

FIAP GRADUAÇÃO

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

Desenvolvimento Avançado iOS

PROF. GUSTAVO CALIXTO

Introdução ao Swift

INTRODUÇÃO AO SWIFT

- Aula de Hoje
 - Entendimento geral sobre funções.
- Git com códigos de exemplo
 - <https://github.com/gmcalixto/ios>

INTRODUÇÃO AO SWIFT

- As funções em Swift permitem diferentes possibilidades de valores de entrada em argumentos e valores de retorno, tais como:
 - Valores simples;
 - Tupla de dados;
 - Funções;
 - Lista indeterminada de valores;
 - Uso do Optional

INTRODUÇÃO AO SWIFT

Funções com retorno simples

- func -> palavra reservada para função
- lista de parâmetros entre () ;
- retorno da função identificado por "->"

```
func sayHello(name: String, day: String, month: String) ->
    String
{
    return "Hello \(name)! Today is \(day) \(month)"
}

print(sayHello(name:"Pessoa", day: "25", month: "April"))
```

INTRODUÇÃO AO SWIFT

Função retornando tupla de dados como retorno

```
func sendNumbers(numbers: [Int]) -> (min: Int, max: Int, sum: Int){  
    var max = numbers[0]  
    var min = numbers[0]  
    var sum = 0  
  
    for number in numbers{  
        if number > max{  
            max = number  
        }  
        else if number < min{  
            min = number  
        }  
  
        sum += number  
    }  
    return (min,max,sum)  
}  
  
let resultado = sendNumbers(numbers: [10,40,70,25,35,76])  
print("Menor:\(resultado.min) Maior: \(resultado.max) Soma: \  
      (resultado.sum)")
```

Tupla no retorno da função

INTRODUÇÃO AO SWIFT

Função retornando tupla de dados como argumento

```
func sendNumbers(numbers: (num1: Int, num2: Int, num3: Int)) ->
    (min: Int, max: Int, sum: Int){
    let max = numbers.num1
    let min = numbers.num2
    return (min*2, max*4, numbers.num1+numbers.num2+numbers.num3)
}

var resulFunc = sendNumbers(numbers: (1,2,3))
print("Menor: \(resulFunc.min) Maior: \(resulFunc.max) Soma: \
      \(resulFunc.sum)")
```

Tupla como argumento

INTRODUÇÃO AO SWIFT

Função com entrada de argumentos variáveis (de mesmo tipo)

```
func avgNumbers(numbers: Float...) -> Float{  
    var sum:Float = 0.0  
    for number in numbers{  
        sum += Float(number)  
    }  
  
    return sum/Float(numbers.count)  
}  
  
let resultadoavg = avgNumbers(numbers:10,5)  
  
print ("Media: \(resultadoavg)")
```

Quanto argumentos desejar. A
entrada se torna uma coleção
de dados

INTRODUÇÃO AO SWIFT

Funções dentro de função

```
func calcEstat(numeros: [Float]) -> (media: Float, soma: Float)
{
    func media(_numeros:[Float]) -> Float{
        var sum:Float = 0.0
        for number in _numeros{
            sum += Float(number)
        }

        return sum/Float(_numeros.count)
    }
    func soma(_numeros:[Float]) -> Float{
        var sum:Float = 0.0
        for number in _numeros{
            sum += Float(number)
        }
        return sum
    }

    return (media(_numeros: numeros),soma(_numeros: numeros))
}

let resultadocalc = calcEstat(numeros: [25,30,40,50])
print("Media: \(resultadocalc.media) Moda: \
(resultadocalc.soma)")
```

Vetor como parâmetro de entrada. Passagem por valor

Funções internas são reconhecidas somente dentro do escopo da função externa

INTRODUÇÃO AO SWIFT

Funções passando função como retorno

- A indicação de uma função como parâmetro é dada por "(argumentos)->retorno".

```
func dobraNumero() -> ((Int)->Int){  
    func dobrar(numero:Int) -> Int{  
        return numero*2  
    }  
    return dobrar  
}  
  
let resultTransf = dobraNumero()  
  
print("Dobro: \(resultTransf(7))")
```

O retorno é o nome da função criada

Criação de variável parametrizável

INTRODUÇÃO AO SWIFT

Função passando funções como argumentos

```
func temMedia(nota:Float) -> Bool{
    return (nota >= 6.0)
}
func tirou10(nota:Float) -> Bool{
    return (nota == 10)
}
func retornaCondicao(fn: (Float)->Bool,notas: [Float]) ->
[Bool]{

    var retorno = [Bool]()

    for nota in notas{
        retorno.append(fn(nota))
    }
    return retorno
}
let notaAlunos:[Float] = [10,9,8,7,6,4,3,7,5,4]

var alunosAprovados = retornaCondicao(fn: temMedia,notas:
notaAlunos)

var alunosCom10 = retornaCondicao(fn: tirou10,notas:
notaAlunos)
print(alunosAprovados)
print(alunosCom10)
```

Funções como argumentos

Permite que qualquer função com
uma determinada assinatura seja
utilizada

INTRODUÇÃO AO SWIFT

Closures: implementação de funções em linha sem a necessidade de assinatura

```
let valor = {(numero:Int) -> Int in
    var numNovo = 3*numero;
    return numNovo
}
print(valor(5))

//passando função já pronta
var saberMedia = temMedia
print(saberMedia(5))

//implementando com closures
saberMedia = {(nota:Float)->Bool in
    return (nota >= 6.0)
}
print(saberMedia(10))
```

{(parâmetros)->retorno in
Implementação
}

Comparando o uso de closures e
função já implementada

INTRODUÇÃO AO SWIFT

Uso de operadores Optional em funções

```
func soma(_numeros:[Float]) -> Float?{
    if _numeros.isEmpty
    {
        return nil
    }
    else
    {
        var sum:Float = 0.0
        for number in _numeros{
            sum += Float(number)
        }
        return sum
    }
}

var vetSoma = [Float]()

print(soma(_numeros: vetSoma) ?? 0.0)

vetSoma.append(10)
vetSoma.append(15)

print(soma(_numeros: vetSoma) ?? 0.0)
```

Indica que a função pode retornar valor nulo

INTRODUÇÃO AO SWIFT

Passagem por referência através da palavra *inout*

```
var vet = [1,2,3,4,5,6,7,8,9]

funcdobraVetor( vetor:<inout[Int]>){
    var count = 0
    for elemento in vetor{
        vetor[count] = elemento * 2
        count += 1
    }
}

dobraVetor(vetor: &vet)

print(vet)
```

Uso da palavra reservada
"inout"

INTRODUÇÃO AO SWIFT

Próxima aula

- Classes e objetos
- Chamadas de função com objetos genéricos
- Extensões e Protocolos

Copyright © 2019 Prof. Gustavo Moreira Calixto

Todos direitos reservados. Reprodução ou divulgação total ou parcial deste documento é expressamente proibido sem o consentimento formal, por escrito, do Professor (autor).