UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

String

QXD0001 - Fundamentos de Programação

Prof. Bruno Góis Mateus (brunomateus@ufc.br)

Fonte: https://ttemporin.dev/imersao/

Agenda

- Introdução
- String
- Percorrendo string
- Manipulando string
- Erros comuns

Introdução

Introdução

String

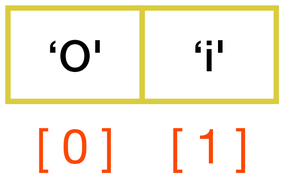
- Em vários programas precisamos de texto:
  - Nome do usuário
  - Senha
  - Email
  - Mensagens
- Tudo isso é representado por String

String

String

O que são?

- Uma string é uma sequência de caracteres
- Um slice de bytes de leitura exclusiva (imutável)



String

Declarando e imprimindo

```
// Declaração e inicialização
var s1 string = "Oi"
s2 := "Mundo"

fmt.Println(s1, s2)
```

String

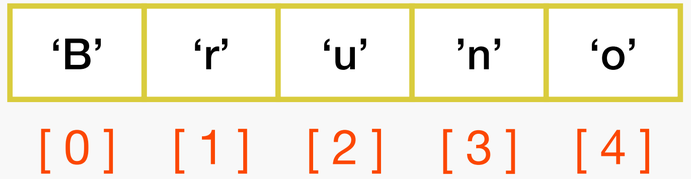
Acessando via índice

```
nome := "Bruno"

fmt.Println(nome[0])
fmt.Println(nome[1])
```

66

114



Uma string é um slice de bytes de leitura onde cada caracteres é representado por um número (ASCII / Unicode).

String

Convertendo para caractere

```
nome := "Bruno"

fmt.Printf("%c\n", nome[0])  ⟶  B
fmt.Printf("%c\n", nome[1])  ⟶  r
```

Percorrendo strings

Percorrendo vetores

A forma tradicional

```
nome := "Bruno"
sobrenome := "Góis"

for i := 0; i < len(nome); i++ {
  fmt.Printf("%c\n", nome[i])
}

for i := 0; i < len(sobrenome); i++ {
  fmt.Printf("%c\n", sobrenome[i])
}
```

Retorna o tamanho da string

‘B’	‘r’	‘u’	’n’	‘o’

‘G’	‘Ã’	‘³’	’i’	’s’

‘G’	‘ó’	’i’	’s’

Retorna 5 por conta do caractere ó que ocupa 2 bytes porque o Go usa UTF-8 por padrão.

Percorrendo strings

O Caractere de Verdade - Rune

- Para lidar com letras reais (independente de quantos bytes ocupam), o Go usa o conceito de Rune
  - Apenas um apelido pro tipo int32
- Podemos usá-lo para contar corretamente o número de caracteres

```
import "unicode/utf8"

totalLetras := utf8.RuneCountInString("Góis") // Retorna 4
```

Percorrendo vetores

Percorrendo do jeito certo

```
sobrenome := "Góis"

for indice, letra := range sobrenome {
    fmt.Printf("Byte: %d | Letra: %c\n", indice, letra)
}
```

Faz a decodificação UTF-8 automaticamente

Manipulando strings

Manipulando strings

- Ao trabalhar com Strings é comum que precisemos manipulá-las de alguma forma
  - Comparar
  - Ignorar letras maiúsculas ou minúsculas
  - Realizar substituições/alterar
  - Concatenar

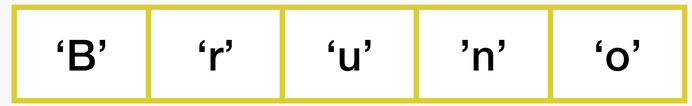
Manipulando strings

Concatenando e comparando

```
nome := "Bruno"
sobrenome := "Góis"

completo := nome + " " + sobrenome

if nome == "bruno" {
    fmt.Println("Usuário encontrado")
}
```



false

Manipulando strings

Imutabilidade

- Por serem parecidas com um Slice é natural que tentemos alterar uma string da seguinte forma:

```
palavra := "Golang"
palavra[0] = 'J' // ERRO DE COMPILAÇÃO!
```

Uma vez criada, uma string não pode ser alterada caractere por caractere

Por que o Go faz isso?

Segurança e performance. Múltiplas partes do programa podem ler a mesma string na memória sem risco de que alguém a corrompa.

Manipulando strings

O pacote strings

Função	Descrição
`Compare(a, b string) int`	Compara a e b e retorna um inteiro. 0 se a == b, -1 se a < b e 1 se a > b
`Contains(s, substr string) bool`	Indica se a substring substr está contida em s
`ContainsRune(s string, r rune) bool`	Indica se o caractere r está contida em s
`Count(s, substr string) int`	Retorna o número de ocorrência de substr em s
`EqualFold(s, t string) bool`	Compara duas strings ignorando maiúsculas e minúsculas
`Fields(s string) []string`	Divide a string s em torno de cada ocorrência de um ou mais caracteres de espaço em branco consecutivos
`HasPrefix(s, prefix string) bool`	Indica se a string inicia com o prefixo indicado

Manipulando strings

O pacote strings

Função	Descrição
`HasSuffix(s, suffix string) bool`	Indica se a string termina o com sufixo fornecido
`Index(s, substr string) int`	Retorna o índice da primeira ocorrência de substr em s, ou -1 se não estiver presente
`IndexRune(s string, r rune) int`	Retorna o índice da primeira ocorrência de caracteres em s, ou -1 se não estiver presente
`Join(elems []string, sep string) string`	Concatena os elementos do seu primeiro argumento para criar uma única string onde a sep é colocada
`LastIndex(s, substr string) int`	Retorna o índice da última ocorrência de substr em s, ou -1 se não estiver presente
`Replace(s, old, new string, n int) string`	Retorna uma string com as primeiras n ocorrências não sobrepostas de old substituídas por new
`ReplaceAll(s, old, new string) string`	Retorna uma string com as todas as ocorrências não sobrepostas de old substituídas por new

Manipulando strings

O pacote strings

Função	Descrição
`Split(s, sep string) []string`	Divide as substrings de s em todas as substrings separadas por sep e retorna um slice
`SplitAfter(s, sep string) []string`	Divide as substrings de s em todas as substrings separadas por sep, mas mantém o separador e retorna um slice
`ToLower(s string) string`	Retorna s com todas as letras minúsculas em Unicode
`ToUpper(s string) string`	Retorna s com todas as letras maiúsculas em Unicode
`Trim(s, cutset string) string`	Retorna uma string com todos as ocorrências de cutset iniciais e finais removidos
`TrimPrefix(s, prefix string) string`	Retorna uma string com o prefixo removido
`TrimSuffix(s, prefix string) string`	Retorna uma string com o sufixo removido

Erros comuns

Erros comuns

❌ Índice fora do limite

```
nome := "Bruno"

fmt.Println(nome[10]) ❌
```

Erros comuns

❌ Erro 2— Acentos

```
fmt.Println(len("á")) // Não é 1   ❌
```

Referências

- Go 101
- Documentação Oficial do Pacote strings