UNIFASB
CENTRO UNIVERSITÁRIO
SÃO FRANCISCO DE BARREIRAS

UNINASSAU

Grupo Ser Educacional

Gente criando o futuro

Desenvolvimento para Dispositivos Móveis

Aula 3 - Funções e Listas (Mutáveis e Imutáveis)

Estruturas de Controle

1) Seleção: if, if-else, if-else if-else

2) Repetição: for, while, do-while

Estruturas de Controle - Seleção

```
i = 47
if (i == 42) {
        println(“olá”)
} else {
        println(“oi”)
}
```

Estruturas de Controle - Seleção

```
idade = 17
if (idade >= 16 && idade <=17)
        println(“voto opcional”)
} else if (idade >= 18 && idade < 65)
        println(“voto obrigatório”)
} else {
        println(“não precisa votar”)
}
```

```
    val nota: Double = 9.3

    // Usando operador range
    if (nota in 9..10) {
        println("Fantástico")
    } else if (nota in 7..8) {
        println("Parabéns")
    } else if (nota in 4..6) {
        println("Tem como recuperar")
    } else if (nota in 0..3) {
        println("Te vejo no próximo semestre")
    } else {
        println("Nota inválida")
    }
}
```

Estruturas de Controle - Seleção

```
val ling = mutableListOf("Kotlin", "Java", "Python")
    if("Kotlin" in ling){
        print("ok")
    }
}
```

Estruturas de Controle - Repetição

```
for (i in 1..3){
        println(i)
}

var t:Int = 50
for (i in 1..t){
        println(i)
}
```

Estruturas de Controle - Repetição

```
for (i in 6 downTo 0){
        println(i)
}

for (i in 6 downTo 0 step 2){
        println(i)
}
```

Estruturas de Controle - Repetição

```
for (i in 1..10){
        if(i == 5){
                break
        }
        println(“Atual: $i”)
}
```

Estruturas de Controle - Repetição

```
var x = 0
while(x > 5){
        print(x)
        x++
}
```

```
var x = 5
while(x > 0){
        print(x)
        x--
}
```

Estruturas de Controle - Repetição

```
var x = 0
do{
        print(x)
        x++
} while(x > 5)
```

Funções

Trecho de Código que pode ser chamado em qualquer ponto de um programa.

Todo programa Kotlin é uma função, a função principal.

Temos dois tipos de função: função com retorno (retorna um resultado) e função sem retorno (somente executa alguma operação).

Funções com Retorno

```
fun main() {
    val r = sum(40, 50)
    println("Resultado: $r")
}

fun sum(a:Int, b:Int): Int{
    return a + b
}
```

Funções sem Retorno

```
fun printSum(a:Int, b:Int){
        var s = a + b
        print(“$a + $b = $s”)
}

printSum(20, 30)                    // Chamada da função
```

Biblioteca Math

```
import kotlin.math.*
```

a) Funções Trigonométricas

sin(x): Seno de x em radianos.
cos(x): Cosseno de x em radianos.
tan(x): Tangente de x em radianos.

Biblioteca Math

b) Funções Exponenciais e Logarítmicas

pow(base, exponente): Potência de base elevada ao expoente.

```
fun main() {
    print(Math.pow(2.0,2.0))
}
```

Biblioteca Math

c) Funções de Arredondamento e Valor Absoluto

abs(x): Retorna o valor absoluto de x.

round(x): Valor arredondado de x.

Biblioteca Math

d) Funções de Comparação:

max(x, y): Retorna o maior valor entre x e y.

min(x, y): Retorna o menor valor entre x e y.

Biblioteca Math

e) Constantes: PI (Valor de π)

Para calcular a área de um círculo: área = π * raio^2

```
val raio = 3.0
val area = Math.PI * raio * raio
println("A área do círculo é: $area")
```

Listas

Uma lista é uma estrutura de dados que representa listas ordenadas de elementos, acessíveis por meio de um índice numérico (o primeiro item da lista ocupa o índice zero).

Também são chamadas de vetores ou arrays e de maneira simples, permitem armazenar diversos dados em uma variável.

Podem ser de qualquer tipo e com vários tipos.

Por que é útil?

Listas são utilizadas na maioria das aplicações profissionais, sendo o conhecimento dessa estrutura de dados fundamental para o programador que deseja atuar profissionalmente.

Listas

Em Kotlin temos dois tipos de listas:

- List
- MutableList

List

- List é imutável, ou seja, uma vez criada a lista nenhum item pode ser adicionado ou removido dela;

- Por ser imutável, List não possui métodos para alteração da lista, como add ou remove, por exemplo.

List – Como Utilizar

Criação:

```
val linguagens = listOf(“Kotlin”, “Java”, “Python”)
```

List – Como Utilizar

Acesso:

```
val linguagens = listOf(“Kotlin”, “Java”, “Python”)

print(linguagens[0])
```

List – Como Utilizar

Acesso (get):

```
val linguagens = listOf(“Kotlin”, “Java”, “Python”)

print(linguagens.get(2))
```

List – Como Utilizar

Percorrendo:

```
val linguagens = listOf(“Kotlin”, “Java”, “Python”)

for (i in linguagens){
        println(i)
}
```

MutableList

- MutableList é a interface para listas mutáveis, nas quais podemos adicionar ou remover itens;

- Por ser mutável, MutableList possui métodos que modificam a lista, como add e remove, por exemplo.

MutableList – Como Utilizar

Criação:

```
val linguagens = mutableListOf(“Kotlin”, “Java”, “Python”)
```

MutableList – Como Utilizar

Acesso:

```
val linguagens = mutableListOf(“Kotlin”, “Java”, “Python”)

print(linguagens[0]) ou

print(linguagens.get(0))
```

List – Como Utilizar

Percorrendo:

```
val linguagens = mutableListOf(“Kotlin”, “Java”, “Python”)

for (i in linguagens){
        println(i)
}
```

Adicionando Elementos no Final (add)

```
val linguagens = mutableListOf(“Kotlin”, “Java”, “Python”)

linguagens.add(“PHP”)

print(linguagens)
```

Adicionando Elementos pela Posição (add)

```
val linguagens = mutableListOf(“Kotlin”, “Java”, “Python”)

linguagens.add(1, “PHP”)

print(linguagens)
```

Removendo Elementos pela Posição (removeAt)

```
val linguagens = mutableListOf(“Kotlin”, “Java”, “Python”)

linguagens.removeAt(1)

print(linguagens)
```

Removendo Elementos (remove)

```
val linguagens = mutableListOf(“Kotlin”, “Java”, “Python”)

linguagens.remove(“Java”)

print(linguagens)
```

Modificando Elementos (set)

```
val linguagens = mutableListOf(“Kotlin”, “Java”, “Python”)

linguagens.set(2, “C++”)

print(linguagens)
```

Removendo Todos Elementos (clear)

```
val linguagens = mutableListOf(“Kotlin”, “Java”, “Python”)

linguagens.clear()

print(linguagens)
```