UNIVERSIDADE FEDERAL DO CEARÁ

DEPARTAMENTO DE COMPUTAÇÃO

CIÊNCIA DA COMPUTAÇÃO

# CK0211 - Fundamentos de Programação: Estrutura de Repetição

Emanuele Santos

**Bibliografia: Ascencio, Cap. 5**

# Objetivos

- Apresentar a Estrutura de Repetição em Algoritmos
- Apresentar a Estrutura de Repetição em Python

# ESTRUTURA DE REPETIÇÃO EM ALGORITMOS

# Estrutura de Repetição

- Usada quando um trecho do algoritmo ou até mesmo o algoritmo inteiro precisa ser repetido
- O número de repetições pode ser fixo ou estar atrelado a uma condição

# Estrutura de repetição para número definido de repetições (estrutura PARA)

- Usada quando se sabe o número de vezes que um trecho do algoritmo deve ser repetido

```
PARA i ← valor_inicial ATÉ valor_final FAÇA [PASSO n]
INÍCIO
    comando1
    comando2
    …
    comandom
FIM
```

# Estrutura de repetição para número definido de repetições (estrutura PARA)

```
PARA i ← valor_inicial ATÉ valor_final FAÇA [PASSO n]
INÍCIO
    comando1
    comando2
    …
    comandom
FIM
```

- Os comandos 1 a m serão executados utilizado a variável **i** como controle
- A variável **i** irá variar de **valor_inicial** até o **valor_final**

# Estrutura de repetição para número definido de repetições (estrutura PARA)

```
PARA i ← valor_inicial ATÉ valor_final FAÇA [PASSO n]
INÍCIO
    comando1
    comando2
    …
    comandom
FIM
```

- **PASSO** está entre colchetes porque é opcional
  - Indica de quanto em quanto i vai variar
  - Quando PASSO for omitido, isso significa que o seu valor é 1

# Estrutura de repetição para número definido de repetições (estrutura PARA): Exemplos

```
PARA i ← 1 ATÉ 10 FAÇA
    ESCREVA i
```

- O comando ESCREVA i será executado **10 vezes**, ou seja, i varia de 1 a 10
- Assim o programa acima mostrará os números 1, 2, 3, 4, 5, 6, 7, 8, 9 e 10, mostrando um número por linha e sem as vírgulas

# Teste de Mesa ou Computador Chinês

- Ferramenta empregada para verificar se um algoritmo está sendo executado corretamente
- Esse teste simula o que o pseudocódigo está executando passo a passo

# Procedimento para executar um computador chinês

- 1. Elaborar uma tabela onde cada coluna se refere a cada variável envolvida e o resultado de uma operação em particular, ou número da linha do algoritmo (ou observação pertinente)
- 2. Executar os passos previstos no algoritmo
- 3. Verificar se os resultados obtidos são coerentes. Senão, corrigir o algoritmo e testar novamente para as entradas anteriores.
- 4. Realizar o teste para diferentes entradas e concluir quando todos os testes forem bem sucedidos

# Estrutura de repetição para número definido de repetições (estrutura PARA): Exemplos

```
PARA j ← 1 ATÉ 9 FAÇA PASSO 2
    ESCREVA j
```

- O comando ESCREVA j será executado **5 vezes**, ou seja, j varia de 1 a 9, de 2 em 2
- Assim o programa acima mostrará os números 1, 3, 5, 7 e 9, mostrando um número por linha e sem as vírgulas

# Estrutura de repetição para número definido de repetições (estrutura PARA): Exemplos

```
PARA i ← 10 ATÉ 5 FAÇA PASSO −1
    ESCREVA i
```

- O comando ESCREVA i será executado **6 vezes**, ou seja, i varia de 10 a 5
- Assim o programa acima mostrará os números 10, 9, 8, 7, 6 e 5, mostrando um número por linha e sem as vírgulas

# Estrutura de repetição para número definido de repetições (estrutura PARA): Exemplos

```
PARA j ← 15 ATÉ 1 FAÇA PASSO –2
    ESCREVA j
```

- O comando ESCREVA j será executado **8 vezes**, ou seja, j varia de 15 a 1, de 2 em 2
- Assim o programa acima mostrará os números 15, 13, 11, 9, 7, 5, 3 e 1, mostrando um número por linha e sem as vírgulas

# Estrutura de repetição para número definido de repetições (estrutura PARA)

- A estrutura de repetição também é chamada de **loop** ou **laço**
- Cada execução do laço é chamada de **iteração**
- Quando a variável de controle do laço aumenta em cada iteração, dizemos que a variável é **incrementada** do valor em PASSO
- Quando a variável de controle do laço diminui em cada iteração, dizemos que a variável é **decrementada** do valor em PASSO

# Usando acumuladores em estruturas de repetição

- Acumuladores devem ser usados quando a realização de um cálculo precisa de valores obtidos em cada iteração, ou seja o cálculo só estará pronto depois que o loop terminar

```
SOMA ← 0 # inicialização da variável SOMA com zero
PARA i ← 1 ATÉ 5 FAÇA
INÍCIO
    ESCREVA “Digite um número: ”
    LEIA num
    SOMA ← SOMA + num # acumulando o valor de num em SOMA
FIM
ESCREVA “Soma = ”, SOMA
```

# Usando contadores em estruturas de repetição

- Contadores podem ser usados para contar situações específicas dentro de um loop
- São usadas outras variáveis (não use a mesma variável de controle do laço)

```
CONT ← 0 # inicialização da variável CONT com zero
PARA i ← 1 ATÉ 5 FAÇA
INÍCIO
    ESCREVA “Digite um número: ”
    LEIA num
    SE num > 5
    ENTÃO CONT ← CONT + 1 # aumentando em 1 o valor de CONT
FIM
ESCREVA “Quantidade de números maiores que 5 = ”, CONT
```

# Exercícios

- Usando a estrutura PARA, escreva um algoritmo que peça para o usuário entrar com 5 números e ao final mostre a quantos desses números eram pares

- Usando a estrutura PARA, peça para o usuário entrar com um número inteiro maior que zero e calcule o fatorial desse número

# Inicialização de acumuladores

- Normalmente, acumuladores para somas (somatórios) são inicializados com zero e acumuladores para produtos (produtórios) são inicializados com 1

- Os acumuladores são inicializados sempre antes de iniciar o laço e são incrementados/decrementados dentro do laço

# Estrutura de repetição para número indefinido de repetições e teste no início (estrutura ENQUANTO)

- Normalmente, é usada quando não se sabe o número de vezes que um trecho do algoritmo deve ser repetido
- Uma **condição** é verificada no **início** de cada iteração
- Enquanto a condição for verdadeira, os comandos são realizados

```
ENQUANTO condição FAÇA
    comando1
```

```
ENQUANTO condição FAÇA
INÍCIO
  comando1
  comando2
  comando3
FIM
```

# Estrutura de repetição para número indefinido de repetições e teste no início (estrutura ENQUANTO): Exemplos

```
x ← 1
y ← 5
ENQUANTO x < y FAÇA
INÍCIO
   x ← x + 2
   y ← y + 1
   ESCREVA x, y
FIM
```

- Como seria o computador chinês para esse algoritmo?

# Estrutura de repetição para número indefinido de repetições e teste no início (estrutura ENQUANTO): Exemplos

```
x ← 1
y ← 1
ENQUANTO x <= 5 FAÇA
INÍCIO
   y ← y * x
   x ← x + 1
   ESCREVA x, y
FIM
```

- Como seria o computador chinês para esse algoritmo?

# Estrutura de repetição para número indefinido de repetições e teste no final (estrutura REPITA)

- Normalmente, é usada quando não se sabe o número de vezes que um trecho do algoritmo deve ser repetido
- Uma **condição** é verificada no **final** de cada iteração
- Os comandos são realizados até a condição se tornar verdadeira
- A diferença entre a estrutura ENQUANTO e a estrutura REPITA é que, em REPITA os comandos serão executados pelo menos uma vez, já que o teste da condição está no final

```
REPITA
   comandos
ATÉ condição
```

# Estrutura de repetição para número indefinido de repetições e teste no final (estrutura REPITA): Exemplos

```
x ← 1
y ← 5
REPITA
   x ← x + 2
   y ← y + 1
   ESCREVA x, y
ATÉ x >= y
```

- Como seria o computador chinês para esse algoritmo?

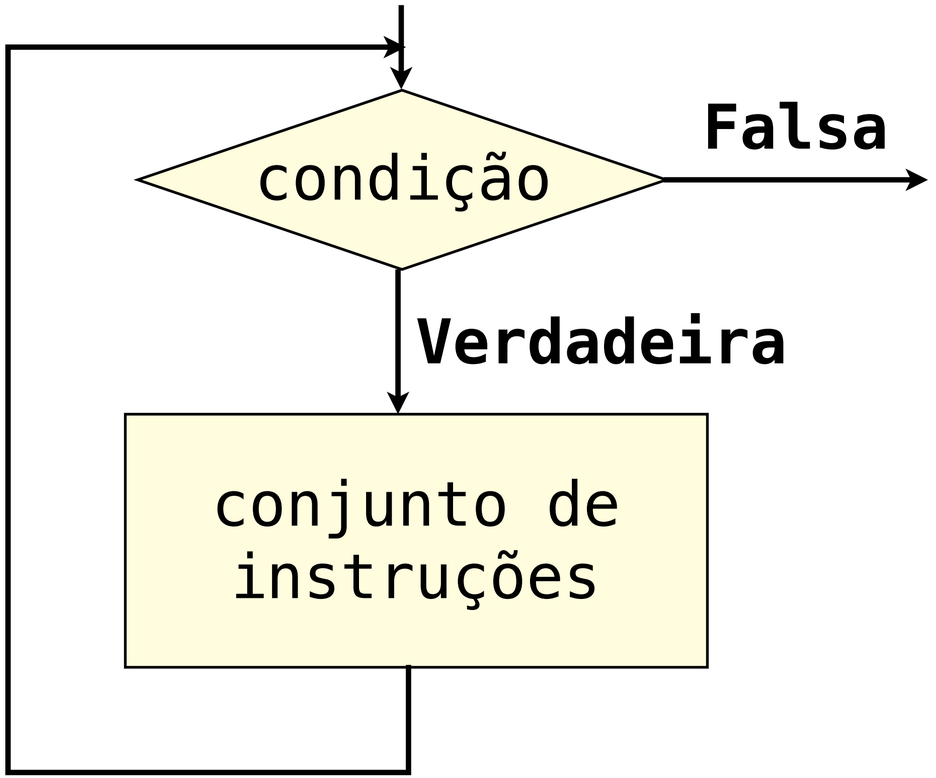
# Estrutura de repetição para número indefinido de repetições e teste no final (estrutura REPITA): Exemplos

```
x ← 1
y ← 1
REPITA
   y ← y * x
   x ← x + 1
   ESCREVA x, y
ATÉ x = 6
```

- Como seria o computador chinês para esse algoritmo?

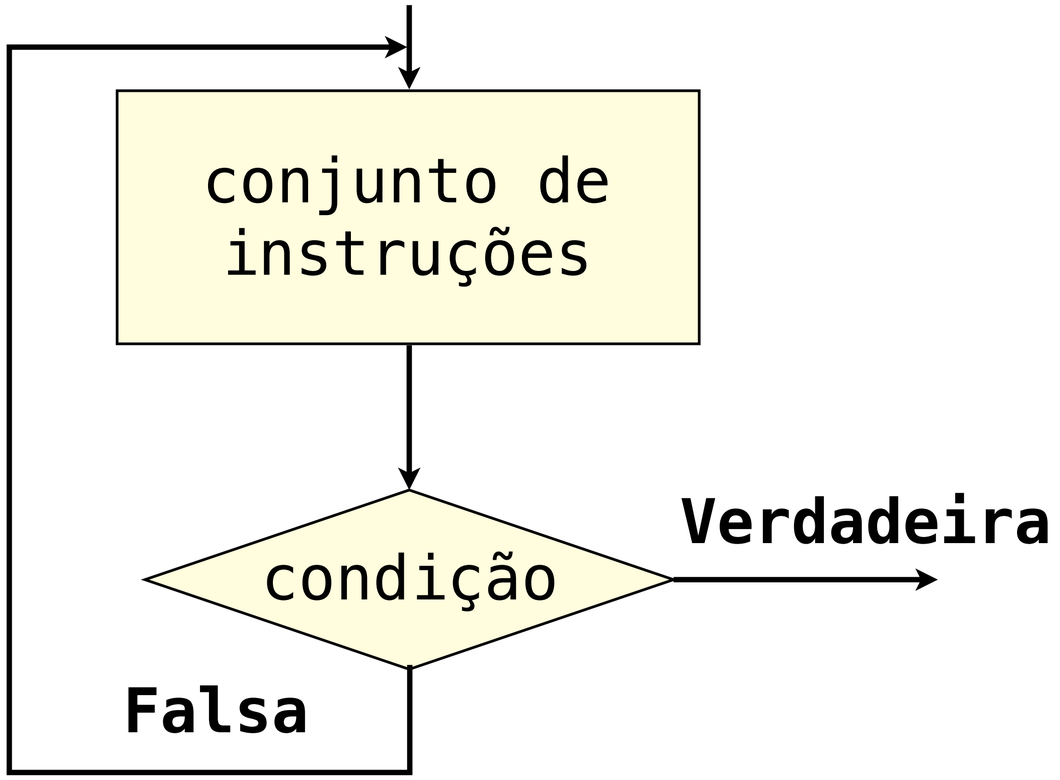
# Estruturas de repetição em fluxograma

Teste no início:



# Estruturas de repetição em fluxograma

Teste no final:
REPITA … ATÉ



# ESTRUTURA DE REPETIÇÃO EM PYTHON

# O comando while

não esquecer os dois pontos

```
while condição :
    comando1
    comando2
```

4 espaços delimitando um novo bloco

- Onde:
  - condição é uma expressão lógica que pode retornar verdadeiro ou falso
  - Os comandos serão executados enquanto a condição for verdadeira (True)

# Repetição com teste no início

Exemplo 1:

```
n = int(input(“Digite um número positivo: ”))
while n <= 0:
    print(‘Número inválido. Tente novamente.’)
    n = int(input(“Digite um número positivo: ”))
print(‘Você digitou o número ’, n)
```

# Usando contadores com while

```
cont = 0
while cont < 10:
    cont = cont + 1
    print(‘valor do contador’, cont)
print(‘fim’)
```

# Terminação do loop precocemente

- Utiliza-se o comando break:

```
while condição1:
    comando1
    if condição2:
        break
    comando2
    comando3
outros comandos
```

# Exercício usando o while

- Escreva um programa que faça a contagem regressiva de 10 a 1, mostrando cada um dos números na tela

# Repetição com teste no final

Em outras linguagens de programação, como Pascal, C e Java, existe uma estrutura equivalente a **REPITA** ... **ATÉ**

Em Python, esse tipo de estrutura precisa ser emulada com o comando **while**:

```
while True:
    comandos ...
    if condição :
        break
```

comandos são executados pelo menos uma vez, mesmo que a condição seja falsa

# Repetição com teste no final

Exemplo 1:

```
while True:
    idade = int(input(“Quantos anos você tem?”))
    if idade > 0:
        break
    print(“Idade inválida. Tente novamente: ”)
print(“ Ok. Você tem ”, idade, “ anos.”)
```

# O comando for

não esquecer os dois pontos

```
for i in lista_de_valores :
    comando1
    comando2
```

4 espaços delimitando um novo bloco

- Onde:
  - lista_de_valores é uma lista crescente ou decrescente de números
  - Os comandos serão executados um número de vezes igual ao número de elementos da lista

# Listas em Python

- Python fornece diversos tipos de dados compostos usados para agrupar outros tipos de dados simples
- O tipo mais versátil é a lista (list), o qual pode ser escrito como uma lista de valores separados por **vírgula** entre **colchetes**

```
#tipo list
numeros = [1, 2, 3, 4, 5]
numeros2 = [5, 4, 3, 2, 1, 0]
numeros3 = [1, 3, 5, 7, 9]
```

# Listas em Python

- Exemplo

```
PARA i ← 1 ATÉ 10 FAÇA
    ESCREVA i
```

```
for i in [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]:
    print(i)
```

# Listas em Python

- Exemplo

```
PARA i ← 10 ATÉ 5 FAÇA PASSO -1
    ESCREVA i
```

```
for i in [10, 9, 8, 7, 6, 5]:
    print(i)
```

# Listas em Python

- Exemplo

```
PARA j ← 15 ATÉ 1 FAÇA PASSO −2
    ESCREVA j
```

```
for i in [15, 13, 11, 9, 7, 5, 3, 1]:
    print(i)
```

# Criando listas de inteiros em Python

- Usa-se a função range para criar uma lista de inteiros começando de inicio, até fim (mas não inclui fim) de passo em passo

```
range(inicio=0, fim, passo=1)
```

- Onde:
  - inicio e passo são opcionais; quando inicio é omitido, possui o valor zero e passo quando é omitido possui o valor 1
  - Os números gerados serão menores que fim, se a lista for crescente; ou maiores que fim, se a lista for decrescente

# Criando listas de inteiros em Python

- Exemplos

```
range(4) #só o fim foi passado
# [0, 1, 2, 3]
range(1,5) #inicio e fim foram passados
# [1, 2, 3, 4]
range(1, 10, 2) #lista de 1 a 9, de 2 em 2
# [1, 3, 5, 7, 9]
range(10, 1, -2) # lista de 10 a 2, de 2 em 2
# [10, 8, 6, 4, 2]
```

# O comando for com a função range

- Exemplo

```
PARA i ← 1 ATÉ 10 FAÇA
    ESCREVA i
```

```
for i in range(1, 11):
    print(i)
```

# Listas em Python

- Exemplo

```
PARA i ← 10 ATÉ 5 FAÇA PASSO -1
    ESCREVA i
```

```
for i in range(10, 4, -1):
    print(i)
```

# Listas em Python

- Exemplo

```
PARA j ← 15 ATÉ 1 FAÇA PASSO -2
    ESCREVA j
```

```
for i in range(15, 0, -2):
    print(i)
```

# Exercício

- Escreva um algoritmo para mostrar a tabuada de um número 0 < inteiro <=10 entrado pelo usuário usando **for**

- Exemplo: 5

5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50