Introdução ao Go

Aula 6: A Importância da Função "main"

Agora que entendemos a importância do pacote `main`, chegou a hora de falar sobre a **função** `main`. Assim como o pacote, a função `main` é essencial em programas Go. Vamos explorar o motivo.

O que é a função `main`?

- `main` **é uma função especial em Go.**
- Uma função é um bloco de código que pode ser executado chamando-a diretamente.
- No caso da função `main`, ela é **automática e obrigatória**: o Go a executa assim que o programa inicia.

Sem a função `main`, o Go não sabe onde começar a executar o código.

Diferença do Go para outras linguagens

Em linguagens como **JavaScript**, o código é executado de cima para baixo no arquivo principal, sem a necessidade de envolver tudo em uma função específica.
Já em Go, **o código principal deve estar dentro da função** `main`, como no exemplo abaixo:

```go
package main

import "fmt"

func main() {
    fmt.Println("Bem-vindo ao meu programa em Go!")
}
```

Regras importantes sobre a função `main`

1. **Um programa executável precisa de uma função `main`**
   - Se estiver construindo um programa que será executado, você **deve** ter essa função.
   - No entanto, se estiver criando uma biblioteca ou pacote (como o pacote `fmt`), a função `main` não é necessária.
2. **A função `main` deve existir apenas uma vez**
   - Se você tiver mais de um arquivo dentro do pacote `main`, **apenas um deles pode conter a função `main`**.
   - Caso contrário, o Go exibirá um erro:

     ```text
     main redeclared in this block
     ```
3. **Código principal deve estar dentro da função `main`**
   - Apenas as importações e declarações especiais podem estar fora de funções no arquivo.
   - Todo código executável deve ser colocado dentro de funções (normalmente, dentro da `main`).
4. **Um programa em Go começa pela função `main`**
   - É o ponto de entrada do programa, onde tudo começa.

Exemplo prático

1. **Código funcional com uma única função `main`**:

   ```go
   package main

   import "fmt"

   func main() {
   ```

```
    fmt.Println("Olá! Este é o início do programa.")
}
```

2. **Erro ao declarar duas funções `main` em diferentes arquivos no mesmo pacote:**

   Arquivo 1: `main.go`

   ```go
   package main

   import "fmt"

   func main() {
       fmt.Println("Executando o primeiro arquivo!")
   }
   ```

   Arquivo 2: `other.go`

   ```go
   package main

   func main() {
       println("Tentando declarar outra função main.")
   }
   ```

   Resultado:

   ```text
   ./other.go:3:6: main redeclared in this block
   ```

3. **Programa sem a função `main`:**

   ```go
   package main

   import "fmt"

   func printMessage() {
   ```

```
    fmt.Println("Esta função não será executada automaticamente.")
}
```

Resultado ao executar `go run`:

```text
cannot run non-main package
```

Bibliotecas Go não precisam de função `main`

Se você estiver criando um pacote para ser usado em outros projetos, como o pacote `fmt`, a função `main` não é necessária. O foco será apenas em expor as funções e funcionalidades que outros desenvolvedores podem importar e usar.

Exemplo de um pacote simples:

Arquivo: `mathutils/mathutils.go`

```go
package mathutils

func Add(a int, b int) int {
    return a + b
}

func Subtract(a int, b int) int {
    return a - b
}
```

Aqui, não existe a função `main`, pois este pacote é uma biblioteca. Ele será importado e usado por outro programa.

Resumo

- A função `main` é o ponto de entrada obrigatório para qualquer programa Go executável.
- Todo código principal do programa deve estar dentro da função `main`.
- Apenas uma função `main` pode existir por programa.
- Pacotes que não são executáveis (como bibliotecas) não precisam de função `main`.

Essa estrutura clara ajuda o Go a manter a organização e a escalabilidade dos projetos. Agora, com a função `main`, sabemos exatamente onde tudo começa!