

# Algoritmos e Lógica de Programação

Prof. Flavius Gorgônio

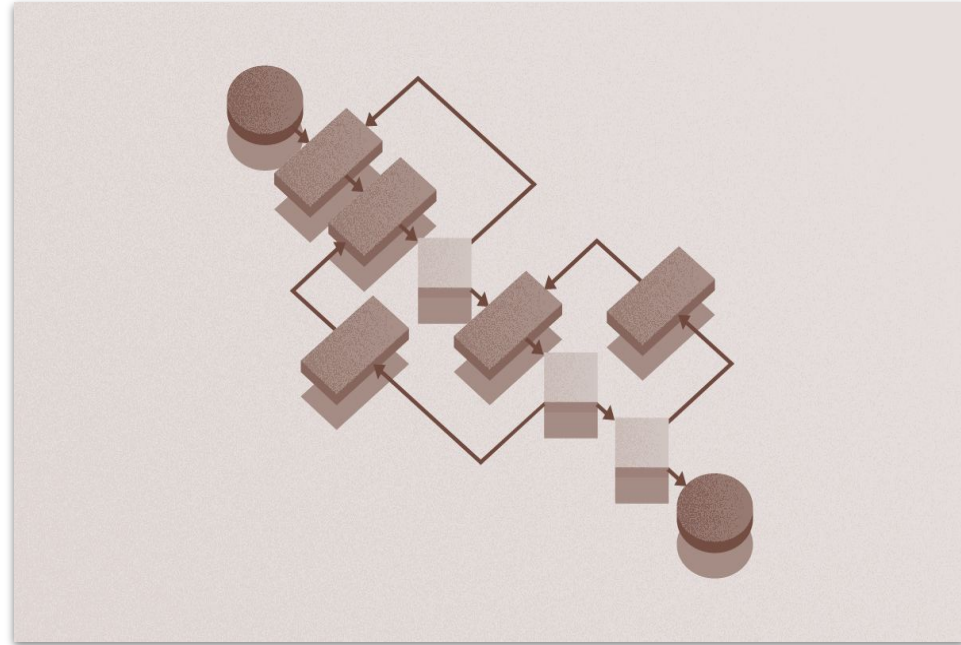


# Semana 8

Estruturas de controle de repetição: A estrutura FOR

# Agenda

- Motivação
- Estruturas de controle de fluxo
  - Estruturas de controle de decisão
  - Estruturas de controle de repetição
- Estruturas de controle de repetição
  - Repetição com teste lógico
  - Repetição com contador
- Aplicações e exemplos

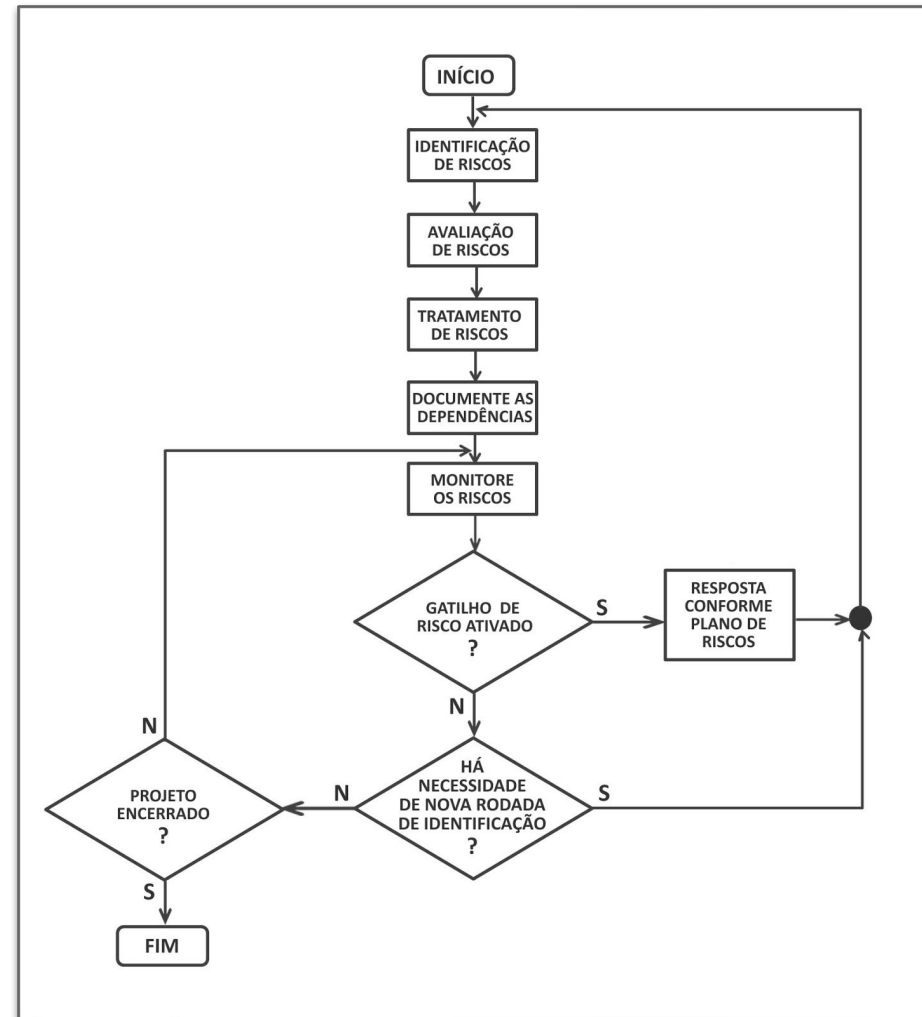


# Motivação

Suponha que você precisa implementar o fluxo de verificação de riscos em projetos apresentado na imagem ao lado através de um programa de computador

Perceba que pode ser necessário que o **programa** REPITA ALGUNS TRECHOS em determinados momentos do fluxo

Alguns setas direcionam o fluxo para trechos que já foram executados antes



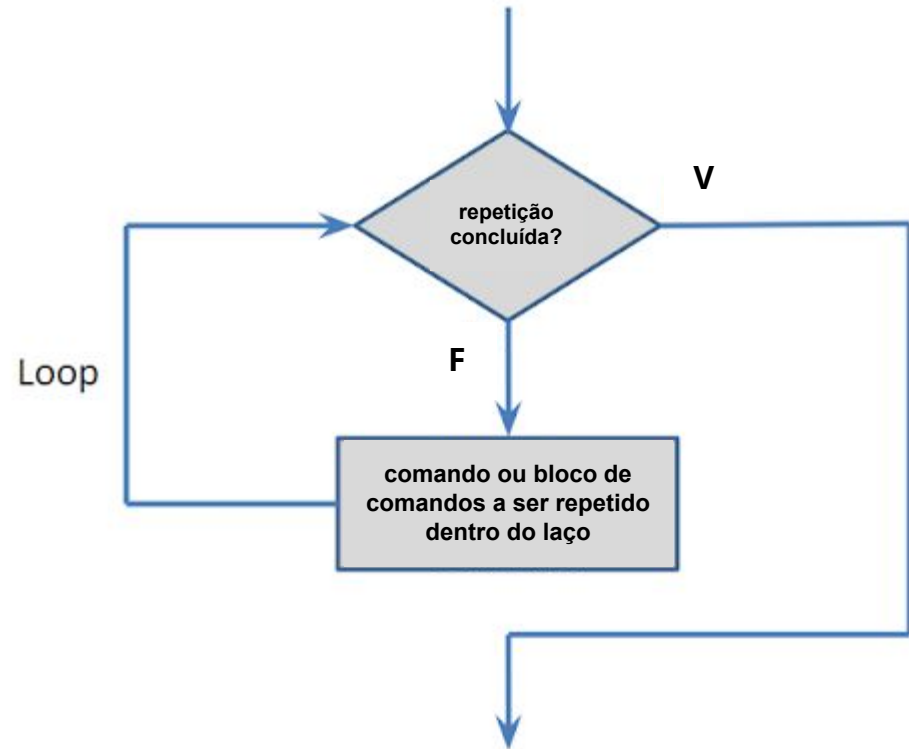
# A estrutura **para ... início ... fim**

Uma estrutura de repetição do tipo **para ... início ... fim** permite executar um ou mais comandos (bloco de comandos) zero ou mais vezes

Se a verificação de repetição concluída resultar em FALSO, o comando (ou bloco de comandos) subordinado à estrutura é executado

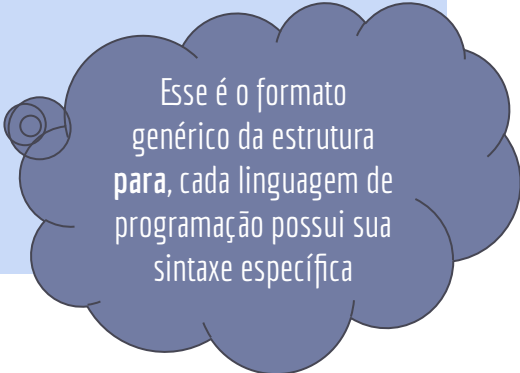
Se resultar em VERDADEIRO, o comando (ou bloco de comandos) subordinado à estrutura não é executado e o laço encerra

Em ambos os casos, a execução continua com os comandos subsequentes à estrutura de decisão



# A forma genérica da estrutura **para ... início ... fim**

```
...  
para <var> de <início> até <fim> faça  
    instrução_1  
    instrução_2  
    ...  
    instrução_n  
fim_do_bloco  
...
```



Esse é o formato genérico da estrutura **para**, cada linguagem de programação possui sua sintaxe específica

As condições de início e fim (cor vermelha) são verificadas e, no caso de não ter sido concluída, o bloco de comandos subordinado à estrutura (cor azul) será executado

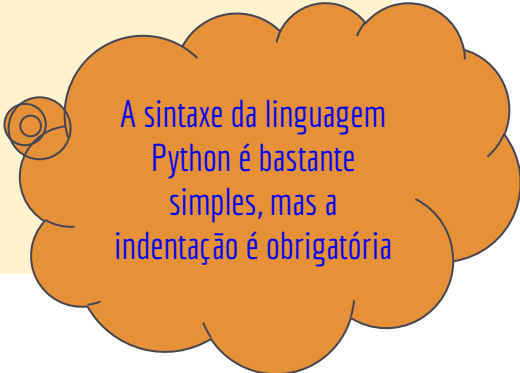
Ao final da execução do bloco, o controle do programa retorna ao **para**, a variável é atualizada e novamente testada

Enquanto o valor final da variável não for atingido, o bloco de comandos (conhecido como laço) continuará a ser executado

Se a variável chegar ao seu valor final, o bloco azul não é executado e o fluxo continua após o término do laço

# A estrutura **para** no Python

```
...  
for <var> in range(<fim>):  
    instrução_1  
    instrução_2  
    ...  
    instrução_n  
...
```



A sintaxe da linguagem Python é bastante simples, mas a indentação é obrigatória

Na linguagem Python, a estrutura **para** é nomeada de **for**

A função **range** cria a sequência de valores (iniciando em zero) a ser percorrida pela variável **var**

O delimitador de início de bloco é o símbolo de dois pontos (:)

Não há delimitador de final de bloco, o fim da indentação delimita o término do bloco

O bloco deve, obrigatoriamente, estar indentado mais à direita em relação à indentação do **for**

# Exemplo de uso da estrutura **for**

```
print("Início do Laço")

for i in range(10):

    print("Valor de i = ",i)

print("Fim do Laço")
```

```
Início do Laço
Valor de i = 0
Valor de i = 1
Valor de i = 2
Valor de i = 3
Valor de i = 4
Valor de i = 5
Valor de i = 6
Valor de i = 7
Valor de i = 8
Valor de i = 9
Fim do Laço
```



# Equivalência entre for e while

```
# Laço com estrutura for
```

```
print("Início do Laço")
```

```
for i in range(10):
```

```
    print("Valor de i = ",i)
```

```
print("Fim do Laço")
```

```
# Laço com estrutura while
```

```
print("Início do Laço")
```

```
i = 0
```

```
while i < 10:
```

```
    print("Valor de i = ",i)
```

```
    i = i + 1
```

```
print("Fim do Laço")
```

# A estrutura **para** com valor inicial diferente

```
...  
for <var> in range(<início>,<fim>):  
    instrução_1  
    instrução_2  
    ...  
    instrução_n  
...
```

Nem sempre, o programador deseja iniciar a contagem a partir de zero

Para estes casos, a estrutura **for** da linguagem Python permite alterar o valor inicial do laço alterando-se os parâmetros da função **range**

Se forem fornecidos dois parâmetros, a função **range** utiliza o primeiro parâmetro como valor inicial e o segundo parâmetro como valor alvo

# Exemplo de uso da estrutura **for** com início e fim

```
# Laço for com outro início
```

```
print("Início do Laço")
```

```
for i in range(3,10):
```

```
    print("Valor de i = ",i)
```

```
print("Fim do Laço")
```

```
Início do Laço
```

```
Valor de i = 3
```

```
Valor de i = 4
```

```
Valor de i = 5
```

```
Valor de i = 6
```

```
Valor de i = 7
```

```
Valor de i = 8
```

```
Valor de i = 9
```

```
Fim do Laço
```

# Equivalência entre **for** e **while** com início diferente

```
# Laço for com outro início
```

```
print("Início do Laço")
```

```
for i in range(3,10):
```

```
    print("Valor de i = ",i)
```

```
print("Fim do Laço")
```

```
# While com outro início
```

```
print("Início do Laço")
```

```
i = 3
```

```
while i < 10:
```

```
    print("Valor de i = ",i)
```

```
    i = i + 1
```

```
print("Fim do Laço")
```

# A estrutura **para** com valor de passo diferente

```
...  
for <var> in range(<início>,<fim>,<passo>) :  
    instrução_1  
    instrução_2  
    ...  
    instrução_n  
...
```

Outras vezes, o programador deseja avançar ou recuar com valor diferente de um

Para estes casos, a estrutura **for** da linguagem Python permite alterar o valor da atualização (passo) alterando-se os parâmetros da função **range**

Se forem fornecidos três parâmetros, a função **range** utiliza o primeiro parâmetro como valor inicial, o segundo parâmetro como valor alvo e o terceiro parâmetro como valor de atualização

# Exemplo de uso da estrutura **for** com passo

```
# Laço for com passo

print("Início do Laço")

for i in range(1,20,3):

    print("Valor de i = ",i)

print("Fim do Laço")
```

```
Início do Laço

Valor de i = 1

Valor de i = 4

Valor de i = 7

Valor de i = 10

Valor de i = 13

Valor de i = 16

Valor de i = 19

Fim do Laço
```

# Equivalência entre **for** e **while** com passo diferente

```
# Laço for com passo
```

```
print("Início do Laço")
```

```
for i in range(1,20,3):
```

```
    print("Valor de i = ",i)
```

```
print("Fim do Laço")
```

```
# Laço while com passo
```

```
print("Início do Laço")
```

```
i = 1
```

```
while i < 20:
```

```
    print("Valor de i = ",i)
```

```
    i = i + 3
```

```
print("Fim do Laço")
```

# Exemplo de uso do **for** com passo regressivo

```
# Laço for regressivo

print("Início do Laço")

for i in range(20,1,-3):

    print("Valor de i = ",i)

print("Fim do Laço")
```

```
Início do Laço

Valor de i = 20

Valor de i = 17

Valor de i = 14

Valor de i = 11

Valor de i = 8

Valor de i = 5

Valor de i = 2

Fim do Laço
```



# Equivalência de **for** e **while** com passo regressivo

```
# Laço for regressivo

print("Início do Laço")

for i in range(20,1,-3):

    print("Valor de i = ",i)

print("Fim do Laço")
```

```
# Laço while regressivo

print("Início do Laço")

i = 20

while i > 1:

    print("Valor de i = ",i)

    i = i - 3

print("Fim do Laço")
```

# Gerando sequências de valores

Escreva laços que gerem e exibam as seguintes sequências:

1. Pares menores que 20, iniciando em zero
2. Pares entre 40 e 20, incluindo-os
3. Ímpares entre 10 e 30
4. Múltiplos de 4 entre 1 e 25
5. Divisores de 30 entre 10 e 20
6. Ímpares menores que x (informado pelo usuário)
7. Múltiplos de 7 entre a e b (informado pelo usuário)

# Sugestões de bibliografia

## Básica

- MENEZES, Nilo Ney Coutinho. Introdução à programação com Python: algoritmos e lógica de programação para iniciantes. 2a ed. São Paulo: Novatec, 2010. 328p. ISBN: 9788575224083
- FARRELL, Joyce. Lógica e design de programação: introdução. São Paulo: Cengage Learning, 2010. xiv, 416p. ISBN: 9788522107575.
- SOUZA, Marco; GOMES, Marcelo; SOARES, Márcio; CONCILIO, Ricardo. Algoritmos e Lógica de Programação. 2a ed. São Paulo: Cengage Learning, 2013. 240 p. ISBN: 9788522111299

## Complementar

- ASCENCIO, Ana Fernanda; CAMPOS, Edilene Aparecida. Fundamentos da programação de computadores: Algoritmos, Pascal, C/C++(Padrão Ansi) e Java. 3a ed. São Paulo: Pearson Prentice Hall. 569 p. ISBN: 9788564574168
- ZELLE, John M. Python programming: an introduction to computer science. 2nd ed. Sherwood, Or.: Franklin, Beedle & Associates, c2010. xiv, 514 p. ISBN: 9781590282410.
- LUTZ, Mark; ASCHER, David. Learning Python. 4nd ed. Sebastopol, CA: O Reilly, c2009. xlv, 1162 p. ISBN: 978059615064.



# Copyright

Este material é para uso exclusivo durante as aulas da disciplina DCT1101 - ALGORITMOS E LÓGICA DE PROGRAMAÇÃO, do curso de Bacharelado em Sistemas de Informação da Universidade Federal do Rio Grande do Norte, não estando autorizada a sua publicação, compartilhamento, divulgação ou utilização em outros contextos diferentes dos aqui apresentados.

Todos os textos, imagens, exemplo de código e demais materiais utilizados nesses slides são apenas para fins didáticos, o autor e a instituição não permitem a sua utilização sem autorização expressa, assim como não se responsabilizam pelo seu uso indevido ou por danos causados pelos mesmos.