



Bora brincar
um pouco?



Variáveis / Constantes

`var`

`let`

Tipos de Dado



- **String** - Textos
- **Int** - Números Inteiros
- **Float** - Número decimal - 32 Bits
- **Double** - Número decimal - (Dobro de Float)
- **Boolean** - Verdadeiro ou Falso

Double vs Float



Nível de precisão de casas decimais

Operações



+ - / *

Print



```
print("Olá, Mundo!")
```

While - (Enquanto)



```
var index = 1
while index <= 10 {
    print("index: \(index)")
    index = index + 1
}
```

Repeat - (Repita enquanto)



```
var index = 1
repeat {
    print("index: \(index)")
    index = index + 1
} while index <= 10
```


Repeat - (Repita enquanto)



```
var index = 1
repeat {
    print("index: \(index)")
    index = index + 1
} while index <= 10
```

For - (para)



```
for index in 1...4 {  
    print("index: \(index)")  
}  
  
for index in 1..<4 {  
  
for index in ["A", "B", "C", "D", "E"] {
```

For - (para)



```
for (index, element) in ["A", "B",  
"C", "D", "E"].enumerated() {
```

If - Else

Comparações

== -> Igual

!= -> Diferente

> -> Maior

< -> Menor

>= -> Maior igual

<= -> Menor igual

Operadores Lógicos

&& -> E

|| -> Ou

! -> Não



Exercícios



1. Declarar a variável “umNumeroA” do tipo **Int** e atribuir um valor a ela.
Declarar a variável “umNumeroB” de tipo **Double** e atribuir um valor a ela.
Declarar la variável “umString” de tipo **String** e atribuir um valor a ela.
Em seguida, imprimir na tela:
 - a. O valor de cada variável.
 - b. A soma de `umNumeroA + umNumeroB`.
 - c. A diferença entre `umNumeroA - umNumeroB`.

2. Imprima na tela o número de segundos existentes em um ano

Exercícios



3. Imprima na tela os dez números naturais elevados ao quadrado

4. Imprima na tela a soma dos primeiros dez números naturais ÍMPARES elevados ao quadrado

5. Imprimir números aleatórios entre 0 e 5, até que se imprima um 3.
Ajuda: A função `arc4random_uniform(_ n: Int)` (definida na biblioteca Foundation) retorna um número aleatório entre 0 e o parâmetro, exemplo.

```
var numero = arc4random_uniform(5)
```



Funções



Uma função é um bloco de código que realiza uma tarefa em específico. Permite encapsular código e reutilizá-lo diversas vezes.

```
func retorneEsseNumero (numero: Int) -> Int { return numero }
```


Exercícios



1. Definir três métodos que imprimam saudações diferentes em tela e sejam executados da seguinte forma:

`cumprimentarA(pessoa: "João")`

`cumprimentar(a: "João")`

`cumprimentarA("João")`

2. Escrever o método **`eMenor(oPrimeiro: Int, oSegundo: Int) -> Bool`** deve usar dois números inteiros como parâmetros e retornar **`true`** se o primeiro número for menor que o segundo número ou **`false`**, caso contrário.

- Invocar o método com os números 3 e 5 e imprimir na tela o resultado.
- Invocar o método com os números 7 e 5 e imprimir na tela o resultado.
- Invocar o método com os números 10 e 10; e imprimir na tela o resultado.

Exercícios



1. Escrever o método ***elmparMaiorQueDez(umNumero: Int) -> Bool*** que deve analisar um número inteiro como parâmetro e retornar ***true*** se o número for ímpar e maior do que dez e ***false***, caso contrário.
 - Invocar o método com o número 3 e imprimir na tela o resultado.
 - Invocar o método com o número 4 e imprimir na tela o resultado.
 - Invocar o método com o número 13 e imprimir na tela o resultado.
 - Invocar o método com o número 14 e imprimir na tela o resultado.
2. Escrever o método ***imprimirlmparesPositivos() -> Void*** que imprima na tela os primeiros 100 números inteiros positivos ímpares.

Links

Cookbook - Hello World

Cookbook - Functions

Cookbook - Array