Introdução ao Go

Aula 21: Criando Strings Formatadas

Nesta aula, aprendemos como usar o pacote `fmt` para criar e armazenar strings formatadas em variáveis, em vez de imprimir imediatamente no console. Essa técnica é útil quando precisamos manipular ou reutilizar o texto em outras partes do programa.

Principais Conceitos

1. **Funções de String do Pacote `fmt`**:
   - `Sprint`: Gera uma string sem formatação.
   - `Sprintf`: Gera uma string com formatação (similar ao `Printf`, mas retorna a string em vez de imprimir).
   - `Sprintln`: Gera uma string com uma quebra de linha no final, mas sem formatação adicional.
2. **Diferença entre `Print` e `Sprintf`**:
   - `Print` e suas variantes (`Println`, `Printf`) imprimem imediatamente no console.
   - `Sprint` e suas variantes (`Sprintf`, `Sprintln`) **retornam uma string** formatada, permitindo que ela seja armazenada em uma variável.
3. **Uso de `Sprintf`**:
   - Permite formatar strings com placeholders (como `%f`, `%s`, etc.).
   - A string formatada pode ser armazenada e utilizada posteriormente.

Exemplo de Código

Criando Strings Formatadas

```
go
```

```go
package main

import (
    "fmt"
)

func main() {
    // Valores fictícios
    futureValue := 12345.6789
    realFutureValue := 9876.54321

    // Criando strings formatadas
    formattedFV := fmt.Sprintf("Future Value: %.2f\n", futureValue)
    formattedRFV := fmt.Sprintf("Future Value (adjusted for inflation): %.2f\n",
realFutureValue)

    // Imprimindo as strings formatadas
    fmt.Print(formattedFV)
    fmt.Print(formattedRFV)
}
```

Saída no Terminal

plaintext

```
Future Value: 12345.68
Future Value (adjusted for inflation): 9876.54
```

Passo a Passo

1. **Formatando uma String**:
    - `fmt.Sprintf` é usado para criar uma string com formatação.
    - Armazene o valor retornado em uma variável para usá-lo posteriormente.

2. **Placeholder e Formatação**:
    - Use placeholders como `%f` para floats, `%s` para strings, `%d` para inteiros, entre outros.
    - Ajuste a precisão de números decimais com `%.2f` ou outra configuração.
3. **Imprimindo Strings Formatadas**:
    - Strings criadas com `Sprintf` podem ser passadas para `fmt.Print` ou outras funções que imprimem no console.

Vantagens do Uso de Strings Formatadas

- **Reutilização**:
    - As strings podem ser reutilizadas em diferentes partes do código, reduzindo repetição.
- **Legibilidade**:
    - O código se torna mais organizado ao separar a formatação da impressão.
- **Flexibilidade**:
    - Permite criar strings formatadas para salvar em arquivos, enviar como respostas em APIs, entre outras aplicações.

Exemplo Completo

Aqui está um exemplo mais elaborado que demonstra como reutilizar strings formatadas em diferentes partes do programa:

```go
package main

import (
    "fmt"
)
```

```
func main() {
    // Valores de entrada
    revenue := 10000.0
    expenses := 3000.0
    taxRate := 20.0 // em %

    // Cálculos
    earningsBeforeTax := revenue - expenses
    earningsAfterTax := earningsBeforeTax * (1 - taxRate/100)

    // Criando strings formatadas
    formattedBeforeTax := fmt.Sprintf("Earnings Before Tax: %.2f\n",
earningsBeforeTax)
    formattedAfterTax := fmt.Sprintf("Earnings After Tax: %.2f\n", earningsAfterTax)
    formattedTaxRate := fmt.Sprintf("Tax Rate: %.2f%%\n", taxRate)

    // Imprimindo no console
    fmt.Print(formattedBeforeTax)
    fmt.Print(formattedAfterTax)
    fmt.Print(formattedTaxRate)
}
```

Saída no Terminal

plaintext

```
Earnings Before Tax: 7000.00
Earnings After Tax: 5600.00
Tax Rate: 20.00%
```

Conclusão

O uso de funções como `Sprintf` permite maior controle sobre como as strings são manipuladas e exibidas. Separar a formatação da saída não apenas melhora a legibilidade

do código, mas também oferece flexibilidade para criar textos reutilizáveis em várias partes do programa.