Desenvolvimento para Dispositivos Móveis

Programação Estruturada, Sequência, Seleção e Repetição

Prof. Bruno Azevedo

UNIP – Universidade Paulista

Avaliando Condições

- Em um algoritmo podemos querer avaliar condições, assim como fazemos no dia-a-dia.
- Por exemplo, se vamos sair de casa, avaliamos se está chovendo ou não, para levar um guarda-chuva.
- Avaliamos se está frio para escolhermos qual roupa iremos vestir.
- Para isso temos **Estruturas Condicionais de Decisão**.

Estruturas Condicionais

- Por exemplo, considere o seguinte **algoritmo** para avaliar se devemos sair de casa com guarda-chuva ou não.

```
Se está chovendo
   Leve o guarda chuva.
```

- Ou para decidir se levaremos nosso casaco.

```
Se está frio
   Leve o casaco.
```

- Podemos também considerar alternativas, caso não esteja chovendo ou não esteja frio.

Estruturas Condicionais

- O **Senão** é utilizado em conjunto o **Se** para fornecer uma alternativa, caso a condição não seja satisfeita.
- Se a condição do **Se** for falsa, o **Senão** é executado.

```
Se condição
    sequência de instruções 1
Senão
    sequência de instruções 2
```

- Exemplo:

```
Se está chovendo
   Leve o guarda-chuva.
Senão
   Deixe o guarda-chuva em casa.
```

Condições

- Estamos avaliando **condições**.
- Se a condição é verdadeira, faça algo. Se é falsa, faça outra coisa.
- Essa é a ideia de estruturas condicionais. Elas avaliam **condições**.
- Mas o que são condições?

Condições

- Uma condição é uma **expressão lógica ou relacional** que quando avaliada gera um resultado de **verdadeiro** ou **falso**.
- Podemos usar **operadores** em nossa condição à ser avaliada.
- Podemos usar variáveis e valores em nossas condições.

Condições

- Vamos ver um exemplo.

```
Se (hora igual a 19:10)
    Vá para a sala de aula.
```

- Mas devemos ir para a sala de aula mesmo no domingo?
- Precisamos portanto **conectar** condições para chegarmos a condição final que desejamos.

Condições

- Podemos utilizar **operadores lógicos** para conectar múltiplas condições e construir uma única condição a ser testada.

```
Se (hora igual a 19:10) E (é dia de semana)
    Vá para a sala de aula.
```

- Conectamos as duas condições (horário == 19:10 e dia == dia de semana) com o **operador lógico** E.

Estrutura Condicional: `if`

- Aprendemos a estrutura condicional **Se** que usamos para exemplos do dia-a-dia.

> Se condição
> sequência de instruções

- Em Kotlin, a estrutura equivalente possui a seguinte sintaxe:

```
if (condição) {
    // sequência de instruções
}
```

if

- Exemplo:

```
val a = 20
val b = 10
if (a > b) {
    println("a é maior que b")
}
```

if

- Outro exemplo:

```
val a = 20
val b = 10
if (a > b) {
    println("a é maior que b")
} else if (a == b) {
    println("a e b são iguais")
} else {
    println("a é menor que b")
}
val c = a + b
if (c < 50) {
    println("c é menor que cinquenta")
}
```

else

- Vimos o uso de **Senão**, alternativa ao **Se**.
- Caso a avaliação da condição do **Se** for falsa, temos um conjunto de instruções alternativas.

```
Se estáChovendo == Verdadeiro
    GuardaChuva = Verdadeiro
Senão
    GuardaChuva = Falso
```

- Em Kotlin temos a mesma construção, utilizando a palavra-chave **else**:

```
val a = 10
val b = 20
if (a > b) {
    println("a é maior que b")
} else {
    println("a não é maior que b")
}
```

else if

- Em Kotlin, podemos também combinar o if com o else.
- A estrutura **else if** é usada quando precisamos testar outra condição após a primeira ser verificada como falsa.
- Sintaxe:

```
if (condição1) {
    // instruções a serem executadas se a condição1
    // for verdadeira
} else if (condição2) {
    // instruções a serem executadas se a condição2
    // for verdadeira
} else {
    // instruções a serem executadas se nenhuma
    // das condições anteriores for verdadeira
}
```

else if

- Exemplo em Kotlin:

```
if (a > b) {
    println("a é maior que b")
} else if (a == b) {
    println("a e b são iguais")
} else {
    println("a é menor que b")
}
```

Estrutura de Seleção: `when`

- A estrutura `when` é usada para seleção múltipla.
- Substitui o uso de vários `if-else if`.

```
val dia = 3
when (dia) {
    1 -> println("Segunda")
    2 -> println("Terça")
    3 -> println("Quarta")
    else -> println("Outro dia")
}
```

- Saída:

```
Quarta
```

Estrutura Condicional Repetitiva: `while`

- A estrutura `if` testa **uma** vez a condição e executa, ou não, um bloco de código.
- A estrutura `while` funciona similar a estrutura `if`, mas permite a repetição do conjunto de instruções enquanto as condições se mantiverem satisfeitas.
- Ou seja, o `while` manterá o fluxo de nosso programa dentro de um **laço** até que as condições não sejam mais verdadeiras.
- A sintaxe é a seguinte:

```
while (condição) {
  // Bloco de código a ser executado enquanto a condição se mantiver verda
}
```

- Após a execução do bloco, a condição do while é reavaliada.
- Se continuar verdadeira, o bloco se repete; caso contrário, a execução segue adiante.
- Assim como no `if`, é possível combinar condições com operadores lógicos.

Estrutura Condicional Repetitiva: `while`

- Vamos conhecer alguns exemplos práticos.
- Contador de 1 até 5.

```
var contador = 1
while (contador <= 5) {
    println(contador)
    contador += 1
}
```

- Saída:

```
1
2
3
4
5
```

Estrutura Condicional Repetitiva: do-while

- A estrutura do-while garante que o bloco seja executado pelo menos uma vez.
- A condição é verificada após a execução.

```
var i = 1
do {
    println("Número: $i")
    i++
} while (i <= 5)
```

Estrutura Condicional Repetitiva: `for`

- A estrutura `for` é usada quando o número de repetições é conhecido.

```
for (i in 1..5) {
    println("Valor: $i")
}
```

Inicialização, Condição e Incremento

- Todo laço de repetição envolve três partes essenciais:
  - Inicialização: variável de controle começa com um valor.
  - Condição de parada: define até quando o laço deve executar.
  - Incremento ou Decremento: atualiza o valor da variável.

```
var i = 0           // inicialização
while (i < 10) {    // condição
    println(i)
    i++             // incremento
}
```

- A estrutura `for` em Kotlin permite essas três partes de forma integrada:

```
for (i in 0..10 step 2) {
    println(i)
}
```

- A palavra-chave `step` define o valor do incremento entre as iterações.

Atividade Prática 4

1. Criar um aplicativo que permita ao usuário inserir um número e, ao pressionar um botão, exiba a tabuada desse número (de 1 a 10) usando um laço for.
2. Ampliação: adicionar as seguintes funcionalidades:
   - Usar um laço while para exibir apenas os números pares da tabuada.
   - Criar um botão que, ao ser pressionado, inicie um contador usando um laço do-while, que será interrompido por outro botão de parada.
   - Personalizar o layout do App para exibir as informações de maneira clara e organizada.
3. Cada aluno deve produzir um relatório de 1 a 3 páginas contendo:
   - Resumo teórico: Explicação conceitual dos laços de repetição (for, while, do-while).
   - Código-fonte comentado: Explicação detalhada do código, destacando o funcionamento de cada estrutura de repetição utilizada.