Покоряем golang

Лекция 2. Основы синтаксиса

Григорий Базилевич

27 ноября 2021 года

Переменная и ее объявление

```
package main

import "fmt"

func main() {
    var a int
    a = 10
```

Переменная и ее объявление

```
package main

import "fmt"

func main() {
    var a int
    a = 10
    var b = 20
```

Переменная и ее объявление

```
package main

import "fmt"

func main() {
    var a int
    a = 10
    var b = 20
    c := a + b
    fmt.Println(a, b, c)
```

Переменная и ее объявление

```
package main

import "fmt"

func main() {
    var a int
    a = 10
    var b = 20
    c := a + b
    fmt.Println(a, b, c)

    var d, e = 1, "ok!"
    cpp, py, js := 23, "Py3.10", false

    fmt.Println(py, e)
}
```

Примитивы

- int, int8, int16, int32, int64
- uint, uint8, uint16, uint32, uint64, uintptr

Примитивы

- int, int8, int16, int32, int64
- uint, uint8, uint16, uint32, uint64, uintptr
- byte (= int8)
- bool

Примитивы

- int, int8, int16, int32, int64
- uint, uint8, uint16, uint32, uint64, uintptr
- byte (= int8)
- bool
- float32, float64
- complex64, complex128

Примитивы

- int, int8, int16, int32, int64
- uint, uint8, uint16, uint32, uint64, uintptr
- byte (= int8)
- bool
- float32, float64
- complex64, complex128
- string
- rune (= int32, представляет Unicode код)

Строки

```
hi := "\tHi!\n"
withoutSpec := `\tHi!\n`

var rawByte byte = '\x27'
var utf8 rune // Unicode (UTF-8) out of the box
```

Строки

```
hi := "\tHi!\n"
withoutSpec := `\tHi!\n`

var rawByte byte = '\x27'
var utf8 rune // Unicode (UTF-8) out of the box

hello := "Hello, "
helloWorld := hello + "World"
```

Строки

```
hi := "\tHi!\n"
withoutSpec := `\tHi!\n`

var rawByte byte = '\x27'
var utf8 rune // Unicode (UTF-8) out of the box

hello := "Hello, "
helloWorld := hello + "World"

// cannot assign to helloWorld[0]
helloWorld[0] = 72 // strings are immutable
```

Строки

```
// str = "Привет, мир!"

byteLen := len(str) // 21
runeLen := utf8.RuneCountInString(str) // 12
```

Строки

```
// str = "Привет, мир!"

byteLen := len(str) // 21
runeLen := utf8.RuneCountInString(str) // 12

hello := str[:12] // Привет
hello1 := str[0] // byte, 72 !! (не 'П')
```

Строки

```
// str = "Привет, мир!"

byteLen := len(str) // 21
runeLen := utf8.RuneCountInString(str) // 12

hello := str[:12] // Привет
hello1 := str[0] // byte, 72 !! (не 'П')

byteString := []byte(str)
strFromByte := string(byteString)
```

Константы

```
const pi = 3.14159265
const (
    e = 2.7182818
    hello = "Hello"
)
```

Константы

```
const pi = 3.14159265
const (
    e = 2.7182818
    hello = "Hello"
)

const (
    zero = iota
    _ // omission of a value
    two // = 2
)

const (
    _ = iota
    kB uint64 = 1 << (10 * iota) // = 1024
    MB // = 1048576
)
```

Массивы

```
var arr [3]int // [0, 0, 0]
fmt.Printf("%v\n", a) // [0 0 0]
fmt.Printf("%#v\n", a) // [3]int{0, 0, 0}
```

Массивы

```
var arr [3]int // [0, 0, 0]
fmt.Printf("%v\n", a) // [0 0 0]
fmt.Printf("%#v\n", a) // [3]int{0, 0, 0}

const size = 2
var b [2 * size]bool
```

Массивы

```
var arr [3]int // [0, 0, 0]
fmt.Printf("%v\n", a) // [0 0 0]
fmt.Printf("%#v\n", a) // [3]int{0, 0, 0}

const size = 2
var b [2 * size]bool

a3 := [...]int{1, 2, 3}
```

Массивы

```
var arr [3]int // [0, 0, 0]
fmt.Printf("%v\n", a) // [0 0 0]
fmt.Printf("%#v\n", a) // [3]int{0, 0, 0}

const size = 2
var b [2 * size]bool

a3 := [...]int{1, 2, 3}

// invalid array index 3 (out of bounds for 3-element array)
fmt.Println(a3[3])
```

Слайсы

```
var buf []int
buf1 := []int
buf2 := []int{42}
buf3 := make([]int, 1, 10) // len, cap
```

Слайсы

```
var buf []int
buf1 := []int
buf2 := []int{42}
buf3 := make([]int, 1, 10) // len, cap

elem := buf2[0]
elem2 := buf2[1] // panic: index out of range
```

Слайсы

```
var buf []int
buf1 := []int
buf2 := []int{42}
buf3 := make([]int, 1, 10) // len, cap

elem := buf2[0]
elem2 := buf2[1] // panic: index out of range

buf = append(buf, 9, 10)
buf2 = append(buf, 12)
buf = append(buf, buf2...)
```

Слайсы

```go
var buf []int
buf1 := []int
buf2 := []int{42}
buf3 := make([]int, 1, 10) // len, cap

elem := buf2[0]
elem2 := buf2[1] // panic: index out of range

buf = append(buf, 9, 10)
buf2 = append(buf, 12)
buf = append(buf, buf2...)

bufLen, bufCap := len(buf), cap(buf)
```

Слайсы

```
buf := []int{0, 1, 2, 3, 4, 5}
// [l; r)
s1 := buf[1:]
s2 := buf[:2]
s3 := buf[1:3]
```

Слайсы

```
buf := []int{0, 1, 2, 3, 4, 5}
// [l; r)
s1 := buf[1:]
s2 := buf[:2]
s3 := buf[1:3]

b1 := buf[:]
b1 = append(b1, 6)
```

Слайсы

```
buf := []int{0, 1, 2, 3, 4, 5}
// [l; r)
s1 := buf[1:]
s2 := buf[:2]
s3 := buf[1:3]

b1 := buf[:]
b1 = append(b1, 6)

copied := make([]int, len(buf), cap(buf))
copy(copied, buf)
```

Слайсы

```
buf := []int{0, 1, 2, 3, 4, 5}
// [l; r)
s1 := buf[1:]
s2 := buf[:2]
s3 := buf[1:3]

b1 := buf[:]
b1 = append(b1, 6)

copied := make([]int, len(buf), cap(buf))
copy(copied, buf)

copy(buf[1:3], []int{9, 10})
```

Мапы

```
var info map[string]string = map[string]string{
    "ping": "pong",
}
withLength := make(map[string]string, 10)
length := len(info)
value := info["key"]
```

Мапы

```
var info map[string]string = map[string]string{
    "ping": "pong",
}
withLength := make(map[string]string, 10)
length := len(info)
value := info["key"]

value, ok := info["key"]
_, ok := info["ping"]
```

Мапы

```
var info map[string]string = map[string]string{
    "ping": "pong",
}
withLength := make(map[string]string, 10)
length := len(info)
value := info["key"]

value, ok := info["key"]
_, ok := info["ping"]

delete(info, "ping")
```

Ставим условия

```
boolValue := true
if boolValue {
    fmt.Print("boolVal is true!")
}
```

Ставим условия

```
boolValue := true
if boolValue {
    fmt.Print("boolVal is true!")
}

if v, ok := mp["key"]; ok {
    fmt.Print("Key exists")
} else {
    fmt.Print("Nope")
}
```

Ставим условия

```
str := "TEST"
switch str {
case "Ping":
    fallthrough
case "Pong", "TEST":
    // some code
default:
    // some code
}
```

Ставим условия

```
a, b := 10, 12
switch {
case a > 10 && b < 12:
     // some code
case a <= 10 && b == 12:
     // some code
case a == 10:
     // some code
}
```

Циклимся

```
for {  // while (true)
    break
}
```

Циклимся

```
for { // while (true)
    break
}

isRunning := true
for isRunning {
    isRunning = false
}
```

Циклимся

```
for { // while (true)
    break
}

isRunning := true
for isRunning {
    isRunning = false
}

for i := 0; i < 2; i++ {
    fmt.Println(i)
    if i == 1 {
        continue
    }
}
```

Циклимся

```
a := []int{1, 2, 3}
for idx, key := range arr {
    fmt.Printf("%d: %d", idx, key)
}
```

Циклимся

```
a := []int{1, 2, 3}
for idx, key := range arr {
    fmt.Printf("%d: %d", idx, key)
}

info := map[string]string{"ping": "pong"}
for k, v := range info {
    fmt.Printf("%s: %s", k, v)
}
```

Циклимся

```
a := []int{1, 2, 3}
for idx, key := range arr {
    fmt.Printf("%d: %d", idx, key)
}

info := map[string]string{"ping": "pong"}
for k, v := range info {
    fmt.Printf("%s: %s", k, v)
}

str := "Hello, world!"
for pos, c := range str {
    fmt.Printf("%d: %c", pos, c)
}
```

Функции

```
func plusOne(a int) int {
    return a + 1
}
```

Функции

```
func plusOne(a int) int {
    return a + 1
}

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

Функции

```
func plusOne(a int) int {
    return a + 1
}

func sumOfThree(a, b int, c int) int {
    return a + b + c
}

func realFunc(x int) (int, error) {
    if x % 2 == 0 {
        return 0, fmt.Errorf("number is even")
    }
    return x, nil
}
```

Функции

```
func namedReturn(x int) (a int, err error) {
    if x % 2 == 0 {
        err = fmt.Errorf("number is even")
        return
    }
    return x, nil
}
```

Функции

```
func namedReturn(x int) (a int, err error) {
    if x % 2 == 0 {
        err = fmt.Errorf("number is even")
        return
    }
    return x, nil
}

func sum(in ...int) (sum int) {
    for _, v := range in {
        sum += v
    }
    return
}
```

Анонимные функции

```
test := func(a int) int {
    return -a
}
test(5) // -5
```

Анонимные функции

```
test := func(a int) int {
    return -a
}
test(5) // -5

type strFuncType func(string) error
prefixer := func(prefix string) strFuncType {
    return func(in string) error {
        fmt.Printf("[%s] %s", prefix, in)
        return nil
    }
}
successLogger := prefixer("LOG_SUCCESS")
successLogger("okay")
```

Отложенное выполнение

```
func main() {
    defer fmt.Println("after main")
    fmt.Println("main")
}
```

Отложенное выполнение

```
func main() {
    defer fmt.Println("after main")
    fmt.Println("main")
}

func getValue() string {
    fmt.Println("getValue")
    return "getValue result"
}

func test() {
    defer fmt.Println("test")
    defer func() {
        fmt.Println(getValue())
    }()
    fmt.Print("work")
}
```

Паника

```
func run() {
    panic("some problems")
    // panic: some problems
}
```

Паника

```
func run() {
    panic("some problems")
    // panic: some problems
}

func start() {
    defer func() {
        if err := recover(); err != nil {
            fmt.Println("err:", err)
        }
    }()

    run()
}
```