

# Vetores e Matrizes

Fundamentos de Lógica e Algoritmos

# Sumário

- Definição
- Operações com Vetores e Matrizes
- Exemplo
- Desafios

# Definição

# Vetores e Matrizes

- Wikipedia [https://pt.wikipedia.org/wiki/Arranjo\\_\(computação\)](https://pt.wikipedia.org/wiki/Arranjo_(computação))
- “...é uma estrutura de dados que **armazena uma coleção de elementos** de tal forma que cada um dos elementos possa ser **identificado** por, pelo menos, um **índice** ou uma chave.”
- Sinônimos
  - Arranjo, Variável indexada, Vetor, Matriz, Array

# Estrutura de dados

- Estrutura de dados [https://pt.wikipedia.org/wiki/Tipo\\_abstrato\\_de\\_dado](https://pt.wikipedia.org/wiki/Tipo_abstrato_de_dado)
  - “Tipo abstrato de dado (TAD) é uma **especificação de um conjunto de dados e operações** que podem ser executadas sobre esses dados.”
- Coleções [https://pt.wikipedia.org/wiki/Coleção\\_\(computação\)](https://pt.wikipedia.org/wiki/Coleção_(computação))
  - “Uma **coleção** é um grupo de um **número variável de itens** de dados (possivelmente zero) que **têm algum significado** compartilhado com o problema a resolver e necessidade de ser **operados em conjunto**, de alguma maneira controlada.”

# Lista

- Wikipedia <https://pt.wikipedia.org/wiki/Lista>
- “... é uma estrutura de dados abstrata que implementa uma **coleção ordenada de valores**, onde o mesmo valor pode ocorrer mais de uma vez.”

# Operações

# Operações com vetores

- Criar e acesso a itens
- Tamanho e posição de item
- Contém
- Adição e remoção de item
- Conversão
- Outras operações
- Alta ordem
- Mapear, Filtrar e Reduzir



# Criar e acessar itens

## Potigol

```
### Criar vetor (unidimensional)
var meu_vetor := [1, 2, "texto"]

### 1º item -> valor 1
escreva meu_vetor[1]

### 3º item -> valor "texto"
escreva meu_vetor[3]

### Laço em todos os itens
para item em meu_vetor faça
    escreva item
fim
```

## Python

```
### Criar vetor (unidimensional)
meu_vetor = [1, 2, "texto"]

### 1º item -> valor 1
print( meu_vetor[0] )

### 3º item -> valor "texto"
print( meu_vetor[2] )

### Laço em todos os itens
for item in meu_vetor:
    print( item )
```

# Tamanho e posição de item

## Potigol

```
### Criar vetor (unidimensional)  
var meu_vetor := [1, 2, "texto"]
```

```
### tamanho  
meu_vetor.tamanho
```

```
### posição de um item  
meu_vetor.posição(2)  
meu_vetor.posição("texto")  
meu_vetor.posição(10)
```

## Python

```
### Criar vetor (unidimensional)  
meu_vetor = [1, 2, "texto"]
```

```
### tamanho  
len( meu_vetor )
```

```
### posição de um item  
try:  
    meu_vetor.posição(2)  
    meu_vetor.posição("texto")  
    meu_vetor.posição(10)  
except:  
    -1
```

# Contém

## Potigol

```
### Criar vetor (unidimensional)
var meu_vetor := [1, 2, "texto"]

### posição de um item
meu_vetor.contém(2)
meu_vetor.contém("texto")
meu_vetor.contém(10)
```

## Python

```
### Criar vetor (unidimensional)
meu_vetor = [1, 2, "texto"]

### posição de um item
2 in meu_vetor
"texto" in meu_vetor
10 in meu_vetor
```

# Adição e remoção de item

## Potigol

```
### Criar vetor (unidimensional)  
var vetor := [1, 2, "texto"]
```

```
### adicionar um item
```

```
vetor := vetor.insira( 1, ":P" )  
vetor := ":D" :: vetor  
vetor := [4] + vetor  
vetor := vetor.insira( 3, ":D" )  
vetor := vetor.insira( 10, ":D" )
```

```
### remover um item (posição)
```

```
vetor := vetor.remove( 1 )  
vetor := vetor.remove( 10 )
```

## Python

```
### Criar vetor (unidimensional)  
vetor = [1, 2, "texto"]
```

```
### adicionar um item
```

```
vetor.insert( 0, ":P" )  
  
vetor = [4] + vetor  
vetor.append( 2, ":D" )  
vetor.insert( 10, ":D" )
```

```
### remover um item
```

```
vetor.pop( 0 )  
vetor.remove( ":D" )
```

# Conversão

## Potigol

```
### Criar vetor (unidimensional)  
vetor = [1, 2, "texto"]
```

```
### conversão para texto  
escreva vetor.junte( )
```

```
escreva vetor.junte( ":" )
```

```
escreva vetor.insira( ":", "::",  
"::::" )
```

## Python

```
### Criar vetor (unidimensional)  
vetor = [1, 2, "texto"]
```

```
### conversão para texto  
print( "".join( str(item) for  
item in vetor ) )
```

```
print( ":".join( str(item) for  
item in vetor ) )
```

```
print( ":" +  
"::".join( str(item) for item in  
vetor ) + ":::" )
```

# Outras operações

## Potigol

```
### Criar vetor (unidimensional)  
var vetor := [10, 30, 20]
```

```
cabeça = vetor.cabeça()  
cauda = vetor.cauda()  
último = vetor.último()  
escreva vetor.inverta()  
vetor = vetor.ordene  
escreva vetor.pegue(2)  
escreva vetor.descarte(2)
```

## Python

```
### Criar vetor (unidimensional)  
vetor = [10, 30, 20]
```

```
cabeca,*cauda = vetor
```

```
ultimo = vetor[-1:]  
print( vetor[::-1] )  
vetor.sort( )  
print( vetor[:2] )  
print( vetor[2:] )
```

# Operações de alta ordem

- Wikipedia [https://en.wikipedia.org/wiki/Higher-order\\_function](https://en.wikipedia.org/wiki/Higher-order_function)
- "operações que operam sobre outras operações, seja tomando-as como argumentos e/ou retornando-as"
- Comum na programação funcional

# Operações de alta ordem

- Mapear (map) <https://en.wikipedia.org/wiki/Map> (higher-order function)
  - Aplica uma operação a cada item da coleção criando uma nova coleção com o mesmo número de itens (novos itens)
- Filtrar (filter) <https://en.wikipedia.org/wiki/Filter> (higher-order function)
  - Aplica uma operação de resultado boelano a cada item da coleção criando um subconjunto de itens da coleção (itens já existentes)
- Reduzir (fold ou reduce) <https://en.wikipedia.org/wiki/Fold> (higher-order function)
  - Aplica uma operação para analisar os itens e “resumir” a coleção em 1 valor



# Alta ordem : Mapa

## Potigol

```
### Criar vetor (unidimensional)  
var vetor := [2, 4, 6, 8, 10]
```

```
escreva vetor.mapeie(item =>  
item div 2)  
# [1, 2, 3, 4, 5]
```

```
escreva vetor.mapeie( item =>  
str( item ))  
# ["1", "2", "3", "4", "5"]
```

## Python

```
### Criar vetor (unidimensional)  
vetor = [2, 4, 6, 8, 10]
```

```
print( list(map( lambda item :  
item // 2, vetor )) )
```

```
print( list(map( str, vetor )) )
```

# Alta ordem : Filtro

## Potigol

```
### Criar vetor (unidimensional)
var vetor := [2, 4, 6, 8, 10]

escreva vetor.selezione(item =>
item mod 4 == 0)
# [4, 8]
```

## Python

```
### Criar vetor (unidimensional)
vetor = [2, 4, 6, 8, 10]

print( list(filter( lambda item:
item % 4 == 0, vetor )) )
```

# Alta ordem : Reduzir

## Potigol

```
### Criar vetor (unidimensional)
var vetor := [2, 4, 6, 8, 10]

escreva
vetor.injete((acumulador, item)
=> acumulador + item)
# 30
```

## Python

```
from functools import reduce
```

```
### Criar vetor (unidimensional)
vetor = [2, 4, 6, 8, 10]

print( reduce(lambda acumulador,
item: acumulador + item,
vetor) )
```

# Exemplos

# URI Online Judge

**URI Online Judge**  
**1238 Combinador**

# 1° Exemplo

# Saída



# Desafios

# Qual a saída?

# Links

- Definições
  - <https://pt.wikipedia.org/wiki/Arranjo> (computação)
  - <https://pt.wikipedia.org/wiki/Lista>
  - [https://pt.wikipedia.org/wiki/Tipo abstrato de dado](https://pt.wikipedia.org/wiki/Tipo_abstrato_de_dado)
  - [https://en.wikipedia.org/wiki/Higher-order function](https://en.wikipedia.org/wiki/Higher-order_function)
  - [https://en.wikipedia.org/wiki/Map \(higher-order function\)](https://en.wikipedia.org/wiki/Map_(higher-order_function))
  - [https://en.wikipedia.org/wiki/Filter \(higher-order function\)](https://en.wikipedia.org/wiki/Filter_(higher-order_function))
  - [https://en.wikipedia.org/wiki/Fold \(higher-order function\)](https://en.wikipedia.org/wiki/Fold_(higher-order_function))
- Tutoriais
  - <https://codeburst.io/here-are-7-ways-higher-order-functions-can-improve-your-life-a392aa6b29d2>
  - <https://book.pythontips.com/>
  -