
cit


Hilton **Pintor**
Desenvolvedor (iOS/tvOS/watchOS)
hiltonpintor@gmail.com
citi

// Aula 02

// Dúvidas da Aula 01

Dúvidas em aberto, esclarecimentos, e correções.

```
// operações de tipos diferentes

var opcional: Int? = 1
var concreto: Int = 1

opcional + concreto ! Value of optional type not unwrapped
```

- Operações devem ser com operandos do mesmo tipo
- Opcional e concretos não são do mesmo tipo
- erro de compilação

```swift
// operações de tipos diferentes

var opcional: Int? = 1
var concreto: Int = 1

opcional! + concreto // 2
```

- é preciso extrair o valor de opcional
- force unwrapping
- para realizar a operação

```
// operações de tipos diferentes


var int: Int = 1
var double: Double = 1

int + double
```
Binary operator '+' cannot be applied to operands of type 'Int' and 'Double

- dois tipos diferentes
- erro de compilação

```swift
// operações de tipos diferentes


var int: Int = 1
var double: Double = 1

Double(int) + double // 2
int + Int(double)  // 2
```

- conversão de um tipo em outro

```
// operações de tipos diferentes


var string: String = "1"
int = 1

string + int
```

Binary operator '+' cannot be applied to operands of type 'String' and ‘Int'

- dois tipos diferentes
- erro de compilação

```swift
// operações de tipos diferentes


var string: String = "1"
int = 1

Int(string)! + int    // 2
string + String(int)  // "11"
```

- converter String em Int
  - gera opcional
  - pode ter ou não um valor numérico na string
- converter de Int para String
  - o + agora é concatenação

```swift
// switch com intervalo

var valor = 11
var mensagem = ""

switch valor {

case Int.min..<0:
    mensagem = "negativo"

case 0...Int.max:
    mensagem = "positivo"

default:
    mensagem = "impossível"
}
```

Int.max = 9223372036854775807
Int.min = -9223372036854775808

https://stackoverflow.com/questions/36476599/swift-switch-statement-considered-all-cases-of-int-but-compiler-still-display-e

- Foi falado sobre usar switch para verificar se um valor pertence a um intervalo.
- Essa é uma forma de se fazer.
- Nunca vai entrar no caso default, mas mesmo assim o compilador obriga.

// compilador não é tão esperto

"Unfortunately, integer operations like **'...'** and **'<'** are just plain functions to Swift, so it'd be difficult to do this kind of analysis.

Even with special case understanding of integer intervals, **I think there are still cases in the full generality of pattern matching for which exhaustiveness matching would be undecidable.**

We may eventually be able to handle some cases, but there will always be special cases involved in doing so."

https://stackoverflow.com/questions/36476599/swift-switch-statement-considered-all-cases-of-int-but-compiler-still-display-e

- Mensagem de um engenheiro da Apple.
- Compilador não é tão inteligente para operadores de intervalo e comparação.
- ele não tem como saber se o switch está exaustivo.

```swift
// switch com intervalo

var valor = 11
var mensagem = ""

switch valor {

case Int.min..<0:
    mensagem = "negativo"

case 0...Int.max:
    mensagem = "positivo"

default:
    mensagem = "impossível"
}
```

Int.max = 9223372036854775807
Int.min = -9223372036854775808

https://stackoverflow.com/questions/36476599/swift-switch-statement-considered-all-cases-of-int-but-compiler-still-display-e

- como o caso default nunca poderá ser atingido
- podemos usar break (ignorar o caso)
- encerra a execução do switch sem realizar nenhuma ação

```swift
// switch com intervalo

var valor = 11
var mensagem = ""

switch valor {

case Int.min..<0:
    mensagem = "negativo"

case 0...Int.max:
    mensagem = "positivo"

default:


}
```

Int.max = 9223372036854775807
Int.min = -9223372036854775808

https://stackoverflow.com/questions/36476599/swift-switch-statement-considered-all-cases-of-int-but-compiler-still-display-e

- como o caso default nunca poderá ser atingido
- podemos usar break (ignorar o caso)
- encerra a execução do switch sem realizar nenhuma ação

```swift
// switch com intervalo

var valor = 11
var mensagem = ""

switch valor {

case Int.min..<0:
    mensagem = "negativo"

case 0...Int.max:
    mensagem = "positivo"

default:
    break
}
```

Int.max = 9223372036854775807
Int.min = -9223372036854775808

https://stackoverflow.com/questions/36476599/swift-switch-statement-considered-all-cases-of-int-but-compiler-still-display-e

- como o caso default nunca poderá ser atingido
- podemos usar break (ignorar o caso)
- encerra a execução do switch sem realizar nenhuma ação

// compilador não é tão esperto

Although `break` is not required in Swift, you can use a `break` statement to match and **ignore a particular case** or to break out of a matched case before that case has completed its execution.

https://developer.apple.com/library/content/documentation/Swift/Conceptual/Swift_Programming_Language/ControlFlow.html

- Break não é necessário, é opcional
- usado para ignorar um caso
  - ou para sair do caso antes do fim da execução

// switch com where

```swift
var someVar = 3

switch someVar {

case let x where x < 0:
   mensagem = "\(x) é menor que zero"

case let x where x == 0:
   mensagem = "\(x) é igual a zero"

case let x where x > 0:
   mensagem = "\(x) é maior que zero"

default:
   break
}
```

https://stackoverflow.com/questions/36476599/swift-switch-statement-considered-all-cases-of-int-but-compiler-still-display-e

- switch com comparações
- é necessário o uso do where
- captura o valor em X, e compara no where
- interpolação de strings
- break

```swift
// switch com Where

switch someVar {

case _ where someVar < 0:
   mensagem = "\(someVar) é menor que zero"

case _ where someVar == 0:
   mensagem = "\(someVar) é igual a zero"

case _ where someVar > 0:
   mensagem = "\(someVar) é maior que zero"

default:
   break
}
```

https://stackoverflow.com/questions/36476599/swift-switch-statement-considered-all-cases-of-int-but-compiler-still-display-e

- não precisamos dar um novo nome ao valor
    - ignoramos com _
    - utilizamos o identificador original

// switch com fallthrough

In Swift, switch statements **don’t fall through** the bottom of each case and into the next one.

That is, the entire switch statement **completes its execution as soon as the first matching case is completed**

https://developer.apple.com/library/content/documentation/Swift/Conceptual/Swift_Programming_Language/ControlFlow.html

- Só cai em um caso.
- finaliza o switch após executar o caso

```swift
// switch com fallthrough

let inteiro = 5
var descricao = "O número \(inteiro) é"
switch inteiro {
case 2, 3, 5, 7, 11, 13, 17, 19:
    descricao += " primo, e também"

default:
    descricao += " um inteiro."
}
descricao // "O número 5 é primo, e também um inteiro."
```

https://developer.apple.com/library/content/documentation/Swift/Conceptual/Swift_Programming_Language/ControlFlow.html

- para fazer com que execute mais de um caso
- que depois de um caia no outro
    - fallthrough

```swift
// switch com fallthrough

let inteiro = 5
var descricao = "O número \(inteiro) é"
switch inteiro {
case 2, 3, 5, 7, 11, 13, 17, 19:
    descricao += " primo, e também"
    fallthrough
default:
    descricao += " um inteiro."
}
descricao // "O número 5 é primo, e também um inteiro."
```

https://developer.apple.com/library/content/documentation/Swift/Conceptual/Swift_Programming_Language/ControlFlow.html

- para fazer com que execute mais de um caso
- que depois de um caia no outro
    - fallthrough

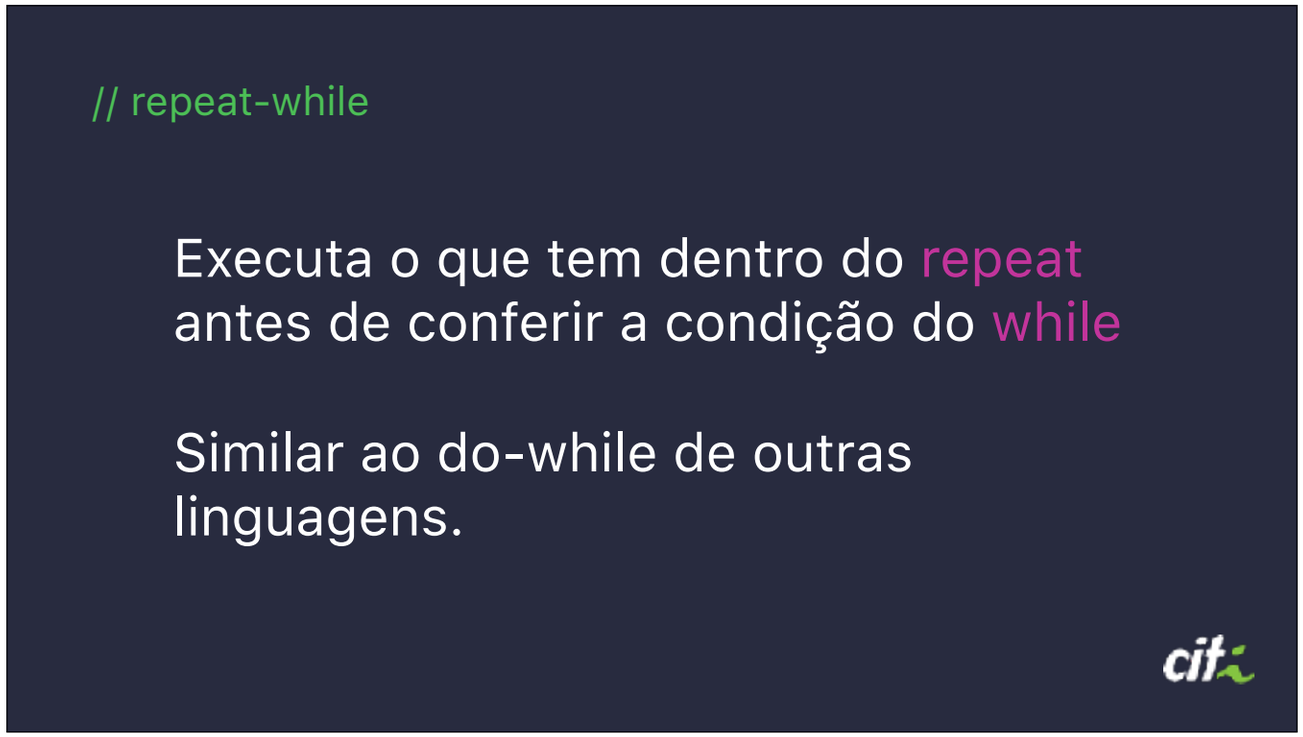
// switch com fallthrough

The fallthrough keyword **does not check** the case **conditions** for the switch case that it causes execution to fall into.

The fallthrough keyword simply causes code execution to **move directly to the statements inside** the next case (or default case) block, as in C's standard switch statement behavior.

https://developer.apple.com/library/content/documentation/Swift/Conceptual/Swift_Programming_Language/ControlFlow.html

- não checa o caso
    - cai direto no corpo do próximo case
    - igual a C



- Foi perguntado sobre do-while
- Repeat
  - Executa a ação
  - depois confere a condição (while)

```swift
// repeat-while

var valorAtual = 0
var valorEsperado = 10

repeat {
    valorAtual = valorAtual + 1

} while valorAtual != valorEsperado

valorAtual  // 10
```


REPETE X **ENQUANTO** Y

- incrementa o valor atual
    - até ser igual ao esperado

// Exercícios

// Exercício 02

Scrabble

No jogo scrabble, procura-se fazer palavras com o máximo de pontos. Cada letra da palavra tem uma pontuação

Letter	Value
A, E, I, O, U, L, N, R, S, T	1
D, G	2
B, C, M, P	3
F, H, V, W, Y	4
K	5
J, X	8
Q, Z	10

Faça uma função que dado uma palavra, retorne a sua pontuação correspondente.

adaptado de: http://exercism.io/exercises/swift/scrabble-score/readme

// Exercício 03

Calculadora Escrita

Avalie o resultado das seguintes operações:

```
let soma = "Quanto é 5 mais 13?”
let subtracao = "Quanto é 15 menos 10?”
let multiplicacao = "Quanto é 1 vezes 13?”
let divisao = "Quanto é 15 menos 5?”
```

O resultado da expressão deve ser impresso (print) no console.

//Dica: procure por split

adaptado de: http://exercism.io/exercises/swift/wordy/readme

// Exercício 04

Dígitos

Implemente uma **função recursiva** chamada digitos, que recebe como entrada um numero positivo e retorna um Array contendo os dígitos desse número, em ordem.

// exemplo:

```
digitos(213)    // retorna [2, 1, 3]
```

adaptado de: https://www.weheartswift.com/recursion/

// Exercício 05-a

Array de Dicionários

Dado um **array de dicionários**, onde cada dicionário contem exatamente duas chaves: "nome", e "sobrenome".

Armazene em uma variável nomes, todos os nomes contidos nos dicionários, e em uma variável sobrenomes, todos os sobrenomes contidos nos dicionários.

// exemplo:

```
var pessoas: [[String:String]] = [
  [
    "nome": "Hilton",
    "sobrenome": "Pintor"
  ],
  [
    "nome": "Elton",
    "sobrenome": "Santana"
  ],
  [
    "nome": "Daniel",
    "sobrenome": "Oliveira"
  ],
  [
    "nome": "Clarissa",
    "sobrenome": "Pessoa"
  ],
  [
    "nome": "Fanny",
    "sobrenome": "Chien"
  ]
]
```

adaptado de: https://www.weheartswift.com/recursion/

// Exercício 05-b

Array de Dicionários

construindo acima do exemplo anterior, construa um array nomeCompleto onde cada elemento é uma string que contem o nome e sobrenome de cada pessoa separados por espaço.

adaptado de: https://www.weheartswift.com/recursion/


DÚVIDAS
?
citi