

Para saber mais

Função com quantidade de parâmetros indeterminados

Conforme estudado nesta aula, uma função em Go pode ter um, muitos ou nenhum parâmetro:

```
package main

import (
    "fmt"
)

func SemParametro() string {
    return "Exemplo de função sem parâmetro!"
}

func UmParametro(texto string) string {
    return texto
}

func DoisParametros(texto string, numero int) (string, int) {
    return texto, numero
}

func main() {
    fmt.Println(SemParametro())
    fmt.Println(UmParametro("Exemplo de função com um parâmetro"))
    fmt.Println(DoisParametros("Passando dois parâmetros: essa string e o número", 10))
}
```

E nossa saída seria:

```
Exemplo de função sem parâmetro!
Exemplo de função com um parâmetro
Passando dois parâmetros: essa string e o número 10

Program exited.
```

[Neste link, você pode executar o código acima. \(https://play.golang.org/p/Kq5ZXTXLvRW\).](https://play.golang.org/p/Kq5ZXTXLvRW)

Porém, é possível que uma função receba um número indeterminado de parâmetros. Funções deste tipo são conhecidas em Go como [variadic functions \(https://en.wikipedia.org/wiki/Variadic_function\)](https://en.wikipedia.org/wiki/Variadic_function), ou função variádica.

Criando um função variádica

Para criar uma variadic function, devemos preceder o tipo do do argumento com reticências, conforme o exemplo abaixo:

```
package main

import (
    "fmt"
)

func Somando(numeros ...int) int {
    resultadoDaSoma := 0
    for _, numero := range numeros {
        resultadoDaSoma += numero
    }
    return resultadoDaSoma
}

func main() {
    fmt.Println(Somando(1))
    fmt.Println(Somando(1,1))
    fmt.Println(Somando(1,1,1))
    fmt.Println(Somando(1,1,2,4))
}
```

[Neste link, você pode executar o código acima. \(https://play.golang.org/p/s-kHUpafjOP\)](https://play.golang.org/p/s-kHUpafjOP)

Note o uso das reticências na declaração do parâmetro número: `numeros ...int`. Portanto, podemos criar uma função sem parâmetro, com um, dois, três, ou uma quantidade indeterminada de parâmetros com Go.