03

Inferência de tipos

Transcrição

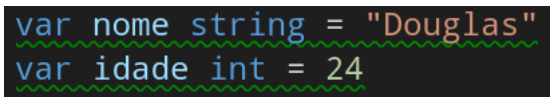
Vamos atribuir um valor à idade e voltar com a sua impressão:

```
package main

import "fmt"

func main() {
    var nome string = "Douglas"
    var idade int = 24
    var versao float32 = 1.1
    fmt.Println("Olá sr.", nome, "sua idade é", idade)
    fmt.Println("Este programa está na versão", versao)
}
```

No Visual Studio Code, podemos perceber que as variáveis `nome` e `idade` estão com um sublinhado verde:



O que isso significa?

Inferindo os tipos das variáveis

O Visual Studio nos dá esse aviso dizendo que nós podemos omitir o tipo da variável, pois o Go consegue inferir o tipo dessas variáveis. Ele consegue entender que, se a variável começa e termina com aspas, ela é uma string. Da mesma forma, se temos um número inteiro, sem casa decimal, o Go entenderá que a variável é do tipo inteiro:

```
package main

import "fmt"

func main() {
    var nome = "Douglas"
    var idade = 24
    var versao float32 = 1.1
    fmt.Println("Olá sr.", nome, "sua idade é", idade)
    fmt.Println("Este programa está na versão", versao)
}
```

Ao executar o programa, ele continua funcionando da mesma forma. Mas por que o Visual Studio não faz a mesma sugestão para a variável `versao` ?

Ele não faz a mesma sugestão pois há dois tipos flutuantes, o `float32` e o `float64` , logo o Go pode inferir qualquer um dos dois tipos à nossa variável, mas nada impede que façamos isso:

```
package main

import "fmt"

func main() {
    var nome = "Douglas"
    var idade = 24
    var versao = 1.1
    fmt.Println("Olá sr.", nome, "sua idade é", idade)
    fmt.Println("Este programa está na versão", versao)
}
```

Para saber se o Go conseguir inferir corretamente o tipo das variáveis, podemos descobri-los, importando o pacote `reflect` e chamando a sua função `TypeOf`, passando para ela a variável que queremos saber o tipo:

```
package main

import "fmt"
import "reflect"

func main() {
    var nome = "Douglas"
    var idade = 24
    var versao = 1.1
    fmt.Println("Olá sr.", nome, "sua idade é", idade)
    fmt.Println("Este programa está na versão", versao)

    fmt.Println("O tipo da variável idade é", reflect.TypeOf(versao))
}
```

E temos a seguinte saída:

```
alura@alura-01:~/go/src/hello$ go run hello.go
Olá sr. Douglas sua idade é 24
Este programa está na versão 1.1
O tipo da variável idade é float64
```

Assim vemos que, por padrão, o Go infere o maior tipo flutuante para a variável, o `float64`. Então, a menos que queiramos especificar que queremos o tipo `float32`, podemos omitir a declaração do tipo, que quando o Go ver um número decimal, ele será capaz de inferir o seu tipo.

Declaração curta de variáveis

Para deixar o nosso código mais limpo ainda, podemos remover a palavra `var` das variáveis. Podemos fazer isso pois o Go possui um segundo operador de atribuição de variáveis, um mais "curto", que é o `:=`. Quando utilizamos esse operador, estamos dizendo ao Go que estamos declarando uma variável e atribuindo um valor a ela:

```
package main

import "fmt"
```

```
func main() {
    nome := "Douglas"
    idade := 24
    versao := 1.1
    fmt.Println("Olá sr.", nome, "sua idade é", idade)
    fmt.Println("Este programa está na versão", versao)
}
```

Quando vermos o operador `:=`, significa que está sendo declarada uma variável, atribuindo o seu valor e inferindo o seu tipo. Além disso, esse operador possui outras vantagens, que veremos ao longo do treinamento.