

Go

Neni

6 de Agosto de 2019

Conteúdo

Listings	2
Lista de Figuras	2
Lista de Tabelas	2
1 O que é GO	3
2 Configurando ambiente	5
2.1 Instalando GO	5
2.2 Editores/plugins e IDEs	5
3 Ferramentas Go	7
4 Funções e packages	9
4.1 Estrutura de um arquivo GO	9
4.2 Funções	9
4.3 Packages	10
5 Documentando	13
6 Main packages	15
6.1 fmt	15
6.2 os	15
6.3 flag	15
6.4 http	15
6.5 log	15
6.6 strings	15
7 Variáveis e tipos	17
8 Arrays e slices	19
9 Fluxo de controle	21

10 Loops	23
11 Estruturas e interfaces	25
12 Ponteiros	27
13 Concorrência	29
14 Testes	31
15 Benchmarks	33
Bibliografia	35

Listings

4.1 Estrutura de um arquivo.go	9
4.2 Funções	9
4.3 Executável	10
4.4 Biblioteca em \$GOPATH/src/github.com/nenitf/abcdef	10
4.5 Executável utilizando biblioteca abcdef	11
5.1 Documentando pacote, variaveis, funcoes e tipos	13
5.2 Documentando pacotes	13

Lista de Figuras

Lista de Tabelas

3.1 Ferramentas go	7
------------------------------	---

Capítulo 1

O que é GO

Golang é uma linguagem de programação criada pela Google para resolver seus problemas [7, 1] . Desenvolvida por Robert Griesemer, Rob Pike e Ken Thompson em 11/2009 [7, 4] .

Características:

- Linguagem compilada [7, 1] ;
- Fortemente Tipada [7, 4] ;
- Código livre e aberto [7, 4] ;
- Tempo de compilação otimizado [7, 4] ;
- Garbage Collector [7, 4] ;
- Linguagem mínima [7, 4] ;
- C-like [7, 4] ;
- Não é OO, porém possui métodos e interfaces [7, 4] ;
- Concorrência com goroutines [7, 4] ;

Capítulo 2

Configurando ambiente

2.1 Instalando GO

Seguir as instruções de acordo com seu sistema operacional em: [Download](#) .

DICA:

GOPATH é a variável de ambiente que indica o local dos códigos go [2, 15] .

GOROOT é a variável de ambiente que indica o local de instalação do go (compilador e ferramentas) [7,

7]

2.2 Editores/plugins e IDEs

- [Visual Studio Code](#) : Linux, Windows e Mac;
- [Vim](#) : Linux, Windows e Mac¹;
- [NeoVim](#) : Linux, Windows e Mac¹;
- [Atom](#) : Linux, Windows e Mac.

¹ [Plugin](#)

Capítulo 3

Ferramentas Go

Tabela 3.1: Ferramentas go

Comando	Descrição
build	Compila pacote main
clean	
doc	
env	
bug	
fix	Formata o arquivo .go [5, 77]
fmt	
generate	
get	
install	
list	Download de pacotes a partir do caminho de importação [7, 9] . Usado para baixar do github [7, 9] .
run	
test	
tool	
version	
vet	Verifica erros no código [7, 9] .
c	
buildmode	
cache	
filetype	
gopath	
enviroment	
importpath	
packages	
testflag	
testfunc	

DICA:

Digitando `go` é possível ver todas as ferramentas da linguagem.

DICA:

O comando `godoc -http=:6060` cria servidor web com documentação de pacotes [5, 78]

Capítulo 4

Funções e packages

4.1 Estrutura de um arquivo GO

A estrutura comum de um arquivo possui 3 partes: Declaração de pacote, importe de pacotes/bibliotecas e funções - nessa ordem. Abaixo exemplos comentados (`//` ou `/* */` comentam códigos Go).

Listing 4.1: Estrutura de um arquivo.go

```
1 // declaração de pacote
2 package main
3
4 // importe de pacotes/bibliotecas
5 import (
6     "fmt"
7     "os"
8 )
9
10 // Funções
11 func main(){
12     // algoritmo
13 }
14
15 func abcde(){
16     // algoritmo
17 }
```

4.2 Funções

Função é um bloco de código nomeado com algoritmo que pode tanto receber (parâmetros) quanto retornar 0 ou mais valores ^[7,15]. Por pasta/pacote não devem haver funções com nome repetido ^[7,8] - Go não possui sobrecarga de métodos.

Listing 4.2: Funções

```
1 package main
2
3 import "fmt"
4
5 func main(){
6     // fmt.Print exibe resultado da função divideValor
7     fmt.Print(divideValor(4, 2))
8
9     // fmt.Print exibe resultado da função somaValores
```

```
10     fmt.Print(somaValores(4, 2))
11 }
12
13 // Recebe dois valores e retorna um
14 func divideValor(n1 int, n2 int) float64 {
15     // Retorna resultado da divisão
16     return n1/n2
17 }
18
19 // Recebe quantidade indefinida de valores (função variática) e retorna um
20 func somaValores(nums ...int) float64 {
21     // Variável incrementada com os valores
22     total := 0
23
24     // Loop pela quantidade de numeros passados
25     for _, num := range nums {
26         // total = total + num
27         total += num
28     }
29
30     // Retorna total
31     return total
32 }
```

4.3 Packages

Pacotes são a maneira de Go organizar e reutilizar código [2, 18] . Há dois tipos de programa em Go: executáveis e bibliotecas [2, 18] .

- Executáveis necessitam de um arquivo com pacote main declarado e uma função main [7, 8] .
- Bibliotecas são arquivos que podem ser reutilizados em qualquer projeto quando importados. Funções, variáveis e tipos podem ser visíveis caso seu nome comece com letra maiúscula [2, 106] .

Listing 4.3: Executável

```
1 package main
2
3 func main(){
4     // ...
5 }
```

Listing 4.4: Biblioteca em \$GOPATH/src/github.com/nenitf/abcdef

```
1 // Nome do package
2 package abcdef
3
4 // Função exposta (letra inicial maiúscula)
5 // Pode ser usada por um executável
6 func ProximaLetraAlfabeto(){
7     // ...
8 }
9
10 // Função oculta (letra inicial minúscula)
11 // Só pode ser usada dentro da própria biblioteca
```

```
12 func loopAlfabeto(){  
13     // ...  
14 }
```

Listing 4.5: Executável utilizando biblioteca abcdef

```
1 package main  
2  
3 // importando pacote  
4 import "github.com/nenitf/abcdef"  
5  
6 func main(){  
7     absdef.ProximaLetraAlfabeto()  
8 }
```

Go possui uma série de pacotes próprios que podem ser importados, no capítulo 6 os mais usuais são explicados.

Capítulo 5

Documentando

Para documentar pacotes, funções, variáveis e tipos visíveis, basta comentar imediatamente acima da declaração [5, 80] .

Listing 5.1: Documentando pacote, variaveis, funcoes e tipos

```
1 package ccord
2
3 // GMStoUTM converte coordenada GMS para UTM
4 func GMStoUTM() {
5     // ...
6 }
```

DICA:

Pode criar arquivo doc.go no diretório do seu pacote unicamente para documentá-lo [5, 81]

Listing 5.2: Documentando pacotes

```
1 /*
2 Pacote responsável por
3 calculos com coordenadas
4 */
5 package ccord
```


Capítulo 6

Main packages

Funções mais usuais dos pacotes mais comuns da biblioteca padrão do Go.

6.1 `fmt`

6.2 `os`

6.3 `flag`

6.4 `http`

6.5 `log`

6.6 `strings`

Capítulo 7

Variáveis e tipos

Capítulo 8

Arrays e slices

Capítulo 9

Fluxo de controle

Capítulo 10

Loops

Capítulo 11

Estruturas e interfaces

Capítulo 12

Ponteiros

Capítulo 13

Concorrência

Capítulo 14

Testes

Capítulo 15

Benchmarks

Bibliografia

- [1] A. Donovan and B. Kernighan. *A Linguagem de Programação Go*. Novatec, 1 edition, 3 2017.
- [2] C. Doxsey. *Introdução à linguagem Go*. Novatec, 1 edition, 4 2016.
- [3] C. Filipini. *Programando em Go*. Casa do Código, 1 edition, 2016.
- [4] Google. *Go Documentation*. <https://golang.org/doc>.
- [5] W. Kennedy, B. Ketelsen, and E. Martin. *Go em Ação*. Novatec, 1 edition, 5 2016.
- [6] E. Körbes. *A linguagem go*. <https://greatercommons.com/learn/golang-ptbr>, 2019. Curso Online.
- [7] L. Leitão. *Go (golang): Explorando a linguagem do google*. <https://www.udemy.com/curso-go>, 2017. Curso Online.
- [8] W. Willians. *Avançando com go lang*. <https://www.schoolofnet.com/curso/go-lang/linguagem-go/avancando-com-go-lang/>, 2016. Curso Online.
- [9] W. Willians. *Iniciando com go lang*. <https://www.schoolofnet.com/curso/go-lang/linguagem-go/iniciando-com-go-lang/>, 2016. Curso Online.
- [10] W. Willians. *Iniciando com go lang - oo*. <https://www.schoolofnet.com/curso/go-lang/linguagem-go/iniciando-com-golang-oo>, 2016. Curso Online.