

CC1612

Fundamentos de Algoritmos

Prof. Danilo H. Perico

Estruturas de Repetição

Repetições

- São utilizadas para executar **várias vezes** a mesma parte do programa
- Normalmente dependem de uma condição
- Repetições são a base de vários programas!

Exemplo

- Fazer um programa para imprimir 10 números sequenciais na tela, começando do número 1.

Exemplo

- Esta é uma solução para o problema do exemplo:
- É uma solução boa?!

```
print(1)
print(2)
print(3)
print(4)
print(5)
print(6)
print(7)
print(8)
print(9)
print(10)
```

```
1
2
3
4
5
6
7
8
9
10
```

Exemplo

- Esta é uma solução para o problema do exemplo:
- É uma solução boa?!
- Não: **Muitos comandos repetidos!**

```
print(1)
print(2)
print(3)
print(4)
print(5)
print(6)
print(7)
print(8)
print(9)
print(10)
```

```
1
2
3
4
5
6
7
8
9
10
```

Repetições

- Idealmente, poderíamos ter o comando `print(x)` imprimindo a variável `x`, onde `x` variasse de 1 até 10.

Repetições

- Idealmente, poderíamos ter o comando `print(x)` imprimindo a variável `x`, onde `x` variasse de 1 até 10.
- A boa notícia é que existem comandos para realizar este tipo de ação!

Estruturas de Repetição

Comando *while*

Comando **while**

- O comando **while** (enquanto) serve para executarmos alguma repetição **enquanto** uma condição for verdadeira (True)
- Sintaxe:

```
while <condição>:  
    #bloco que será repetido enquanto a condição for verdadeira
```

Comando **while**

- Exemplo: imprimir os números de 1 até 10


```
x = 1
while x <= 10:
    print(x)
    x = x + 1
```

```
1
2
3
4
5
6
7
8
9
10
```

Exercício

1. Escreva um programa que faça a contagem regressiva de 10 até 0. O programa deve imprimir cada número da contagem a cada um segundo.

```
from time import sleep  
  
print(1)  
sleep(1)  
print(2)
```



1
2

Função **sleep(x)** - pausa o código por **x** segundos

sleep(1) - pausa o código por **1** segundo

Exemplo

- Impressão do número 1 até o número digitado pelo usuário:

```
ultimo = int(input("Digite o último dígito da contagem: "))  
i = 1  
while i <= ultimo:  
    print(i)  
    i += 1
```

Equivalente a $i = i + 1$

Digite o último dígito da contagem: 5

1
2
3
4
5

- O lado direito do sinal de igual (=) é executado primeiro!
- O resultado é atribuído para a variável que estiver do lado esquerdo do sinal de igual (=)

Exemplo

- Impressão do número 1 até o número digitado pelo usuário:

```
ultimo = int(input("Digite o último dígito da contagem: "))  
  
i = 1  
  
while i <= ultimo:  
    print(i)  
    i += 1
```

i é um contador

um contador é uma
variável que conta o
número de ocorrências de
um evento: neste caso, o
número de repetições!

Digite o último dígito da contagem: 5

1
2
3
4
5

Exemplo - combinando repetição com **if**

- Programa que imprime todos os números pares de 0 até um número digitado pelo usuário.

```
ultimo = int(input("Digite o último dígito da contagem: "))  
  
i = 0  
  
while i <= ultimo:  
    if i % 2 == 0:  
        print(i)  
    i += 1
```

Exemplo - combinando repetição com **if**

- Programa que imprime todos os números pares de 0 até um número digitado pelo usuário.

```
ultimo = int(input("Digite o último dígito da contagem: "))  
  
i = 0  
  
while i <= ultimo:  
    if i % 2 == 0:  
        print(i)  
    i += 1
```

Digite o último dígito da contagem: 6

0
2
4
6

Exercício

2. Escreva um programa que imprime a tabuada do número digitado pelo usuário.
3. Faça um programa que solicita um número entre 0 e 10. Mostre uma mensagem de erro caso o valor seja inválido e continue pedindo até que o usuário informe um valor válido. Quando o valor for válido dê a mensagem “número aceito!”

Dica: Você pode utilizar operadores lógicos (**and** ou **or**) na condição do while também!

Exemplo

- Programa para calcular a somatória de 10 números que devem ser digitados pelo usuário.

```
n = 1
soma = 0

while n <= 10:
    x = int(input("Digite o %d número:" % n))
    soma = soma + x
    n += 1
print("Soma", soma)
```

```
Digite o 1 número:5
Digite o 2 número:6
Digite o 3 número:4
Digite o 4 número:7
Digite o 5 número:8
Digite o 6 número:9
Digite o 7 número:5
Digite o 8 número:2
Digite o 9 número:1
Digite o 10 número:3
Soma 50
```

Exemplo

- Programa para calcular a somatória de 10 números que devem ser digitados pelo usuário.

```
n = 1
soma = 0

while n <= 10:
    x = int(input("Digite o %d número:" % n))
    soma = soma + x
    n += 1
print("Soma", soma)
```

n é um contador

soma é um acumulador

um acumulador é uma
variável que guarda uma
somatória

while infinito

- Muitas vezes, queremos que nossos programas sejam executados infinitamente
- Nesses casos, podemos utilizar uma condição que nunca deixe de ser verdadeira (True)

while infinito

```
while True:  
    #bloco que sempre será executado,  
    #nunca sai do loop de repetição
```

Comando **break**

- Porém, mesmo quando utilizamos um **while** infinito, é possível que em determinadas situações o programa precise sair do loop de repetição.
- Esta interrupção pode ser alcançada com o comando **break**
- O comando **break** pode ser utilizado para interromper o while, independentemente da condição

Comando **break** - Exemplo

- Somatória de valores digitados pelo usuário até que o número 0 (zero) seja digitado; quando 0 for digitado o resultado da somatória é exibido:

```
somatoria = 0

while True:
    entrada = int(input("Digite um número a somar ou 0 para sair:"))
    if entrada == 0:
        break
    else:
        somatoria = somatoria + entrada

print("Somatória", somatoria)
```

```
Digite um número a somar ou 0 para sair:5
Digite um número a somar ou 0 para sair:6
Digite um número a somar ou 0 para sair:4
Digite um número a somar ou 0 para sair:0
Somatória 15
```

Repetições aninhadas

- Podemos combinar vários **while**, um dentro do outro!
- Com isso, conseguimos alterar automaticamente o valor de mais do que somente uma variável

Exemplo

- Programa para calcular as tabuadas do número 1 até o número 10.

```
tabuada = 1
while tabuada <= 10:
    print()
    print("Tabuada do", tabuada)
    multiplicador = 0
    while multiplicador <= 10:
        print(tabuada, "x", multiplicador, "=", (tabuada*multiplicador))
        multiplicador += 1
    tabuada += 1
```

Tabuada do 1

| | | | | |
|---|---|----|---|----|
| 1 | x | 0 | = | 0 |
| 1 | x | 1 | = | 1 |
| 1 | x | 2 | = | 2 |
| 1 | x | 3 | = | 3 |
| 1 | x | 4 | = | 4 |
| 1 | x | 5 | = | 5 |
| 1 | x | 6 | = | 6 |
| 1 | x | 7 | = | 7 |
| 1 | x | 8 | = | 8 |
| 1 | x | 9 | = | 9 |
| 1 | x | 10 | = | 10 |

Tabuada do 2

| | | | | |
|---|---|---|---|---|
| 2 | x | 0 | = | 0 |
| 2 | x | 1 | = | 2 |
| 2 | x | 2 | = | 4 |
| 2 | x | 3 | = | 6 |

Estruturas de Repetição

Comando *for*

Comando **for**

- **for** é a estrutura de repetição mais utilizada
- Sintaxe:

```
for <referência> in <sequência>:  
    #bloco de código que será repetido  
    #a cada iteração
```

- Durante a execução, a cada *iteração**, a *referência* aponta para um elemento da *sequência*.
- Uma vantagem do **for** com relação ao **while** é que o *contador* não precisa ser explícito!

* *iteração*: ato de iterar**; repetição.

** *iterar*: tornar a fazer; repetir.

Comando **for** - Exemplo

- Calcular a somatória dos números de 0 a 99

```
somatoria = 0  
  
for x in range(0,100):  
    somatoria = somatoria + x  
print(somatoria)
```

4950

A função **range(*i, f, p*)** é bastante utilizada nos laços com **for**

Ela gera um conjunto de valores inteiros:

- Começando de ***i***
- Até valores menores que ***f***
- Com passo ***p***

Se o passo ***p*** não for definido, o padrão de 1 será utilizado.

Exercício

4. Faça um programa que gera 100 vezes um número aleatório entre 1 e 100 e, então, exiba qual foi o maior número gerado e quantas vezes o maior número foi atualizado no seu código.

Para isso, você deve comparar o número gerado na iteração presente com o maior número armazenado até o momento.

A função `randrange(i, f)` gera números inteiros de `i` até valores menores que `f`

```
from random import randrange  
numero = randrange(1, 101)
```

Comando **else** na repetição

- É possível a utilização do comando **else** nas estruturas de repetição
- Tanto no **while** quanto no **for**
- A cláusula **else** só é executada quando a condição do *loop* se torna falsa.
- Se você sair do *loop* com o comando **break**, por exemplo, ela não será executada.

```
i = 0
while i < 11:
    print(i)
    i+=2
else:
    print("Os números pares de 0 a 10 foram exibidos")
```

```
0
2
4
6
8
10
Os números pares de 0 a 10 foram exibidos
```

```
for i in range(0,11,2):
    print(i)
else:
    print("Os números pares de 0 a 10 foram exibidos")
```

```
0
2
4
6
8
10
Os números pares de 0 a 10 foram exibidos
```

Comando **else** na repetição

- É possível a utilização do comando **else** nas estruturas de repetição
- Tanto no **while** quanto no **for**
- A cláusula **else** só é executada quando a condição do *loop* se torna falsa.
- Se você sair do *loop* com o comando **break**, por exemplo, ela não será executada.

```
i = 0
while i < 11:
    print(i)
    i+=2
else:
    print("Os números pares de 0 a 10 foram exibidos")
```

```
0
2
4
6
8
10
Os números pares de 0 a 10 foram exibidos
```

```
for i in range(0,11,2):
    print(i)
else:
    print("Os números pares de 0 a 10 foram exibidos")
```

```
0
2
4
6
8
10
Os números pares de 0 a 10 foram exibidos
```

Comando **else** na repetição

- Exemplo com **break**: o **else** não é executado

```
1 i = 0
2
3 while i < 11:
4     print(i)
5     i+=2
6     if i == 8:
7         break
8 else:
9     print("Os números pares de 0 a 10 foram exibidos")
10
```

0
2
4
6


Comando **continue** na repetição

- O comando **continue** funciona de maneira parecida com o **break**, porém o **break** interrompe e sai do *loop*;
- Já o **continue** faz com que a próxima iteração comece a ser executada, não importando se existem mais comandos depois dele ou não
- O **continue** não sai do *loop*
- O **continue** faz com que a próxima iteração seja executada imediatamente

Comando **continue** na repetição

- Exemplo com **continue**

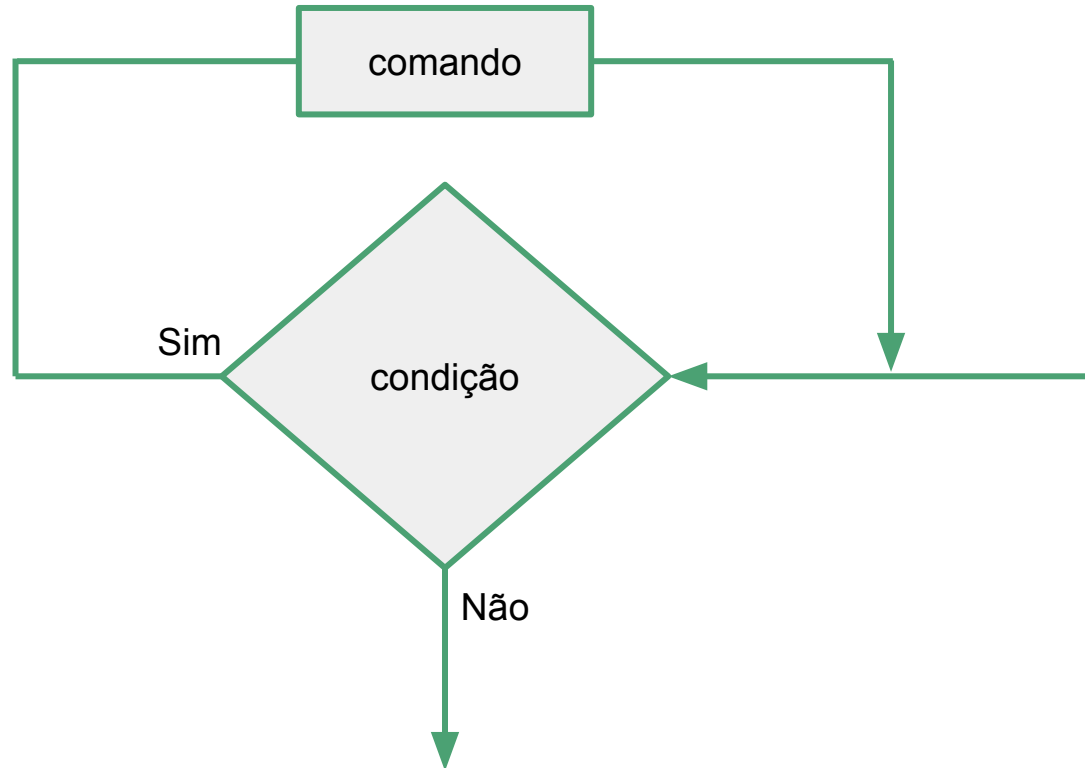
```
1 i = 0
2
3 while i < 12:
4     i+=2
5     if i == 8:
6         continue
7     print(i)
8 else:
9     print("Os números pares de 2 a 12 foram exibidos, com exceção do 8")
```



- Chama a próxima iteração
- Não executa os comandos da própria iteração: neste caso pula o `print()` com `i = 8`

```
2
4
6
10
12
Os números pares de 2 a 12 foram exibidos, com exceção do 8
```

Símbolo no fluxograma - repetição



Pseudocódigo

ENQUANTO ($a < b$) **FAÇA**

Comandos para condição verdadeira

FIM-ENQUANTO

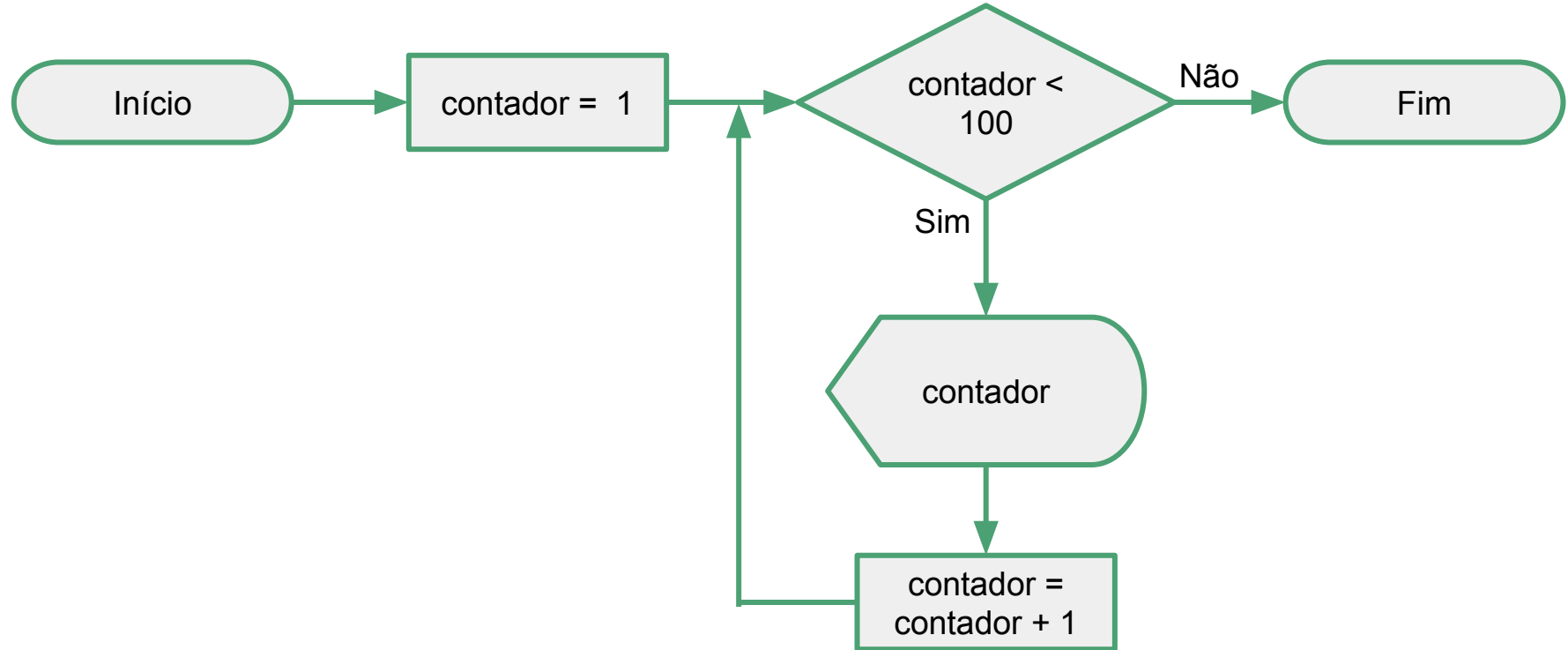
ou

WHILE ($a < b$) **DO**

Comandos para condição verdadeira

END-WHILE

Símbolo no fluxograma - Exemplo



Pseudocódigo - Exemplo

INÍCIO

contador = 1

ENQUANTO (contador < 100) **FAÇA**

ESCREVA contador

contador = contador + 1

ENQUANTO-FIM

FIM

Exercícios

5. Escreva um programa que calcula a média aritmética de 5 números digitados pelo usuário. Utilize contadores e acumuladores.

considerando que o primeiro número digitado é n_1 , o segundo n_2 , ...

$$media = \frac{(n_1+n_2+n_3+n_4+n_5)}{5}$$

6. Escreva um programa que leia números digitados pelo usuário. O programa deve ler os números até que 0 (zero) seja digitado. Quando 0 for digitado, o programa deve exibir a quantidade de dígitos que foram digitados, a somatória destes dígitos e a média aritmética.

7. Um zoológico em particular determina o preço da entrada com base na idade do visitante. Os visitantes com 2 anos de idade ou menos têm entrada gratuita. Crianças entre 3 e 12 anos pagam R\$ 14,00. Idosos com 65 anos ou mais pagam R\$ 18,00. A entrada para todos os outros visitantes custa R\$ 23,00.

Crie um programa que comece lendo as idades de todos os visitantes de um mesmo grupo, sendo uma idade informada em cada linha. O usuário digitará uma linha em branco para indicar que não há mais pessoas no grupo. Em seguida, seu programa deve exibir o custo total para o grupo. O custo deve ser exibido usando duas casas decimais.

8. Escreva um programa que calcule o perímetro de um polígono. Comece recebendo do usuário os valores de x e y para o primeiro ponto do polígono. Em seguida, continue lendo pares de valores x e y até que o usuário insira uma linha em branco para a coordenada x. Cada vez que você lê uma coordenada adicional, você deve calcular a distância até o ponto anterior e adicioná-la ao perímetro. Quando uma linha em branco for inserida para a coordenada x, seu programa deve adicionar a distância do último ponto de volta ao primeiro ponto. Em seguida, ele deve exibir o perímetro total. Um exemplo de entrada e saída é mostrado abaixo, com a entrada do usuário em negrito:

Digite o x da coordenada: **0**

Digite o y da coordenada: **0**

Digite o x da coordenada (em branco para sair): **1**

Digite o y da coordenada: **0**

Digite o x da coordenada (em branco para sair): **0**

Digite o y da coordenada: **1**

Digite o x da coordenada (em branco para sair):

O perímetro desse polígono é 3.414213562373095

9. Escreva um programa que leia um valor inteiro n ($1 < n < 1000$). $n*2$ linhas de saída serão apresentadas na execução do programa, seguindo a lógica do exemplo abaixo.

Saída: Imprima a saída conforme o exemplo fornecido.

| Exemplo de Entrada | Exemplo de Saída |
|--------------------|---|
| 5 | 1 1 1 1 2 2 2 4 8 2 5 9 3 9 27 3 10 28 4 16 64 4 17 65 5 25 125 5 26 126 |

10. A seguinte sequência de números 0 1 1 2 3 5 8 13 21... é conhecida como série de Fibonacci. Nesta sequência, cada número, depois dos 2 primeiros, é igual à soma dos 2 anteriores. Escreva um algoritmo que leia um inteiro N ($N < 46$) e mostre os N primeiros números dessa série.

Saída: Os valores devem ser mostrados na mesma linha, separados por um espaço em branco. Não deve haver espaço após o último valor.

Exemplo:

| Exemplo de Entrada | Exemplo de Saída |
|--------------------|------------------|
| 5 | 0 1 1 2 3 |