

# Condicionais

---

DCC 119 – Algoritmos



# Para começar a aula de hoje...

**SE** você fez os exercícios 5 **E** 6 da aula passada  
 “Parabéns! Seu esforço valerá a pena.”

## **SENÃO**

**SE** você fez apenas um **OU** tentou fazer os dois  
 “Continue se esforçando e procure ajuda  
 do professor ou dos monitores, se precisar.”

## **SENÃO**

“É muito importante que você estude,  
 tente fazer os exercícios sozinho e  
 não deixe a matéria acumular...”

# Para começar a aula de hoje...

**SE** você fez os exercícios 5 **E** 6 da aula passada  
 “Parabéns! Seu esforço valerá a pena.”

**SENÃO**

**SE** você fez apenas um **OU** tentou fazer os dois  
 “Continue se esforçando!”  
 do professor ou da professora

**SENÃO**

“É muito importante  
 tente fazer os exercícios  
 não deixe a matéria

Esse “recado” ilustra bem a  
 aula de hoje:

**CONDIÇÕES** permitem que  
 um mesmo programa se  
 comporte de forma distinta  
 em situações distintas.

# Para começar a aula de hoje...

**SE** você fez os exercícios 5 **E** 6 da aula passada  
 “Parabéns! Seu esforço valerá a pena.”

**SENÃO**

**SE** você fez apenas um **OU** tentou fazer os dois  
 “Continue se esforçando”  
 do professor ou c

Esse “recado” ilustra bem a aula de hoje:

**SENÃO**

“É muito importante que você tente fazer os exercícios. Se não, não deixe a matéria passar.”

***EXPRESSÕES LÓGICAS***

permitem que as diferentes situações sejam identificadas.

# Expressões lógicas

- De modo geral, as expressões lógicas são usadas para verificar se:
  - o **valor** de uma ou mais variáveis satisfaz uma **determinada condição** durante a **execução** do programa.

# Expressões lógicas

O resultado de uma expressão lógica é sempre VERDADEIRO ou FALSO:

*A nota do aluno é menor que 60?*

Quando a nota do aluno é 75, `nota < 60` é falso.

Quando a nota é 39, `nota < 60` é verdadeiro.

O resultado da expressão lógica  
`nota < 60`,  
 depende do valor da variável `nota`  
 durante a execução.

# Expressões lógicas

Vale lembrar que:

- Na linguagem C, não existe o tipo de dados booleano (verdadeiro ou falso).
- O valor **zero** é interpretado como **falso** e qualquer valor **diferente de zero** é considerado **verdadeiro**.
- Assim, se o resultado de uma comparação for falso, produz-se o valor 0, caso contrário, produz-se o valor 1.

# Expressões lógicas

Geralmente, expressões lógicas são compostas por:

- valores numéricos (constantes, variáveis, etc),
- operadores relacionais e
- operadores lógicos.



# Operadores relacionais

- Os operadores relacionais em C são:
  - < menor que
  - > maior que
  - <= menor ou igual que
  - >= maior ou igual que
  - == igual a
  - != diferente de
- Estes operadores comparam dois valores.
- O resultado produzido por um operador relacional é verdadeiro (1) ou falso (0).

# Operadores relacionais

Assumindo que....	qual o resultado de... ?
-------------------	--------------------------

---

idade=17	idade < 18
----------	------------

---

nota1=95	nota1 >= 60
----------	-------------

---

a=1, b=2, c=1	(b*b-4*a*c) > 0
---------------	-----------------

---

x=-4, y=2	abs(x) >= abs(y)
-----------	------------------

---

n1=7	n1%2 != 0
------	-----------

---

denominador=1	denominador != 0
---------------	------------------

---

hora=10, fimExp=18	hora == fimExp
--------------------	----------------

---

sexo='F'	sexo == 'F'
----------	-------------

---

# Operadores lógicos

- Os operadores lógicos combinam expressões lógicas (ou booleanas).
- Operadores:
  - &&** operador binário E (AND)
  - ||** operador binário OU (OR)
  - !** operador unário de NEGAÇÃO (NOT)
- Expressões compostas por && ou || são avaliadas da esquerda para a direita.

# Operadores lógicos

Assumindo que.... qual o resultado de... ?

---

`idade=17`

`idade >= 18 && idade < 60`

`idade < 18 || idade >= 60`

---

`nota=95,faltas=5`

`nota >= 60 && faltas < 7`

`nota < 60 || faltas >= 7`

---

`a=1,b=2,c=1`

`(b*b-4*a*c) > 0 && a != 0`

---

`x=-4`

`abs(x) > 1 && abs(x) <=2`

---

`idade=18,sexo='F'`

`idade == 18 && sexo == 'M'`

`idade != 18 || sexo == 'F'`

---

# Operadores lógicos

Assumindo que....

qual o resultado  
de... ?

---

```
n1=7, impar=(n1%2)
```

```
!impar
```

---

```
idade=17
```

```
meia=(idade < 18 || idade >= 60)
```

```
!meia
```

---

```
nota=95, faltas=5
```

```
ri=(faltas > 7)
```

```
aprovado=(nota >= 60 && !ri)
```

```
!aprovado
```

---

```
idade=18, sexo='F'
```

```
alistar=(idade == 18 && sexo == 'M') !alistar
```

---

# Operadores relacionais e lógicos

Tabela Verdade para operadores lógicos.

<b>a</b>	<b>b</b>	<b>a &amp;&amp; b</b>	<b>a    b</b>
V	V	V	V
V	F	F	V
F	V	F	V
F	F	F	F

<b>a</b>	<b>! a</b>
V	F
F	V

# Operadores relacionais e lógicos

Uma expressão com o operador **E** é **FALSA** se ao menos um operando é **FALSO**.

a	b	a && b	a    b
V	V	V	V
V	<b>F</b>	<b>F</b>	V
<b>F</b>	V	<b>F</b>	V
<b>F</b>	<b>F</b>	<b>F</b>	F

a	! a
V	F
F	V

# Operadores relacionais e lógicos

Uma expressão com **OU** é **VERDADEIRA** se ao menos um operando é **VERDADEIRO**.

a	b	a && b	a    b
V	V	V	V
V	F	F	V
F	V	F	V
F	F	F	F

a	! a
V	F
F	V



# Operadores relacionais e lógicos

- Esses operadores são normalmente utilizados para tomada de decisões.
- Podem ser utilizados para atribuir valores **lógicos** a variáveis.
- Exemplo:

```
int a, b, c;
int duasRaizesReais;
int umaRaizReal;
scanf ("%d%d%d", &a, &b, &c);
duasRaizesReais = (b*b-4*a*c) > 0;
umaRaizReal = (b*b-4*a*c) == 0;
```

As variáveis `umaRaizReal` e `duasRaizesReais` receberão FALSO (valor 0) ou VERDADEIRO (valor 1), dependendo do resultado de  $(b*b-4*a*c)$ .

# Combinando operadores relacionais

- Operadores relacionais precisam de operadores lógicos para serem combinados.
- Um erro comum é utilizar operadores relacionais em sequência, sem o uso de operadores lógicos.

# Combinando operadores relacionais

Suponha que a variável `numero` tem valor -2.  
Observe a avaliação das expressões abaixo:

$$\begin{array}{l}
 \underbrace{0 \leq \text{numero} < 10} \\
 \text{FALSO} < 10 \\
 \underbrace{\phantom{\text{FALSO}}} \\
 0 < 10 \\
 \underbrace{\phantom{0}} \\
 \text{VERDADEIRO}
 \end{array}$$

$$\begin{array}{l}
 \underbrace{0 \leq \text{numero} \ \&\& \ \text{numero} < 10} \\
 \text{FALSO} \ \&\& \ \text{numero} < 10 \\
 \underbrace{\phantom{\text{FALSO}}} \\
 0 \ \&\& \ \text{numero} < 10 \\
 \phantom{0} \ \&\& \ \underbrace{\phantom{\text{numero} < 10}} \\
 0 \ \&\& \ 1 \\
 \underbrace{\phantom{0 \ \&\&}} \\
 \text{FALSO}
 \end{array}$$

# Combinando operadores relacionais

## ERRADO

`0 <= numero < 10`

`a == b == c`

`x != y != z`

## CORRETO

`0 <= numero && numero < 10`

`a == b && b == c`

`x != y && y != z && x != z`

# Combinando operadores lógicos

- Operadores lógicos podem ser combinados.
- Expressões compostas por apenas um tipo de operador lógico podem ser escritas normalmente.

```
idade <= 12 || idade >= 65 || estudante
```

```
larg > 10 && comp > 10 && altura > 3
```

- Mas...

# Combinando operadores lógicos

**ATENÇÃO:** Operadores lógicos têm precedências diferentes!

- Sem conhecer sua precedência, o uso de parênteses é necessário.

```
( timeA < timeB && timeA < timeC ) || timeA == 0
```

```
( ! brasileiro ) || idade < 16
```

# Exercício



1) Indique quais variáveis recebem valor 0, assumindo que `diaSemana` tem valor 2 e `hora` tem valor 10? Tente identificar as diferenças sutis entre as expressões.

```
ocupado1 = (diaSemana == 2 || diaSemana == 5)
           && (hora >= 14 && hora < 16)
```

```
ocupado2 = diaSemana == 2 || (diaSemana == 5
                               && (hora >= 14 && hora < 16))
```

```
ocupado3 = (diaSemana == 2 || (diaSemana == 5
                               && hora >= 14)) && hora < 16
```

```
ocupado4 = ((diaSemana == 2 || diaSemana == 5)
            && hora >= 14) && hora < 16
```

**DESAFIO:** Elabore uma expressão que indique que você está ocupado nas 2<sup>as</sup> até às 16h, nas 5<sup>as</sup> o dia inteiro e nos outros dias após às 14h.

# Condicionais

Uma ***EXPRESSÃO LÓGICA*** sempre resulta em:  
**VERDADEIRO** ou **FALSO**

Uma ***ESTRUTURA CONDICIONAL*** (ou alternativa dupla) permite que se execute:  
 um conjunto de ações **quando** o resultado for **VERDADEIRO** ou um outro conjunto de ações **quando** ele for **FALSO**



# Condicionais

```
SE nota < 60,
    Imprima "aluno reprovado"
SENÃO,
    Imprima "aluno aprovado"
```

Uma **ESTRUTURA CONDICIONAL** (ou alternativa dupla) permite que se execute:

um conjunto de ações  
quando o resultado for  
VERDADEIRO

ou

um outro conjunto de  
ações quando ele for  
FALSO

# Condicionais

```
SE nota < 60,
    Imprima "aluno reprovado"
SENÃO,
    Imprima "aluno aprovado"
```

Suponha que a variável `nota` tem valor 39.

Neste caso, o algoritmo acima vai imprimir ***SOMENTE*** o texto `"aluno reprovado"`

# Condicionais

```
SE nota < 60,  
    Imprima "aluno reprovado"  
SENÃO,  
    Imprima "aluno aprovado"
```

Suponha que a variável `nota` tem valor 75.

Neste caso, o algoritmo acima vai imprimir ***SOMENTE*** o texto `"aluno aprovado"`

# Sintaxe

```

if (condicao)
{
    bloco de comandos1;
}
else
{
    bloco de comandos2;
}

```

# Exemplo 1

```
int valor1, valor2, maior;
printf ("\nDigite dois valores: ");
scanf ("%d%d", &valor1, &valor2);
if (valor1 > valor2)
{
    maior = valor1;
}
else
{
    maior = valor2;
}
printf ("\nMAIOR = %d", maior);
```

O que será impresso pelo trecho de código acima se o usuário digitar 7 e 9?

# Exemplo 2

```
int quantidade;
float precoUnitario, preco;
scanf ("%d%f", &quantidade, &precoUnitario);
preco = quantidade * precoUnitario;
if (quantidade > 10 && precoUnitario > 50.0 )
{
    preco = preco * 0.85;
    printf ("Ganhou um desconto de 15%%!\n");
}
else
{
    preco = preco * 0.95;
    printf ("Toda a loja com 5%% de desconto!\n");
}
printf ("\nPreco final: %f", preco);
```

O que será impresso se o usuário digitar 10 e 100.0?

# Resolvendo um exercício...

Construa um algoritmo para ler os coeficientes  $A$ ,  $B$  e  $C$  de uma equação do segundo grau e:

- **se delta for maior ou igual a zero:**  
calcular e imprimir as raízes da equação.
- **caso contrário** (delta negativo):  
imprimir a mensagem “Não há solução real”.

# Resolução – Passo 1/5

## ***Enunciado está entendido?***

- Para resolver o problema é necessário conhecer a fórmula de Bhaskara.  
Esta equação tem a seguinte forma:

$$x = \frac{-b \pm \sqrt{\Delta}}{2a}$$

onde

$$\Delta = b^2 - 4ac$$



# Resolução – Passo 2/5

## *Quais variáveis serão necessárias?*

- **a**, **b**, **c** e **delta** serão as variáveis reais que comporão a fórmula da equação acima.
- Serão necessárias mais duas variáveis reais **x1** e **x2** que serão as raízes da equação.

# Resolução – Passo 3/5

## ***Como resolver o problema proposto?***

1. Criar as variáveis do programa
2. Ler o valor das variáveis
3. Calcular o valor do delta
4. Se delta maior ou igual a zero:
  - 4.1. Calcular o valor das raízes reais
  - 4.2. Imprimir o valor das raízes reais
5. Senão
  - 5.1. Avisar o usuário que não há raízes reais

# Resolução – Passo 3/5

## *Como resolver o problema proposto?*

1. Criar as variáveis do programa
2. Ler o valor das variáveis
3. Calcular o valor do delta
4. **Se** delta maior ou igual a zero:
  - 4.1. Calcular o valor das raízes reais
  - 4.2. Imprimir o valor das raízes reais
5. **Senão**
  - 5.1. Avisar o usuário que não há raízes reais

```

if (condicao)
{
    comandos1;
}
else
{
    comando2;
}
  
```

# Resolução – Passo 4/5

```

#include <stdio.h>
#include <math.h>
int main()
{
    float a, b, c, delta, x1, x2;
1 → printf("Digite os coeficientes da equacao (A B C): ");
2 → scanf("%f %f %f", &a, &b, &c);
3 → delta = (b*b - 4*a*c);

4 → if (delta >= 0)
    {
4.1 → |   x1 = (-b + sqrt(delta)) / (2*a);
      |   x2 = (-b - sqrt(delta)) / (2*a);
4.2 → |   printf("Raizes da equacao: %f, %f", x1, x2);
    }
5 → else
    {
5.1 → |   printf ("Nao existem raizes reais");
    }
    return 0;
}

```

# Resolução – Passo 5/5

## ***Testar o algoritmo***

Por exemplo, faça o teste de mesa para a seguinte entrada

- $a = 1$
- $b = 5$
- $c = 4$

Saída esperada: -1 e -4.

```

1  #include <stdio.h>
2  #include <math.h>
3  int main()
4  {
5      float a, b, c, delta, x1, x2;
6      printf("Digite os coeficientes (A B C)");
7      scanf("%f %f %f", &a, &b, &c);
8      delta = (b*b - 4*a*c);
9
10     if (delta >= 0)
11     {
12         x1 = (-b + sqrt(delta)) / (2*a);
13         x2 = (-b - sqrt(delta)) / (2*a);
14         printf("Raizes: %f, %f", x1, x2);
15     }
16     else
17     {
18         printf ("Nao existem raizes reais");
19     }
20     return 0;
21 }

```

## TESTE DE MESA

linha	a	b	c	delta	x1	x2	Condição
3	?	?	?	?	?	?	

## TESTE DE MESA

```

1  #include <stdio.h>
2  #include <math.h>
3  int main()
4  {
5      float a, b, c, delta, x1, x2;
6      printf("Digite os coeficientes (A B C)");
7      scanf("%f %f %f", &a, &b, &c);
8      delta = (b*b - 4*a*c);
9
10     if (delta >= 0)
11     {
12         x1 = (-b + sqrt(delta)) / (2*a);
13         x2 = (-b - sqrt(delta)) / (2*a);
14         printf("Raizes: %f, %f", x1, x2);
15     }
16     else
17     {
18         printf ("Nao existem raizes reais");
19     }
20     return 0;
21 }

```

linha	a	b	c	delta	x1	x2	Condição
3	?	?	?	?	?	?	

[illegible]

Digite os coeficientes (A B C): 1 5 4



**Digite os coeficientes (A B C): 1 5 4**

41

**Digite os coeficientes (A B C): 1 5 4**

42

## TESTE DE MESA

```

1  #include <stdio.h>
2  #include <math.h>
3  int main()
4  {
5      float a, b, c, delta, x1, x2;
6      printf("Digite os coeficientes (A B C)
7      scanf("%f %f %f", &a, &b, &c);
8      delta = (b*b - 4*a*c);
9
10     if (delta >= 0)
11     {
12         x1 = (-b + sqrt(delta)) / (2*a);
13         x2 = (-b - sqrt(delta)) / (2*a);
14         printf("Raizes: %f, %f", x1, x2);
15     }
16     else
17     {
18         printf ("Nao existem raizes reais");
19     }
20     return 0;
21 }

```

Digite os coeficientes (A B C): 1 5 4

linha	a	b	c	delta	x1	x2	Condição
3	?	?	?	?	?	?	
7	1	5	4	?	?	?	
8	1	5	4	9	?	?	
10	1	5	4	9	?	?	V
12	1	5	4	9	-1	?	

## TESTE DE MESA

```

1  #include <stdio.h>
2  #include <math.h>
3  int main()
4  {
5      float a, b, c, delta, x1, x2;
6      printf("Digite os coeficientes (A B C)
7      scanf("%f %f %f", &a, &b, &c);
8      delta = (b*b - 4*a*c);
9
10     if (delta >= 0)
11     {
12         x1 = (-b + sqrt(delta)) / (2*a);
13         x2 = (-b - sqrt(delta)) / (2*a);
14         printf("Raizes: %f, %f", x1, x2);
15     }
16     else
17     {
18         printf ("Nao existem raizes reais");
19     }
20     return 0;
21 }

```

Digite os coeficientes (A B C): 1 5 4

linha	a	b	c	delta	x1	x2	Condição
3	?	?	?	?	?	?	
7	1	5	4	?	?	?	
8	1	5	4	9	?	?	
10	1	5	4	9	?	?	V
12	1	5	4	9	-1	?	
13	1	5	4	9	-1	-4	

## TESTE DE MESA

```

1  #include <stdio.h>
2  #include <math.h>
3  int main()
4  {
5      float a, b, c, delta, x1, x2;
6      printf("Digite os coeficientes (A B C)
7      scanf("%f %f %f", &a, &b, &c);
8      delta = (b*b - 4*a*c);
9
10     if (delta >= 0)
11     {
12         x1 = (-b + sqrt(delta)) / (2*a);
13         x2 = (-b - sqrt(delta)) / (2*a);
14         printf("Raizes: %f, %f", x1, x2);
15     }
16     else
17     {
18         printf ("Nao existem raizes reais");
19     }
20     return 0;
21 }

```

linha	a	b	c	delta	x1	x2	Condição
3	?	?	?	?	?	?	
7	1	5	4	?	?	?	
8	1	5	4	9	?	?	
10	1	5	4	9	?	?	V
12	1	5	4	9	-1	?	
13	1	5	4	9	-1	-4	
14	1	5	4	9	-1	-4	

Digite os coeficientes (A B C): 1 5 4  
Raizes: -1.000000, -4.000000

## TESTE DE MESA

```

1  #include <stdio.h>
2  #include <math.h>
3  int main()
4  {
5      float a, b, c, delta, x1, x2;
6      printf("Digite os coeficientes (A B C)");
7      scanf("%f %f %f", &a, &b, &c);
8      delta = (b*b - 4*a*c);
9
10     if (delta >= 0)
11     {
12         x1 = (-b + sqrt(delta)) / (2*a);
13         x2 = (-b - sqrt(delta)) / (2*a);
14         printf("Raizes: %f, %f", x1, x2);
15     }
16     else
17     {
18         printf ("Nao existem raizes reais");
19     }
20     return 0;
21 }

```

linha	a	b	c	delta	x1	x2	Condição
3	?	?	?	?	?	?	
7	1	5	4	?	?	?	
8	1	5	4	9	?	?	
10	1	5	4	9	?	?	V
12	1	5	4	9	-1	?	
13	1	5	4	9	-1	-4	
14	1	5	4	9	-1	-4	
20	1	5	4	9	-1	-4	

Digite os coeficientes (A B C): 1 5 4  
 Raizes: -1.000000, -4.000000

# Exercícios

2) Faça uma função que receba como parâmetro um número inteiro e imprima se este número é par ou ímpar.

Em seguida, faça um programa que leia um número inteiro do teclado e chame a função.

Faça um teste de mesa com o valor 4 e outro com o valor 5.

# Condicionais

Frequentemente, ocorrem situações em que:

um conjunto de ações precisa ser executado se a condição for satisfeita,  
mas **não existe qualquer ação** a ser tomada se a condição não for satisfeita.

A ***ESTRUTURA CONDICIONAL SIMPLES*** (ou alternativa simples), neste caso, executa:

um conjunto de ações  
**quando** o resultado for  
VERDADEIRO

ou

nenhuma ação  
**quando** ele for  
FALSO



# Condicionais

```

Imprima "Para a matricula, traga:"
Imprima " - Documento de identidade;"
Imprima " - CPF;"
SE sexo == 'M' ,
    Imprima " - Cert. de serviço militar;"
Imprima " - Foto 3x4."
    
```

A **ESTRUTURA CONDICIONAL SIMPLES** (ou alternativa simples), neste caso, executa:

um conjunto de ações  
**quando** o resultado for  
 VERDADEIRO

ou

nenhuma ação  
**quando** ele for  
 FALSO

# Sintaxe

```
if ( condição )
{
    bloco de comandos;
}
```

# Exemplo 1 (outra versão)

```
int valor1, valor2, maximo;
scanf ("%d%d", &valor1, &valor2);

maximo = valor1;

if (valor2 > maximo)
{
    maximo = valor2;
}

printf ("\nMAIOR = %d", maximo);
```

# Exemplo 3

```

int diaPagamento;
float multaAtraso, total;

multaAtraso = 5.0;

scanf ("%d%f", &diaPagamento, &total);

if (diaPagamento > 5)
{
    total = total + multaAtraso;
    printf ("Multa por atraso: %f", multaAtraso);
}

printf ("\nTotal: %f", total);

```

# Exercícios

3) Elabore uma função que receba como parâmetros dois números inteiros e imprima uma mensagem se um for divisível pelo outro.

Em seguida, faça um programa que leia um número inteiro do teclado e chame a função para verificar se este número é divisível por 2, por 3, por 5 e por 7 (serão 4 chamadas).

Faça um teste de mesa com o valor 14.

# Condicionais aninhadas

- Às vezes, é necessário usar estruturas condicionais aninhadas

```

if (condicao)
{
    comandos1;
    if (outraCondicao)
    {
        comandos2;
    }
}
else
{
    comandos3;
}
    
```

# Condicionais aninhadas

- Ou ainda usar um **if** dentro de outro **else**:

```

if (condicao)
{
    comandos1;
}
else
{
    if (condicao2)
    {
        comandos2;
    }
    else
    {
        comandos3;
    }
}

```

# Atenção

- Todo **else** precisa estar imediatamente depois do bloco de comandos de um **if**:

```
if (condicao)
{
    comandos1;
}
else
{
    if (condicao2)
    {
        comandos2;
    }
    else
    {
        comandos3;
    }
}
```



# Atenção

- Todo **else** precisa estar imediatamente depois do bloco de comandos de um **if**:

```
if (condicao)
{
    bloco de comandos 1;
}
else
    if (condicao2)
        unico comando 2;
    else
        if (condicao3)
        {
            bloco de comandos 3;
        }
        else
        {
            comandos4;
        }
```

# Exemplo 4

Determine se um número inteiro é zero, negativo ou positivo.

```
#include <stdio.h>
int main()
{
    int num;
    printf("Digite um numero inteiro: ");
    scanf("%d", &num);
    if(num == 0)
        printf("Valor zero");
    else
    {
        if(num > 0)
            printf("Valor positivo");
        else
            printf("Valor negativo");
    }
    return 0;
}
```

Se **if** ou **else** tem apenas um único comando no bloco de comandos, as chaves podem ser omitidas.

# Importância da indentação

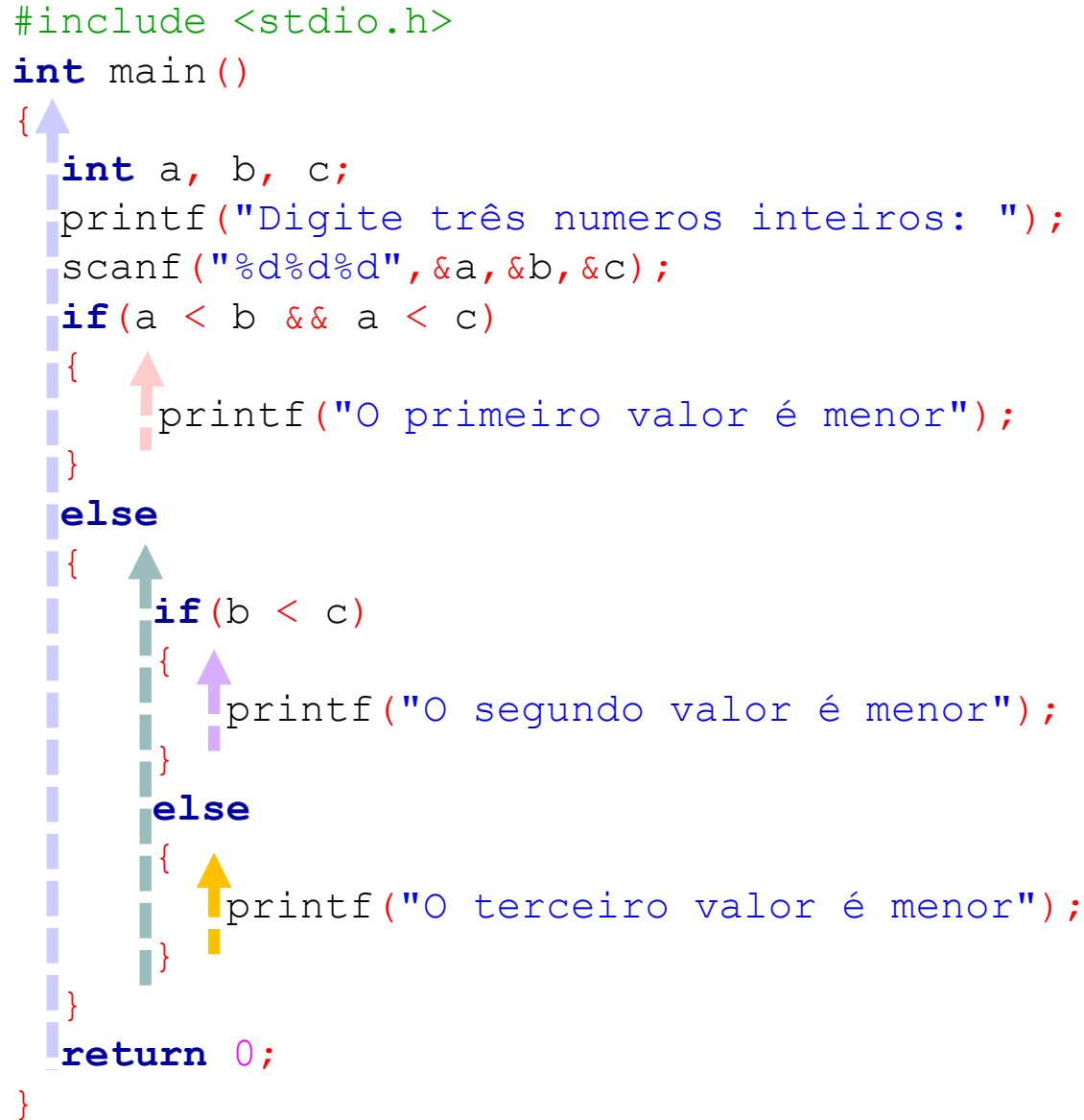
```
#include <stdio.h>
int main()
{
    int a, b, c;
    printf("Digite três numeros inteiros: ");
    scanf("%d%d%d", &a, &b, &c);
    if(a < b && a < c) {
        printf("O primeiro valor é menor");
    } else {
        if(b < c) {
            printf("O segundo valor é menor");
        } else {
            printf("O terceiro valor é menor");
        }
    }
    return 0;
}
```

# Importância da indentação

```
#include <stdio.h>
int main()
{
    int a, b, c;
    printf("Digite três numeros inteiros: ");
    scanf("%d%d%d", &a, &b, &c);
    if(a < b && a < c)
    {
        printf("O primeiro valor é menor");
    }
    else
    {
        if(b < c)
        {
            printf("O segundo valor é menor");
        }
        else
        {
            printf("O terceiro valor é menor");
        }
    }
    return 0;
}
```

# Importância da indentação

```
#include <stdio.h>
int main()
{
    int a, b, c;
    printf("Digite três numeros inteiros: ");
    scanf("%d%d%d", &a, &b, &c);
    if(a < b && a < c)
    {
        printf("O primeiro valor é menor");
    }
    else
    {
        if(b < c)
        {
            printf("O segundo valor é menor");
        }
        else
        {
            printf("O terceiro valor é menor");
        }
    }
    return 0;
}
```



# Exercício

## 4) O que é impresso neste programa?

```
int main()
{
    int a = 2, b = 4, c = 4;
    if ((a < 2) && ((b != 3) || (c == 3)))
        printf("saida 1");
    if ((c == 3) || (c == 4))
        printf("saida 2");
    if ((a < 3) && (b > 4))
        printf("saida 3");
    else if (a == 2)
        if (b == 5)
            printf("saida 4");
        else printf("saida 5");
    else printf("saida 6");
    return 0;
}
```

# Exercício

## 4) O que é impresso neste programa?

```
int main()
{
    int a = 2, b = 3, c = 4;
    if ((a<3) && ((b !=3) || (c == 4)))
        printf("saida 1");
    if ((c ==3) || (b ==4))
        printf("saida 2");
    if ((a < 3) && (b > 4))
        printf("saida 3");
    else
        if (a == 2)
            if (b==5)
                printf("saida 4");
            else
                printf("saida 5");
        else
            printf("saida 6");
    return 0;
}
```

Lembre-se, a indentação de um código facilita o seu desenvolvimento e a sua leitura.

# Exercícios

5) Faça um programa para ler dois números e um caractere: ' + ', ' - ', ' \* ' e ' / '.

O programa deve imprimir o resultado da operação efetuada sobre os números lidos.

Teste com 3.6, 4 e ' / '.

**DESAFIO:** Teste com 7.8, 0.0 e ' / ' e corrija seu programa para imprimir uma mensagem caso não seja possível efetuar um cálculo.



# Exercícios

- 6) Elabore uma função que receba como parâmetro a idade de uma pessoa (inteiro) e imprima:

*se idade < 13: "Criança";*

*se  $13 \leq \text{idade} < 20$ : "Adolescente";*

*se  $20 \leq \text{idade} < 65$ : "Adulto"; e,*

*se idade  $\geq 65$ : "Idoso" .*

Para fazer o programa, combine o uso de 3 pares de **if-else**.

Faça um programa que leia do teclado a idade da pessoa e imprima a mensagem.

Teste com o valor 15 e com o valor 70.

# Múltipla escolha

- Além do uso de `if-else`, há um outro tipo de estrutura que permite executar um bloco de comandos dependendo do valor de uma variável ou expressão.

# Múltipla escolha

É útil quando:

- Há um grande número de alternativas;
- Todas as alternativas dependem da avaliação de uma mesma expressão (na maioria das vezes, dependem do valor de uma única variável);
- O resultado da expressão (ou a variável da condição) é do tipo **int** ou **char**

# Múltipla escolha: Sintaxe

```

switch (expressao)
{
    case Valor1: Comandos1;
                break;
    case Valor2: Comandos2;
                break;
    . . .
    case ValorN: ComandosN;
                break;
    default: Comandos;
}

```

# Múltipla escolha: Sintaxe

```

switch (expressao)
{
    case Valor1: Comandos1;
                break;
    case Valor2: Comandos2;
                break;
    ...
    case ValorN: ComandosN;
                break;
    default: Comandos;
}
    
```

Variável (ou expressão) cujo valor definirá a opção que será executada.

# Múltipla escolha: Sintaxe

```

switch (expressao)
{
    case Valor1: Comandos1;
                break;
    case Valor2: Comandos2;
                break;
    . . .
    case ValorN: ComandosN;
                break;
    default: Comandos;
}
    
```

Cada valor que a expressão pode assumir.

# Múltipla escolha: Sintaxe

```

switch (expressao)
{
    case Valor1: Comandos1;
                break;
    case Valor2: Comandos2;
                break;
    ...
    case ValorN: ComandosN;
                break;
    default: Comandos;
}

```

Comandos que devem ser executados em cada caso.

# Múltipla escolha: Sintaxe

```

switch (expressao)
{
    case Valor1: Comandos1;
                break;
    case Valor2: Comandos2;
                break;
    ...
    case ValorN: ComandosN;
                break;
    default: Comandos;
}

```

O comando **break** encerra a sequência de comandos do **case** e sai do **switch**.



# Múltipla escolha: Sintaxe

```

switch (expressao)
{
    case Valor1: Comandos1;
                break;
    case Valor2: Comandos2;
                break;
    ...
    case ValorN: ComandosN;
                break;
    default: Comandos;
}

```

A ausência do comando **break** faz com que os comandos dos casos seguintes sejam executados até o próximo **break** ou fim do **switch**.

# Múltipla escolha: Sintaxe

```

switch (expressao)
{
    case Valor1 : Comandos1;
                break;
    case Valor2 : Comandos2;
                break;
    . . .
    case ValorN : ComandosN;
                break;
    default : Comandos;
}
    
```

O comando **default** permite que uma sequência de comandos seja executada se o valor da expressão for diferente de todos os casos especificados.

# Exemplo

```
#include <stdio.h>

int main()
{
    int epoca;
    printf("Digite o numero do trimestre:");
    scanf("%d", &epoca);
    switch (epoca)
    {
        case 1: printf("verao");
                break;
        case 2: printf ("outono");
                break;
        case 3: printf ("inverno");
                break;
        case 4: printf ("primavera");
                break;
        default: printf("Trimestre invalido");
    }
    return 0;
}
```

# Exemplo

```
#include <stdio.h>

int main()
{
    int mes;
    printf("Digite o numero do mes:");
    scanf("%d", &mes);
    switch (mes)
    {
        case 1: case 2: case 3:
            printf("verao");
            break;
        case 4: case 5: case 6:
            printf ("outono");
            break;
        case 7: case 8: case 9:
            printf ("inverno");
            break;
        case 10: case 11: case 12:
            printf ("primavera");
            break;
        default: printf("Mes invalido");
    }
    return 0;
}
```

# Exercícios

- 7) Elabore uma função que recebe como parâmetro um inteiro representando um dia da semana e imprime o seu respectivo nome por extenso. Considere que o número 1 representa o domingo; 2, a segunda, etc. Caso o número não corresponda a um dia da semana, a função deve exibir a mensagem “Dia da semana inválido”.  
Faça um programa para chamar a função.  
Teste seu programa com um valor válido e outro inválido.

# Exercícios

- 8) Elaborar um programa para ler o código de um produto e informar a sua origem:
- a) Código do produto entre 1 e 20: Europa
  - b) Código do produto entre 21 e 40: Ásia
  - c) Código do produto entre 41 e 60: América
  - d) Código do produto entre 61 e 80: África
  - e) Código do produto maior que 80: Paraguai

# Exercícios

- 9) O dono de um supermercado quer modificar o cálculo do preço final das mercadorias usando como base o seu valor de custo:
- Até R\$2,00, o acréscimo deve ser um valor fixo de R\$0,15;
  - Entre R\$2,00 e R\$5,00, o acréscimo deve ser proporcional, de 2%;
  - Entre R\$5,00 e R\$20,00, deve ser proporcional, de 10%;
  - Acima de R\$20,00, o acréscimo deve ser proporcional, de 8%.

Assim, um produto que custe R\$ 28,00, terá acréscimo de R\$ 2,35:

- R\$ 0,15 => referente aos primeiros R\$ 2,00;
- + R\$ 0,06 => referente aos 2% sobre a faixa R\$ 2,00 a R\$ 5,00 (R\$ 3);
- + R\$ 1,50 => referente aos 10% sobre a faixa R\$ 5,00 a R\$ 20,00 (R\$ 15);
- + R\$ 0,64 => referente aos 8% acima de R\$ 20,00.

Faça um programa que leia o valor de custo do produto e imprima o seu preço final.

# Exercícios

**DESAFIO:** Uma empresa de telefonia quer que você desenvolva um programa para calcular o preço de uma ligação internacional. Para isso, o programa precisa **ler o prefixo** que identifica o país da ligação e a **duração da ligação** em segundos (número inteiro).

O preço de cada minuto é:

- R\$1,90 para Argentina (54), Uruguai (598), Paraguai (595), Chile (56);
- R\$2,00 para Alemanha (49), França (33), Inglaterra (44);
- R\$2,10 para Canadá e Estados Unidos (1);
- R\$2,30 para outros países.

Após 5 minutos, o preço dos minutos adicionais tem redução de 5%.

Desenvolva o programa usando, ao menos, **duas funções** (além da função main).

Teste seu programa com os valores 39 e 415.



# Condicionais

---

DCC 120 – Laboratório de Programação



# Comando Condicional - `if`

- O comando `if` é uma estrutura de decisão que decide se uma sequência de comandos será ou não executada. Sua sintaxe é:

```
if (expressão)
{
    sequencia de comandos;
}
```

- Ou

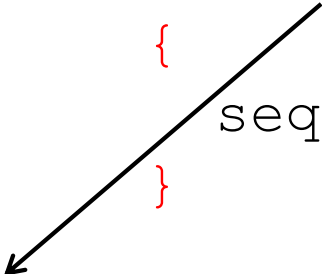
```
if (expressão)
    unico comando;
```

# Comando Condicional - `if`

```

if (expressão)
{
    sequencia de comandos;
}

```



- A expressão sempre será avaliada, e o resultado lógico (verdadeiro ou falso), na linguagem C corresponde a
  - **FALSO:** o valor zero (`==0`)
  - **VERDADEIRO:** os demais (`!=0`)

# if - Exemplos

- Programa para determinar o maior de dois números fornecidos pelo usuário.

```
int main()
{
    int a, b, maior;
    scanf ("%d%d", &a, &b);
    maior = a;
    if (b > maior) {
        maior = b;
    }
    printf ("\nMAIOR = %d", maior);
    return 0;
}
```

# Condicional: `if-else`

- O comando `if` pode decidir entre duas sequências de comandos qual vai ser a executada e tem a seguinte sintaxe:

```
if (expressão)
{
    // caso a expressão retorne verdadeiro
    sequencia de comandos;
}
else
{
    // caso a expressão retorne falso
    sequencia de comandos;
}
```

# if-else

- Exemplo: Verificar se um número é par.

```
#include <stdio.h>

int main()
{
    int x;
    printf("Digite o numero: ");
    scanf("%d", &x);
    if (x % 2 == 0)
        printf("%d e' par \n", x);
    else
        printf("%d e' impar \n", x);
    return 0;
}
```

- Obs.: No exemplo acima não são utilizadas chaves no `if` e no `else` pois há apenas um comando a ser executado. Em casos como esse pode-se ou não utilizar as chaves.

# if else , if , ...

```

if (condição1)
{
    comandos if1;
    if (condição2)
    {
        comandos if2;
    }
    else
    {
        comandos else2;
    }
}
else
{
    comandos else1;
    if (condição3)
    {
        comandos if3;
    }
}

```

# Comando switch

- Utilizado quando uma determinada variável pode ser igual a diferentes valores que se deseja avaliar
- Sintaxe:

```
switch (variavel)
{
    case constante1: comandos;
    break;
    case constante2: comandos;
    break;
    default: comandos;
}
```



# Comando switch - Exemplo

```
int main()
{
    int epoca;
    printf("Digite o trimestre do ano em que estamos: ");
    scanf("%d", &epoca);
    switch (epoca)
    {
        case 1: printf("verao");
                break;
        case 2: printf("outono");
                break;
        case 3: printf("inverno");
                break;
        case 4: printf("primavera");
                break;
        default: printf("periodo invalido");
    }
    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 as funções `imprimeDivisaoInteira` e `imprimeDivisaoReal` que recebem dois números inteiros e imprimem o resultado da divisão do primeiro número pelo segundo. Se não for possível fazer a divisão (se o segundo valor for igual a zero), imprima uma mensagem informando o problema.

Faça um programa para chamar as funções e teste ambas usando como entrada os valores 9 e 4.

# Exercícios

2. Desenvolva a função `saoPositivos` que recebe como parâmetros dois números reais, informados pelo usuário e verifica se ambos são maiores que zero. A função deve retornar 1 (verdadeiro) se ambos forem positivos e 0 (falso) caso contrário.

Faça um programa que leia dois números e chame a função, imprimindo "Ambos os valores sao positivos" ou "Ao menos um dos valores eh negativo.", dependendo do caso.

Teste o programa com 1 e 2 e com 3 e -4.

# Exercícios

3. Construa a função `calculaPesoIdeal` que recebe o sexo e a altura de uma pessoa como parâmetros. A função deve calcular e retornar o peso ideal da pessoa, utilizando uma das seguintes fórmulas:

- masculino:  $(72.7 * \text{alt}) - 58;$
- feminino:  $(62.1 * \text{alt}) - 44.7.$

Faça um programa que lê o sexo, a altura e o peso de uma pessoa e imprime se esta pessoa está acima, abaixo ou com o peso ideal.

Teste seu programa com os valores F, 1.71 e 59.5.

# Exercícios

4. Elabore um programa que leia 3 valores reais (x, y e z) de comprimento e imprima na tela se tais valores formam os lados de um triângulo ou não. Para formar um triângulo, os valores devem atender às seguintes condições:

$$x < y + z \quad \text{e} \quad y < x + z \quad \text{e} \quad z < x + y.$$

Teste seu programa com os valores 4, 2.2 e 1.4.

# Exercícios

5. Faça uma função chamada `leNumeroPositivo`.  
 A função deve ler um número inteiro e, se for positivo, deve retorná-lo. Se não for positivo, a função deve exibir uma mensagem informando que o usuário terá mais uma chance para digitar um valor. Se novamente o valor for inválido, a função deve exibir uma mensagem e retornar o valor zero.  
 Faça um programa que chame a função e imprima o valor retornado.  
 Teste seu programa com os valores -1 e 5 e com 0 e -4.

# Exercícios

6. Desenvolva a função `classificaCaractere` que lê um caractere e imprime uma das seguintes mensagens:

- "Operador matematico" (+, -, \*, /, %);
- "Operador relacional" (<, >);
- "Operador logico" (!);
- "Outro simbolo valido em C" (&, =, ", ', parênteses e chaves);
- "Caractere nao identificado".

Faça um programa em C que chame a função 3 vezes. Teste seu programa com <, \$ e ).



# Exercícios

7. Construa a função `classificaNadador` que recebe a idade de um nadador (número inteiro) como parâmetro e imprime sua categoria, de acordo com a tabela abaixo:

CATEGORIA	FAIXA ETÁRIA
infantil A	5 a 7 anos
infantil B	8 a 10 anos
juvenil A	11 a 13 anos
juvenil B	14 a 17 anos
adulto	18 a 30 anos
sênior	maiores de 30 anos

Faça um programa que leia a idade de um nadador e imprima sua categoria.

# Exercícios

8) Para auxiliar os vendedores de uma loja na orientação aos clientes sobre as diversas formas de pagamento, desenvolver um algoritmo para:

a) Imprimir o seguinte menu:

```
Forma de pagamento:
- À vista.
- Cheque para trinta dias.
- Em duas vezes.
- Em três vezes.
- Em quatro vezes.
- A partir de cinco vezes.
Entre com sua opção:
```

b) Ler o código da opção de pagamento.

# Exercícios

- c) Imprimir uma das mensagens de acordo com a opção lida:

Opção = 1: Desconto de 20%

Opção = 2, 3 ou 4: Mesmo preço a vista

Opção = 5: Juros de 3% ao mês

Opção = 6: Juros de 5% ao mês

Opção <1 ou opção >6: Opção inválida