Introdução ao Go

Aula 24: Functions - Return Values & Variable Scope

Nesta aula, exploramos como as funções em Go podem retornar valores e como o escopo das variáveis afeta o código. Vamos entender como criar funções que produzem resultados e como gerenciar o acesso a variáveis dentro do programa.

Funções Podem Retornar Valores

As funções não apenas executam operações, mas também podem retornar resultados que podem ser usados em outras partes do código.

Exemplo: Função que Retorna um Valor

go

```go
package main

import (
    "fmt"
    "math"
)

// Função que calcula o valor futuro de um investimento
func calcularValorFuturo(investimento float64, taxa float64, anos float64) float64 {
    return investimento * math.Pow(1+taxa/100, anos)
}

func main() {
    resultado := calcularValorFuturo(1000, 5, 10)
    fmt.Println("Valor futuro do investimento:", resultado)
}
```

Saída:

plaintext

```
Valor futuro do investimento: 1628.89
```

Retornando Múltiplos Valores

Go permite retornar mais de um valor separando-os por vírgula. Isso é útil para cálculos que exigem múltiplos resultados.

Exemplo: Retornando Dois Valores

go

```go
package main

import (
    "fmt"
    "math"
)

// Função que calcula o valor futuro e ajusta pela inflação
func calcularValoresFuturos(investimento float64, taxa float64, anos float64)
(float64, float64) {
    fv := investimento * math.Pow(1+taxa/100, anos)
    rfv := fv / 1.05 // Ajuste considerando uma inflação de 5%
    return fv, rfv
}

func main() {
    futuro, futuroReal := calcularValoresFuturos(1000, 5, 10)
    fmt.Println("Valor futuro:", futuro)
    fmt.Println("Valor futuro ajustado pela inflação:", futuroReal)
}
```

Saída:

plaintext

```
Valor futuro: 1628.89
Valor futuro ajustado pela inflação: 1552.27
```

Escopo de Variáveis

O escopo determina onde uma variável pode ser acessada no código.

- **Escopo Local:** Variáveis declaradas dentro de uma função só podem ser usadas ali.
- **Escopo Global:** Variáveis declaradas fora de qualquer função podem ser acessadas em todo o código.

Exemplo: Escopo Local e Global

go

```go
package main

import "fmt"

// Variável global (acessível em todo o código)
const taxaInflacao = 5.0

func calcularAjuste(valor float64) float64 {
    // Variável local (acessível apenas dentro desta função)
    ajuste := valor / (1 + taxaInflacao/100)
    return ajuste
}

func main() {
    valor := 2000.0
    ajustado := calcularAjuste(valor)
    fmt.Println("Valor ajustado:", ajustado)
}
```

Saída:

plaintext

```
Valor ajustado: 1904.76
```

> **Observação:** A variável `ajuste` só existe dentro de `calcularAjuste()`, enquanto `taxaInflacao` é global e pode ser usada em qualquer função.

Especificando Tipos de Retorno

Como Go é uma linguagem fortemente tipada, é obrigatório especificar os tipos de valores que uma função retornará.

Sintaxe:

```go
func nomeDaFuncao(param1 tipo, param2 tipo) (tipoRetorno1, tipoRetorno2) {
    // Código da função
    return valor1, valor2
}
```

Exemplo: Retorno com Tipos Definidos

```go
func somar(a int, b int) int {
    return a + b
}
```

Se a função não retorna nada, não é necessário definir um tipo de retorno.

Chamando uma Função com Retorno

Ao chamar uma função que retorna valores, podemos armazená-los em variáveis ou usá-los diretamente.

Exemplo: Armazenando o Retorno

```go
resultado := somar(3, 7)
fmt.Println(resultado) // Saída: 10
```

Exemplo: Usando Diretamente

```go

```

```
fmt.Println(somar(3, 7)) // Saída: 10
```

Pontos-Chave Sobre Retorno de Funções e Escopo

✅ **Funções podem retornar valores**, que podem ser armazenados ou usados diretamente.
✅ **Go permite retornar múltiplos valores**, facilitando operações mais complexas.
✅ **O escopo das variáveis define onde elas podem ser acessadas**, garantindo segurança e organização no código.
✅ **É obrigatório definir o tipo de retorno de uma função**, garantindo coerência nos valores processados.

Resumo

- Funções podem **retornar valores**, permitindo maior reutilização do código.
- É possível **retornar múltiplos valores** em Go.
- **Variáveis locais** só existem dentro da função onde foram declaradas, enquanto **variáveis globais** podem ser acessadas de qualquer função.
- Sempre **especifique os tipos de retorno** ao definir funções em Go.

Com essas técnicas, seu código fica mais modular, organizado e eficiente!