



Основы

Tinkoff.ru





Сергей Лутошин

Backend разработка

В инвестициях с момента запуска

Отвечаю за многие нагруженные сервисы в
этом направлении



План занятия

1. Немного истории
2. Как golang появился в tinkoff.ru/invest
3. Синтаксические конструкции и типы данных
4. Слайсы
5. Мапы



Немного истории



A long time ago in a Google
far far away ...

Краткая история языка. Кто и когда?



Роберт Гризмер, Роб Пайк, Кен Томпсон



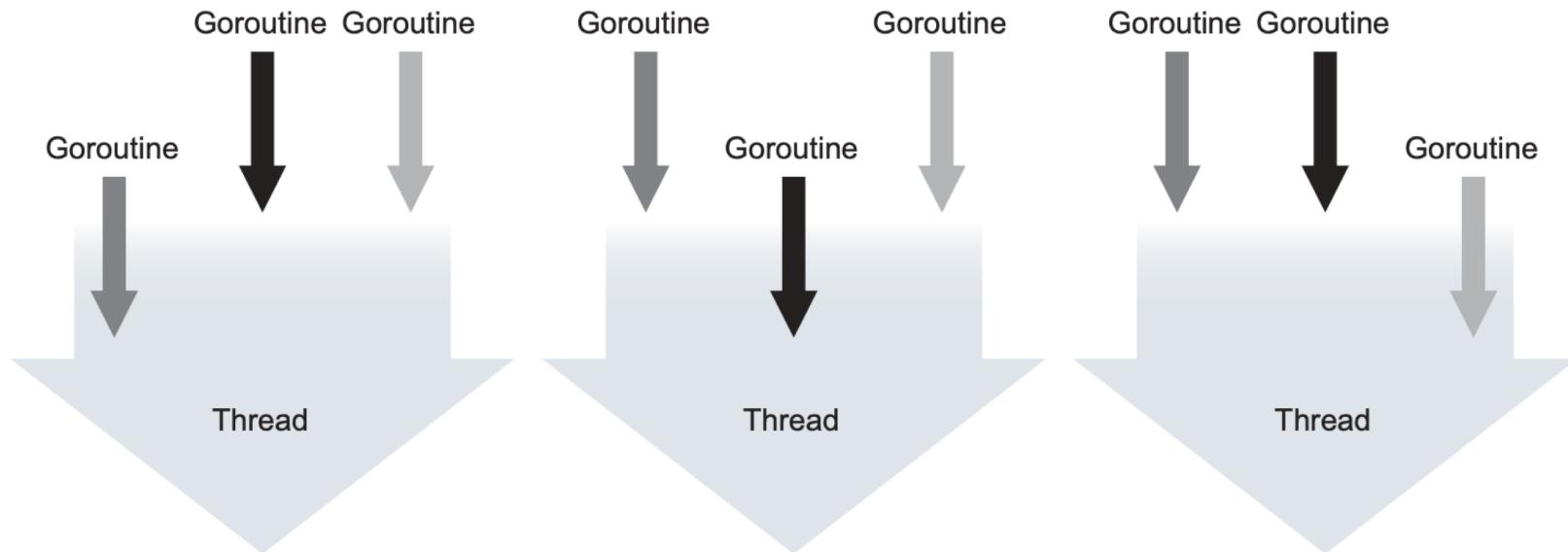
Краткая история языка. Зачем?



- Скорость разработки
- Скорость компиляции
- Асинхронная M:N модель
- Горутины, каналы, пайплайны
- Сборщик мусора

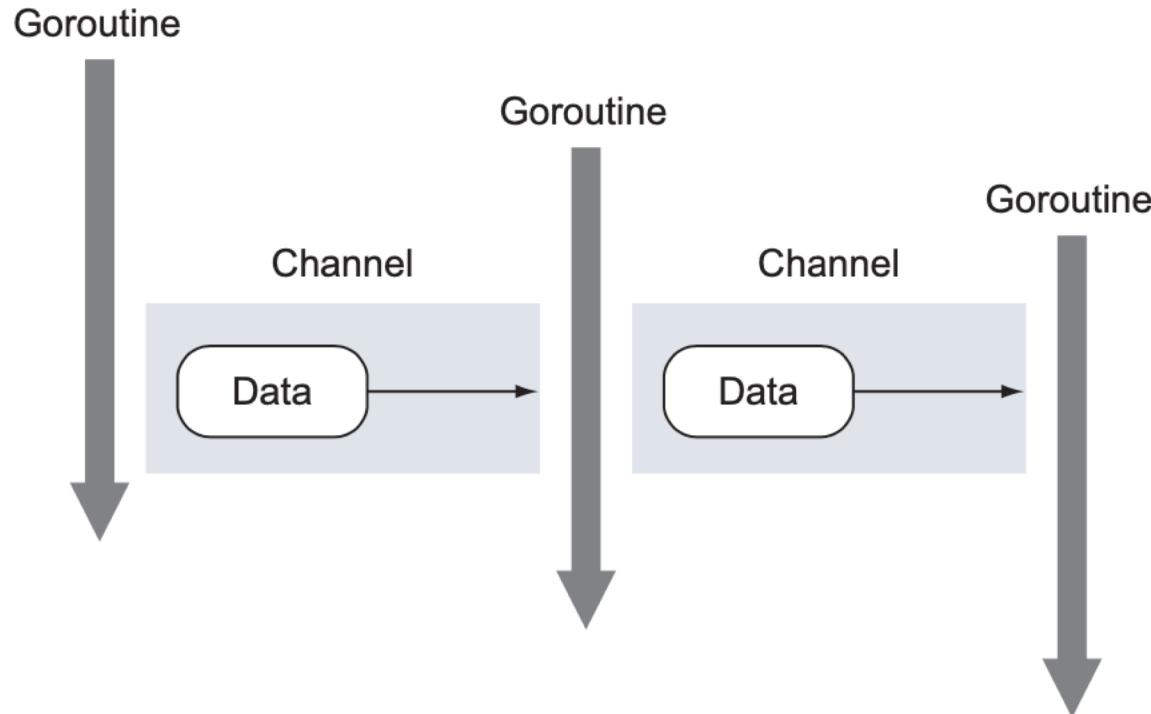


Асинхронная M:N модель





Горутины, пайплайны, каналы





Golang в tinkoff.ru/invest



Тинькофф Инвестиции





Рыночные данные

TCS Group (Tinkoff Bank holder) TCS ☆

Акции Есть идеи Финансовый сектор

Последняя сделка

20,15 \$

За день

▲ 0,41 \$ 2,08%

За день

Неделю

Месяц

Полгода

Год

Всё время

▲ 3,88 \$ 19,26%



07 января 19

16 января 19

28 января 19



Показатели по бумагам

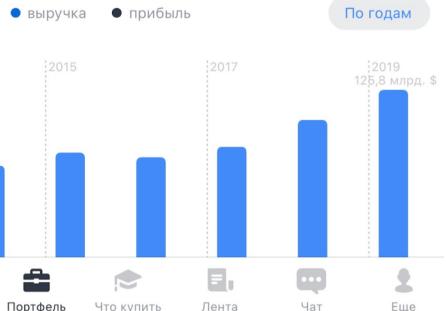
Microsoft Corporation
MSFT

Стакан Пульс Показатели Прогнозы Новости

Основные показатели

Цена открытия	185 \$
Цена закрытия	183,89 \$
Дневной диапазон	181,4 - 187,2 \$
Годовой диапазон (52w)	105 - 187,2 \$
Дивидендная доходность	1,11 %
Цена/Прибыль (P/E)	34,01704

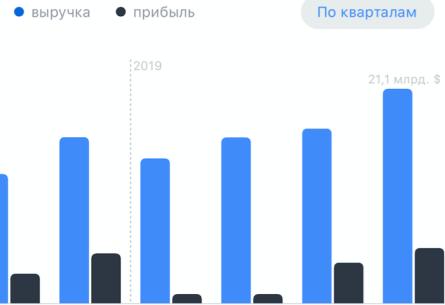
Финансовые показатели



Facebook
FB

Стакан Пульс Показатели Прогнозы Новости

Финансовые показатели



Денежные средства





Лента новостей и идей

Все

Идеи

Новости



Google

Россия стала лидером по числу запросов госорганов к Google об удалении контента

В первом полугодии 2018 года от российских госорганов к Google поступило 19,2 тыс. запросов на удаление контента, что составляет больше 70% от всех мировых запросов, следует из ежегодного отчета компании.

Вчера в 19:37 · RNS

Избранное



TCS Group (Tink... 20,15 \$ ▲ 2,08%



Стриминговая доставка котировок

Инструменты														Стакан: AAPL			Стакан: DHR			Стакан: SBER		
Symbol	Bid	Ask	Last	Trades	Volume	Turnvo	Last Vc	Bid Am	Ask Ar	Bid Qty	Ask Qt	Chg, %	MA	Bid, \$	Объём	Ask, \$	Bid, \$	Объём	Ask, \$	Bid, \$	Объём	Ask, \$
														0,62 (0,194%)		0,46 (0,283%)		0,04 (0,016%)		0,04 (0,016%)		0,04 (0,016%)
AAPL	\$ 319,91	320,55	319,91	146	1,12k	357,77	3	50	43	1,21k	1,30k	-0,04 %	S/L	319,90	60 99	320,53	162,31	50 106	162,77	254,56	117 794	254,60
AFLT	₽ 120,68	120,72	120,72	2,00k	119,17k	143,67	20	320	173	106,24k	75,02k	0,92 %	S/L	319,90	44 1	320,54	162,21	56 85	162,87	254,55	671 130	254,61
AMZN	\$ 2078,1	2084,8	2078,1	14	41	85,12k	2	50	33	168	147	-0,05 %	S/L	319,85	110 116	321,00	162,20	85 275	164,07	254,54	250 50	254,63
BANE	₽ 2017,5	2019,5	2017,0	38	377	761,75k	2	144	125	6,72k	2,91k	0,22 %	S/L	319,00	11 36	322,00	161,00	275 275	164,45	254,53	135 1	254,64
FB	\$ 212,66	212,82	212,82	33	419	89,40k	5	46	41	1,11k	1,83k	0,23 %	S/L	318,00	63 100	323,06	160,62	275 6	166,80	254,52	250 678	254,65
FXMM	₽ 1592,0	1594,0	1592,0	19	129	205,44	1	83	24	5,41k	3,79k	-0,03 %	-	317,91	152 1	323,23	155,45	1		254,51	1106 1	254,66
GAZP	₽ 227,98	228,03	228,01	1,46k	91,43k	208,88	200	670	817	189,85k	281,35k	-0,48 %	S/L	317,90	2 1	323,64	254,50	1888 145		254,46	250 12	254,67
GOOG	\$ 1481,76	1482,6	1487,17	1	1	1,49k	1	17	17	163	261	0,54 %	S/L	317,50	2 53	323,80	254,45	138 120		254,45	138 120	254,70
MSFT	\$ 184,56	184,83	184,83	48	366	67,59k	1	50	21	1,85k	1,43k	0,51 %	S/L	317,20	21 30	324,00	254,44	98 510		254,44	98 510	254,71
NFLX	\$ 366,57	367,19	367,30	4	7	2,57k	3	35	27	609	631	0,14 %	S/L	316,91	100 35	324,80	254,43	298 201		254,43	298 201	254,72
Кардаде вы...Р	₽ 171,63	172,36	0,000	0	0	0	0	10	15	560	21,70k	0,00 %	-	316,50	5 13	325,00	254,42	273 200		254,42	273 200	254,73
ОФЗ 26223	₽ 254,56	254,60	254,56	1,69k	105,90	269,92	1	564	542	191,21k	224,61k	0,10 %	S/L	316,00	24 100	325,06	254,41	555 200		254,41	555 200	254,74
ОФЗ 26223	₽ 1030,8	1031,95	1030,8	6	47	48,49k	1	25	14	62,88k	57,74k	0,00 %	L	315,91	100 2	325,11	254,40	210 364		254,40	210 364	254,75
SBER	₽ 254,56	254,60	254,56	1,69k	105,90	269,92	1	564	542	191,21k	224,61k	0,10 %	S/L	315,10	1 193	325,41	254,39	200 392		254,39	200 392	254,76
SU29011RMF...														315,03	2 1	325,64	254,38	200 418		254,38	200 418	254,77
TWTR	\$ 36,79	36,89	36,74	18	262	9,65k	34	50	50	5,76k	9,76k	-0,78 %	S/L	315,00	42 1	325,68	254,37	200 2350		254,37	200 2350	254,78
														314,97	193 1	325,75	254,36	611 298		254,36	611 298	254,79
														314,80	2 2	326,00	254,35	2 923		254,35	2 923	254,80



Синтаксис и типы данных



Go – язык программирования со статической типизацией.

uint8	unsigned 8-bit integers (0 to 255)
uint16	unsigned 16-bit integers (0 to 65535)
uint32	unsigned 32-bit integers (0 to 4294967295)
uint64	unsigned 64-bit integers (0 to 18446744073709551615)
int8	signed 8-bit integers (-128 to 127)
int16	signed 16-bit integers (-32768 to 32767)
int32	signed 32-bit integers (-2147483648 to 2147483647)
int64	signed 64-bit integers (-9223372036854775808 to 9223372036854775807)
float32	IEEE-754 32-bit floating-point numbers
float64	IEEE-754 64-bit floating-point numbers
complex64	floating point numbers with float32 real and imaginary parts
complex128	floating point numbers with float64 real and imaginary parts
byte	alias uint8
rune	alias for int32
uint	either 32 or 64 bits
int	same size as uint
uintptr	an unsigned integer large enough to store the uninterpreted bits of a pointer value

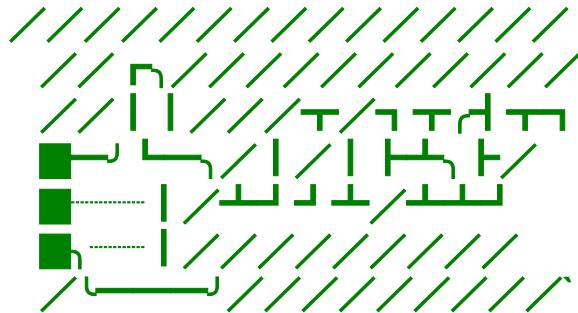
64 бита на 64-х битной платформе, 32 бита на 32-х битной платформе



Тип данных string

- Строки не мутабильны
- Они всегда в UTF-8 по умолчанию
- Существует 2 способа инициал. строку(raw & interpret literal)

a := `Я строка, в которой не нужно экранировать спец. символы`



<<< RAW string literals

a := "Скажи \"Привет\" финтеху"

<<< interpret string literals



Тип данных string

```
1 s := "Hello, Сергей!"  
2 for i, c := range s {  
3     fmt.Printf("%d: %s\n", i, string(c))  
4 }  
5 fmt.Printf("length of '%s': %d", s, len(s))
```

```
0: H  
1: e  
2: l  
3: l  
4: o  
5: ,  
6:  
7: C  
9: e  
11: р  
13: г  
15: е  
17: ю  
19: !  
  
length of 'Hello, Сергей!': 20
```

Rune это особенный тип!

Rune alias int32

Может занимать до 3-х байт



Приведем string к []rune

```
1 s := "Hello, Сергей!"
2 sb := []byte(s)
3 sr := []rune(s)
4
5 j := 1
6 for _, r := range sr {
7     fmt.Printf("%d: %s\n", j, string(r))
8     j++
9 }
10 fmt.Printf("length of '%s': %d(string), %d([]byte), %d([]rune)", s, len(s),
len(sb), len(sr))
```

```
1: H
2: e
...
13: ў
14: !
length of 'Hello, Сергей!': 20(string), 20([]byte), 14([]rune)
```



Как инициализировать переменную

Without initialization

```
var i int
```

With initialization, explicit type

```
var i int = 1  
j := int64(1)
```

With initialization, implicit type

```
i := 1
```



Что такое zero value?

```
1 var a int
2 var b string
3 var c float64
4 var d bool
5
6 fmt.Printf("var a %T = %v\n", a, a)
7 fmt.Printf("var b %T = %q\n", b, b)
8 fmt.Printf("var c %T = %v\n", c, c)
9 fmt.Printf("var d %T = %v\n", d, d)
```

```
var a int = 0
var b string = ""
var c float64 = 0
var d bool = false
```



Константы

```
1 const (
2     a = "A"
3     b = 'A'
4     c = 2
5     d = 3.141592
6     e = int64(7)
7 )
8 func main() {
9     fmt.Printf("a is type %T with value %q\n", a, a)
10    fmt.Printf("b is type %T with value %q\n", b, b)
11    fmt.Printf("c is type %T with value %v\n", c, c)
12    fmt.Printf("d is type %T with value %v\n", d, d)
13    fmt.Printf("e is type %T with value %v\n", e, e)
14 }
```

```
a is type string with value "A"
b is type int32 with value 'A'
c is type int with value 2
d is type float64 with value 3.141592
e is type int64 with value 7
```



fmt.Printf

```
1 // Use %v to print the value
2 // Use %s to print string
3 // Use %q to print quote string
4 // Use %d to print a decimal
5 // Use %T to print a type
6 // Use \ to escape the character
7 // Use \n to print a new line
8 // Use %+v to print name and value (later for structs)
```



iota

```
1 const (
2     _ int = iota
3     Red
4     Blue
5     Green
6 )
7
8 func main() {
9     fmt.Printf("The value of Red is %v\n", Red)
10    fmt.Printf("The value of Blue is %v\n", Blue)
11    fmt.Printf("The value of Green is %v\n", Green)
12 }
```

```
The value of Red is 1
The value of Blue is 2
The value of Green is 3
```



```
1 const (
2     Read = 1 << iota // 00000001 = 1
3     Write          // 00000010 = 2
4     Remove         // 00000100 = 4
5     // admin have all permissions
6     Admin = Read | Write | Remove
7 )
8 func main() {
9     fmt.Printf("The value of Read is %v\n", Read)
10    fmt.Printf("The value of Write is %v\n", Write)
11    fmt.Printf("The value of Remove is %v\n", Remove)
12    fmt.Printf("The value of Admin is %v\n", Admin)
13 }
```

```
The value of Read is 1
The value of Write is 2
The value of Remove is 4
The value of Admin is 7
```



Обычный цикл for

```
1 fmt.Printf("Обычный цикл for: ")
2 for i := 0; i < 2; i++ {
3     fmt.Printf("%d ", i)
4 }
5 fmt.Println()
```

Обычный цикл for: 0 1



Бесконечный цикл for

```
1 var counter int
2 fmt.Printf("Бесконечный цикл: ") // Выход по return либо break
3 for {
4     fmt.Printf("%d ", counter)
5     if counter == 2 {
6         break
7     }
8     counter++
9 }
```

Бесконечный цикл: 0 1 2



Цикл range по слайсу

```
1 fmt.Println("Range по слайсу: ")
2 slice := []int{0, 1, 2, 3}
3 for i, value := range slice {
4     // value - это переменная цикла, значение которой совпадает с элементом i-ой итерации
5     fmt.Printf("\tslice[%d]=%v, &value=%p, &slice[i]=%p\n", i, value, &value, &slice[i])
6 }
```

Range по слайсу:

```
slice[0]=0, &value=0xc000098028, &slice[i]=0xc00009a020
slice[1]=1, &value=0xc000098028, &slice[i]=0xc00009a028
slice[2]=2, &value=0xc000098028, &slice[i]=0xc00009a030
slice[3]=3, &value=0xc000098028, &slice[i]=0xc00009a038
```



Switch

```
1 fmt.Println("Go runs on ")
2 switch os := runtime.GOOS; os {
3 case "darwin":
4     fmt.Println("OS X.")
5 case "linux":
6     fmt.Println("Linux.")
7 default:
8     // freebsd, openbsd,
9     // plan9, windows...
10    fmt.Sprintf("%s.", os)
11 }
12 }
```

Go runs on OS X.



Слайсы



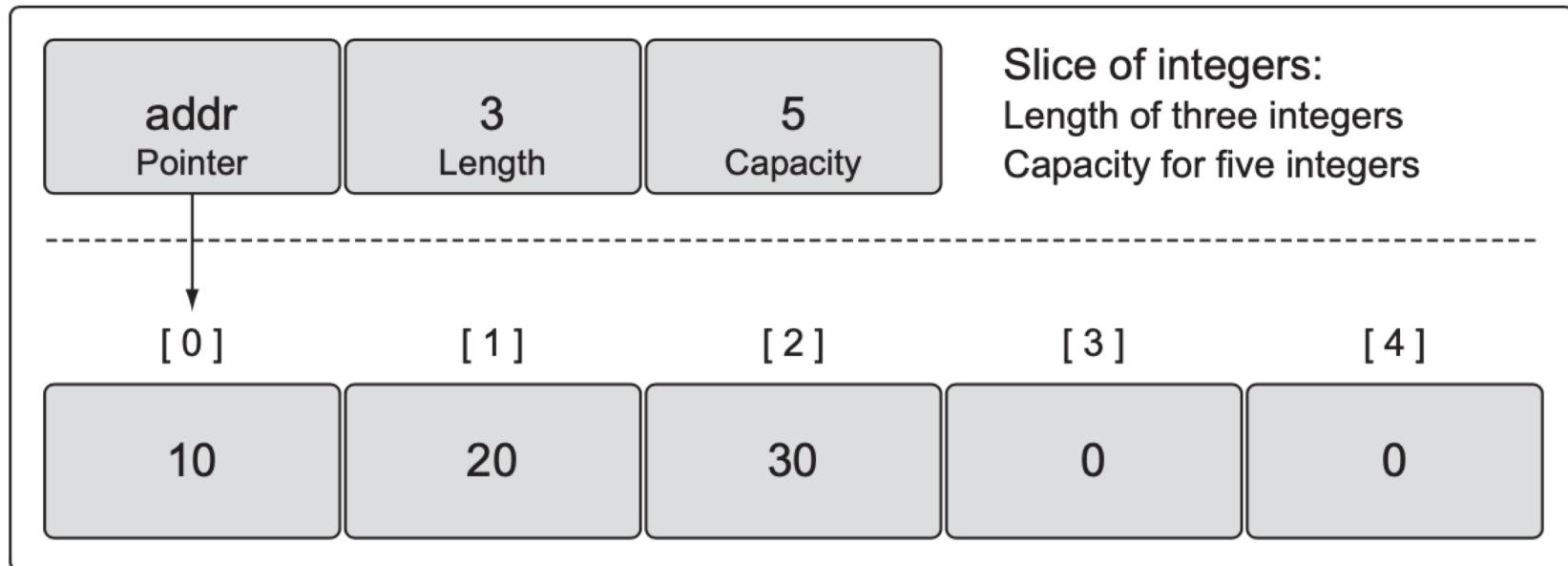
Почему слайсы?

- Легко менять размер (есть встроенная функция append)
- Возможность работать с подмножеством слайса
- Он представлен непрерывным участком в памяти
- Слайс как и массив всегда сохраняет последовательность

```
var array [5]string  
var slice []string
```



Внутренне устройство слайса





Добавляем элементы в слайс

```
1 var stocks []string
2 stocks = append(stocks, "AAPL") // appends one element
3 stocks = append(stocks, "FB", "TCS") // appends variadic elements
4
5 moreStocks := []string{"YNDX", "MSFT"}
6 stocks = append(stocks, moreStocks...) // appends other slice
7
8 fmt.Println("stocks: ", stocks)
```

```
stocks: [AAPL FB TCS YNDX MSFT]
```



Добавляем элементы в слайс

```
1 var stocks []string
2 fmt.Printf("len=%d, cap=%d\n", len(stocks), cap(stocks))
3 stocks = append(stocks, "AAPL") // appends one element
4 fmt.Printf("len=%d, cap=%d\n", len(stocks), cap(stocks))
5 stocks = append(stocks, "FB", "TCS") // appends variadic elements
6 fmt.Printf("len=%d, cap=%d\n", len(stocks), cap(stocks))
7 moreStocks := []string{"YNDX", "MSFT"}
8 stocks = append(stocks, moreStocks...) // appends other slice
9 fmt.Printf("len=%d, cap=%d\n", len(stocks), cap(stocks))
10 fmt.Println("stocks: ", stocks)
```

```
len=0, cap=0
len=1, cap=1
len=3, cap=3
len=5, cap=6
stocks: [AAPL FB TCS YNDX MSFT]
```



ФУНКЦИЯ make

```
1 otherStocks := []string{"YNDX", "MSFT", "FB", "TCS"}  
2 stocks := make([]string, 0, len(otherStocks))  
3 fmt.Printf("len=%d, cap=%d\n", len(stocks), cap(stocks))  
4 for _, stock := range otherStocks {  
5     stocks = append(stocks, stock[:1])  
6     fmt.Printf("len=%d, cap=%d\n", len(stocks), cap(stocks))  
7 }  
8  
9 fmt.Println("stocks: ", stocks)
```

```
len=0, cap=4  
len=1, cap=4  
len=2, cap=4  
len=3, cap=4  
len=4, cap=4  
stocks: [Y M F T]
```



Типичная ошибка make+append

```
1 otherStocks := []string{"YNDX", "MSFT", "FB", "TCS"}
2 stocks := make([]int, len(otherStocks), len(otherStocks))
3 fmt.Printf("len=%d, cap=%d\n", len(stocks), cap(stocks))
4 for _, stock := range otherStocks {
5     stocks = append(stocks, int(stock[0]))
6     fmt.Printf("len=%d, cap=%d\n", len(stocks), cap(stocks))
7 }
8
9 fmt.Println("stocks: ", stocks)
```

```
len=4, cap=4
len=5, cap=8
len=6, cap=8
len=7, cap=8
len=8, cap=8
stocks: [0 0 0 0 89 77 70 84]
```



Подмножества слайса

```
1 stocks := []string{"YNDX", "MSFT", "FB", "TCS"}  
2  
3 fmt.Println(stocks[1:3])  
4  
5 fmt.Println(stocks[2:len(stocks)])  
6 fmt.Println(stocks[2:])  
7  
8 fmt.Println(stocks[0:3])  
9 fmt.Println(stocks[:3])
```

```
[MSFT FB]  
[FB TCS]  
[FB TCS]  
[YNDX MSFT FB]  
[YNDX MSFT FB]
```



ФУНКЦИЯ COPY

```
1 original := []string{ "YNDX" , "MSFT" , "FB" , "TCS" }
2
3 ref := original
4
5 cpy := make([]string, len(original))
6
7 copy(cpy, original)
8 ref[0] = "_Y_"
9 original[1] = "_M_"
10
11 fmt.Println("original: ", original)
12 fmt.Println("ref: ", ref)
13 fmt.Println("cpy: ", cpy)
```

```
original:  [_Y_ _M_ FB TCS]
ref:    [_Y_ _M_ FB TCS]
cpy:    [YNDX MSFT FB TCS]
```



Slice == nil

```
1 var stocks []string
2 if stocks == nil {
3     fmt.Println("slice is nil")
4 }
```

```
slice is nil
```

nil
Pointer

0
Length

0
Capacity

nil slice:
Length of zero
Capacity of zero

```
var slice []int
```



Мапы



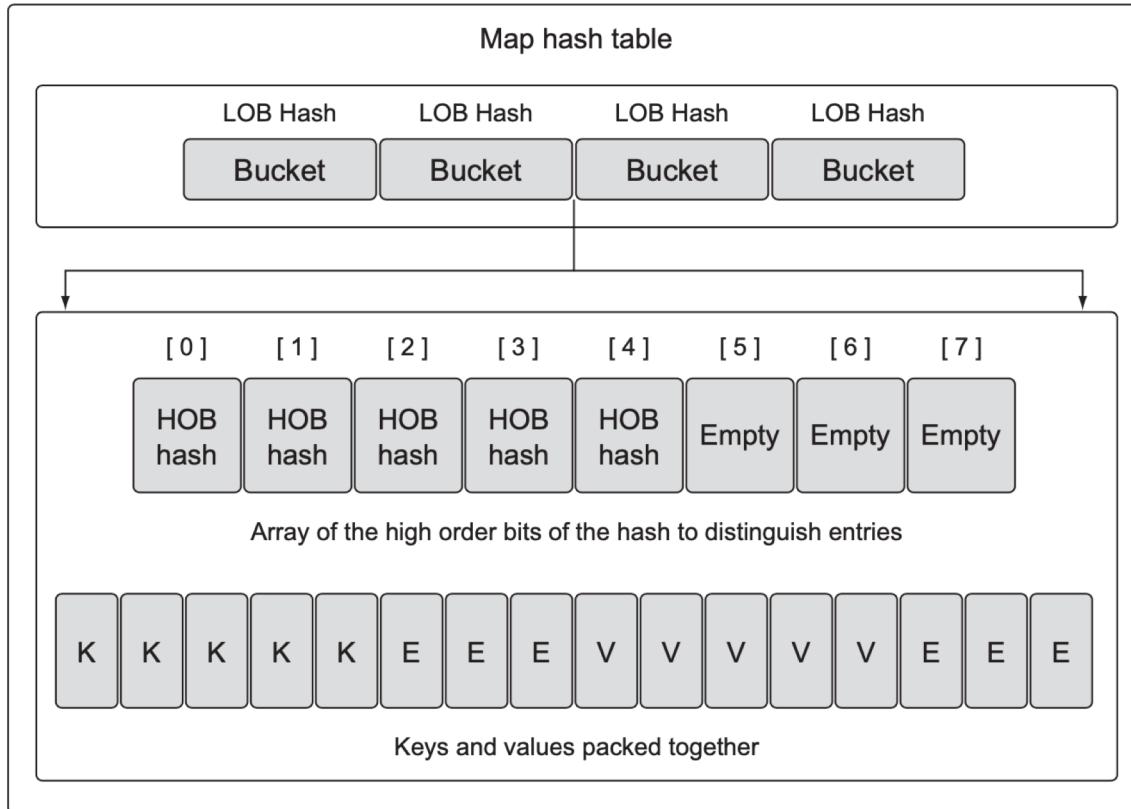
Когда мапы?

- Когда последовательность элементов не важна
- Когда известен ключ, по которому нужно достать значение

```
yield := make(map[string]string)  
  
yield := map[string]int{  
    "YNDX": 1000,  
    "TCS": 10000,  
    "FB": 1000000,  
}
```



Что такое мапа в go?



K = Key

V = Value

E = Empty



Добавить/прочитать/удалить ключ

```
1 yield := make(map[string]int)
2 yield[ "YNDX" ] = 1000
3 yield[ "TCS" ] = 10000
4 fmt.Println( "TCS yield is", yield[ "TCS" ] )
5 delete(yield, "YNDX")
6 fmt.Println(yield)
```

```
TCS yield is 10000
map[TCS:10000]
```



Проверка существования ключа

```
1 stocks := []string{"YNDX", "FB", "TCS"}
2 yield := map[string]int{
3     "YNDX": 1000,
4     "TCS": 10000,
5 }
6 for _, stock := range stocks {
7     y, ok := yield[stock]
8     if !ok {
9         continue
10    }
11    fmt.Println("yield for", stock, "is", y)
12 }
```

```
yield for YNDX is 1000
yield for TCS is 10000
```



for-range для мапы

```
1 yield := map[string]int{
2     "YNDX": 1000,
3     "TCS":   10000,
4 }
5 for stock, yield := range yield {
6     fmt.Println("yield for", stock, "is", yield)
7 }
```

```
yield for YNDX is 1000
yield for TCS is 10000
```



Типичная ошибка при работе с мапами

```
1 var yield map[string]int
2 yield[ "YNDX" ] = 1000
```

```
panic: assignment to entry in nil map

goroutine 1 [running]:
main.main()
```

PS: как мы делимся знаниями?



1. Реализуем гипотезу в play.golang.org
2. Нажимаем кнопку [share] и, получив ссылку, шарим ее товарищам в телеграм-канал
3. Расписываем свою идею, например какой был ожидаемый результат и какой получили фактически
4. Ждем фидбека от коллег и кураторов



Обратная связь

Tinkoff.ru



Спасибо за внимание

Tinkoff.ru