

ЦЕНТР ЦИФРОВОГО РАЗВИТИЯ

STUD iT

Лекция 2

Основы Golang

Боренко Анастасия Денисовна
Выпускник иу7, Java-разработчик



Языки программирования

Какие вы знаете?



Языки программирования















Языки программирования

По уровню абстракции	низкоуровневый	высокоуровневый
По методу сборки и запуска	компилируемый	интерпретируемый
По парадигме программирования	процедурный, объектноориентированный, функциональный, аспектноориентированный и т.д.	

Почему Go?



GOLANG

Golang (или просто Go) – это компилируемый, статически типизированный язык программирования общего назначения, разработанный компанией Google в 2007 году и представленный публике в 2009 году.

- простой
- функциональный
- популярный





План лекции

1. Инструменты разработки
2. Программа на Go
3. Синтаксис
4. Массивы и слайсы
5. Указатели, ссылки, значения
6. Структуры и интерфейсы



Инструменты разработки

Компилятор - Go.

Актуальная версия - 1.27.1.



Команда	Синтаксис	Значение
clean	go clean	удаление объектных и кэшированных файлов
build	go build ***	компиляция пакетов и зависимостей
run	go run ***	компиляция и запуск Go программу
test	go test ***	запуск тестов к пакетам
tmt	go tmt ***	форматирование исходников в соответствии с styleGuides
install	go install	компиляция и установка пакетов и зависимостей
list	go list	вывод списка пакетов или модулей
version	go version	напечатать версию Go



Команда	Синтаксис	Значение
download	go mod download ***	загрузка модулей в локальный кеш
edit	go mod edit ***	редактирование go.mod из инструментов или скриптов
graph	go mod graph ***	распечатать граф требований модуля
init	go mod init ***	инициализировать новый модуль в текущем каталоге
tidy	go mod tidy	добавить отсутствующие и удалить неиспользуемые модули
verity	go mod verity	проверка достоверности зависимостей



Базовые типы

Типы		Обозначение	Пример значения
Числа	Целочисленные со знаком	int int8, int16, int32, int64 (rune=int16)	0, 1, -1, 5,...
	Целочисленные без знака	uint, uintptr uint8, uint16, uint32, uint64 (byte=uint8)	0, 1, 5,...
	Вещественные	float float32, float64	0.0, 1.23467
	Комплексные	complex64, complex128	1.203 и мнимая часть
Булевые		bool	true|false
Строки		string	“строка”



Базовые типы

```
var i int          // 0
var s string       // ""
var ok bool        // false

a := 42
b := 3.14

fmt.Println(float64(a) + b) // явное преобразование

type UserID int
id := UserID(1001)
```



Общий синтаксис

```
var name string = "Go"    // явное объявление
age := 15                 // короткое: тип выводится сам

const pi = 3.14
const (
    StatusOK    = 200
    StatusError = 500
)

if n := len(name); n > 0 {   // короткая инициализация
    fmt.Println("длина:", n)
}
switch age {                 // без break — выход из case сам
case 15:
    fmt.Println("молодой язык")
default:
    fmt.Println("неизвестно")
}
```



Общий синтаксис

```
for i := 1; i <= 5; i++ {     // классический
    sum += i
}
for i, ch := range name {     // по коллекции
    fmt.Printf("[%d]=%c\n", i, ch)
}
for condition { ... }         // как while
for { ... }                   // бесконечный
```

Интересный факт: в Go нет цикла while!!!



Массивы

```go
var arr [3]int                  // [0 0 0] — zero value
arr2 := [3]int{10, 20, 30}  // литерал
arr3 := [...]int{1, 2, 3}   // длина выведется сама → [3]int

arr := [3]int{10, 20, 30}
copy := arr
copy[0] = 99
fmt.Println(arr)  // [10 20 30] — не изменился
fmt.Println(copy) // [99 20 30]

arr[0] = 42
fmt.Println(len(arr)) // 3
for i, v := range arr {
    fmt.Println(i, v)
}
```



Слайсы

```go
nums := []int{1, 2, 3}              // литерал
buf := make([]string, 0, 4)         // len=0, cap=4
var nilSlice []int                  // nil, len=0

nums = append(nums, 4, 5)
fmt.Println(len(nums), cap(nums))

part := nums[1:4]  // элементы с индексами 1, 2, 3
part[0] = 200      // меняет и nums — общий underlying array!

a := []int{1, 2, 3}
b := append(a, 4)                   // может выделить новый массив
dst := make([]int, len(a))
copy(dst, a)                        // независимая копия элементов

var nilSlice []int   // nilSlice == nil → true
empty := []int{}     // empty == nil → false
// len у обоих = 0; range и append работают с обоими
```



Функции

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

fmt.Println(add(2, 3)) // 5
```



ФУНКЦИИ

```
func divmod(a, b int) (int, int) {
    return a / b, a % b
}

q, r := divmod(17, 5) // q=3, r=2
```



ФУНКЦИИ

```
func divmod(a, b int) (quot int, rem int) {
    quot = a / b
    rem = a % b
}

q, r := divmod(17, 5) // q=3, r=2
```



Функции

Аргументы всегда передаются по значению

Для игнорирования ненужного результата можем использовать “_”:

```
q, _ := divmod(17, 5)
```



Указатели, ссылки, значения

Символ	Смысл
&x	взять адрес переменной
*p	разыменовать: взять значение по адресу
*int	тип «указатель на int»



Указатели, ссылки, значения

```
x := 10
p := &x             // адрес x
fmt.Println(*p)     // 10

*p = 42             // меняем x через указатель
fmt.Println(x)      // 42
```



Указатели, ссылки, значения

```go
func incrementByValue(n int)  { n++ }
// копия — оригинал не меняется

func incrementByPointer(n *int) { *n++ }
// меняем по адресу

n := 5
incrementByValue(n)      // n всё ещё 5
incrementByPointer(&n)   // n стало 6
```



Указатели, ссылки, значения

```
var p *int              // nil
// *p — паника! сначала проверяй: if p != nil
```



Структуры

```
type User struct {
    Name string
    Age  int
}

u := User{Name: "Анна", Age: 20}
// или позиционно: User{"Анна", 20}
```



Структуры

```
// Value receiver — работает с копией
func (u User) Greet() string {
    return fmt.Sprintf("Привет, я %s", u.Name)
}
// Pointer receiver — может менять исходный объект
func (u *User) HaveBirthday() {
    u.Age++
}
u.Greet()
u.HaveBirthday() // Go сам возьмёт &u
```



Интерфейсы

```
type Shape interface {
    Area() float64
}
```



ИНТЕРФЕЙСЫ

```
type Circle struct{ Radius float64 }
func (c Circle) Area() float64 {
    return 3.14159 * c.Radius * c.Radius
}
type Rectangle struct{ Width, Height float64 }
func (r Rectangle) Area() float64 {
    return r.Width * r.Height
}
func printArea(s Shape) {
    fmt.Println(s.Area())
}
printArea(Circle{Radius: 2})
printArea(Rectangle{Width: 3, Height: 4})
```



Интерфейсы

Утиная типизация



Кря-кря-кря



Пакеты в Go

```
03_packages/
  main.go              → package main
  greeter/
    greeter.go         → package greeter
```

```
package main
import (
    "fmt"
    "lection2.gobase/03_packages/greeter"
)

func main() {
    fmt.Println(greeter.Hello("студент"))
}
```



Видимость

```
// package greeter
func Hello(name string) string { ... } // экспортирована
func secret() string { ... }           // только внутри
greeter

// package main
greeter.Hello("студент") // ОК
greeter.secret()         // ошибка компиляции
```



Общая структура проекта на Go

```
myproject/
  go.mod              ← имя модуля и версия Go
  go.sum              ← checksums зависимостей (после go get)
  main.go             ← или cmd/.../main.go
  …

// go.mod
module lection2.gobase
go 1.22.0
```



Общая структура проекта на Go

```
myapp/
  go.mod
  cmd/
    myapp/
      main.go          ← package main, точка входа
  internal/            ← код только для этого модуля (снаружи не
импортируют)
    service/
    storage/
  pkg/                 ← опционально: публичные библиотеки наружу
  api/ / docs/         ← контракты, OpenAPI и т.п.
```



СПАСИБО ЗА ВНИМАНИЕ!



ОБРАТНАЯ СВЯЗЬ