Introdução ao Go

Aula 32: Nested "if" Statements & Using "return" To Stop Function Execution

Nesta aula, o professor introduziu o conceito de **"if" aninhado (nested if)** e a utilização do **"return"** para interromper a execução de uma função. O objetivo foi adicionar validações ao sistema de depósito e saque, garantindo que o usuário não insira valores inválidos, como números negativos ou saques superiores ao saldo disponível.

Principais Conceitos

1. If Aninhado (Nested If)

- Permite adicionar verificações dentro de outras verificações existentes.
- Usado para validar o valor antes de modificar o saldo da conta.

Exemplo de Verificação no Depósito:

```go
if depositAmount <= 0 {
    fmt.Println("Valor inválido! O depósito deve ser maior que zero.")
    return
}
```

✔ **Se o valor for menor ou igual a zero**, a mensagem de erro é exibida e a função é interrompida com `return`, impedindo a execução do restante do código.

---

2. Uso do "return" para Interromper a Função

- `return` sem valor serve para interromper a função e impedir que o código continue.
- Evita que um depósito ou saque inválido seja processado.

Exemplo de Verificação no Saque:

```go
if withdrawalAmount <= 0 {
    fmt.Println("Valor inválido! O saque deve ser maior que zero.")
    return
}
if withdrawalAmount > accountBalance {
    fmt.Println("Valor inválido! Você não pode sacar mais do que tem na conta.")
    return
}
```

✔ **Se o valor for inválido (negativo ou maior que o saldo), a função para imediatamente.**

Código Atualizado com Validações

```go
package main

import (
    "fmt"
)

func main() {
    var accountBalance float64 = 1000
    var choice int

    fmt.Println("Escolha uma opção:")
    fmt.Println("1 - Ver saldo")
    fmt.Println("2 - Depositar dinheiro")
    fmt.Println("3 - Sacar dinheiro")
    fmt.Println("4 - Sair")

    fmt.Print("Sua escolha: ")
    fmt.Scan(&choice)

    if choice == 1 {
        fmt.Println("Seu saldo atual é:", accountBalance)
    } else if choice == 2 {
```

```
        var depositAmount float64
        fmt.Print("Valor do depósito: ")
        fmt.Scan(&depositAmount)

        // Validação do depósito
        if depositAmount <= 0 {
            fmt.Println("Valor inválido! O depósito deve ser maior que zero.")
            return
        }

        accountBalance += depositAmount
        fmt.Println("Saldo atualizado! Novo saldo:", accountBalance)
    } else if choice == 3 {
        var withdrawalAmount float64
        fmt.Print("Valor do saque: ")
        fmt.Scan(&withdrawalAmount)

        // Validação do saque
        if withdrawalAmount <= 0 {
            fmt.Println("Valor inválido! O saque deve ser maior que zero.")
            return
        }
        if withdrawalAmount > accountBalance {
            fmt.Println("Valor inválido! Você não pode sacar mais do que tem na
conta.")
            return
        }

        accountBalance -= withdrawalAmount
        fmt.Println("Saldo atualizado! Novo saldo:", accountBalance)
    } else {
        fmt.Println("Goodbye!")
    }
}
```

Explicação do Código

✅ **Validações foram adicionadas** para evitar valores negativos e saques acima do saldo.
✅ **O uso de `return` impede que operações inválidas sejam realizadas**, encerrando a

execução antes de modificar o saldo.
✅ **Os ifs aninhados permitem verificações detalhadas** dentro de cada operação.

Pontos Importantes

✔ **If aninhado** ajuda a organizar verificações dentro de blocos de código específicos.
✔ **"return" sem valor** encerra a execução da função atual.
✔ **Melhoria na experiência do usuário**, impedindo operações incorretas e exibindo mensagens explicativas.

Próximos Passos

🔹 Implementar um **loop** para manter o programa rodando até que o usuário escolha sair.
🔹 Melhorar a **interface de interação**, permitindo que o usuário veja o saldo atualizado sem reiniciar o programa.
🔹 **Salvar o saldo em um arquivo ou banco de dados** para que ele não seja perdido após o fechamento do programa.

Se precisar de mais detalhes, é só perguntar! 🚀

Aula 32: Usando "if" Aninhados e Usando "return" para Parar a Execução da Função