Introdução ao Go

Aula 26: Exercício - Trabalhando com Funções em Go

O objetivo deste exercício foi refatorar o código de um **calculador de lucro** para usar funções personalizadas. Isso melhora a organização, modularidade e clareza do programa.

Tarefa 1: Criar a Função `getUserInput`

Essa função:

- Exibe um texto ao usuário para solicitar uma entrada.
- Lê e retorna o valor inserido como `float64`.

Código da Função `getUserInput`:

```go
package main

import (
    "fmt"
)

func getUserInput(infoText string) float64 {
    var userInput float64
    fmt.Println(infoText)
    fmt.Scan(&userInput)
    return userInput
}
```

Tarefa 2: Refatorar o Código da Função `main`

1. Substituir o código para capturar `revenue`, `expenses` e `taxRate` por chamadas à função `getUserInput`.

2. Melhorar a legibilidade removendo repetições.

Código Refatorado da `main`:

```go
func main() {
    revenue := getUserInput("Digite a receita (revenue):")
    expenses := getUserInput("Digite as despesas (expenses):")
    taxRate := getUserInput("Digite a taxa de imposto (taxRate):")

    // Chamando a função para calcular os resultados
    EBT, profit, ratio := calculateFinancials(revenue, expenses, taxRate)

    // Exibindo os resultados formatados
    fmt.Printf("EBT: %.1f\n", EBT)
    fmt.Printf("Lucro: %.1f\n", profit)
    fmt.Printf("Margem de lucro: %.3f\n", ratio)
}
```

Tarefa 3: Criar a Função `calculateFinancials`

Essa função realiza:

- O cálculo do **EBT (Earnings Before Taxes)**: `revenue - expenses`.
- O cálculo do **lucro líquido**: `EBT - (EBT * taxRate / 100)`.
- O cálculo da **margem de lucro**: `profit / revenue`.

Código da Função `calculateFinancials`:

```go
func calculateFinancials(revenue, expenses, taxRate float64) (float64, float64, float64) {
    EBT := revenue - expenses
    profit := EBT - (EBT * taxRate / 100)
    ratio := profit / revenue
```

```
    return EBT, profit, ratio
}
```

Código Completo

go

```go
package main

import (
    "fmt"
)

func getUserInput(infoText string) float64 {
    var userInput float64
    fmt.Println(infoText)
    fmt.Scan(&userInput)
    return userInput
}

func calculateFinancials(revenue, expenses, taxRate float64) (float64, float64,
float64) {
    EBT := revenue - expenses
    profit := EBT - (EBT * taxRate / 100)
    ratio := profit / revenue
    return EBT, profit, ratio
}

func main() {
    revenue := getUserInput("Digite a receita (revenue):")
    expenses := getUserInput("Digite as despesas (expenses):")
    taxRate := getUserInput("Digite a taxa de imposto (taxRate):")

    EBT, profit, ratio := calculateFinancials(revenue, expenses, taxRate)

    fmt.Printf("EBT: %.1f\n", EBT)
    fmt.Printf("Lucro: %.1f\n", profit)
```

```
    fmt.Printf("Margem de lucro: %.3f\n", ratio)
}
```

Dicas de Bônus

1. **Formatar números:** Use `fmt.Printf` com especificadores de formato como `%0.1f` para exibir floats com casas decimais precisas.
2. **Adicionar espaçamento:** Use `\n` para criar quebras de linha.
3. **Validação de entradas:** Para melhorar a robustez, você pode verificar se o valor inserido pelo usuário é válido (ex.: maior que zero).

Resumo

Este exercício mostrou como:

- Criar funções reutilizáveis em Go.
- Passar e retornar valores com parâmetros e tipos de retorno.
- Organizar melhor o código, reduzindo repetições.

Você agora tem um programa mais limpo e modular que segue boas práticas de desenvolvimento!