesa

```
1  #include <stdio.h>
2
3  float celsius_fahrenheit ( float tc );
4
5  int main() {
6    float cels;
7    float fahr;
8
9    printf("Entre com temperatura Celsius: ");
10   scanf("%f", &cels);
11
12   fahr =              68              ;
13
14   printf("Temperatura em Fahrenheit: ");
15   printf("%f", fahr);
16
17   return 0;
18 }
19
20 float celsius_fahrenheit ( float tc ) {
21   float tf;
22   tf = 1.8 * tc + 32;
23   return tf;
24 }
```

	main		funcao	
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
```

Teste de mesa

```
1  #include <stdio.h>
2
3  float celsius_fahrenheit ( float tc );
4
5  int main() {
6    float cels;
7    float fahr;
8
9    printf("Entre com temperatura Celsius: ");
10   scanf("%f", &cels);
11
12   fahr = celsius_fahrenheit(cels);
13
14   printf("Temperatura em Fahrenheit: ");
15   printf("%f", fahr);
16
17   return 0;
18 }
19
20 float celsius_fahrenheit ( float tc ) {
21   float tf;
22   tf = 1.8 * tc + 32;
23   return tf;
24 }
```

	main		funcao	
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
```

Teste de mesa

```
1  #include <stdio.h>
2
3  float celsius_fahrenheit ( float tc );
4
5  int main() {
6    float cels;
7    float fahr;
8
9    printf("Entre com temperatura Celsius: ");
10   scanf("%f", &cels);
11
12   fahr = celsius_fahrenheit(cels);
13
14   printf("Temperatura em Fahrenheit: ");
15   printf("%f", fahr);
16
17   return 0;
18 }
19
20 float celsius_fahrenheit ( float tc ) {
21   float tf;
22   tf = 1.8 * tc + 32;
23   return tf;
24 }
```

	main		funcao	
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:
```

Teste de mesa

```
1  #include <stdio.h>
2
3  float celsius_fahrenheit ( float tc );
4
5  int main() {
6    float cels;
7    float fahr;
8
9    printf("Entre com temperatura Celsius: ");
10   scanf("%f", &cels);
11
12   fahr = celsius_fahrenheit(cels);
13
14   printf("Temperatura em Fahrenheit: ");
15   printf("%f", fahr);
16
17   return 0;
18 }
19
20 float celsius_fahrenheit ( float tc ) {
21   float tf;
22   tf = 1.8 * tc + 32;
23   return tf;
24 }
```

	main		funcao	
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
```

Teste de mesa

```
1  #include <stdio.h>
2
3  float celsius_fahrenheit ( float tc );
4
5  int main() {
6    float cels;
7    float fahr;
8
9    printf("Entre com temperatura Celsius: ");
10   scanf("%f", &cels);
11
12   fahr = celsius_fahrenheit(cels);
13
14   printf("Temperatura em Fahrenheit: ");
15   printf("%f", fahr);
16
17   return 0;
18 }
19
20 float celsius_fahrenheit ( float tc ) {
21   float tf;
22   tf = 1.8 * tc + 32;
23   return tf;
24 }
```

	main		funcao	
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
```

Teste de mesa

```
1  #include <stdio.h>
2
3  float celsius_fahrenheit ( float tc );
4
5  int main() {
6    float cels;
7    float fahr;
8
9    printf("Entre com temperatura Celsius: ");
10   scanf("%f", &cels);
11
12   fahr = celsius_fahrenheit(cels);
13
14   printf("Temperatura em Fahrenheit: ");
15   printf("%f", fahr);
16
17   return 0;
18 }
19
20 float celsius_fahrenheit ( float tc ) {
21   float tf;
22   tf = 1.8 * tc + 32;
23   return tf;
24 }
```

	main		funcao	
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
```

Função sem retorno de valor

- Função que não devolve valor para o programa ou função que a chamou.
- Em algumas linguagens, funções com esta característica são denominadas procedimentos.
- Um procedimento normalmente executa uma ação que não gera dados.

Função sem retorno de valor

```
1  #include <stdio.h>
2
3  void imprimeFormatoHora(int qtdMinutos)
4  {
5    int hora, min;
6    hora = qtdMinutos / 60;
7    min  = qtdMinutos % 60;
8    printf("%02d:%02d",hora,min);
9    return;
10 }
11
12 int main()
13 {
14   int minutos;
15   printf("Entre com os minutos para converter em hora:");
16   scanf("%d",&minutos);
17   printf("%d minutos equivale a ", minutos);
18   imprimeFormatoHora(minutos);
19   return 0;
20 }
```

Função sem retorno de valor

```
1  #include <stdio.h>
2
3  void imprimeFormatoHora(int qtdMinutos)
4  {
5     int hora, min;
6     hora = qtdMinutos / 60;
7     min  = qtdMinutos % 60;
8     printf("%02d:%02d",hora,min);
9     return;
10 }
11
12 int main()
13 {
14    int minutos;
15    printf("Entre com os minutos para converter em hora:");
16    scanf("%d",&minutos);
17    printf("%d minutos equivale a ", minutos);
18    imprimeFormatoHora(minutos);
19    return 0;
20 }
```

Tipo de retorno

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

Função sem retorno de valor

```
1  #include <stdio.h>
2
3  void imprimeFormatoHora(int qtdMinutos)
4  {
5    int hora, min;
6    hora = qtdMinutos / 60;
7    min  = qtdMinutos % 60;
8    printf("%02d:%02d",hora,min);
9    return;
10 }
11
12 int main()
13 {
14   int minutos;
15   printf("Entre com os minutos para converter em hora:");
16   scanf("%d",&minutos);
17   printf("%d minutos equivale a ", minutos);
18   imprimeFormatoHora(minutos);
19   return 0;
20 }
```

Comando de retorno

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

Função sem retorno de valor

```
1  #include <stdio.h>
2
3  void imprimeFormatoHora(int qtdMinutos)
4  {
5    int hora, min;
6    hora = qtdMinutos / 60;
7    min  = qtdMinutos % 60;
8    printf("%02d:%02d",hora,min);
9    return;
10   printf("Esta linha de comando nunca sera executada! ");
11 }
12
13 int main()
14 {
15   int minutos;
16   printf("Entre com os minutos para converter em hora:");
17   scanf("%d",&minutos);
18   printf("%d minutos equivale a ", minutos);
19   imprimeFormatoHora(minutos);
20   return 0;
21 }
```

Comando de retorno

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

Função sem retorno de valor

```
1  #include <stdio.h>
2
3  void imprimeFormatoHora(int qtdMinutos)
4  {
5    int hora, min;
6    hora = qtdMinutos / 60;
7    min  = qtdMinutos % 60;
8    printf("%02d:%02d",hora,min);
9    return;
10 }
11
12 int main()
13 {
14   int minutos;
15   printf("Entre com os minutos para converter em hora:");
16   scanf("%d",&minutos);
17   printf("%d minutos equivale a ", minutos);
18   imprimeFormatoHora(minutos);
19   return 0;
20 }
```

Chamada

A chamada a uma função que não retorna valores normalmente aparece sozinha em uma linha de comando.

Função sem retorno de valor

```
1  #include <stdio.h>
2
3  void imprimeFormatoHora(int qtdMinutos)
4  {
5    int hora, min;
6    hora = qtdMinutos / 60;
7    min  = qtdMinutos % 60;
8    printf("%02d:%02d",hora,min);
9    return;
10 }
11
12 int main()
13 {
14   int minutos;
15   printf("Entre com os minutos:");
16   scanf("%d",&minutos);
17   printf("%d minutos equivale a ", minutos);
18   imprimeFormatoHora(minutos);
19   return 0;
20 }
```

TESTE DE MESA

linha	minutos	hora	min	qtdMinutos

Função sem retorno de valor

```
1  #include <stdio.h>
2
3  void imprimeFormatoHora(int qtdMinutos)
4  {
5    int hora, min;
6    hora = qtdMinutos / 60;
7    min  = qtdMinutos % 60;
8    printf("%02d:%02d",hora,min);
9    return;
10 }
11
12 int main()
13 {
14   int minutos;
15   printf("Entre com os minutos:");
16   scanf("%d",&minutos);
17   printf("%d minutos equivale a ", minutos);
18   imprimeFormatoHora(minutos);
19   return 0;
20 }
```

TESTE DE MESA

linha	minutos	hora	min	qtdMinutos
12	?			

Função sem retorno de valor

```
1  #include <stdio.h>
2
3  void imprimeFormatoHora(int qtdMinutos)
4  {
5    int hora, min;
6    hora = qtdMinutos / 60;
7    min  = qtdMinutos % 60;
8    printf("%02d:%02d",hora,min);
9    return;
10 }
11
12 int main()
13 {
14   int minutos;
15   printf("Entre com os minutos:");
16   scanf("%d",&minutos);
17   printf("%d minutos equivale a ", minutos);
18   imprimeFormatoHora(minutos);
19   return 0;
20 }
```

```
Entre com os minutos:
```

TESTE DE MESA

linha	minutos	hora	min	qtdMinutos
12	?			

Função sem retorno de valor

```
1  #include <stdio.h>
2
3  void imprimeFormatoHora(int qtdMinutos)
4  {
5    int hora, min;
6    hora = qtdMinutos / 60;
7    min  = qtdMinutos % 60;
8    printf("%02d:%02d",hora,min);
9    return;
10 }
11
12 int main()
13 {
14   int minutos;
15   printf("Entre com os minutos:");
16   scanf("%d",&minutos);
17   printf("%d minutos equivale a ", minutos);
18   imprimeFormatoHora(minutos);
19   return 0;
20 }
```

```
Entre com os minutos: 345
```

TESTE DE MESA

linha	minutos	hora	min	qtdMinutos
12	?			
16	345			

Função sem retorno de valor

```
1  #include <stdio.h>
2
3  void imprimeFormatoHora(int qtdMinutos)
4  {
5    int hora, min;
6    hora = qtdMinutos / 60;
7    min  = qtdMinutos % 60;
8    printf("%02d:%02d",hora,min);
9    return;
10 }
11
12 int main()
13 {
14   int minutos;
15   printf("Entre com os minutos:");
16   scanf("%d",&minutos);
17   printf("%d minutos equivale a ", minutos);
18   imprimeFormatoHora(minutos);
19   return 0;
20 }
```

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

TESTE DE MESA

linha	minutos	hora	min	qtdMinutos
12	?			
16	345			

Função sem retorno de valor

```
1  #include <stdio.h>
2
3  void imprimeFormatoHora(int qtdMinutos)
4  {
5    int hora, min;
6    hora = qtdMinutos / 60;
7    min  = qtdMinutos % 60;
8    printf("%02d:%02d",hora,min);
9    return;
10 }
11
12 int main()
13 {
14   int minutos;
15   printf("Entre com os minutos:");
16   scanf("%d",&minutos);
17   printf("%d minutos equivale a ", minutos);
18   imprimeFormatoHora(minutos);
19   return 0;
20 }
```

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

TESTE DE MESA

linha	minutos	hora	min	qtdMinutos
12	?			
16	345			

Função sem retorno de valor

```
1  #include <stdio.h>
2
3  void imprimeFormatoHora(int qtdMinutos)
4  {
5    int hora, min;
6    hora = qtdMinutos / 60;
7    min  = qtdMinutos % 60;
8    printf("%02d:%02d",hora,min);
9    return;
10 }
11
12 int main()
13 {
14   int minutos;
15   printf("Entre com os minutos:");
16   scanf("%d",&minutos);
17   printf("%d minutos equivale a ", minutos);
18   imprimeFormatoHora(minutos);
19   return 0;
20 }
```

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

TESTE DE MESA

linha	minutos	hora	min	qtdMinutos
12	?			
16	345			
3		?	?	345

Função sem retorno de valor

```
1  #include <stdio.h>
2
3  void imprimeFormatoHora(int qtdMinutos)
4  {
5    int hora, min;
6    hora = qtdMinutos / 60;
7    min  = qtdMinutos % 60;
8    printf("%02d:%02d",hora,min);
9    return;
10 }
11
12 int main()
13 {
14   int minutos;
15   printf("Entre com os minutos:");
16   scanf("%d",&minutos);
17   printf("%d minutos equivale a ", minutos);
18   imprimeFormatoHora(minutos);
19   return 0;
20 }
```

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

TESTE DE MESA

linha	minutos	hora	min	qtdMinutos
12	?			
16	345			
3		?	?	345
6		5	?	345

Função sem retorno de valor

```
1  #include <stdio.h>
2
3  void imprimeFormatoHora(int qtdMinutos)
4  {
5    int hora, min;
6    hora = qtdMinutos / 60;
7    min  = qtdMinutos % 60;
8    printf("%02d:%02d",hora,min);
9    return;
10 }
11
12 int main()
13 {
14   int minutos;
15   printf("Entre com os minutos:");
16   scanf("%d",&minutos);
17   printf("%d minutos equivale a ", minutos);
18   imprimeFormatoHora(minutos);
19   return 0;
20 }
```

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

TESTE DE MESA

linha	minutos	hora	min	qtdMinutos
12	?			
16	345			
3		?	?	345
6		5	?	345
7		5	45	345

Função sem retorno de valor

```
1  #include <stdio.h>
2
3  void imprimeFormatoHora(int qtdMinutos)
4  {
5    int hora, min;
6    hora = qtdMinutos / 60;
7    min  = qtdMinutos % 60;
8    printf("%02d:%02d",hora,min);
9    return;
10 }
11
12 int main()
13 {
14   int minutos;
15   printf("Entre com os minutos:");
16   scanf("%d",&minutos);
17   printf("%d minutos equivale a ", minutos);
18   imprimeFormatoHora(minutos);
19   return 0;
20 }
```

```
Entre com os minutos: 345
345 minutos equivale a 05:45
```

TESTE DE MESA

linha	minutos	hora	min	qtdMinutos
12	?			
16	345			
3		?	?	345
6		5	?	345
7		5	45	345

Função sem retorno de valor

```
1  #include <stdio.h>
2
3  void imprimeFormatoHora(int qtdMinutos)
4  {
5    int hora, min;
6    hora = qtdMinutos / 60;
7    min  = qtdMinutos % 60;
8    printf("%02d:%02d",hora,min);
9    return;
10 }
11
12 int main()
13 {
14   int minutos;
15   printf("Entre com os minutos:");
16   scanf("%d",&minutos);
17   printf("%d minutos equivale a ", minutos);
18   imprimeFormatoHora(minutos);
19   return 0;
20 }
```

```
Entre com os minutos: 345
345 minutos equivale a 05:45
```

TESTE DE MESA

linha	minutos	hora	min	qtdMinutos
12	?			
16	345			
3		?	?	345
6		5	?	345
7		5	45	345

Função sem retorno de valor

```
1  #include <stdio.h>
2
3  void imprimeFormatoHora(int qtdMinutos)
4  {
5    int hora, min;
6    hora = qtdMinutos / 60;
7    min  = qtdMinutos % 60;
8    printf("%02d:%02d",hora,min);
9    return;
10 }
11
12 int main()
13 {
14   int minutos;
15   printf("Entre com os minutos:");
16   scanf("%d",&minutos);
17   printf("%d minutos equivale a ", minutos);
18   imprimeFormatoHora(minutos);
19   return 0;
20 }
```

```
Entre com os minutos: 345
345 minutos equivale a 05:45
```

TESTE DE MESA

linha	minutos	hora	min	qtdMinutos
12	?			
16	345			
3		?	?	345
6		5	?	345
7		5	45	345

Função sem retorno de valor

```
1  #include <stdio.h>
2
3  void imprimeFormatoHora(int qtdMinutos)
4  {
5    int hora, min;
6    hora = qtdMinutos / 60;
7    min  = qtdMinutos % 60;
8    printf("%02d:%02d",hora,min);
9    return;
10 }
11
12 int main()
13 {
14   int minutos;
15   printf("Entre com os minutos:");
16   scanf("%d",&minutos);
17   printf("%d minutos equivale a ", minutos);
18   imprimeFormatoHora(minutos);
19   return 0;
20 }
```

```
Entre com os minutos: 345
345 minutos equivale a 05:45
```

TESTE DE MESA

linha	minutos	hora	min	qtdMinutos
12	?			
16	345			
3		?	?	345
6		5	?	345
7		5	45	345

Função sem retorno de valor

```
1  #include <stdio.h>
2
3  void imprimeFormatoHora(int qtdMinutos)
4  {
5    int hora, min;
6    hora = qtdMinutos / 60;
7    min  = qtdMinutos % 60;
8    printf("%02d:%02d",hora,min);
9    return;
10 }
11
12 int main()
13 {
14   int minutos;
15   printf("Entre com os minutos:");
16   scanf("%d",&minutos);
17   printf("%d minutos equivale a ", minutos);
18   imprimeFormatoHora(minutos);
19   return 0;
20 }
```

```
Entre com os minutos: 345
345 minutos equivale a 05:45
```

TESTE DE MESA

linha	minutos	hora	min	qtdMinutos
12	?			
16	345			
3		?	?	345
6		5	?	345
7		5	45	345

Função com retorno de valor

- Função que retorna um único valor para o programa ou função que a chamou.

- Funções com retorno de valor são utilizadas para realizar uma operação e devolver alguma resposta relativa à operação realizada.

Função com retorno de valor

```
1 #include <stdio.h>
2
3 int converteEmMinutos(int numHoras, int numMinutos)
4 {
5   int totalMinutos;
6   totalMinutos = numHoras * 60 + numMinutos;
7   return totalMinutos;
8 }
9
10 int main()
11 {
12   int hora,min,qtdMinutos;
13   printf("Entre com horas e minutos para conversao:");
14   scanf("%d %d",&hora,&min);
15   qtdMinutos = converteEmMinutos(hora,min);
16   printf("%02d:%02d equivale a %d minutos.",
17          hora, min, qtdMinutos);
18   return 0;
19 }
```

Função com retorno de valor

```
1  #include <stdio.h>
2
3  int converteEmMinutos(int numHoras, int numMinutos)
4  {
5    int totalMinutos;
6    totalMinutos = numHoras * 60 + numMinutos;
7    return totalMinutos;
8  }
9
10 int main()
11 {
12   int hora,min,qtdMinutos;
13   printf("Entre com horas e minutos para conversao:");
14   scanf("%d %d",&hora,&min);
15   qtdMinutos = converteEmMinutos(hora,min);
16   printf("%02d:%02d equivale a %d minutos.",
17          hora, min, qtdMinutos);
18   return 0;
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.

Função com retorno de valor

```
1  #include <stdio.h>
2
3  int converteEmMinutos(int numHoras, int numMinutos)
4  {
5    int totalMinutos;
6    totalMinutos = numHoras * 60 + numMinutos;
7    return totalMinutos;
8  }
9
10 int main()
11 {
12   int hora,min,qtdMinutos;
13   printf("Entre com horas e minutos para conversao:");
14   scanf("%d %d",&hora,&min);
15   qtdMinutos = converteEmMinutos(hora,min);
16   printf("%02d:%02d equivale a %d minutos.",
17          hora, min, qtdMinutos);
18   return 0;
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`.

Função com retorno de valor

```
1  #include <stdio.h>
2
3  int converteEmMinutos(int numHoras, int numMinutos)
4  {
5    int totalMinutos;
6    totalMinutos = numHoras * 60 + numMinutos;
7    return totalMinutos;
8  }
9
10 int main()
11 {
12   int hora,min,qtdMinutos;
13   printf("Entre com horas e minutos para conversao:");
14   scanf("%d %d",&hora,&min);
15   qtdMinutos = converteEmMinutos(hora,min);
16   printf("%02d:%02d equivale a %d minutos.",
17          hora, min, qtdMinutos);
18   return 0;
19 }
```

Chamada

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

Função com retorno de valor

```
1  #include <stdio.h>
2
3  int converteEmMinutos(int numHoras, int numMinutos)
4  {
5    int totalMinutos;
6    totalMinutos = numHoras * 60 + numMinutos;
7    return totalMinutos;
8  }
9
10 int main()
11 {
12   int hora,min;
13   printf("Entre com horas e minutos para conversao:");
14   scanf("%d %d",&hora,&min);
15   printf("%02d:%02d equivale a %d minutos.",
16          hora, min, converteEmMinutos(hora,min) );
17   return 0;
18 }
```

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.

Função com retorno de valor

```
1  #include <stdio.h>
2
3  int converteEmMinutos(int numHoras, int numMin)
4  {
5    int totalMinutos;
6    totalMinutos = numHoras * 60 + numMin;
7    return totalMinutos;
8  }
9
10 int main()
11 {
12   int hora,min;
13   printf("Entre com horas e minutos:");
14   scanf("%d %d",&hora,&min);
15   printf("%02d:%02d equivale a %d minutos.",
16      hora, min, converteEmMinutos(hora,min) );
17   return 0;
18 }
```

TESTE DE MESA

linha	hora	min	num Horas	num Min	totalMinutos

Função com retorno de valor

```
1  #include <stdio.h>
2
3  int converteEmMinutos(int numHoras, int numMin)
4  {
5    int totalMinutos;
6    totalMinutos = numHoras * 60 + numMin;
7    return totalMinutos;
8  }
9
10 int main()
11 {
12   int hora,min;
13   printf("Entre com horas e minutos:");
14   scanf("%d %d",&hora,&min);
15   printf("%02d:%02d equivale a %d minutos.",
16      hora, min, converteEmMinutos(hora,min) );
17   return 0;
18 }
```

TESTE DE MESA

linha	hora	min	num Horas	num Min	totalMinutos
10	?	?			

Função com retorno de valor

```
1  #include <stdio.h>
2
3  int converteEmMinutos(int numHoras, int numMin)
4  {
5    int totalMinutos;
6    totalMinutos = numHoras * 60 + numMin;
7    return totalMinutos;
8  }
9
10 int main()
11 {
12   int hora,min;
13   printf("Entre com horas e minutos:");
14   scanf("%d %d",&hora,&min);
15   printf("%02d:%02d equivale a %d minutos.",
16      hora, min, converteEmMinutos(hora,min) );
17   return 0;
18 }
```

```
Entre com horas e minutos:
```

TESTE DE MESA

linha	hora	min	num Horas	num Min	totalMinutos
10	?	?			

Função com retorno de valor

```
1  #include <stdio.h>
2
3  int converteEmMinutos(int numHoras, int numMin)
4  {
5    int totalMinutos;
6    totalMinutos = numHoras * 60 + numMin;
7    return totalMinutos;
8  }
9
10 int main()
11 {
12   int hora,min;
13   printf("Entre com horas e minutos:");
14   scanf("%d %d",&hora,&min);
15   printf("%02d:%02d equivale a %d minutos.",
16      hora, min, converteEmMinutos(hora,min) );
17   return 0;
18 }
```

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

TESTE DE MESA

linha	hora	min	num Horas	num Min	totalMinutos
10	?	?			
14	5	?			

Função com retorno de valor

```
1  #include <stdio.h>
2
3  int converteEmMinutos(int numHoras, int numMin)
4  {
5    int totalMinutos;
6    totalMinutos = numHoras * 60 + numMin;
7    return totalMinutos;
8  }
9
10 int main()
11 {
12   int hora,min;
13   printf("Entre com horas e minutos:");
14   scanf("%d %d",&hora,&min);
15   printf("%02d:%02d equivale a %d minutos.",
16      hora, min, converteEmMinutos(hora,min) );
17   return 0;
18 }
```

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

TESTE DE MESA

linha	hora	min	num Horas	num Min	totalMinutos
10	?	?			
14	5	45			

Função com retorno de valor

```
1  #include <stdio.h>
2
3  int converteEmMinutos(int numHoras, int numMin)
4  {
5    int totalMinutos;
6    totalMinutos = numHoras * 60 + numMin;
7    return totalMinutos;
8  }
9
10 int main()
11 {
12   int hora,min;
13   printf("Entre com horas e minutos:");
14   scanf("%d %d",&hora,&min);
15   printf("%02d:%02d equivale a %d minutos.",
16      hora, min, converteEmMinutos(hora,min) );
17   return 0;
18 }
```

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

TESTE DE MESA

linha	hora	min	num Horas	num Min	totalMinutos
10	?	?			
14	5	45			

Função com retorno de valor

```
1  #include <stdio.h>
2
3  int converteEmMinutos(int numHoras, int numMin)
4  {
5    int totalMinutos;
6    totalMinutos = numHoras * 60 + numMin;
7    return totalMinutos;
8  }
9
10 int main()
11 {
12   int hora,min;
13   printf("Entre com horas e minutos:");
14   scanf("%d %d",&hora,&min);
15   printf("%02d:%02d equivale a %d minutos.",
16      hora, min, converteEmMinutos(hora,min) );
17   return 0;
18 }
```

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

TESTE DE MESA

linha	hora	min	num Horas	num Min	totalMinutos
10	?	?			
14	5	45			
3			5	45	?

Função com retorno de valor

```
1  #include <stdio.h>
2
3  int converteEmMinutos(int numHoras, int numMin)
4  {
5     int totalMinutos;
6     totalMinutos = numHoras * 60 + numMin;
7     return totalMinutos;
8  }
9
10 int main()
11 {
12    int hora,min;
13    printf("Entre com horas e minutos:");
14    scanf("%d %d",&hora,&min);
15    printf("%02d:%02d equivale a %d minutos.",
16       hora, min, converteEmMinutos(hora,min) );
17    return 0;
18 }
```

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

TESTE DE MESA

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

Função com retorno de valor

```
1  #include <stdio.h>
2
3  int converteEmMinutos(int numHoras, int numMin)
4  {
5     int totalMinutos;
6     totalMinutos = numHoras * 60 + numMin;
7     return totalMinutos;
8  }
9
10 int main()
11 {
12    int hora,min;
13    printf("Entre com horas e minutos:");
14    scanf("%d %d",&hora,&min);
15    printf("%02d:%02d equivale a %d minutos.",
16       hora, min, converteEmMinutos(hora,min) );
17    return 0;
18 }
```

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

TESTE DE MESA

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

Função com retorno de valor

```
1  #include <stdio.h>
2
3  int converteEmMinutos(int numHoras, int numMin)
4  {
5    int totalMinutos;
6    totalMinutos = numHoras * 60 + numMin;
7    return totalMinutos;
8  }
9
10 int main()
11 {
12   int hora,min;
13   printf("Entre com horas e minutos:");
14   scanf("%d %d",&hora,&min);
15   printf("%02d:%02d equivale a %d minutos.",
16      hora, min, converteEmMinutos(hora,min) );
17   return 0;
18 }
```

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

TESTE DE MESA

linha	hora	min	num Horas	num Min	totalMinutos
10	?	?			
14	5	45			
3			5	45	?
6			5	45	345
15	5	45			

Função com retorno de valor

```
1  #include <stdio.h>
2
3  int converteEmMinutos(int numHoras, int numMin)
4  {
5    int totalMinutos;
6    totalMinutos = numHoras * 60 + numMin;
7    return totalMinutos;
8  }
9
10 int main()
11 {
12   int hora,min;
13   printf("Entre com horas e minutos:");
14   scanf("%d %d",&hora,&min);
15   printf("%02d:%02d equivale a %d minutos.",
16      hora, min,          345                );
17   return 0;
18 }
```

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

TESTE DE MESA

linha	hora	min	num Horas	num Min	totalMinutos
10	?	?			
14	5	45			
3			5	45	?
6			5	45	345
15	5	45			

Função com retorno de valor

```
1  #include <stdio.h>
2
3  int converteEmMinutos(int numHoras, int numMin)
4  {
5    int totalMinutos;
6    totalMinutos = numHoras * 60 + numMin;
7    return totalMinutos;
8  }
9
10 int main()
11 {
12   int hora,min;
13   printf("Entre com horas e minutos:");
14   scanf("%d %d",&hora,&min);
15   printf("%02d:%02d equivale a %d minutos.",
16      hora, min, converteEmMinutos(hora,min) );
17   return 0;
18 }
```

```
Entre com horas e minutos: 5 45
05:45 equivale a 345 minutos.
```

TESTE DE MESA

linha	hora	min	num Horas	num Min	totalMinutos
10	?	?			
14	5	45			
3			5	45	?
6			5	45	345
15	5	45			

Função com retorno de valor

```
1  #include <stdio.h>
2
3  int converteEmMinutos(int numHoras, int numMin)
4  {
5    int totalMinutos;
6    totalMinutos = numHoras * 60 + numMin;
7    return totalMinutos;
8  }
9
10 int main()
11 {
12   int hora,min;
13   printf("Entre com horas e minutos:");
14   scanf("%d %d",&hora,&min);
15   printf("%02d:%02d equivale a %d minutos.",
16       hora, min, converteEmMinutos(hora,min) );
17   return 0;
18 }
```

```
Entre com horas e minutos: 5 45
05:45 equivale a 345 minutos.
```

TESTE DE MESA

linha	hora	min	num Horas	num Min	totalMinutos
10	?	?			
14	5	45			
3			5	45	?
6			5	45	345
15	5	45			

Função com retorno de valor

```
1  #include <stdio.h>
2
3  int converteEmMinutos(int numHoras, int numMin)
4  {
5    int totalMinutos;
6    totalMinutos = numHoras * 60 + numMin;
7    return totalMinutos;
8  }
9
10 int main()
11 {
12   int hora,min;
13   printf("Entre com horas e minutos:");
14   scanf("%d %d",&hora,&min);
15   printf("%02d:%02d equivale a %d minutos.",
16      hora, min, converteEmMinutos(hora,min) );
17   return 0;
18 }
```

```
Entre com horas e minutos: 5 45
05:45 equivale a 345 minutos.
```

TESTE DE MESA

linha	hora	min	num Horas	num Min	totalMinutos
10	?	?			
14	5	45			
3			5	45	?
6			5	45	345
15	5	45			

Exercícios

2. Faça o teste de mesa do programa abaixo:

```
01 #include <stdio.h>
02 int calculo (int p, int q)
03 {
04   p = p * 10;
05   q = q + 10;
06   return(p + q);
07 }
08 int main()
09 {
10   int x = 2, y = 5;
11   printf("%d %d %d",x,y,calculo(x,y));
12   return 0;
13 }
```

	main		calculo	
linha	x	y	p	q
	?	?	?	?

Não se esqueça de mostrar o valor impresso no final da execução.

Exercícios

3. a) Escreva uma função que recebe dois números inteiros e imprime a soma, o produto, a diferença, o quociente e o resto entre esses dois números.

   b) Faça um programa em C (função principal) que leia dois inteiros do teclado e chame a função da letra a).

   c) Teste seu programa com os valores 11 e 3.

4. a) Elabore uma função que receba três valores reais e retorne a média aritmética destes valores.

   b) Em seguida, faça um programa (função principal) que leia três valores do teclado e imprima sua média, utilizando a função da letra a).

   c) Teste seu programa com os valores 2, 6 e 7.

Vantagem do uso de funções

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 função (chamada) pode ser feita de forma simples;
- o código fica legível, mais fácil de ser entendido e mais compacto;
- o uso de funções possibilita a modularização do código de um programa, isto é, o desenvolvimento do código organizado em módulos funcionais.

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

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 função;
- todos os parâmetros recebidos pela função.

O escopo de uma variável local corresponde apenas ao bloco de comandos de sua função.

Dentro de uma função não se tem acesso a variáveis declaradas em outra função.

Variáveis e escopo

A reserva de memória para uma variável local é condicionada à execução da função:

- A reserva é refeita cada vez que a função é executada (podem ser reservados endereços distintos a cada execução).
- Quando a execução da função 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 função `volume_cilindro`.

```c
#include <stdio.h>

#define PI 3.14159

float volume_cilindro(float raio, float altura)
{
  float volume = PI * raio * raio * altura;
  return volume;
}

int main()
{
  float raio, altura, volume;
  printf("Entre com o valor do raio: ");
  scanf("%f", &raio);
  printf("Entre com o valor da altura: ");
  scanf("%f", &altura);

  volume = volume_cilindro(raio, altura);

  printf("Volume do cilindro = ");
  printf("%f", volume);
  return 0;
}
```

Variáveis e escopo

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

```c
#include <stdio.h>

#define PI 3.14159

float volume_cilindro(float r, float a)
{
  float v = PI * r * r * a;
  return v;
}

int main()
{
  float raio, altura, volume;
  printf("Entre com o valor do raio: ");
  scanf("%f", &raio);
  printf("Entre com o valor da altura: ");
  scanf("%f", &altura);

  volume = volume_cilindro(raio, altura);

  printf("Volume do cilindro = ");
  printf("%f", volume);
  return 0;
}
```

Variáveis e escopo

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

```c
#include <stdio.h>

#define PI 3.14159

float volume_cilindro(float raio, float altura)
{
  float volume = PI * raio * raio * altura;
  return volume;
}

int main()
{
  float raio, altura, volume;
  printf("Entre com o valor do raio: ");
  scanf("%f", &raio);
  printf("Entre com o valor da altura: ");
  scanf("%f", &altura);

  volume = volume_cilindro(raio, altura);

  printf("Volume do cilindro = ");
  printf("%f", volume);
  return 0;
}
```

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.

```c
#include <stdio.h>

#define PI 3.14159

float volume_cilindro(float raio, float altura)
{
  float volume = PI * raio * raio * altura;
  return volume;
}

int main()
{
  float raio, altura, volume;
  printf("Entre com o valor do raio: ");
  scanf("%f", &raio);
  printf("Entre com o valor da altura: ");
  scanf("%f", &altura);

  volume = volume_cilindro(raio, altura);

  printf("Volume do cilindro = ");
  printf("%f", volume);
  return 0;
}
```

Variáveis e escopo

- Existem alguns tipos de variáveis que não serão abordadas neste curso:
  - Variável Global:
    - declarada fora das funções;
    - vive ao longo de toda execução do programa;
    - visível por todas as funções subsequentes.
  - Variável Estática:
    - existe durante toda a execução do programa;
    - só é visível dentro da função que a declara.

Chamadas e parâmetros

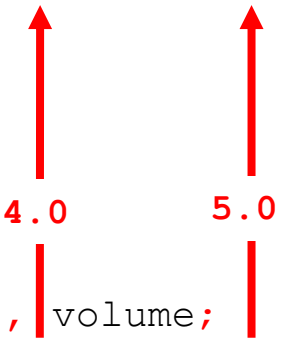
- Quando uma função é chamada, é necessário indicar um valor de entrada para cada um de seus parâmetros.
- Este valor pode ser obtido diretamente de uma constante, de uma variável, de uma expressão ou, até mesmo, do valor de retorno de outra função.

```
volume = volume_cilindro( 4.0, 5.0 );
volume = volume_cilindro( raio, 10.0 / 2 );
volume = volume_cilindro( diametro / 2, altura );
volume = volume_cilindro( raio, sqrt( 9 ) + 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 função e utilizado na inicialização do parâmetro.

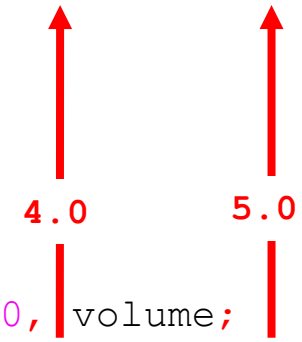
```
#include <stdio.h>
#define PI 3.14159
float volume_cilindro (float r, float a)
{
  float v = PI * r * r * a;
  return v;
}
int main()
{
  float raio=4.0, altura=5.0, volume;
  volume = volume_cilindro(raio, altura);
  printf("Volume do cilindro: %f", volume);
  return 0;
}
```



Chamadas e parâmetros

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

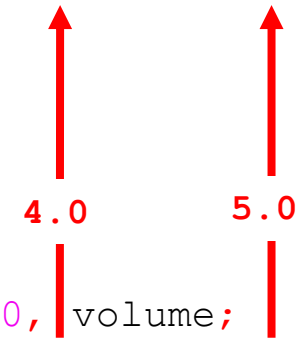
```
#include <stdio.h>
#define PI 3.14159
float volume_cilindro (float r, float a)
{
  float v = PI * r * r * a;
  return v;
}
int main()
{
  float raio=4.0, altura=5.0, volume;
  volume = volume_cilindro(raio, altura);
  printf("Volume do cilindro: %f", volume);
  return 0;
}
```



Chamadas e parâmetros

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

```
#include <stdio.h>
#define PI 3.14159
float volume_cilindro (float r, float a)
{
  float v = PI * r * r * a;
  return v;
}
int main()
{
  float raio=4.0, altura=5.0, volume;
  volume = volume_cilindro(raio, altura);
  printf("Volume do cilindro: %f", volume);
  return 0;
}
```



Chamadas e parâmetros

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

```
int main()
{
  int minutos;
  printf("Entre com os minutos para converter em hora:");
  scanf("%d",&minutos);
  printf("%d minutos equivale a ", minutos);
  imprimeFormatoHora(minutos);
  return 0;
}
```

Chamadas e parâmetros

- O endereço de uma variável é obtido com o uso de **&** seguido pelo nome da variável.
- 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 função.

```
int main()
{
  int minutos;
  printf("Entre com os minutos para converter em hora:");
  scanf("%d",&minutos);
  printf("%d minutos equivale a ", minutos);
  imprimeFormatoHora(minutos);
  return 0;
}
```

Chamadas e parâmetros

No exemplo abaixo:

1) O endereço da variável **minutos** é passado como parâmetro para a função **scanf**.

```
int main()
{
  int minutos;
  printf("Entre com os minutos para converter em hora:");
  scanf("%d",&minutos);
  printf("%d minutos equivale a ", minutos);
  imprimeFormatoHora(minutos);
  return 0;
}
```

Chamadas e parâmetros

No exemplo abaixo:

2) A função **scanf** lê um valor do teclado e o armazena no endereço passado para o parâmetro (no endereço de **minutos**). Mesmo sem fazer parte da definição de **scanf**, a variável **minutos** foi alterada.

```
int main()
{
  int minutos;
  printf("Entre com os minutos para converter em hora:");
  scanf("%d",&minutos);
  printf("%d minutos equivale a ", minutos);
  imprimeFormatoHora(minutos);
  return 0;
}
```

Chamadas e parâmetros

No exemplo abaixo:

3) Após o término de **scanf**, a variável **minutos** conterá o valor digitado durante a execução de **scanf**.

```
int main()
{
  int minutos;
  printf("Entre com os minutos para converter em hora:");
  scanf("%d",&minutos);
  printf("%d minutos equivale a ", minutos);
  imprimeFormatoHora(minutos);
  return 0;
}
```

Chamadas e parâmetros

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

Exercícios

5. O identificador dos parâmetros do exercício 2 foi modificado no código abaixo.

```
01 #include <stdio.h>
02 int calculo (int x, int y)
03 {
04   x = x * 10;
05   y = y + 10;
06   return(x + y);
07 }
08 int main()
09 {
10   int x = 2, y = 5;
11   printf("%d %d %d",x,y,calculo(x,y));
12   return 0;
13 }
```

Como fica o teste de mesa após esta modificação?

Exercícios

6. Considerando a fórmula para o cálculo da distância entre dois pontos (x1, y1) e (x2, y2):

a) Escreva uma função que receba como parâmetros as coordenadas de dois pontos e retorne a distância entre eles.

b) Escreva um programa em C (função principal) que capture do teclado as coordenadas dos 3 vértices de um triângulo, calcule e imprima o perímetro deste triângulo, chamando a função anterior.

c) Teste seu programa, simulando sua execução com as seguintes coordenadas: (4,1), (1,1), (4,5).

Exercícios

7. a) Considerando Pi = 3,14159, para cada opção abaixo, escreva uma função que recebe como parâmetro o raio de um círculo e:

- retorne seu diâmetro;
- retorne sua circunferência;
- retorne sua área;
- imprima o diâmetro, a circunferência e a área chamando as funções anteriores.

b) Elabore um programa que leia do teclado o valor do raio de dois círculos e, para cada círculo, chame a função que imprime as informações.

c) Teste seu programa com os valores 1 e 3.

Exercícios

DESAFIO: Faça um programa para calcular quantas latas de verniz serão necessárias para cobrir um deque de madeira. O usuário do programa informará a largura e o comprimento da superfície a ser coberta e o programa deverá imprimir o número de latas necessárias (valor inteiro), dado que cada lata de verniz cobre até 3 m^2 de superfície. O programa deverá ter no mínimo 3 funções. Teste o programa calculando o necessário para cobrir uma superfície de 4.5 x 5m.

Observação: tente identificar as tarefas que poderão constituir diferentes funções e, para cada tarefa, especifique os dados de entrada (parâmetros) necessários para sua execução e defina se esta tarefa produzirá ou não um resultado (retorno).

Funções

DCC 120

Revisão

- A forma geral de uma função é:

```
tipoDeRetorno nomeDaFuncao ( listaParametros )
{
   corpo da funcao;
   return xxx;
}
```

Exemplos de funções

- Função sem retorno de valor

```
void imprimeSoma (int a, int b)
{
  printf("%d", a+b);
}
```

- Função com retorno de valor

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

Exemplos de funções

```c
#include ...

int Func1(int a)//definição
{
   ...
   return val;
}

int main()
{
   ...

   x=Func1(10);   //chamada

   ...
}
```

```c
#include....

int Func1(int a); //declaração

int main()
{
   ...
   x=Func1(10);    //chamada
   ...
}

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

Exemplo de função sem retorno

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

O cálculo e a impressão da média devem ser feitos por uma função.

```
void calculaMedia(float nota1, float nota2, float nota3)
{
  float media;
  media = (nota1 + nota2 + nota3) / 3;
  printf("Media = %.2f\n", media);
}
int main()
{
  float n1, n2, n3;
  printf("Digite as 3 notas: ");
  scanf("%f %f %f", &n1, &n2, &n3);
  calculaMedia (n1, n2, n3);
  return 0;
}
```

Exemplo de função com retorno

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

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

int main ()
{
  int numero, res;
  printf("Digite um numero inteiro: ");
  scanf("%d", &numero);

  res = quadrado(numero);
  printf("O quadrado de %d eh %d.\n", numero, res);

  return 0;
}
```

Exercícios

Para cada exercício:

- Leia atentamente o enunciado até que o problema seja completamente entendido;
- Enumere os passos necessários para a solução do problema;
- “Traduza” os passos listados para a linguagem de programação C;
- Compile e corrija eventuais erros de sintaxe;
- Teste seu programa com diferentes entradas.

Exercícios

1) Faça uma função que receba como parâmetros o valor de uma compra e o número de parcelas e imprima o valor da parcela a ser paga a prazo.
Ao ser executada em um programa com as entradas 3530.8 e 14, sua função deverá imprimir:

```
COMPRA A PRAZO
Valor da compra: R$ 3530.8
Numero de parcelas: 14
Valor da parcela a prazo: R$ 252.20
```

2) Faça uma função que receba por parâmetro um tempo expresso em segundos e imprima na tela esse mesmo tempo em horas, minutos e segundos.
Elabore também um programa para ler este valor do teclado e chamar a função.
Para a entrada 13579, o programa deverá imprimir:

```
CONVERSAO DE SEGUNDOS EM HORAS, MINUTOS E SEGUNDOS
Tempo total em segundos: 13579
Equivale a: 3 horas, 46 minutos e 19 segundos
```

Exercícios

3) 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:

media = (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 na terceira prova para que seja aprovado.

b) Escreva um programa que leia do teclado as duas primeiras notas de um aluno, chame a função do item anterior e imprima a nota mínima que o aluno precisa tirar na p3 para que seja aprovado.

```
CALCULO DE NOTA PARA APROVACAO
Nota na 1a prova: 5.5
Nota na 2a prova: 3.5
Nota necessaria:  5.50
```

Exercícios

DESAFIO: A empresa CDA produz caixas-d'água para casas, prédios, indústrias, etc. Por atender clientes com características bastante distintas, a CDA constrói as caixas d'água sob encomenda, permitindo que cada cliente especifique as medidas desejadas (largura, altura e profundidade) conforme sua necessidade. Há algum tempo, os diretores da empresa notaram que alguns funcionários têm dificuldade na hora de calcular o preço do produto quando um orçamento é solicitado. Por isso, contrataram você para construir um programa que calcule o preço de uma caixa d'água (sem tampa) dadas as suas dimensões.

O preço varia de acordo com a espessura da "parede" da caixa-d'água, que, por sua vez, varia com o volume de água armazenado:

- Se a caixa-d'água armazena menos que 1 metro cúbico de água, precisa ter a espessura mínima de 0.8 cm.
- A cada 1 metro cúbico (ou fração) adicional, a espessura precisa ser aumentada em 0.3 cm.
- O preço de uma superfície de 1 metro quadrado corresponde a R$45,00 vezes a espessura (em cm).

Exercícios

a) Construa uma função para calcular o volume de água que a caixa encomendada pode armazenar. A função deve receber como parâmetros as medidas desejadas e retornar o volume.

b) Construa uma função que receba como parâmetro o volume de água e calcule e retorne a espessura mínima que a caixa deve ter. Você pode usar a função `ceil` da biblioteca math.h para arredondar para cima o valor do volume.

c) Escreva uma função que recebe como parâmetros duas medidas de comprimento e a espessura e retorne o preço da respectiva superfície.

d) Escreva uma função que recebe como parâmetros as três medidas desejadas e retorna o preço da caixa d'água. Sua função vai precisar chamar as funções das letras a, b e c.

e) Escreva um programa que leia as medidas desejadas pelo cliente e imprima o preço da caixa d'água.

```
PRECO DA CAIXA D'AGUA
Largura: 0.8
Altura: 1
Profundidade: 1.15
Valor da caixa d'agua:  R$ 173.52
```