Introdução ao Go

Aula 27: Estruturas de Controle em Go

Nesta aula, aprendemos sobre **estruturas de controle** em Go. Para isso, começamos a criar um **banco fictício**, onde o usuário pode verificar o saldo, depositar, sacar dinheiro e sair do programa.

1️⃣ Criando o Projeto

Primeiramente, inicializamos um novo módulo Go:

```sh
go mod init example.com/bank
```

Em seguida, criamos o arquivo `bank.go` e adicionamos o código inicial.

2️⃣ Estrutura do Programa

O banco terá um menu onde o usuário pode escolher entre:

1. **Verificar saldo**
2. **Depositar dinheiro**
3. **Sacar dinheiro**
4. **Sair**

Além disso, o programa deve:

- Rodar em **loop infinito** até que o usuário escolha sair.
- **Validar entradas**, garantindo que apenas opções válidas sejam aceitas.
- **Evitar saques inválidos**, como valores negativos ou superiores ao saldo disponível.

3️⃣ Código Inicial

O código começa imprimindo um menu e capturando a escolha do usuário.

```go
package main

import (
    "fmt"
)

func main() {
    fmt.Println("Bem-vindo ao Go Bank!")
    fmt.Println("O que deseja fazer?")
    fmt.Println("1 - Verificar saldo")
    fmt.Println("2 - Depositar dinheiro")
    fmt.Println("3 - Sacar dinheiro")
    fmt.Println("4 - Sair")

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

    fmt.Println("Você escolheu:", choice)
}
```

Agora, ao executar, o programa exibe o menu e captura a opção do usuário.

4️⃣ Adicionando Estruturas de Controle

Agora vamos: ✅ Criar um loop **while (simulado com `for`)** para manter o programa rodando.
✅ Usar um **switch case** para tratar as opções do menu.
✅ Armazenar o **saldo da conta** em uma variável.

```go
package main
```

```go
import (
    "fmt"
)

func main() {
    var balance float64 = 1000.0 // Saldo inicial
    var choice int

    for {
        // Exibir o menu
        fmt.Println("\n=== Go Bank ===")
        fmt.Println("1 - Verificar saldo")
        fmt.Println("2 - Depositar dinheiro")
        fmt.Println("3 - Sacar dinheiro")
        fmt.Println("4 - Sair")

        // Capturar a escolha do usuário
        fmt.Print("Sua escolha: ")
        fmt.Scan(&choice)

        // Estrutura de controle para processar a escolha
        switch choice {
        case 1:
            fmt.Printf("\nSeu saldo atual é: R$ %.2f\n", balance)
        case 2:
            var deposit float64
            fmt.Print("Digite o valor para depósito: ")
            fmt.Scan(&deposit)
            if deposit > 0 {
                balance += deposit
                fmt.Printf("Depósito realizado! Novo saldo: R$ %.2f\n", balance)
            } else {
                fmt.Println("Valor inválido. O depósito deve ser maior que zero.")
            }
        case 3:
            var withdraw float64
            fmt.Print("Digite o valor para saque: ")
            fmt.Scan(&withdraw)
            if withdraw > 0 && withdraw <= balance {
                balance -= withdraw
                fmt.Printf("Saque realizado! Novo saldo: R$ %.2f\n", balance)
            } else {
```

```
                fmt.Println("Saque inválido! Verifique o valor inserido.")
            }
        case 4:
            fmt.Println("Saindo... Obrigado por usar o Go Bank!")
            return
        default:
            fmt.Println("Opção inválida! Escolha entre 1 e 4.")
        }
    }
}
```

5️⃣ Explicação do Código

✅ O **saldo inicial** é **R$ 1000.00**.
✅ O programa roda em um **loop infinito** (`for {}`), só saindo quando o usuário escolhe a opção 4.
✅ `switch case` é usado para tratar cada opção.
✅ O programa verifica:

- Se o **depósito** é um número válido.
- Se o **saque** é possível (não pode ser negativo ou maior que o saldo).
  ✅ A opção 4 **finaliza o programa** com `return`.

6️⃣ Testando o Programa

Exemplo de execução:

yaml

```
=== Go Bank ===
1 - Verificar saldo
2 - Depositar dinheiro
3 - Sacar dinheiro
4 - Sair
Sua escolha: 1
Seu saldo atual é: R$ 1000.00
```

```
Sua escolha: 2
Digite o valor para depósito: 500
Depósito realizado! Novo saldo: R$ 1500.00

Sua escolha: 3
Digite o valor para saque: 2000
Saque inválido! Verifique o valor inserido.

Sua escolha: 4
Saindo... Obrigado por usar o Go Bank!
```

7️⃣ Resumo

✔ **Aprendemos sobre estruturas de controle (`for`, `switch case`).**
✔ **Criamos um sistema bancário simples com validação de entrada.**
✔ **Praticamos loops e condições para um programa interativo.**

Agora, o programa está funcional e bem estruturado! 🚀

Resumo da Aula 27 - Estruturas de Controle em Go

Nesta aula, foi introduzido um conceito fundamental da linguagem Go: **estruturas de controle**. Elas permitem definir quais trechos de código serão executados dependendo de determinadas condições.

Pontos principais da aula:

1. **Introdução às Estruturas de Controle**
   - Permitem controlar o fluxo do programa.
   - São essenciais para criar interações dinâmicas no código.
2. **Projeto Prático: Simulação de um Banco**
   - Criado um novo projeto Go (`go mod init example.com/bank`).
   - Arquivo principal: `bank.go`, dentro do pacote `main`.

- O programa simulará operações bancárias simples.

3. **Fluxo do Programa**
    - O usuário verá um menu com opções:
        1. Ver saldo
        2. Depositar dinheiro
        3. Sacar dinheiro
        4. Sair
    - O programa rodará em um **loop infinito** até que o usuário escolha sair.
    - Validação de entrada será implementada (ex.: impedir saques inválidos).
4. **Interação com o Usuário**
    - Exibição de mensagens no terminal (`fmt.Println`).
    - Leitura de entrada do usuário (`fmt.Scan`).
    - Armazenamento da escolha em uma variável (`choice int`).
5. **Próximos Passos**
    - Implementação das **estruturas de controle** (`if` , `switch` , `for`).
    - Tratamento adequado das opções escolhidas pelo usuário.

Essa aula serviu como introdução ao projeto e ao conceito de controle de fluxo. Nos próximos passos, serão adicionadas funcionalidades como **condições e loops** para tornar o programa mais interativo.