Introdução ao Go

Aula 19: Formatação de Strings (Texto) - Básico

Nesta aula, aprendemos como melhorar a saída de nossos programas usando ferramentas de formatação de texto no pacote `fmt` do Go. Vamos revisar os conceitos apresentados para criar saídas mais informativas e organizadas.

Principais Pontos

1. **Melhorando a Saída de Dados**:
   - Usar texto adicional para identificar os valores exibidos.
   - Exemplo com `fmt.Println`:

     ```go
     fmt.Println("Future Value:", futureValue)
     ```

2. **Problema com Precisão de Números**:
   - Por padrão, números em Go podem ser exibidos com muitas casas decimais, o que pode ser desnecessário.
3. **Usando `fmt.Printf` para Formatar**:
   - `fmt.Printf` permite formatar texto de forma mais controlada, usando placeholders.
   - Exemplo básico:

     ```go
     fmt.Printf("Future Value: %v\n", futureValue)
     ```

     Aqui, `%v` é um **placeholder** para exibir o valor de uma variável.
4. **Adicionando Quebra de Linha**:
   - Em `Printf`, não há quebra de linha automática.

- Use `\n` para adicionar uma quebra manualmente:

```go
fmt.Printf("Future Value: %v\n", futureValue)
```

5. **Formatando Valores Numéricos**:
   - Podemos exibir números com casas decimais controladas usando `%f`:

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

   Aqui, `%.2f` limita a exibição a **duas casas decimais**.
6. **Vários Placeholders em um Só `Printf`**:
   - Exemplo com múltiplos valores:

```go
fmt.Printf("Future Value: %.2f\nAdjusted Value: %.2f\n", futureValue, adjustedValue)
```

7. **Comentando Código**:
   - Use `//` para comentários de uma linha:

```go
// Este é um comentário
```

   - Comentar partes do código ajuda a desativar temporariamente trechos sem deletá-los.

Exemplo de Código

Melhorando a Saída

```go

```

```go
package main

import (
    "fmt"
)

func main() {
    // Valores fictícios para demonstração
    futureValue := 1234.56789
    adjustedValue := 987.654321

    // Melhorando a saída com Println
    fmt.Println("Future Value:", futureValue)
    fmt.Println("Adjusted Value:", adjustedValue)

    // Usando Printf para formatação
    fmt.Printf("Future Value: %.2f\n", futureValue) // Limita a 2 casas decimais
    fmt.Printf("Adjusted Value: %.2f\n", adjustedValue)

    // Múltiplos placeholders em um só Printf
    fmt.Printf("Future Value: %.2f\nAdjusted Value: %.2f\n", futureValue,
adjustedValue)
}
```

Saída no Terminal

plaintext

```
Future Value: 1234.56789
Adjusted Value: 987.654321
Future Value: 1234.57
Adjusted Value: 987.65
Future Value: 1234.57
Adjusted Value: 987.65
```

Conclusão

O uso de `fmt.Printf` é essencial para melhorar a legibilidade e usabilidade dos nossos programas. Com ele, é possível:

- Adicionar rótulos informativos;
- Controlar a precisão de números exibidos;
- Combinar múltiplos valores formatados em uma única linha.

Você pode explorar mais sobre placeholders consultando a documentação oficial do pacote `fmt`.