

# Estrutura de controle repetitiva: Enquanto

Patrícia de Siqueira Ramos

UNIFAL-MG, *campus* Varginha

17 de Agosto de 2020

- Imagine que queremos imprimir todas as potências de 2 ( $2^{**0}, 2^{**1}, \dots, 2^{**10}$ ) na tela
- Faríamos algo do tipo:

Início

```
Escreva(2 ** 0)
```

```
Escreva(2 ** 1)
```

```
Escreva(2 ** 2)
```

```
.
```

```
.
```

```
.
```

```
Escreva(2 ** 10)
```

Fim

- Mas, dessa forma seria muito trabalhoso

# Estruturas repetitivas

- Quando queremos repetir uma mesma ação várias vezes (ou com poucas modificações) podemos usar a estrutura repetitiva
- As estruturas repetitivas mais utilizadas são:
  - enquanto (*while*)
  - repita (*repeat*)
  - para (*for*)
- Primeiro veremos o enquanto: enquanto uma condição permanece verdadeira, as ações são executadas e, quando ela se torna falsa, o comando é abandonado

# Sintaxe do Enquanto

```
Enquanto <condição a ser avaliada> faça  
    comando 1  
    comando 2  
    .  
    .  
    .  
FimEnquanto
```

## Ex.: Potências de 2 usando Enquanto

Para implementar um algoritmo que imprima as potências de 2 (elevando 2 a 0, depois 2 a 1, e assim por diante até 2 elevado a 10) usando o Enquanto seria da seguinte forma:

## Ex.: Potências de 2 usando Enquanto

Para implementar um algoritmo que imprima as potências de 2 (elevando 2 a 0, depois 2 a 1, e assim por diante até 2 elevado a 10) usando o Enquanto seria da seguinte forma:

- Iniciar uma variável contadora (que eu chamei de  $i$ ) como 0, pois é a primeira potência a ser usada

## Ex.: Potências de 2 usando Enquanto

Para implementar um algoritmo que imprima as potências de 2 (elevando 2 a 0, depois 2 a 1, e assim por diante até 2 elevado a 10) usando o Enquanto seria da seguinte forma:

- Iniciar uma variável contadora (que eu chamei de  $i$ ) como 0, pois é a primeira potência a ser usada
- Quando a estrutura Enquanto começa, uma condição sempre é avaliada e, nesse caso, a condição é que a variável contadora seja menor ou igual a 10

## Ex.: Potências de 2 usando Enquanto

Para implementar um algoritmo que imprima as potências de 2 (elevando 2 a 0, depois 2 a 1, e assim por diante até 2 elevado a 10) usando o Enquanto seria da seguinte forma:

- Iniciar uma variável contadora (que eu chamei de  $i$ ) como 0, pois é a primeira potência a ser usada
- Quando a estrutura Enquanto começa, uma condição sempre é avaliada e, nesse caso, a condição é que a variável contadora seja menor ou igual a 10
- Sendo a condição verdadeira, as instruções que estão dentro do Enquanto são executadas:
  - Escrever na tela o valor de  $2 * 2^i$
  - Alterar o valor de  $i$  para o próximo



## Ex.: Potências de 2 usando Enquanto

Para implementar um algoritmo que imprima as potências de 2 (elevando 2 a 0, depois 2 a 1, e assim por diante até 2 elevado a 10) usando o Enquanto seria da seguinte forma:

- Iniciar uma variável contadora (que eu chamei de  $i$ ) como 0, pois é a primeira potência a ser usada
- Quando a estrutura Enquanto começa, uma condição sempre é avaliada e, nesse caso, a condição é que a variável contadora seja menor ou igual a 10
- Sendo a condição verdadeira, as instruções que estão dentro do Enquanto são executadas:
  - Escrever na tela o valor de  $2 * 2^i$
  - Alterar o valor de  $i$  para o próximo
- A condição dentro do Enquanto é avaliada novamente (se for verdadeira, o processo continua, senão, ele termina)

## Ex.: Potências de 2 usando Enquanto

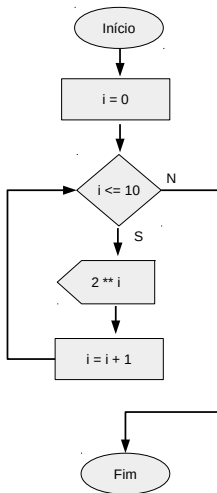
Um exemplo em pseudocódigo do algoritmo seria:

```
Início
  Inteiro: i
  i = 0
  Enquanto i <= 10 faça
    Escreva(2 ** i)
    i = i + 1
  FimEnquanto
Fim
```

Obs.: Como a condição é avaliada antes de entrar no *loop* (laço), o comportamento é parecido com o Se Então

## Ex.: Potências de 2 usando Enquanto

O mesmo algoritmo em fluxograma seria:



# Ex.: Potências de 2 usando Enquanto

Ao executar esse algoritmo:

- A variável começa valendo 0 ( $i = 0$ )
- Como a condição é verdadeira ( $i \leq 10$ ), as instruções dentro do Enquanto são executadas
  - Na tela aparece 1 ( $2 * 0$ )
  - $i$  passa a ser 1 ( $i = i + 1$ )
- A condição dentro do Enquanto é avaliada novamente e como  $1 \leq 10$  é verdadeira, as instruções dentro do Enquanto são executadas
  - Na tela aparece 2 ( $2 * 1$ )
  - $i$  passa a ser 2 ( $i = i + 1$ )
- A condição dentro do Enquanto é avaliada novamente e como  $2 \leq 10$  é verdadeira, as instruções dentro do Enquanto são executadas
  - Na tela aparece 4 ( $2 * 2$ )
  - $i$  passa a ser 3 ( $i = i + 1$ )
- Esse mesmo processo vai sendo repetido até  $i$  ser igual a 11:
  - A condição dentro do Enquanto é avaliada novamente e, como  $11 \leq 10$  é falsa, o algoritmo é finalizado

# Exemplo da potência de 2: python

Abaixo a implementação em *python* e como ficaria a execução:

```
[ ] # exemplo 1
    i = 0
    while i <= 10:
        print(2 ** i)
        i = i + 1
```

```
↳ 1
   2
   4
   8
  16
  32
  64
 128
 256
 512
1024
```

# Situações a se pensar

a) se a condição for falsa da primeira vez?

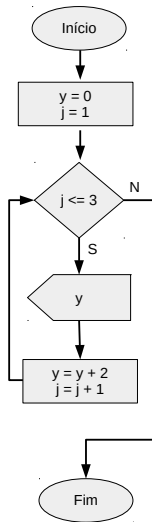
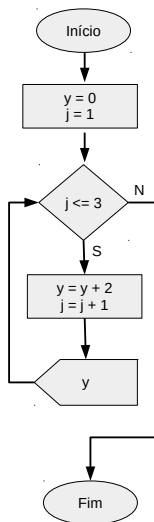
```
i = 0
Enquanto i != 0 faça
    Escreva(i)
    i = i + 1
FimEnquanto
```

b) se a condição for sempre verdadeira?

```
i = 0
Enquanto i >= 0 faça
    Escreva(i)
    i = i + 1
FimEnquanto
```

# Diferentes saídas do Enquanto

Quais serão as saídas dos dois fluxogramas?



# Exemplo: python

```
[ ] # primeiro
    y = 0
    j = 1
    while j <= 3:
        y = y + 2
        j = j + 1
    print(y)
```

2  
4  
6

```
▶ # segundo
   y = 0
   j = 1
   while j <= 3:
       print(y)
       y = y + 2
       j = j + 1
```

0  
2  
4



# O que esse algoritmo faz? Quando ele pára?

Início

Real: z

Escreva('Insira um valor:')

Leia(z)

Enquanto z >= 0 faça

    Escreva('Raiz =', z \*\* 0.5)

    Escreva('Insira um valor:')

    Leia(z)

FimEnquanto

Fim

## Exemplo: python

```
# recebe números e retorna a raiz enquanto não negativos
z = int(input('Insira um valor: '))
while z >= 0:
    print('raiz:', round(z ** 0.5, 2))
    z = int(input('Insira um valor: '))
```

```
↳ Insira um valor: 3
   raiz: 1.73
   Insira um valor: 9
   raiz: 3.0
   Insira um valor: -2
```

Obs.: a função `round` arredonda o número passado como argumento (no exemplo eu quero arredondar o valor da raiz quadrada de `z` para 2 casas decimais)

# Exercício 1

Implemente um algoritmo que faça o seguinte: recebe uma sequência de números terminada em zero informada pelo usuário e retorna a soma dos números. A sequência pode ser de qualquer tamanho.

Por exemplo:

- se o usuário inserir

2, 5, 7, -3, 4, 5, 0

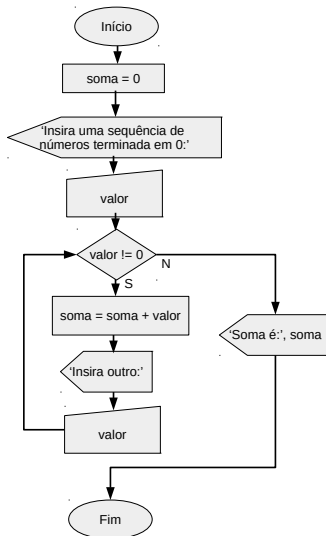
o algoritmo deve retornar que a soma é 20

- se o usuário inserir

0

o algoritmo deve retornar 0

# Exercício 1: soma de sequência terminada em 0



# Exercício 1: python

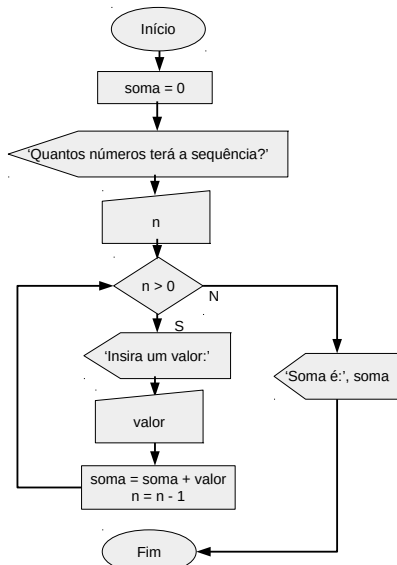
```
# soma de uma sequência de números terminada em 0
# executar a célula e inserir um número da sequência de cada vez
# na caixa que aparecer abaixo do código e
# pressionar ENTER
print('Insira uma sequência de números terminada em 0')
soma = 0
valor = float(input('Insira um valor: '))
while valor != 0:
    soma = soma + valor
    valor = float(input('Insira um valor: '))
print('A soma é:', soma)
```

```
↳ Insira uma sequência de números terminada em 0
Insira um valor: 3
Insira um valor: -3
Insira um valor: 9
Insira um valor: 0
A soma é: 9.0
```

## Exercício 2

Modifique o algoritmo anterior para que o usuário informe, no início, quantos números vai inserir. Depois ele insere os valores (sem o zero no fim) e o algoritmo retorna a soma.

## Exercício 2: soma de sequência ( $n$ definido)



## Exercício 2: python

```
▶ # exercício 2: o usuário informa quantos números vai inserir
n = int(input('Quantos números vai inserir? '))
soma = 0
while n > 0:
    valor = int(input('Informe o valor a ser somado: '))
    soma = soma + valor
    n = n - 1
print('A soma dos valores digitados é:', soma)
```

```
↳ Quantos números vai inserir? 3
Informe o valor a ser somado: 4
Informe o valor a ser somado: -2
Informe o valor a ser somado: 3
A soma dos valores digitados é: 5
```



## Exercício 3

Implemente um algoritmo que faça o seguinte: recebe uma sequência de números terminada em zero informada pelo usuário e retorna o produto dos números, desconsiderando o zero.

Por exemplo:

- se o usuário inserir

3, 2, -3, 0

o algoritmo deve retornar que o produto é  $-18$

- se o usuário inserir

0

o algoritmo deve retornar 0

## Exercício 3: produto de sequência terminada em 0

```
Início
Real: valor, produto
Escreva('Insira uma sequência terminada por 0:')
Leia(valor)
Se valor == 0 Então
    produto = 0
Senão
    produto = 1
Enquanto valor != 0 faça
    produto = produto * valor
Escreva('Insira um valor:')
Leia(valor)
FimEnquanto
FimSe
Escreva('O produto é', produto)
Fim
```

## Exercício 3: python

```
[ ] # exercício 3: produto - corrigindo o erro
print('Digite uma sequência de valores terminada por 0')
valor = float(input('Informe o valor a ser multiplicado: '))
if valor == 0:
    produto = 0
else:
    produto = 1
    while valor != 0:
        produto = produto * valor
        valor = int(input('Informe o valor a ser multiplicado: '))
print('O produto dos valores digitados é:', produto)
```

☞ Digite uma sequência de valores terminada por 0  
Informe o valor a ser multiplicado: 3  
Informe o valor a ser multiplicado: 4  
Informe o valor a ser multiplicado: -2  
Informe o valor a ser multiplicado: 0  
O produto dos valores digitados é: -24.0

# Desafio

Usando a estrutura `Enquanto`, somar os dígitos de um número inteiro inserido pelo usuário. Por exemplo: se o usuário inserir 25332, o algoritmo deve retornar 15, que é a soma dos dígitos ( $2 + 5 + 3 + 3 + 2$ ).

# Desafio

- A chave para resolver essa questão é pensar na divisão por 10
- Por exemplo: pense no número 34

# Desafio

- A chave para resolver essa questão é pensar na divisão por 10
- Por exemplo: pense no número 34

$$\begin{array}{r} 34 \overline{) 10} \\ \underline{30} \phantom{0} \\ 4 \phantom{0} \end{array}$$

resto da divisão  
por 10

$$\begin{array}{r} 34 \overline{) 10} \\ \underline{30} \phantom{0} \\ 4 \phantom{0} \end{array}$$

resultado  
da divisão  
por 10

# Desafio

- A chave para resolver essa questão é pensar na divisão por 10
- Por exemplo: pense no número 34

$$\begin{array}{r} 34 \overline{) 10} \\ \underline{30} \phantom{0} \\ 4 \phantom{0} \end{array}$$

resto da divisão por 10

$$\begin{array}{r} 3 \overline{) 10} \\ \underline{0} \\ 3 \end{array}$$

resultado da divisão por 10

- a divisão inteira de 34 por 10 é 3
- o resto da divisão de 34 por 10 é 4
- a divisão inteira de 3 por 10 é 0
- o resto da divisão de 3 por 10 é 3
- 3 e 4 são os dígitos do número e devem ser somados

# Desafio

- agora pense no número 347

$$\begin{array}{r} 347 \overline{) 10} \\ \underline{340} \quad \text{34} \overline{) 10} \\ \quad 7 \quad \underline{30} \quad \text{3} \overline{) 10} \\ \quad \quad 4 \quad \underline{0} \quad \text{0} \\ \quad \quad \quad 3 \quad \underline{\phantom{0}} \end{array}$$

resto da divisão por 10

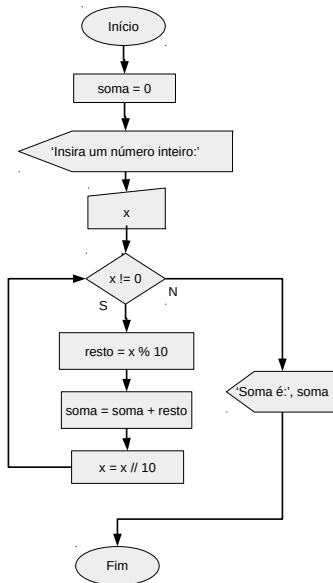
resultado da divisão por 10

Dígitos: 3, 4, 7



- Assim, para resolver o problema é necessário, a cada repetição, verificar se o resultado da divisão inteira do número por 10 é diferente de 0
- Quando esse resultado for 0, quer dizer que o número tem apenas um dígito e a soma já deve ser retornada
- A seguir, um exemplo de algoritmo que resolve o problema

# Desafio: soma dos dígitos de um número



# Desafio

- Um exemplo de execução do algoritmo para  $x = 347$

soma = ~~0~~ ~~7~~ ~~11~~ 14

x = ~~347~~ ~~34~~ ~~3~~ 0

resto = ~~7~~ ~~4~~ 3

# Desafio: python

```
[ ] x = int(input('Digite um número inteiro: '))
    soma = 0
    while x != 0:
        resto = x % 10 # resto da divisão
        soma = soma + resto
        x = x // 10 # divisão inteira
    print(soma)
```

```
☞ Digite um número inteiro: 23456
   20
```