



---

---

---

---

---

---

---

### Introdução ao Swift

- Criada pela Apple em 2014
- Substitui o ObjectiveC
- Mais rápida e mais segura
- Novos recursos
- Principal linguagem de desenvolvimento para o Mac
- Utilização do Xcode neste módulo do curso

---

---

---

---

---

---

---

### Principais tipos de dados

- String (textos)

```
var minhaString : String = String() // variável String apenas declarada
var minhaString : String = "" // variável String inicializada vazia
var minhaString : String = "texto" // variável String com algum texto
minhaString = minhaString + " escrito" // variável adicionando texto a ela mesma
```
- int

```
var meuInt : Int // variável do tipo Int apenas declarada
var meuInt : Int = 50 // variável do tipo Int inicializada
```
- float

```
var meuFloat : Float // variável do tipo Float apenas declarada
var meuFloat : Float = 50.25 // variável do tipo Float já inicializada
```
- double

```
var meuDouble : Double // variável do tipo Double apenas declarada
var meuDouble : Double = 50.25252 // variável do tipo Double já inicializada
```
- bool

```
var meuBool : Bool // variável do tipo Bool apenas declarada
var meuBool : Bool = true // variável do tipo Bool já inicializada
```

---

---

---


---

---

---

---

Arrays



- Utilização de colchetes e o número de posições

```

var meuArrayInt : [Int] = [54, 55, 56, 57, 58, 0, 0, 0, 0, 0]
// Cria um array de 10 posições

var meuInt : Int = 2 * meuArrayInt[2]
// Armazena o valor 56 * 2 (112) em meuInt

```

0	1	2	3	4	5	6	7	8	9
54	55	56	57	58	0	0	0	0	0

---

---

---

---


---

---

---

---

Constantes



- Uso do comando let no lugar de var

```

let MINHA_CONSTANTE : Int = 1024
let PI : Float = 3.14

```

---

---

---

---


---

---

---

---

Operadores e comandos



- Operadores matemáticos
- Operadores condicionais
- Operadores lógicos
- Comandos de controle
- Comandos de repetição

---

---

---

---


---

---

---

---

## Operadores matemáticos



Para os exemplos apresentados, assuma:

```
var x: Double // Variável "x" declarada como do tipo de dado Double
```

- = Atribui o valor da sua direita à variável a sua esquerda
 

```
x = 3 // Resultado: x armazena o valor 3
```
- + Soma dois valores numéricos
 

```
x = 3 + 5 // Resultado: x armazena o valor 8
```
- Subtrai dois valores numéricos
 

```
x = 3 - 5 // Resultado: x armazena o valor -2
```
- \* Multiplica dois valores numéricos
 

```
x = 2 * 6 // Resultado: x armazena o valor 12
```
- / Divide dois valores numéricos
 

```
x = 18 / 3 // Resultado: x armazena o valor 6
```
- % Obtém o resto da divisão entre dois valores numéricos
 

```
x = 19 % 3 // Resultado: x armazena o valor 1
```

---

---

---

---


---

---

---

---

## Operadores matemáticos



- ++ Incrementa em 1 o valor da variável acoplada
 

```
x = 3
x++ // Resultado: x neste momento armazena o valor 4
```
- Decrementa em 1 o valor da variável acoplada
 

```
x = 3
x-- // Resultado: x neste momento armazena o valor 2
```
- É possível utilizar estes operadores em expressões e outros comandos
 

```
x = 3
y = 2 + x++ // Resultado: y = 5 e x = 4
```
- Diferença entre ++x e x++
 

```
x = 3
y = 2 ++x // Resultado: y = 6 e x = 4
```

---

---

---

---


---

---

---

---

## Operadores matemáticos



Assumir x = 5 para o início de cada exemplo apresentado neste slide

- += Soma à variável à sua esquerda o valor apresentado em sua direita
 

```
x += 3 // Resultado: x armazena o valor 8
```
- = Subtrai da variável à sua esquerda o valor apresentado em sua direita
 

```
x -= 3 // Resultado: x armazena o valor 2
```
- \*= Multiplica a variável à sua esquerda pelo valor apresentado em sua direita
 

```
x *= 3 // Resultado: x armazena o valor 15
```
- /= Divide à variável à sua esquerda o valor apresentado em sua direita
 

```
x /= 3 // Resultado: x armazena o valor 1.6666666666667
```
- %= Atribui à variável da esquerda o resto de sua divisão pelo valor apresentado em sua direita
 

```
x %= 3 // Resultado: x armazena o valor 2
```

---

---

---

---

---

---

---

---

## Operadores condicionais



- Utilizados para avaliar uma condição

Operador	Significado
==	Compara se dois valores tem o mesmo valor, mesmo em tipos distintos
===	Compara se dois valores são idênticos
!=	Compara se dois valores são diferentes
!==	Compara se dois valores são diferentes inclusive no tipo de dado
!	Inverte o resultado (negação)
<	Compara se o valor da esquerda é menor que o da direita
>	Compara se o valor da esquerda é maior que o da direita
<=	Compara se o valor da esquerda é menor ou igual que o da direita
>=	Compara se o valor da esquerda é maior ou igual que o da direita

---

---

---

---

---

---

---

---

## Operadores lógicos



- Utilizados para montar uma condição

Operador	Significado
!	Nega a condição informada (inverte seu resultado lógico)
&	Retorna verdadeiro se ambas as condições da esquerda e direita forem satisfeitas
	Retorna verdadeiro se pelo menos uma das condições (da direita ou da esquerda) for satisfeita
&&	O mesmo que &, porém é otimizado: Se a condição do lado esquerdo for falsa, não verifica o lado direito da expressão. Resulta a operação toda em FALSA neste caso.
	O mesmo que  , porém é otimizado: Se a condição do lado esquerdo for verdadeira, não verifica o lado direito da expressão. Resulta a operação toda em VERDADEIRA neste caso.

---

---

---

---

---

---

---

---

## Comandos de controle e repetição



- `if(condição) {} [else if(condição) {}] [else {}]`
  - Permite executar um bloco de código se determinada condição for verdadeira

```
if(3 > 5) {  
    // Não entra no primeiro if  
}  
  
if(1 < 10) {  
    // Entra no segundo if  
}  
else {  
    // Não entra no else do segundo if  
}
```

```
var i : Int = 5  
if(i == 3)  
{  
    // O valor de i é 3  
}  
else if(i == 4)  
{  
    // O valor de i é 4  
}  
else {  
    // O valor de i não é 3 nem 4.  
}
```

---

---

---

---

---

---

---

---

## Comandos de controle e repetição



- Short if (if reduzido)
  - Forma reduzida de escrever o comando if
  - Deve ser utilizado somente em casos onde o bloco de código for curtíssimo, geralmente com definição de variáveis
- `condição ? bloco_true : bloco_false`

```
var x : Int = 5
var y : Int

if(x % 2 == 0) {
  // Caso x seja par
  y = 60;
} else {
  // Caso x seja ímpar
  y = 61;
}
```

```
var x : Int = 5
var y : Int

y = x % 2 == 0 ? 60 : 61

// Se x for par, atribuirá 60, caso contrário 61
```

## Comandos de controle e repetição



- `switch(valor) { cases }`
  - Permite executar um bloco de código específico dependendo do valor avaliado

```
var i : Int = 1
switch(i)
{
  case 0:
    // O valor de i é 0
    break
  case 1:
    // O valor de i é 1
    break
  case 2:
    // O valor de i é 2
    break
  default:
    // Nenhum
    break
}
```

## Comandos de controle e repetição



- `for inicialização; condição; incremento`
  - Permite criar um laço de repetição, executando para cada valor o mesmo bloco de código

```
var i : Int

for i = 0; i <= 10; ++i
{
  // Exibir i
}

// Resultado: 0 1 2 3 4 5 6 7 8 9 10
```

- Este mesmo comando poderia ser escrito desta forma:

```
for i in 1..10
{
  // Exibir i
}
```

## Comandos de controle e repetição



- **while(condição)**

- Permite executar várias vezes um bloco de código enquanto sua condição for verdadeira

```
var i : Int = 0
while(i < 5)
{
    // Exibir i
    i++
}
// Resultado: 0 1 2 3 4
```

---

---

---

---

---

---

---

## Comandos de controle e repetição



- **do { } while (condição)**

- Similar ao while, com a diferença de que o do while tem o bloco de código executado obrigatoriamente uma vez antes de avaliar a condição

```
var i : Int = 0
do
{
    // Exibir i
    i++
}
while(i < 5)
// Resultado: 0 1 2 3 4
```

---

---

---

---

---

---

---

## Comandos de controle e repetição



- **break**

- Permite interromper suspender o comando de repetição e ir para o próximo comando

```
var i : Int
for i = 0; i <= 10; ++i
{
    if(i == 4)
    {
        break
    }
    // Exibir i
}
// Resultado: 0 1 2 3
```

---

---


---

---

---

---

---

Comandos de controle e repetição


- continue**
  - Permite instruir ao comando de repetição que interrompa esta iteração e avance para a próxima iteração da repetição

```

var i : Int
for i = 0; i <= 10; ++i
{
  if(i == 4)
  {
    continue
  }
  // Exibir i
}
// Resultado: 0 1 2 3 5 6 7 8 9

```

---

---

---


---

---

---

---

---

Criando funções


- Permite centralizar blocos de código

Sintaxe

```
func nomeDaFunção([parâmetros]) -> tipoDeRetorno { }
```

Exemplo

```

func retornaDobro(valor: Int) -> Int
{
  return valor * 2
}

var i : Int = 6
var z : Int = retornaDobro(i)

// Exibir z
// Resultado: 12

```

---

---

---


---


---

---

---

---

Aulas práticas e manuais on-line




Assista agora às aulas práticas, que abordam os temas tratados nesta aula teórica, para consolidar seus conhecimentos.

[Clique aqui](#) para visualizar as aulas práticas disponíveis

---

---

---

---

---

---

---

---