

# Algoritmos: Estruturas de Controle



## Introdução à Ciência da Computação

Rosane Minghim

Apoio na confecção: Carlos Elias A. Zampieri

Danilo Medeiros Eler

Rogério Eduardo Garcia

Renato Rodrigues



# Algoritmo

- **Execução sequencial:**

- Uma vez executado um comando, os demais são executados na sequência;

- **Execução condicional: Deseja-se “selecionar” os comandos a serem executados:**

- Executar os comandos em determinadas condições;

- **Repetição:**

- Repetir blocos de comandos pela mudança do fluxo de execução.

# Algoritmo

## ■ Execução sequencial:

- Uma vez executado um comando, os demais são executados na sequência;

## ■ Execução condicional:

OS COMANDOS

- Execução dos comandos somente se determinadas condições;

## ■ Repetição:

- Repetir blocos de comandos pela mudança do fluxo de execução.

**Estruturas de Controle**

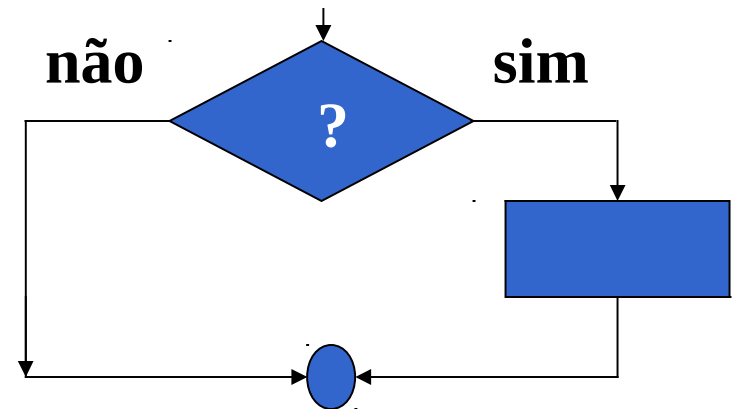
“selecionar”

“repetir”

# Escolha Simples

- Pode-se seleccionar a sequência de comandos a ser executada
- Formato:

*se condição então*  
*comandos*  
*fim se*





# Exemplo de Escolha Simples

constante

FATOR\_CATEGORIA\_1 = 1,0

TAXA\_BÁSICA = 20,00

variável

categoria: inteiro

taxa: real

leia(categoria)

taxa ← taxa\_básica

**se** categoria = 1 **então**

taxa ← taxa - (TAXA\_BÁSICA \* FATOR\_CATEGORIA\_1)

**fim se**

escreva(taxa)

# Escolha Composta

■ Formato:

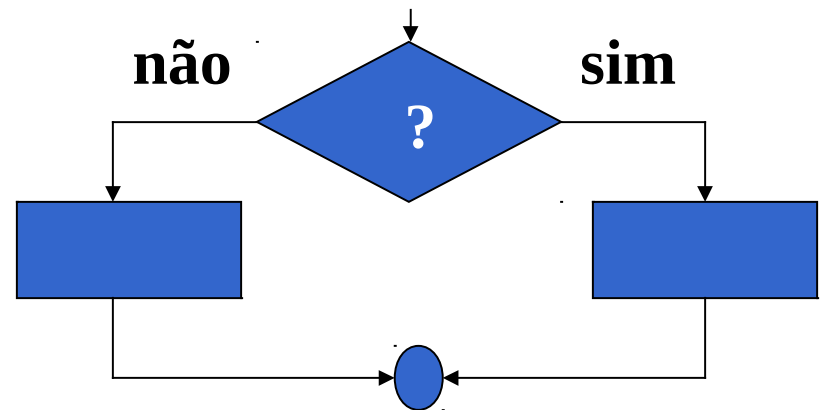
*se condição então*

*comandos*

*senão*

*comandos*

*fim se*





# Exemplo

constante

FATOR\_CATEGORIA\_1 = 1,0

FATOR\_GERAL = 0,2

TAXA\_BÁSICA = 20,00

variável

categoria: inteiro

taxa: real

leia(categoria)

taxa ← taxa\_básica

**se** categoria = 1 **então**

taxa ← taxa - (TAXA\_BÁSICA \* FATOR\_CATEGORIA\_1)

**senão**

taxa ← taxa - (TAXA\_BÁSICA \* FATOR\_GERAL)

**fim se**

escreva(taxa)

# Estruturas ‘Aninhadas’

## Algoritmo maior\_de\_tres

*{algoritmo para obter o maior  
entre três valores inteiros}*

variável

maior: inteiro

numero1, numero2, numero3: inteiro

leia(numero1, numero2, numero3)

```
se (numero1 > numero2) então
  se (numero2 ≥ numero3) então
    maior ← numero1
  senão
    se (numero1 > numero3) então
      maior ← numero1
    senão
      maior ← numero3
  fim se
fim se
senão
  se (numero2 ≥ numero3) então
    maior ← numero2
  senão
    maior ← numero3
  fim se
fim se

escreva(maior)
fim
```



# Estruturas ‘Aninhadas’ (simplificação)

Algoritmo maior\_de\_tres\_melhor

*{algoritmo para obter o maior  
entre três valores inteiros}*

variável

maior: inteiro

numero1, numero2, numero3: inteiro

leia(numero1, numero2, numero3)

maior ← numero1

se (numero2 > maior) então

maior ← numero2

fim se

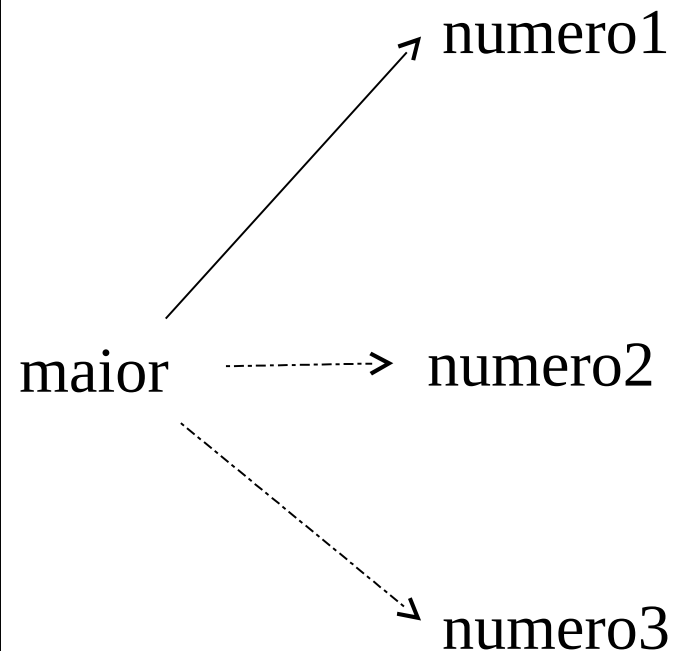
se (numero3 > maior) então

maior ← numero3

fim se

escreva(maior)

fim





# Exercício

Ref. Minghim, R., Telles, G.P. – Curso Introdutório de Programação

- 1) Desenvolver um algoritmo para, dados dois times de futebol (cada time identificado por um número inteiro), seus pontos ganhos e seu saldo de gols no campeonato, decidir qual dos dois está em melhor colocação (armazenando o resultado na variável ganhador). O resultado deve ser impresso. A regra diz que está na frente no campeonato o time que tiver mais pontos ganhos, com desempate pelo saldo de gols.



# Escolhas Múltiplas

- Permite escolher uma entre várias alternativas expressas por valor inteiro ou caractere
- Formato:

```
selecione expressão entre
    constante:
        comandos
constante:
    comandos
...
constante:
    comandos
senão
    comandos
fim seleção
```



# Escolhas Múltiplas

- Na seleção múltipla, a **expressão** é calculada e os **comandos** relacionados abaixo da constante com o mesmo valor da expressão são executados.
- Se não houver valor igual ao da expressão, os comandos subordinados à palavra **senão** são executados.
- A cláusula **senão** é opcional.
  - Se ela não existir e o valor da expressão não for igual a nenhuma constante, nenhum comando da estrutura é executado.
- A seleção é exclusiva
  - No máximo uma das opções é executada.

# Exemplo

## Algoritmo desconto\_taxa

constante

FATOR\_CATEGORIA\_1 = 1,0

FATOR\_CATEGORIA\_2 = 0,8

FATOR\_CATEGORIA\_3 = 0,6

FATOR\_GERAL = 0,2

TAXA\_BÁSICA = 20,00

variável

categoria: inteiro

taxa: real

leia(categoria)

taxa ← taxa\_básica

selecione categoria entre

1:

taxa ← taxa -  
          (taxa\_básica \*  
          FATOR\_CATEGORIA\_1)

2:

taxa ← taxa -  
          (taxa\_básica \*  
          FATOR\_CATEGORIA\_2)

3:

taxa ← taxa -  
          (taxa\_básica \*  
          FATOR\_CATEGORIA\_3)

senão

taxa ← taxa -  
          (taxa\_básica \*  
          FATOR\_GERAL)

fim seleção

escreva(taxa)

fim



# Exercício

Ref. Minghim, R., Telles, G.P. – Curso Introdutório de Programação

- 1) Desenvolver um algoritmo para calcular o valor de uma cartela de passes de ônibus para um passageiro. Uma cartela pode ter 50 ou 100 passes. Determinados tipos de usuários possuem desconto na compra de passes, de acordo com a tabela abaixo:

idosos 50%

estudantes 50%

trabalhadores faixa I 50%

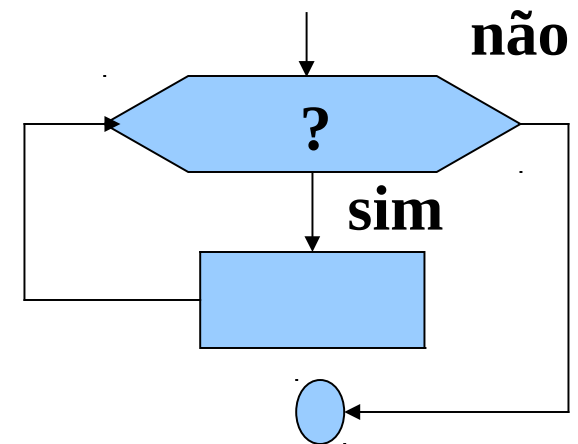
trabalhadores faixa II 25%

trabalhadores faixa I e estudante 75%

# Repetição por Condição

- Uma das formas de repetir um conjunto de comandos de um algoritmo é subordiná-lo a um comando de repetição usando uma estrutura da forma:

**enquanto *condição* faça**  
**comandos**  
**fim enquanto**





# Repetição por Condição

- Os comandos serão repetidos **zero** ou **mais vezes**, enquanto o valor da condição for verdadeiro.
- Essa estrutura normalmente é denominada **laço** ou **loop**.





# Repetição por Condição – Funcionamento

- A condição da cláusula **enquanto** é testada.
- Se ela for verdadeira os comandos seguintes são executados em sequência como em qualquer algoritmo, até a cláusula **fim enquanto**.
- O fluxo nesse ponto é desviado de volta para a cláusula **enquanto**.
- Se a condição agora for falsa (ou quando finalmente for), o fluxo do algoritmo é desviado para o primeiro comando após a cláusula **fim enquanto**.
- Se a **condição** ainda for verdadeira, o processo se repete.



# Repetição por Condição

- A **condição** pode ser qualquer expressão que resulte em um valor do tipo lógico e pode envolver operadores aritméticos, lógicos, relacionais e resultados de funções.

# Exemplo

## Algoritmo calcula\_senos

variável

n, i: inteiro

leia(n)

i ← 0

enquanto i ≤ n faça

    escreva(i, seno(i))

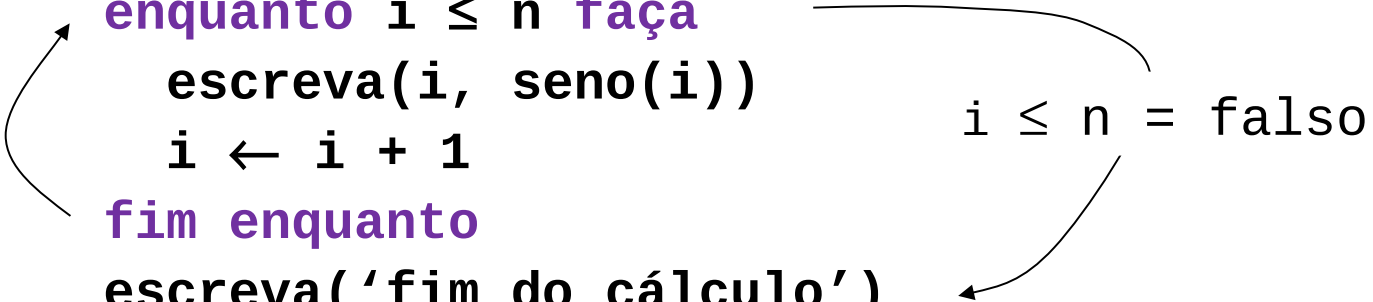
    i ← i + 1

fim enquanto

escreva('fim do cálculo')

fim

i ≤ n = falso



# Exemplo

## Algoritmo calcula\_senos

variável

$n, i$ : inteiro

leia( $n$ )

$i \leftarrow 0$

enquanto ( $i < 45$ ) e ( $i < n + 1$ ) faça

    escreva( $i, \text{seno}(i)$ )

$i \leftarrow i + 1$

fim enquanto

escreva('fim do cálculo')

fim



# Loop Infinito

```
enquanto verdadeiro faça  
    comandos  
fim enquanto
```

- Um loop infinito pode acontecer também quando cometemos algum erro ao especificar a condição lógica que controla a repetição ou ao esquecer de algum comando dentro da iteração.

# Exemplo

## Algoritmo calcula\_senos

**variável**

**n, i: inteiro**

**leia(n)**

**i ← 0**

**enquanto (i < 45) e (i < n + 1) faça**

**escreva(seno(i))**

**fim enquanto**

**fim**

A variável i não é incrementada



# Exemplo – soma consecutiva de dados sem estrutura de repetição

## Algoritmo somasimples

**variável**

```
valor1, valor2, valor2, valor4: real
```

```
soma: real
```

```
leia(valor1, valor2, valor3, valor4)
```

```
soma ← (valor1 + valor2 + valor3 + valor4)/4
```

```
escreva(soma)
```

```
fim
```



# Exemplo – soma consecutiva de dados

## Algoritmo somasimples

variável

valor, soma: real

soma ← 0

leia(número)

enquanto número > -100 faça

    soma ← soma + número

    leia(número)

fim enquanto

escreva(soma)

fim

**Exercício:** Além da soma, calcular a média aritmética





# Outra forma de Repetição

- Com teste no final do bloco de comandos.
- Uma diferença dessa forma para a anterior é que os comandos dentro da estrutura são executados uma vez antes que a condição seja testada pela primeira vez, e serve para os processos iterativos onde existe garantia de execução correta do bloco pelo menos uma vez.



# Outra forma de Repetição

- Outra diferença com relação à forma anterior de repetição é que enquanto aquela estabelecia uma condição de continuidade, esta estabelece a condição de parada para a repetição.
- O seu formato é dado por:

*repita*  
*comandos*  
*até condição*

# Exemplo

- Nesse caso, os comandos são repetidos uma ou mais vezes, até que a condição se torne verdadeira (isto é, enquanto a condição for falsa).

número = 0 falso

```
Algoritmo calcula_quadrado  
  
variável  
    número: real  
  
    repita  
        leia(número)  
        escreva(número*número)  
    até número = 0  
  
fim
```



# Exercício

- Fazer a soma consecutiva de valores inteiros lidos até a soma atingir 25500 ou até que tenham sido lidos 150 termos.
- Imprimir o número de termos da soma. Se a soma excedeu o limite, imprimir a diferença, senão imprimir a soma.
- Casos de teste



# Repetição por Contagem

- Na iteração baseada em contagem, sabe-se **antecipadamente** quantas vezes um conjunto de comandos vai ser repetido.
- Formato:

```
para variável de valor_inicial até valor_final  
    passo valor_do_passo faça  
        comandos  
fim para
```



# Repetição por Contagem

- Inicialmente a **variável**, que chamamos de variável controladora, é inicializada com o **valor\_inicial**. Devem ser do tipo inteiro.
- Em seguida, os comandos são repetidos zero ou mais vezes, enquanto o valor da **variável** estiver entre o **valor\_inicial** e o **valor\_final**, inclusive.
- No final de cada repetição do conjunto de comandos, a variável controladora é automaticamente acrescida do **valor\_do\_passo** e o teste do limite é repetido



# Exemplo

## Algoritmo conta\_com\_para

variável

n, i: inteiro

leia(n)

para i de 1 até n passo 1 faça

    escreva(i)

fim para

fim



# Repetição por Contagem

- A estrutura de controle **para** admite uma variação em que a repetição se dá em ordem decrescente. Basta que o valor do passo seja negativo.

## Algoritmo conta\_decrescente

```
variável
  i,n: inteiro

leia(n)
para i de n até 1 passo -1 faça
  escreva(i)
fim para
fim
```





# Exemplo

## Algoritmo expoente

variável

base, expoente, resultado: inteiro

i: inteiro

leia(base, expoente)

resultado ← 1

para i de 1 até expoente passo 1 faça

    resultado ← resultado\*base

fim para

escreva(resultado)

fim

# Exemplos – dos and don'ts

## Algoritmo laço\_ruim

**constante**

LIMITANTE = 25500

**variável**

n, soma, i: inteiro

**leia(n)**

soma ← 0

**para i de 1 até n passo 1 faça**

**leia(termo)**

soma ← soma + termo

**se soma > LIMITANTE então**

i ← n + 1

**fim se**

**fim para**

**escreva(soma)**

**fim**



## Algoritmo laço\_bom

**constante**

LIMITANTE = 25500

**variável**

n, soma, i: inteiro

**leia(n)**

soma ← 0

**i ← 1**

**enquanto (i ≤ n) e (soma ≤ LIMITANTE) faça**

**leia(termo)**

soma ← soma + termo

**i ← i + 1**

**fim enquanto**

**escreva(soma)**

**fim**



# Equivalência entre as três formas de repetição

- É possível perceber que apenas uma forma de iteração (por exemplo, aquela da cláusula **enquanto**), seria suficiente para desenvolver qualquer algoritmo baseado em repetição.
- As demais formas existem para facilitar a estruturação dos algoritmos e aumentar a clareza do código.



# Exemplo

- Qualquer laço de repetição baseado em contagem é equivalente a uma estrutura **enquanto** no seguinte formato:

```
variável ← valor_inicial  
enquanto variável ≤ valor_final faça  
    comandos  
    variável ← variável + valor_do_passo  
fim enquanto
```



# Exemplo

## Algoritmo expoente

variável

```
base, expoente, resultado: inteiro;  
i: inteiro;
```

```
leia(base, expoente)
```

```
resultado ← 1
```

```
i ← 1
```

```
enquanto i ≤ expoente faça
```

```
    resultado ← resultado*base
```

```
    i ← i + 1
```

```
fim para
```

```
escreva(resultado)
```

```
fim
```



# Percorrendo um Algoritmo: Casos de Teste

- Um algoritmo deve ser revisado buscando melhorias. Além disso, é preciso verificar se sua execução está correta.
- Um recurso para iniciar esse processo é percorrer o algoritmo.
  - Simular manualmente a execução de cada passo do algoritmo até chegar ao fim, assumindo valores para aqueles dados que são lidos do usuário.

# Exemplo

```
1 Algoritmo maiordetres
2
3   variável
4   maior: inteiro
5   número1, número2, número3: inteiro
6   i: inteiro
7
8
9   Para i de 1 até 3 passo 1 faça
10    leia(número1, número2, número3)
11    se número1>número2 então
12     se (número2>número3) ou (número2 = número3) então
13      maior ← número1
14    senão
15     se número1>número3 então
16      maior ← número1
17    senão
18     maior ← número3
19    fim se
20  fim se
21 senão
22  se (número2 > número3) ou (número2 = número3) então
23   maior ← número2
24  senão
25   maior ← número3
26  fim se
27 fim se
28 escreva(maior)
29 fim para
30 fim
```

Passo	Linha	i	numero1	numero2	número3	maior	condição
1	9	1	?	?	?	?	$1 \leq 3$
2	10	1	3	1	2	?	
3	11	1	3	1	2	?	verdadeiro
4	12	1	3	1	2	?	falso
5	13	1	3	1	2	3	
6	28	1	3	1	2	3	
7	29	2	3	1	2	3	
8	9	2	3	1	2	3	$2 \leq 3$
9	10	2	1	4	4	3	
10	11	2	1	4	4	3	falso
11	22	2	1	4	4	3	verdadeiro
12	23	2	1	4	4	4	
13	28	2	1	4	4	4	
14	29	3	1	4	4	4	
15	9	3	1	4	4	4	$3 \leq 3$
16	10	3	5	4	1	4	
17	11	...					



# Exercício

- Calcule a tabuada dos números 1, 2, 3, 4 e 13 usando estruturas de repetição por condição e contagem e imprima em tela os resultados.