Introdução ao Go

Aula 16: Melhorando a Captura de Entrada do Usuário

Nesta aula, o professor mostrou como tornar o programa mais **amigável ao usuário**, exibindo mensagens informativas antes de capturar os valores de entrada. A abordagem torna claro o que é esperado do usuário e melhora significativamente a experiência de uso.

Principais Melhorias Introduzidas

1. **Mensagens informativas antes da entrada do usuário**:
   - Utilizando `fmt.Print` (ao invés de `fmt.Println`) para exibir uma mensagem na **mesma linha** da entrada do usuário.
2. **Declaração explícita de variáveis sem valor inicial**:
   - Quando não há um valor inicial, é necessário usar `var` e declarar o **tipo** explicitamente.
3. **Ordem das capturas de entrada**:
   - A entrada de valores pode seguir uma ordem lógica, como "Valor do Investimento", "Taxa de Retorno" e "Anos".

Código Melhorado: Captura de Entradas

Aqui está o código ajustado com as melhorias explicadas:

go

```go
package main

import (
    "fmt"
    "math"
)
```

```go
func main() {
    // Declaração das variáveis
    var investmentAmount float64
    var years int
    expectedReturnRate := 5.0 // Valor padrão para a taxa de retorno

    // Captura de entrada do usuário
    fmt.Print("Enter investment amount: ")
    fmt.Scan(&investmentAmount)

    fmt.Print("Enter expected return rate (%): ")
    fmt.Scan(&expectedReturnRate)

    fmt.Print("Enter number of years: ")
    fmt.Scan(&years)

    // Cálculo do valor futuro
    futureValue := investmentAmount * math.Pow(1+(expectedReturnRate/100),
float64(years))

    // Exibição dos resultados
    fmt.Printf("\nInvestment Details:\n")
    fmt.Printf("Investment Amount: %.2f\n", investmentAmount)
    fmt.Printf("Expected Return Rate: %.2f%%\n", expectedReturnRate)
    fmt.Printf("Number of Years: %d\n", years)
    fmt.Printf("Future Value: %.2f\n", futureValue)
}
```

Detalhes Implementados

1. Melhor mensagem de entrada com `fmt.Print`

Utilizamos `fmt.Print` para que o texto e a entrada fiquem na **mesma linha**:

```go
fmt.Print("Enter investment amount: ")
fmt.Scan(&investmentAmount)
```

Isso torna o programa mais intuitivo. Por exemplo:

```yaml
Enter investment amount: 1000
```

2. Declarando variáveis sem valor inicial

Quando não há um valor inicial, como no caso de `investmentAmount` e `years`, usamos a palavra-chave `var` e declaramos o **tipo** explicitamente:

```go
var investmentAmount float64
var years int
```

O Go é uma linguagem **estaticamente tipada**, ou seja:

- O **tipo** de cada variável deve ser conhecido em tempo de compilação.
- Se não há um valor inicial, o Go **não consegue inferir o tipo automaticamente**.

3. Variáveis com valores padrão

Para a variável `expectedReturnRate`, foi utilizado um valor padrão (`5.0`), demonstrando o uso da notação curta `:=`:

```go
expectedReturnRate := 5.0
```

Isso permite que o usuário substitua o valor padrão ao digitar uma nova taxa.

4. Resultados mais organizados

Ao exibir os resultados, utilizamos o formato `%.2f` para limitar os números decimais a duas casas:

```go
fmt.Printf("Investment Amount: %.2f\n", investmentAmount)
fmt.Printf("Future Value: %.2f\n", futureValue)
```

Exemplo de Execução

Quando o programa é executado, o terminal apresenta o seguinte fluxo de interação:

1. **Entrada do Usuário**:

```mathematica
Enter investment amount: 1000
Enter expected return rate (%): 7
Enter number of years: 10
```

2. **Saída do Programa**:

```yaml
Investment Details:
Investment Amount: 1000.00
Expected Return Rate: 7.00%
Number of Years: 10
Future Value: 1967.15
```

Resumo

1. **Melhor experiência do usuário**:
   - Mensagens claras para orientar a entrada de dados.
   - Entradas organizadas e exibidas no formato esperado.
2. **Tipos de declaração de variáveis**:
   - **Com valor inicial**: Inferência automática do tipo (`:=`).
   - **Sem valor inicial**: Declaração explícita do tipo com `var` .
3. **Formatação de saída**:
   - Uso de `fmt.Printf` para personalizar o formato dos resultados.

Se precisar de ajuda para adicionar novos cálculos ou outras melhorias ao programa, é só pedir!