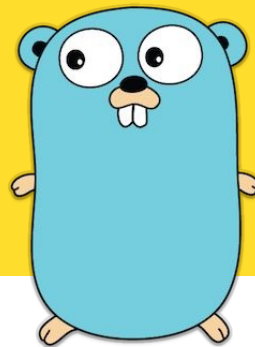


Занятие 1

# ОСНОВЫ



# План

---

- История языка
- Типы данных
- Структура кода
- Домашнее задание

История

# Цель создания

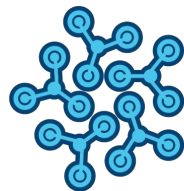
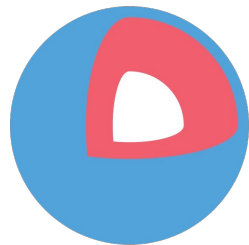
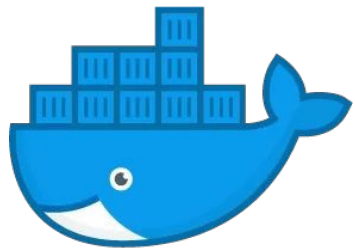
---

Простота быстрой разработки  
больших распределенных систем

[talks.golang.org/2012/splash.article](https://talks.golang.org/2012/splash.article)

[golang.org/doc/faq#What is the purpose of the project](https://golang.org/doc/faq#What_is_the_purpose_of_the_project)

# Что получилось



InfluxDB



Grafana

CoreOS



Prometheus



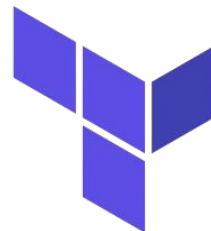
Cockroach DB



Dropbox



etcd



NATS



HashiCorp

Consul

Syncthing



Vault

kubernetes

# История

начало  
разработки

2007



# История

начало  
разработки



2007



2010



# Команда Go

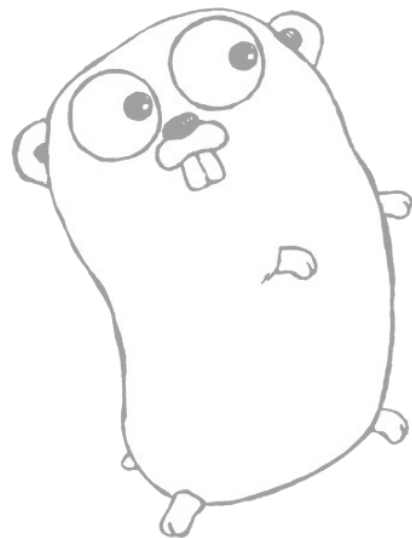
---

Роберт Гризмер  
V8, HotSpot

Роб Пайк  
Plan 9, UTF-8

Кен Томпсон  
Unix, Plan 9, C, UTF-8, регулярные выражения

Брэд Фицпатрик  
LiveJournal, Memcached, OpenID, Gearman






# История

---

начало  
разработки

2007

A horizontal timeline line starting from a small circle at the year 2007 and extending to the right.

# История

---

начало  
разработки



2007

open source  
анонс



2009



# История

---

начало  
разработки



2007

open source  
анонс



2009

Go 1.0

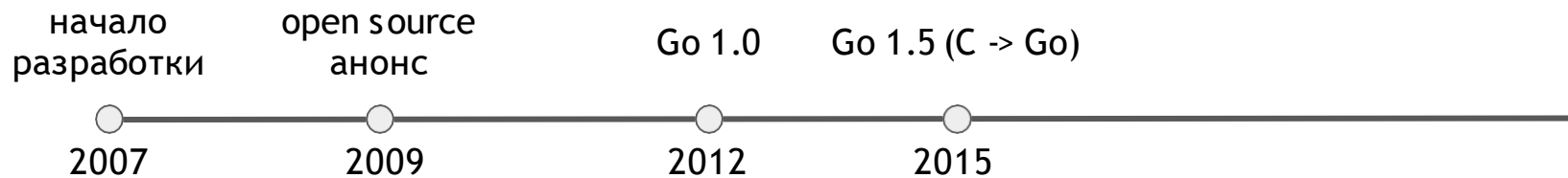


2012



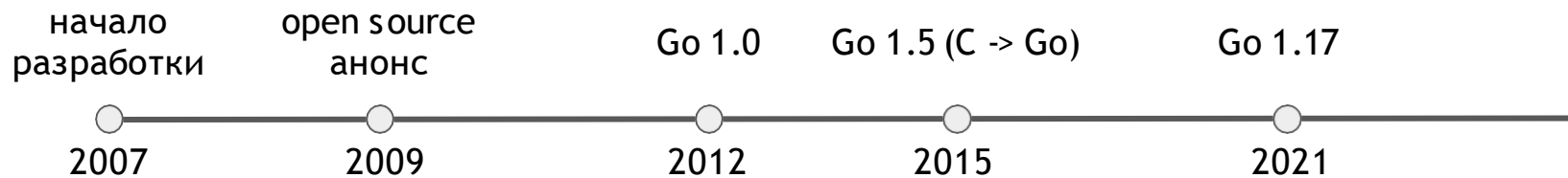
# История

---



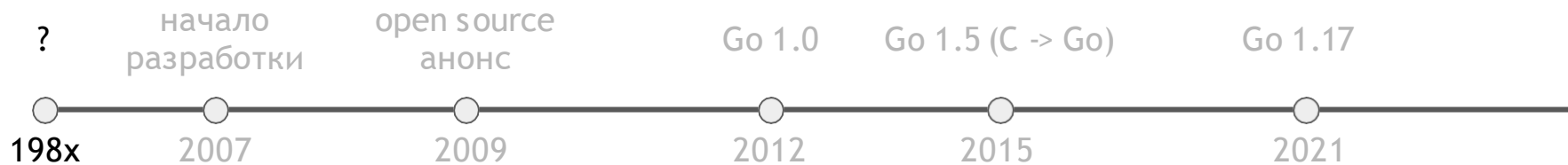
# История

---



# История

---



Подробнее о истории



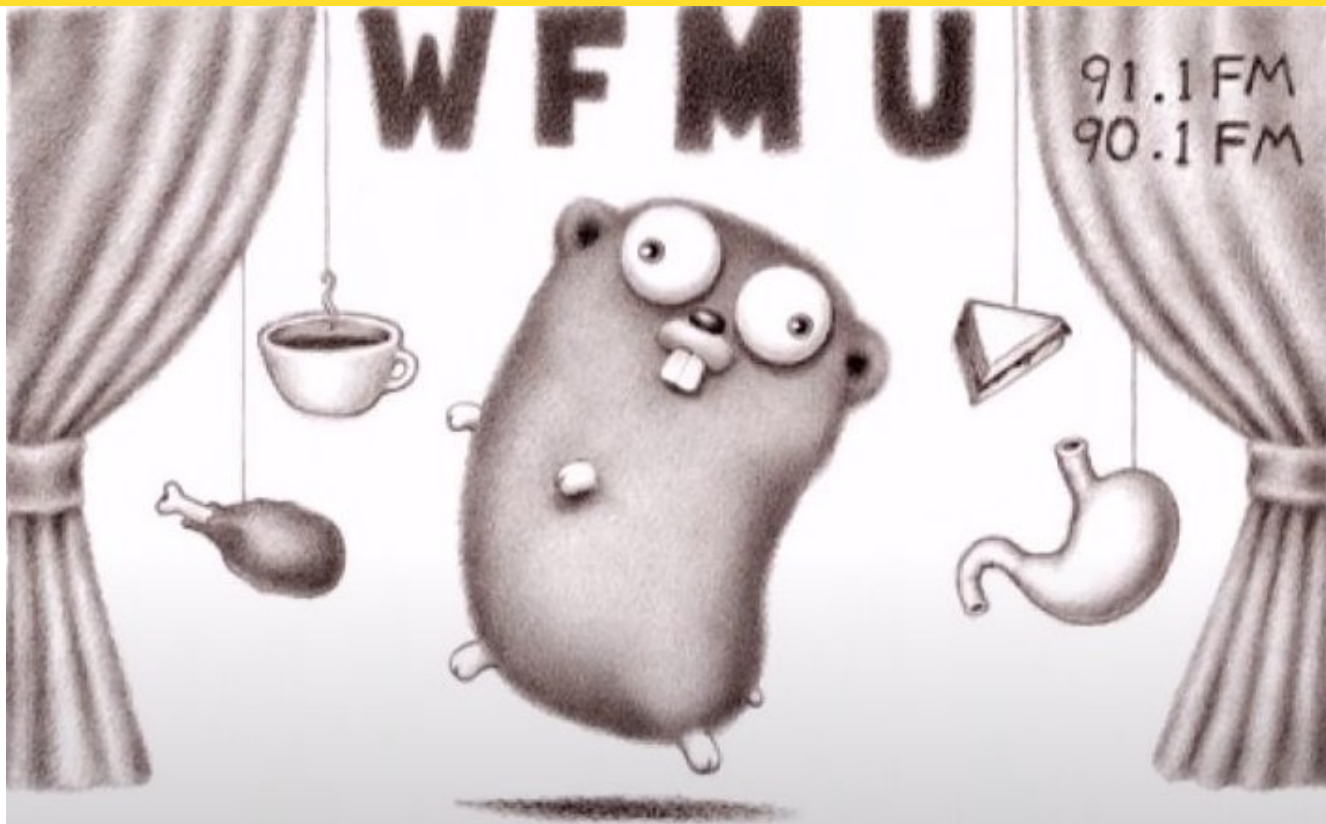
Почему Golang такой  
странный  
Филипп Кулин  
(Дремучий лес)



Golang  
Conf 2019

[youtu.be/ql-uncsqoAU](https://youtu.be/ql-uncsqoAU)

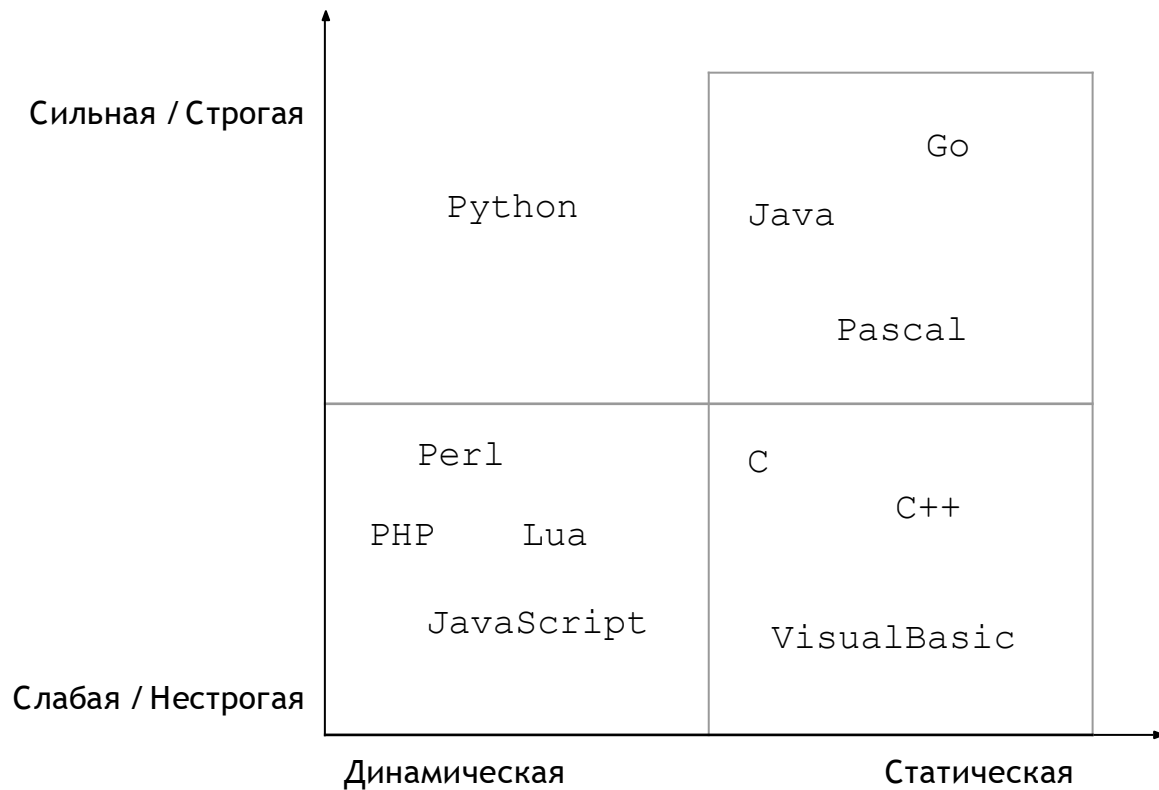
# Маскот





Типизация

# Типизация



# Типизация

---

Сильная / Строгая

Все переменные имеют тип

Типы нельзя смешивать

```
42 + 't'
```

```
int(42) * float64(0.5)
```

# Типизация

Сильная / Строгая

Все переменные имеют тип

Типы нельзя смешивать

```
42 + 't'
```

```
int(42) * float64(0.5)
```

Статическая

Тип определяется при компиляции

Тип не меняется никогда

```
a := 42  
a = 0.5  
a = `hello`
```

# Типы

Скалярные				Композитные (агрегатные)		
Числа	Строки	Символы (руны)	Булев (логический)	Функции	Массивы Слайсы	Остальные типы
Целочисленные <code>int, uint16, ...</code>	<code>string</code> <code>"hello</code> <code>"</code>	<code>rune</code> <code>'w'</code>	<code>bool</code> <code>true false</code>	<code>func ()</code>		Словари
Вещественные <code>float32, float64</code>	<code>`world`</code>					
Комплексные <code>complex64,</code> <code>complex128</code> <code>40 + 2i</code>						Структуры
						Каналы
						Интерфейсы

# Типы

Скалярные				Композитные (агрегатные)		
Числа	Строки	Символы (руны)	Булев (логический)	Функции	Массивы Слайсы	Остальные типы
Целочисленные <code>int, uint16, ...</code>	<code>string</code> <code>"hello"</code> <code>`world`</code>	<code>rune</code> <code>'w'</code>	<code>bool</code> <code>true false</code>	<code>func ()</code>		Словари
Вещественные <code>float32, float64</code>						
Комплексные <code>complex64,</code> <code>complex128</code> <code>40 + 2i</code>						Структуры  Каналы  Интерфейсы

# Типы - логический

---

# Типы - логический

---

```
var a bool
```

```
b := true
```

```
c := false
```

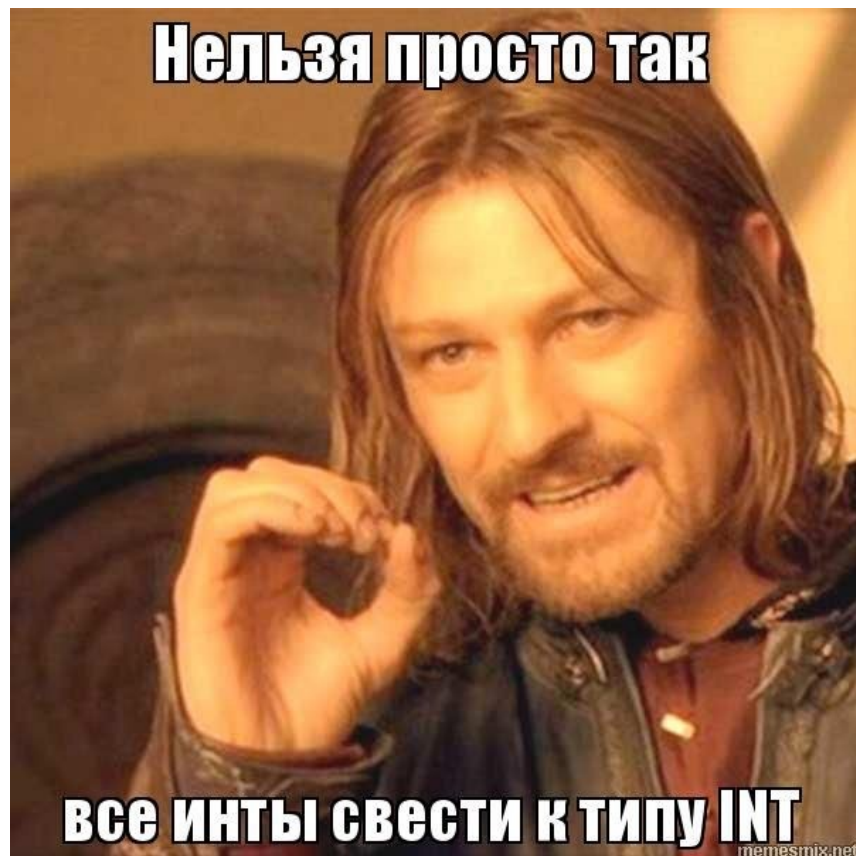
```
d := True // ошибка
```

```
e := FALSE // ошибка
```

```
println(true, false)      true false
```



# Типы - числа



# Типы - числа

---

int8    int16    int32    int64

uint8   uint16   uint32   uint64

# Типы - числа

---

int8    int16    int32    int64    int

uint8   uint16   uint32   uint64   uint

# Типы - числа

---

int8    int16    int32    int64    int

uint8   uint16   uint32   uint64   uint

```
a := int8(42)
```

```
var b int64
```

```
c := 7 // int
```

```
b = a // ошибка
```

```
b = int64(a)
```

# Типы - числа

---

float32   float64

[wikipedia.org/wiki/IEEE 754](https://wikipedia.org/wiki/IEEE_754)

```
d := float32(42)
```

```
var e = float64(0.5)
```

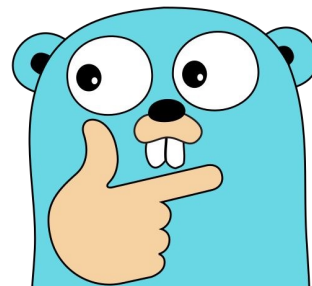
```
e * d // ошибка
```

```
f := 4.2 // float64
```

# Типы - числа

---

```
var (  
    a uint = 1  
    b  int = 2  
)  
  
println(a + b)
```

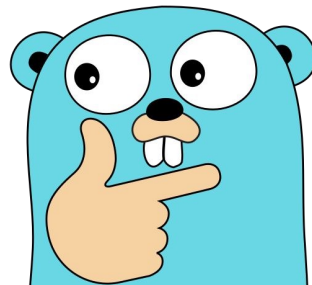


Картинка из  
[clipartmax.com/middle/m2i8Z5Z5Z5N4G6i8\\_view-golang-think-logo-golang/](https://clipartmax.com/middle/m2i8Z5Z5Z5N4G6i8_view-golang-think-logo-golang/)

# Типы - числа

---

```
import (  
    "math"  
)  
  
func main() {  
    a := math.MaxInt64  
    println(a)  
    a += 1  
    println(a)  
}
```



# Типы - строки

---



# Типы - строки

---

```
var s string
```

```
a := "hel \n lo"
```

```
b := `hel \n lo`
```

```
c := "hel \\n lo"
```

```
println(a)      hel  
                  lo
```

```
println(b)      hel \n lo
```

```
println(c)      hel \n lo
```

# Типы - строки

```
var s string
```

```
a := "hel \n lo"
```

```
b := `hel \n lo`
```

```
c := "hel \\n lo"
```

```
println(a)      hel  
                  lo
```

```
println(b)      hel \n lo
```

```
println(c)      hel \n lo
```

```
a.:= "hel
```

```
lo"           // ошибка
```

```
b.:= `hel
```

```
lo`
```

```
println(b)      hel  
                  lo
```

# Типы - строки

---

```
a := "Привет!"
```

```
println(len(a), a)      13 Привет!
```

# Типы - строки

---

```
a := "Привет!"
```

```
println(len(a), a)
```

```
13 Привет!
```

Все строки - UTF-8

# Типы - строки

```
import "encoding/hex"
```

```
func main() {
```

```
    a := "Привет!"
```

```
    println(a[1])
```

```
    println(hex.Dump([]byte(a)))
```

```
}
```

```
159 // или 9f
```

```
00000000 d0 9f d1 80 d0 b8 d0 b2 d0 b5 d1 82 21
```

```
|.....!|
```

# Типы - строки

---

## Весь код Go - UTF-8

```
func привет(имя string) {  
    print("Привет, ", имя)  
}
```

```
func main() {  
    имя := "Лёша"  
    привет(имя)  
}
```

[play.golang.org/p/99nZabgIyBQ](https://play.golang.org/p/99nZabgIyBQ)

# Типы - строки

## Весь код Go - UTF-8

```
func привет(имя string) {  
    print("Привет, ", имя)  
}  
  
func main() {  
    имя := "Лёша"  
    привет(имя)  
}
```

[play.golang.org/p/99nZabgIyBQ](https://play.golang.org/p/99nZabgIyBQ)



Картинка из  
[github.com/lpkg/gohalt](https://github.com/lpkg/gohalt)

# Типы - строки

---

```
s := "Hello"
```

```
a := s
```

```
a += " World"
```

```
fmt.Println(s + "\n" + a)
```

```
Hello
```

```
Hello World
```



# Типы - символы

---

# Типы - символы

---

```
a := 'F'
a = 'Щ'
a = '0f' }
```

UTF-8 символы (character code point)

Максимальное\* значение UTF-8 символа?

4 байта  $\Rightarrow$  int32

\* [pkg.go.dev/unicode/utf8@go1.17#pkg-constants](https://pkg.go.dev/unicode/utf8@go1.17#pkg-constants)

# Типы - символы

---

```
a := 'F'
```

```
a = 'Щ'
```

```
a = '0ґ'
```

```
fmt.Printf("%T %T %T %T %T", 'F', 'Щ', '0ґ', "F", `0ґ`)
```

```
// %T - "выведи ТИП значения"
```

```
int32 int32 int32 string string
```

# Типы - СИМВОЛЫ

---

```
import (  
    "fmt"  
    "unicode/utf8"  
)  
  
func main() {  
    var c rune = utf8.MaxRune // '\U0010FFFF'  
  
    fmt.Println(c)  
}
```

1114111

# Типы - символы

---

```
a := int32(40)
```

```
b := rune(2)
```

```
println(a + b)    42
```

# Типы - символы

---

## Очень популярная ошибка

```
a := 70  
println(string(a))
```

```
b := 7000  
println(string(b))
```

F

```
go vet:  
conversion from int to string yields a string of one rune,  
not a string of digits (did you mean fmt.Sprint(x)?)
```

# Типы - символы

---



# Типы - СИМВОЛЫ

---

[pkg.go.dev/unicode](https://pkg.go.dev/unicode)

[pkg.go.dev/unicode/utf8](https://pkg.go.dev/unicode/utf8)

```
import "unicode/utf8"
func main() {
    println(utf8.RuneCountInString("Привет!"))
}
```

7



# Типы - СИМВОЛЫ

---

[pkg.go.dev/unicode](https://pkg.go.dev/unicode)

[pkg.go.dev/unicode/utf8](https://pkg.go.dev/unicode/utf8)

```
import "unicode/utf8"
func main() {
    println(utf8.RuneCountInString("Привет!"))
}
```

Функция `strings.TrimSpace()` и подобные поддерживают unicode

# Типы - СИМВОЛЫ

[pkg.go.dev/unicode](https://pkg.go.dev/unicode)

[pkg.go.dev/unicode/utf8](https://pkg.go.dev/unicode/utf8)

```
import "unicode/utf8"
func main() {
    println(utf8.RuneCountInString("Привет!"))
}
```

7

Функция `strings.TrimSpace()` и подобные поддерживают unicode

```
for _, char := range "Привет!" {
    println(char, string(char))
}
```

П 1055

р 1088

и 1080

в 1074

е 1077

т 1090

! 33

# Типы - значения по умолчанию

---

# Типы - значения по умолчанию

---

```
var (  
    s string  
    a int32  
    b bool  
)  
  
println(s, a, b)
```

# Типы - значения по умолчанию

---

```
var (                                /* zero values
    s string                          ``
    a int32                           0
    b bool                            false
)

println(s, a, b)                    0 false
*/
```

# Типы - указатели

---

# Типы - указатели

---

```
a := 42
```

```
b := a
```

```
a = 100500
```

```
println(a, b) // 100500 42
```

# Типы - указатели

---

```
a := 42
```

```
b := &a
```

```
a = 100500
```

```
println(a, b) // 100500 0xc000034768
```

```
println(a, *b) // 100500 100500
```



# Типы - указатели

---

```
a := 42
```

```
var b *int = &a
```

```
a = 100500
```

```
println(a, *b) // 100500 100500
```

# Типы - указатели

---

`&var` - взятие адреса (получение указателя)

`&42, &"hello" // ошибка компиляции`

# Типы - указатели

---

`&var` - взятие адреса (получение указателя)

```
&42, &"hello" // ошибка компиляции
```

`*ptr` - разыменование указателя (получение значения)

```
var b *int // == nil  
println(*b)
```

```
panic: runtime error: invalid memory address or nil pointer dereference  
[signal SIGSEGV: segmentation violation code=0x1 addr=0x0 pc=0x454ef6]
```

# Типы - массивы

---

# Типы - массивы

---

```
var a [50]int // помним про zero values
```

# Типы - массивы

---

```
var a [50]int // помним про zero values
```

```
a := [...]int{1, 2, 3, 4}
```

# Типы - массивы

---

```
var a [50]int // помним про zero values
```

```
a := [...]int{1, 2, 3, 4}
```

```
a := [2]int{1, 2, 3, 4} // ошибка
```

```
a := [2]int{1}
```

```
println(a[1])    0
```

# Типы - массивы

---

```
var a [4][2]int
```

```
a[2][1] = 42
```

```
fmt.Println(a)    [[0 0] [0 0] [0 42] [0 0]]
```



# Типы - массивы

---

```
var a [4][2]int
```

```
a[2][1] = 42
```

```
fmt.Println(a)    [[0 0] [0 0] [0 42] [0 0]]
```

```
a[2][100] = 42 // ошибка компиляции
```

# Типы - массивы

---

```
var a [4][2]int
```

```
a[2][1] = 42
```

```
fmt.Println(a)    [[0 0] [0 0] [0 42] [0 0]]
```

```
a[2][100] = 42 // ошибка компиляции
```

```
i := 100
```

```
a[2][i] = 42 // ошибка при исполнении (check)
```

# Типы - массивы

---

```
var a [5]int
fmt.Println(a)    [0 0 0 0 0]
println(len(a))   5
```

```
a[3] = 1
fmt.Println(a)    [0 0 0 1 0]
```

```
b := a
fmt.Println(b)    [0 0 0 1 0]
```

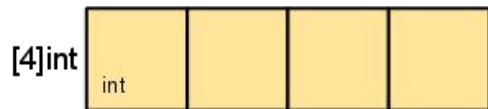
```
b[2] = 7
fmt.Println(b)    [0 0 7 1 0]
```

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

# Типы - массивы

---

```
var a [4]int
```



# Типы - слайсы

---



Картинка из  
[yahoo.com/entertainment/let-the-sounds-of-a-dune-sandworm-sooth-your-102309629482.html](http://yahoo.com/entertainment/let-the-sounds-of-a-dune-sandworm-sooth-your-102309629482.html)

# Типы - слайсы

---

```
var a []int  
fmt.Println(a)
```

```
[]
```

# Типы - слайсы

---

```
var a []int
```

```
fmt.Println(a)
```

```
[]
```

```
a = append(a, 1)
```

```
fmt.Println(a)
```

```
[1]
```

```
a = append(a, 2, 3, 4)
```

```
fmt.Println(a)
```

```
[1 2 3 4]
```

```
a = append(a, []int{5, 6}...)
```

```
fmt.Println(a)
```

```
[1 2 3 4 5 6]
```

# Типы - слайсы

---

```
a := []int{1, 2, 3}
```

```
fmt.Println(a)
```

```
println(len(a))
```

```
[1 2 3]
```

```
3
```

```
a = append(a, 42)
```

```
fmt.Println(a)
```

```
println(len(a))
```

```
[1 2 3 42]
```

```
4
```

```
b := a
```

```
b[1] = 7
```

```
fmt.Println(a)
```

```
fmt.Println(b)
```

```
[1 7 3 42]
```

```
[1 7 3 42]
```



# Типы - слайсы

---

```
var a []int
a = append(a, 1, 2)
fmt.Println(a)
println(len(a))
```

```
[1 2]
2
```

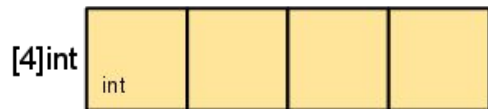
```
b := a
b = append(b, 3, 4, 5, 6, 7, 8)
fmt.Println(a)
fmt.Println(b)
println(len(a), len(b))
```

```
[1 2]
[1 2 3 4 5 6 7 8]
2 8
```

# Типы - слайсы

---

```
var a [4]int
```



# Типы - слайсы

```
a := []byte{1, 2, 3, 4, 5}
```

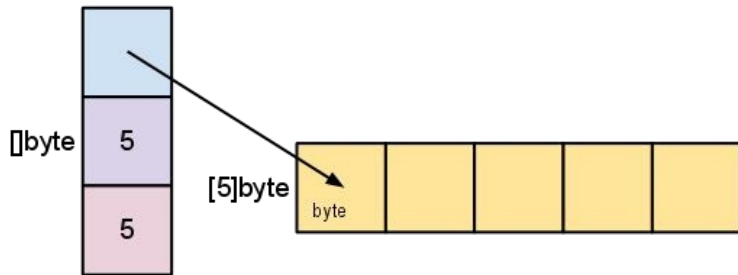
```
type SliceHeader struct {
```

```
    Data uintptr
```

```
    Len  int
```

```
    Cap  int
```

```
}
```



# Типы - слайсы

```
a := []byte{1, 2, 3, 4, 5}
```

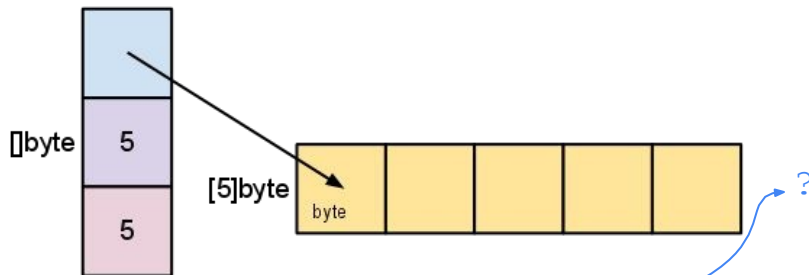
```
type SliceHeader struct {
```

```
    Data uintptr
```

```
    Len  int
```

```
    Cap  int
```

```
}
```



```
a = append(a, 6)
```

# Типы - свои

---

	len(a)	cap(a)	len(b)	cap(b)
a := []byte{}	0	0		

# Типы - СВОИ

---

	len(a)	cap(a)	len(b)	cap(b)
a := []byte{}	0	0		
a = append(a, 1, 2)	2	2		

# Типы - свои

	len(a)	cap(a)	len(b)	cap(b)
a := []byte{}	0	0		
a = append(a, 1, 2)	2	2		
b := a	2	2	2	2

# Типы - свои

	len(a)	cap(a)	len(b)	cap(b)
a := []byte{}	0	0		
a = append(a, 1, 2)	2	2		
b := a	2	2	2	2
b = append(b, 3, 4, 5, 6, 7, 8)	2	2	8	8



# Типы - слайсы

---

```
var a []int
```

```
a = append(a, 1, 2)
```

# Типы - слайсы

---

```
var a []int
```

```
a = append(a, 1, 2)
```

```
append(a, 3, 4) // ошибка компиляции
```

# Типы - слайсы

---

```
var a []int
```

```
a = append(a, 1, 2)
```

```
append(a, 3, 4) // ошибка компиляции
```

```
_ = append(a, 3, 4)
```

```
fmt.Println(len(a), cap(a), a)      2 2 [1 2]
```

# Типы - слайсы

---

```
a := make([]int, 5, 10)
```

```
fmt.Println(len(a), cap(a), a)      5 10 [0 0 0 0 0]
```

# Типы - слайсы

---

```
a := make([]int, 5, 10)
```

```
fmt.Println(len(a), cap(a), a)      5 10 [0 0 0 0 0]
```

```
a[2] = 2
```

```
a[7] = 7 // ошибка исполнения
```

# Типы - слайсы

---

```
a := make([]int, 5, 10)
```

```
fmt.Println(len(a), cap(a), a)      5 10 [0 0 0 0 0]
```

```
a[2] = 2
```

```
a[7] = 7 // ошибка исполнения
```

```
a = append(a, 0, 0, 0)
```

```
a[7] = 7
```

```
fmt.Println(len(a), cap(a), a)      8 10 [0 0 2 0 0 0 0 7]
```

# Типы - слайсы

```
a := make([]int, 0, 100)
```

```
a = append(a, 1, 2)
```

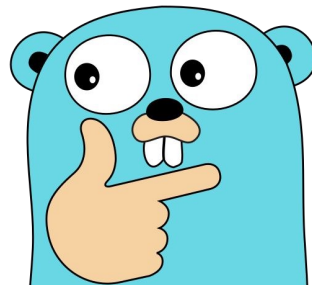
```
b := a
```

```
b = append(b, 3, 4, 5, 6, 7, 8)
```

```
a[1] = 0
```

```
fmt.Println(len(a), cap(a), a)
```

```
fmt.Println(len(b), cap(b), b)
```



# Типы - слайсы

---

```
a := make([]int, 0, 100)
```

```
a = append(a, 1, 2)
```

```
b := a
```

```
b = append(b, 3, 4, 5, 6, 7, 8)
```

```
a[1] = 0
```

```
fmt.Println(len(a), cap(a), a)
```

```
fmt.Println(len(b), cap(b), b)
```

```
2 100 [1 0]
```



# Типы - слайсы

```
a := make([]int, 0, 100)
```

```
a = append(a, 1, 2)
```

```
b := a
```

```
b = append(b, 3, 4, 5, 6, 7, 8)
```

```
a[1] = 0
```

```
fmt.Println(len(a), cap(a), a)
```

```
fmt.Println(len(b), cap(b), b)
```

```
2 100 [1 0]
```

```
8 100 [1 0 3 4 5 6 7 8]
```



# Типы - слайсы

```
a := make([]int, 0, 100)
```

```
a = append(a, 1, 2)
```

```
b := a
```

```
b = append(b, 3, 4, 5, 6, 7, 8)
```

```
a[1] = 0
```

```
fmt.Println(len(a), cap(a), a)      2 100 [1 0]
```

```
fmt.Println(len(b), cap(b), b)      8 100 [1 0 3 4 5 6 7 8]
```

```
fmt.Println(&a[0], &b[0], &a[0] == &b[0]) 0xc0000b8000 0xc0000b8000 true
```

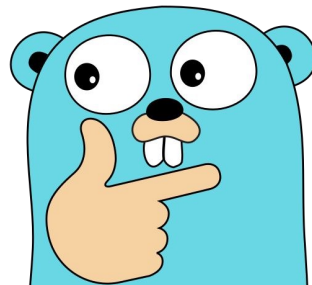
# Типы - слайсы

---

## Очень популярная ошибка

```
a := make([]string, 100)
a = append(a, `hello`)

println(len(a), cap(a))
```



# Типы - слайсы

---

## Очень популярная ошибка

```
a := make([]string, 100)
```

```
a = append(a, `hello`)
```

```
println(len(a), cap(a))
```

```
101 200
```

# Типы - слайсы

---

```
a := []byte{1, 2, 3, 4, 5}
```

```
b := a[2:4] // a[2, 4)
```

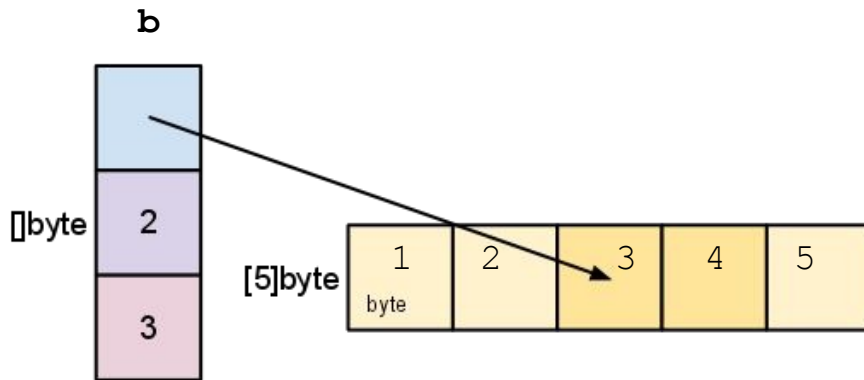
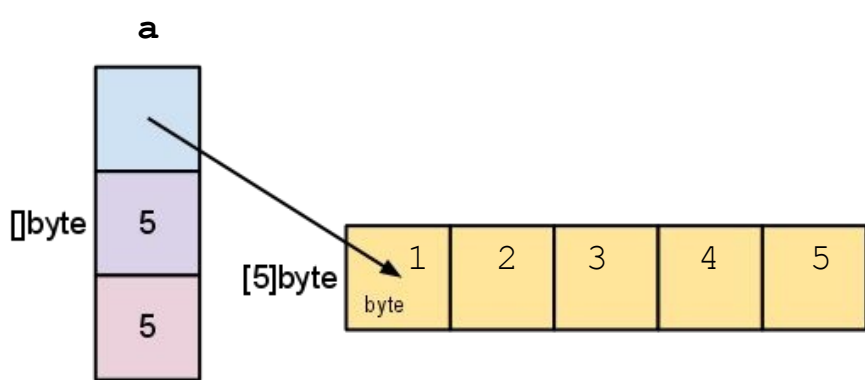
```
fmt.Println(len(b), cap(b), b)      2 3 [3 4]
```

# Типы - слайсы

```
a := []byte{1, 2, 3, 4, 5}
```

```
b := a[2:4] // a[2, 4)
```

```
fmt.Println(len(b), cap(b), b)      2 3 [3 4]
```



# Типы - слайсы

---

```
a := []byte{1, 2, 3, 4, 5}
```

```
a[:] == a[0:5] == a[0:len(a)]
```

```
a[1:] == a[1:5]
```

```
a[:3] == a[0:3]
```

# Типы - слайсы

---

```
a := [...]byte{6, 7, 8}
```

```
b := a[:]
```

```
fmt.Println(len(b), cap(b), b)      3 3 [6 7 8]
```

```
c := a[:2]
```

```
fmt.Println(len(c), cap(c), c)      2 3 [6 7]
```

```
c[1] = 9
```

```
fmt.Println(a)                      [6 9 8]
```



# Типы - слайсы

---

[go.dev/blog/slices-intro](https://go.dev/blog/slices-intro)

[go.dev/blog/slices](https://go.dev/blog/slices)

# Типы - пользовательские

---

# Типы - пользовательские

---

```
var a int  
b := 2 // int
```

```
a += b
```

# Типы - пользовательские

---

```
type A int
```

```
var a int  
b := 2 // int
```

⇒

```
var a A  
b := 2 // int
```

```
a += b
```

```
a += b // ошибка компиляции
```

# Типы - пользовательские

---

```
type НАЗВАНИЕ ОПИСАНИЕ
```

```
type Lol string
```

```
type kek []int
```

```
type Cheburek [][][2][]Lol
```

# Типы - пользовательские

---

```
type A int
```

```
func (a *A) Add(n int) {  
    *a += A(n)  
}
```

```
func main() {  
    var a A  
  
    a.Add(5)  
    println(a)    5  
}
```

# Типы - пользовательские

---

```
type A int
```

```
func (a A) String() string {  
    return fmt.Sprintf("[%d]", a)  
}
```

```
func main() {  
    var a A = 5  
  
    fmt.Println(a)      [5]  
}
```

[tour.golang.org/methods/17](https://tour.golang.org/methods/17)

[pkg.go.dev/fmt@go1.17#Stringer](https://pkg.go.dev/fmt@go1.17#Stringer)

# Типы - пользовательские

---

```
type A int  
type B = A
```

```
func (a A) String() string {  
    return fmt.Sprintf(`[%d]`, a)  
}
```

```
func (b B) Hello() {  
    fmt.Println(b)  
}
```

```
func main() {  
    var b B = 5  
    b.Hello()    [5]  
}
```



# Типы

---

```
type byte = uint8
```

```
type rune = int32
```

[github.com/golang/go/blob/master/src/builtin/builtin.go#L85-L92](https://github.com/golang/go/blob/master/src/builtin/builtin.go#L85-L92)

# Константы

---

# Константы

---

```
var (  
  a = 42  
  b = "hello"  
)
```

⇒

```
const (  
  a = 42  
  b = "hello"  
)
```

# Константы

```
var (  
    a = 42  
    b = "hello"  
)
```

⇒

```
const (  
    a = 42  
    b = "hello"  
)
```

```
var (  
    d int32 = a  
    e float64 = a  
)
```

```
println(d, e)    42 +4.200000e+001
```

# Константы

---

```
const (  
    a = 9223372036854775807000 % 10_000  
    b = 6493247042348742047206323874264455267669785693 % 88888888  
)  
  
println(a, b)      7000 70956925
```

# Константы

---

```
const (  
    a = 42  
    b = int8(42)  
  
)
```

# Константы

---

```
const (  
    a = 42  
    b = int8(42)  
  
    c = int8(9223372036854775807000 % 10_000) // ошибка компиляции overflow  
    d = int16(9223372036854775807000 % 10_000)  
)  
  
var (  
    e.int32 = a  
    f.int32 = b // ошибка компиляции  
)
```

# Константы

---

```
const (  
    a = int8(42)  
    b = int32(100500)  
)
```

```
var (  
    c = a  
    d = b  
)
```

```
fmt.Printf("%T %T", c, d)
```

```
int8 int32
```



# Константы

```
type (  
    PartOfTheDay uint8  
)
```

```
const (  
    Night    = PartOfTheDay(1)  
    Morning  = PartOfTheDay(2)  
    // ...  
)
```

```
fmt.Printf("%T %T\n", Night, Morning)  
fmt.Println(Night, Morning)
```

```
PartOfTheDay PartOfTheDay  
1 2
```

```
var part PartOfTheDay  
part = Morning  
if part == 2 { // ошибка компиляции
```

# Константы - iota

---

```
type (  
    PartOfTheDay uint8  
)  
  
const (  
    Night    = PartOfTheDay(iota)    // iota == 0  
    Morning  // iota == 1  
    Day      // iota == 2  
    Evening  // iota == 3  
)  
  
fmt.Println(Night, Morning, Day, Evening)           0 1 2 3
```

# Константы - iota

---

```
type (  
    PartOfTheDay uint8  
)  
  
const (  
    Night    = PartOfTheDay(iota+1) // iota == 0  
    Morning  // iota == 1  
    Day      // iota == 2  
    Evening  // iota == 3  
)
```

```
fmt.Println(Night, Morning, Day, Evening)
```

1 2 3 4

# Константы - iota

---

```
const (  
    Bit0    = 1 << iota  
    Bit1  
    Bit2  
  
     $\overline{\text{Bit4}}$   
)
```

```
fmt.Println(Bit0, Bit1, Bit2, Bit4)
```

1 2 4 16

# Константы - iota

---

```
const (  
    FieldABit0    = 1 << iota  
    FieldABit1  
)
```

```
const (  
    _             = 1 << iota  
    FieldBBit1  
    FieldBBit2  
)
```

```
fmt.Println(FieldABit0, FieldABit1, FieldBBit1, FieldBBit2) 1 2 2 4
```

# Операторы и приоритеты

---

[golang.org/ref/spec#Arithmetic operators](https://golang.org/ref/spec#Arithmetic_operators)

[golang.org/ref/spec#Operators](https://golang.org/ref/spec#Operators)

[golang.org/ref/spec#Comparison operators](https://golang.org/ref/spec#Comparison_operators)

# Функции

---

# Функции

---

```
package main
```

```
func main() {  
    // ...  
}
```



# Функции

---

```
func НАЗВАНИЕ (ПАРАМЕТРЫ) РЕЗУЛЬТАТ {  
    ТЕЛО  
}
```

# Функции

---

```
package main
```

```
func hello() {  
    println("Hello!")  
}
```

```
func main() {  
    hello()  
}
```

# Функции

---

```
func НАЗВАНИЕ(ПАРАМЕТРЫ) РЕЗУЛЬТАТ {
```

```
func foo( a int, b string, c []float64, d *[3]int ) {
```

# Функции

---

```
func НАЗВАНИЕ(ПАРАМЕТРЫ) РЕЗУЛЬТАТ {
```

```
func bar( vals ...int ) {...}
```

```
bar(1, 2, 3, 4)
```

```
bar([]int{1, 2, 3, 4}...)
```

```
fmt.Printf("hello %s", "world") ⇒ func Printf(msg string, args ...T)
```

# Функции

---

```
func НАЗВАНИЕ (ПАРАМЕТРЫ) РЕЗУЛЬТАТ {
```

```
func spam(a, b int) == func spam(a int, b int)
```

# Функции

---

```
func НАЗВАНИЕ(ПАРАМЕТРЫ) РЕЗУЛЬТАТ {
```

```
func spam(a int, _ string) {  
}
```

# Функции

---

```
func НАЗВАНИЕ (ПАРАМЕТРЫ) РЕЗУЛЬТАТ {
```

```
func sum(a, b int) int {  
    return a + b  
}
```

```
func toString(a int) string {  
    return strconv.Itoa(a)  
}
```

# Функции

---

```
func НАЗВАНИЕ (ПАРАМЕТРЫ) РЕЗУЛЬТАТ {
```

```
func maxFirst(a, b int) (int, int, bool) {  
    flipped := a < b  
  
    if flipped {  
        a, b = b, a  
    }  
  
    return a, b, flipped  
}
```

```
fmt.Println(maxFirst(1, 5))    5 1 true  
fmt.Println(maxFirst(5, 1))    5 1 false  
// func Println(args ...T)
```



# Функции

---

```
func НАЗВАНИЕ (ПАРАМЕТРЫ) РЕЗУЛЬТАТ {
```

```
func maxFirst(a, b int) (max int, min int, flipped bool) {  
    max, min = a, b  
  
    if flipped {  
        a, b = b, a  
    }  
  
    return  
}
```

# Функции

---

```
butler := func(name string) {  
    println("Hello, " + name + "!!")  
}
```

```
spamer := butler
```

```
butler("Alice")  
spamer("Bob")
```

# Функции

---

```
type Spamer func(name string)
```

```
var butler Spamer = func(name string) {  
    println("Hello, " + name + "!!")  
}
```

```
var flooder Spamer = func(name string) bool { // ошибка  
    println("Hello, " + name + "!!")  
    return false  
}
```

# Функции

---

```
type Mod func(s string) string

func upper(s string) string {
    return strings.ToUpper(s)
}

func doubler(s string) string {
    return s + s
}

func applyer(s string, mods ...Mod) string {
    for _, mod := range mods {
        s = mod(s)
    }
    return s
}

println(applyer("Go", upper, doubler))  GOGO
```

# Функции - замыкания

---

```
var a = 5
```

```
func doIt(val int) {  
    a += val  
}
```

```
b := 10
```

```
doIt(b)
```

```
doIt(b)
```

```
println(a)      25
```

# Функции - замыкания

---

```
func suffix(suff string) func(s string) string {  
    return func(s string) string {  
        return s + suff  
    }  
}
```

```
s := suffix(" Heeee!") // s == func(string) string
```

```
println(s("Поработаем?"))    Поработаем? Heeee!  
println(s("Отдохнем?"))      Отдохнем? Heeee!
```

# Функции - CoW

---

```
var a = 5
```

```
func doIt(val int) {  
    a += val  
  
}
```

```
b := 10  
doIt(b)  
doIt(b)  
println(a)    25
```

# Функции - CoW

---

```
var a = 5
```

```
func doIt(val int) {  
    a += val  
    val += a  
}
```

```
b := 10  
doIt(b)  
doIt(b)  
println(a)      25
```



# Функции - CoW

---

```
func doubler(nums [2]int) {  
    nums[0] *= 2  
    nums[1] *= 2  
}  
  
func main() {  
    a := [2]int{1, 2}  
    doubler(a)  
    fmt.Println(a)      [1 2]  
}
```

# Функции - CoW

---

```
func doubler(nums *[2]int) {  
    (*nums)[0] *= 2  
    (*nums)[1] *= 2  
}  
  
func main() {  
    a := [2]int{1, 2}  
    doubler(&a)  
    fmt.Println(a)           [2 4]  
}
```

# Функции - CoW

---

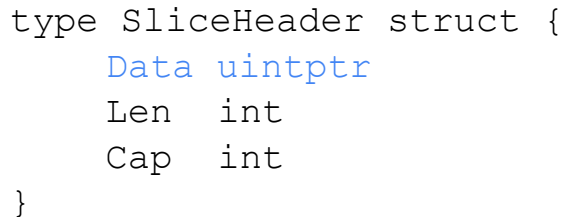
```
func doubler(nums [2]int) [2]int {  
    nums[0] *= 2  
    nums[1] *= 2  
    return nums  
}
```

```
func main() {  
    a := [2]int{1, 2}  
    a = doubler(a)  
    fmt.Println(a)      [2 4]  
}
```

# Функции - CoW

```
func doubler(nums []int) {  
    nums[0] *= 2  
    nums[1] *= 2  
}
```

```
func main() {  
    a := []int{1, 2}  
    doubler(a)  
    fmt.Println(a)    [2 4]  
}
```



```
type SliceHeader struct {  
    Data uintptr  
    Len    int  
    Cap    int  
}
```

# Словари

---

# Словари

---

map [ТИП\_КЛЮЧА] ТИП\_ЗНАЧЕНИЯ

# Словари

---

`map[ТИП_КЛЮЧА] ТИП_ЗНАЧЕНИЯ`

`map[int]string`

# Словари

---

```
map[ТИП_КЛЮЧА]ТИП_ЗНАЧЕНИЯ
```

```
map[int]string
```

```
map[[6]int][]byte
```

```
map[[]int]int // ошибка
```



# Словари

---

```
map[ТИП_КЛЮЧА] ТИП_ЗНАЧЕНИЯ
```

```
map[int]string
```

```
map[[6]int][]byte
```

```
map[[]int]int // ошибка
```

```
type User struct {  
    FirstName string  
    LastName  string  
}  
map[User]struct{}
```

# Словари

```
map[ map[int]int ]int // ошибка
```



# Словари

---

```
m := map[int]int  
m[1] = 1 // runtime ошибка
```

```
var m map[int]int  
m[1] = 1 // runtime ошибка
```

# Словари

---

```
m := map[int]int
m[1] = 1 // runtime ошибка
```

```
var m map[int]int
m[1] = 1 // runtime ошибка
```

```
m := map[int]int{}
m[1] = 1
```

```
m := make(map[int]int)
m[1] = 1
```

```
var m map[int]int
m = make(map[int]int)
m[1] = 1
```

```
type MyMap map[int]int
var m MyMap
m = make(MyMap)
m[1] = 1
```

# Словари

---

```
m := map[int]int{}  
println(len(m))
```

0

```
m := make(map[int]int)  
println(len(m))
```

0

```
m := make(map[int]int, 100)  
println(len(m))
```

0

# Словари

---

```
var m map[int]int
```

```
// Добавление
```

```
m[1] = 2
```

```
// Удаление
```

```
delete(m, 1)
```

```
// Проверки наличия
```

```
val := m[1]           // Если нет - zero value
```

```
val, ok := m[1]       // ok == exists
```

# Словари

---

```
m := map[int]string{
```

```
    42: `world`,
```

```
}
```

```
m[1] = `hello`
```

```
val := m[1]
```

```
val += ` world`
```

```
println(m[1])
```

```
hello
```

# Словари

---

```
type T struct {  
    Count int  
}
```

```
m := map[string]T{}
```

```
const key = `kek`
```

```
m[key] = T{Count: 1}
```

```
m[key].Count++ // ошибка компиляции
```

```
t := m[key]
```

```
t.Count++
```

```
fmt.Printf("%v\n", m[key])    {Count:1}
```



# Словари

---

```
type T struct {  
    Count int  
}  
  
m := map[string]T{}  
  
const key = `kek`  
  
m[key] = T{Count: 1}  
  
// m[key].Count++  
  
t := m[key]  
t.Count++  
m[key] = t  
  
fmt.Printf("%+v\n", m[key])    {Count:2}
```

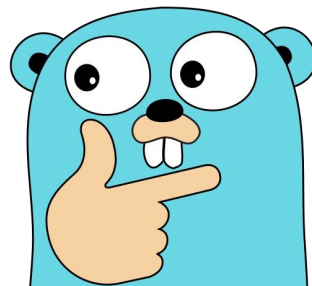
# Словари

---

```
type T struct {  
    Count int  
}  
  
m := map[string]*T{}  
  
const key = `kek`  
  
m[key] = &T{Count: 1}  
  
m[key].Count++ // может быть runtime ошибка  
  
fmt.Printf("%+v\n", m[key])    &{Count:2}
```

# Словари

```
type T struct {  
    Count int  
}  
  
m := map[string]*T{}  
  
const key = `kek`  
  
m[key] = &T{Count: 1}  
  
t, ok := m[key]  
  
if ok {  
    t.Count++  
}  
  
fmt.Printf("%+v\n", m[key])    &{Count:2}
```



# Словари

---

```
type T struct {  
    Count int  
}  
  
m := map[string]*T{}  
  
const key = `kek`  
  
m[key] = nil  
  
t, ok := m[key]  
if ok {  
    t.Count++ // runtime ошибка  
}  
  
fmt.Printf("%+v\n", m[key])
```

# Словари

```
type T struct {  
    Count int  
}  
  
m := map[string]*T{}  
  
const key = `kek`  
  
m[key] = nil  
  
t, ok := m[key]  
if ok {  
    t.Count++ // runtime ошибка  
}  
  
fmt.Printf("%+v\n", m[key])
```

```
type T struct {  
    Count int  
}  
  
m := map[string]*T{}  
  
const key = `kek`  
  
m[key] = nil  
  
t := m[key]  
if t != nil {  
    t.Count++  
}  
  
fmt.Printf("%+v\n", m[key])    <nil>
```

# Структура кода

# Структура кода

---

- Ветвления
- Циклы

# Ветвления

---



# Ветвления

---

```
if УСЛОВИЕ {  
    // ТЕЛО  
}
```

# Ветвления

---

```
count := getCount()  
if count == 0 {
```

```
    // ТЕЛО
```

```
}
```

```
// count можно использовать
```

```
if (count := getCount()) == 0 { // ошибка компиляции
```

```
    // ТЕЛО
```

```
}
```

# Ветвления

---

```
count := getCount()  
if count == 0 {
```

```
    // ТЕЛО
```

```
}
```

```
// count можно использовать
```

```
if count := getCount() ; count == 0 {
```

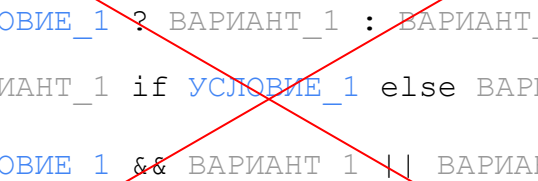
```
    // ТЕЛО
```

```
}
```

```
// count недоступна
```

# Ветвления

```
if УСЛОВИЕ_1 {  
  
    // ВАРИАНТ_1  
  
} else if УСЛОВИЕ_2 {  
  
    // ВАРИАНТ_2  
  
} else {  
  
    // ВАРИАНТ_3  
  
}
```



```
УСЛОВИЕ_1 ? ВАРИАНТ_1 : ВАРИАНТ_2  
ВАРИАНТ_1 if УСЛОВИЕ_1 else ВАРИАНТ_2  
УСЛОВИЕ_1 && ВАРИАНТ_1 || ВАРИАНТ_2
```

# Ветвления

---

```
if УСЛОВИЕ_1 {  
    // ВАРИАНТ_1  
} else if УСЛОВИЕ_2 {  
    // ВАРИАНТ_2  
} else {  
    // ВАРИАНТ_3  
}
```



# Ветвления

```
if УСЛОВИЕ_1 {  
    // ВАРИАНТ_1  
}  
else if УСЛОВИЕ_2 {  
    // ВАРИАНТ_2  
}  
else {  
    // ВАРИАНТ_3  
}
```



```
switch {  
case УСЛОВИЕ_1:  
    // ВАРИАНТ_1  
case УСЛОВИЕ_2:  
    // ВАРИАНТ_2  
default:  
    // ВАРИАНТ_3  
}
```



# Ветвления

---

```
switch [ПРОВЕРЯЕМОЕ_ЗНАЧЕНИЕ] {  
  
case ЗНАЧЕНИЕ_1, ЗНАЧЕНИЕ_2:  
  
    // ВАРИАНТ_1  
  
    fallthrough  
  
case ЗНАЧЕНИЕ_3:  
  
    // ВАРИАНТ_2  
  
default:  
  
    // ВАРИАНТ_3  
  
}
```

# Ветвления

---

```
switch [ПРОВЕРЯЕМОЕ_ЗНАЧЕНИЕ] {  
  
  case ЗНАЧЕНИЕ_1, ЗНАЧЕНИЕ_2:  
  
    // ВАРИАНТ_1  
  
    fallthrough  
  
  case ЗНАЧЕНИЕ_3:  
  
    // ВАРИАНТ_2  
  
  default:  
  
    // ВАРИАНТ_3  
  
}
```



```
case ЗНАЧЕНИЕ_1 .. ЗНАЧЕНИЕ_2:
```



# Ветвления

---

```
switch {  
  case i%3 == 0 && i%5 == 0:  
    println("FizzBuzz")  
  
  case i%3 == 0:  
    println("Fizz")  
  
  case i%5 == 0:  
    println("Buzz")  
  
  default:  
    println(i)  
}
```

# Циклы

---

# Циклы

---

```
for ИНИЦИАЛИЗАЦИЯ ; ПРОВЕРКА ; МОДИФИКАЦИЯ {  
    // ТЕЛО  
}
```

```
for КЛЮЧ, ЗНАЧЕНИЕ := range МАССИВ_СЛОВАРЬ_СТРОКА {  
    // ТЕЛО  
}
```

```
for КЛЮЧ := range МАССИВ_СЛОВАРЬ_СТРОКА {  
    // ТЕЛО  
}
```

```
for {  
    // ТЕЛО  
}
```

# Циклы

---

```
for ИНИЦИАЛИЗАЦИЯ ; ПРОВЕРКА ; МОДИФИКАЦИЯ {  
    // ТЕЛО  
}
```

```
for i, j := 0, 0; i < 10; i, j = i+1, 1 {  
    println(i, j)  
}
```

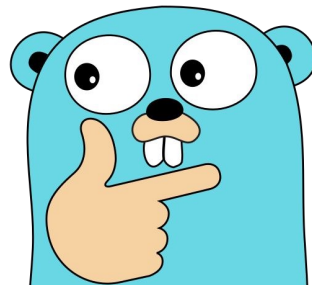
# Циклы

```
for КЛЮЧ, ЗНАЧЕНИЕ := range МАССИВ_СЛОВАРЬ_СТРОКА {  
    // ТЕЛО  
}
```

```
src := []int{1, 2, 3}  
dst := make([]*int, len(src))
```

```
for i, val := range src {  
    dst[i] = &val  
}
```

```
for _, val := range dst {  
    fmt.Println(*val)  
}
```



# Циклы

```
for КЛЮЧ, ЗНАЧЕНИЕ := range МАССИВ_СЛОВАРЬ_СТРОКА {  
    // ТЕЛО  
}
```

```
src := []int{1, 2, 3}  
dst := make([]*int, len(src))
```

```
for i, val := range src {  
    dst[i] = &val  
}
```

```
for _, val := range dst {  
    fmt.Println(*val)  
}
```

3

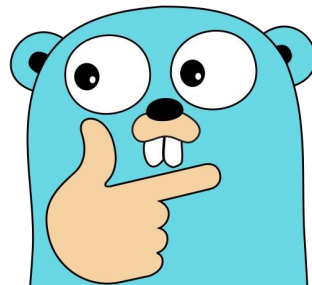
3

3

# Циклы

```
for КЛЮЧ := range МАССИВ {  
    // ТЕЛО  
}
```

```
func main() {  
    nums := []int{1, 2, 3}  
  
    for i := range nums {  
        nums = append(nums, i*2)  
    }  
  
    fmt.Println(nums)  
}
```



# Циклы

---

```
for КЛЮЧ := range МАССИВ {  
    // ТЕЛО  
}
```

```
func main() {  
    nums := []int{1, 2, 3}  
  
    for i := range nums {  
        nums = append(nums, i*2)  
    }  
  
    fmt.Println(nums)      [1 2 3 0 2 4]  
}
```



# Циклы

---

```
for {  
    // ТЕЛО  
}
```

```
loop:  
    for {  
        for {  
            if false {  
                continue  
            } else {  
                break loop  
            }  
        }  
    }  
}
```

# Самое важное

---

# Самое важное

---

- Всё передаётся по значению

# Самое важное

---

- Всё передаётся по значению
- Но без DeepCopy

# Самое важное

---

- Всё передаётся по значению
- Но без DeepCopy
- Используйте линтеры

Домашнее задание

# Вот и всё

