Peter the Great
Saint-Petersburg Polytechnic University

Тема работы

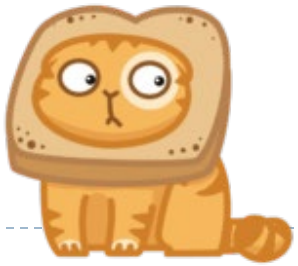
Пользовательские типы в Golang

Исполнители: учащиеся группы 5030102/00201
Руководитель: Иванов Денис Юрьевич

26.09.23

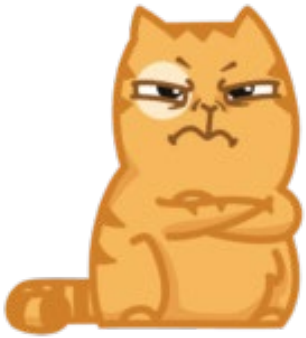
POLYTECH

Классы мои классы



- Только структуры
  - Объявляем через создание типа
  - Нет конструктора, но мы можем его имитировать
  - Внутри ничего, кроме полей
  - Нет значений по умолчанию
- А как быть?..

Создание объекта



```go
type User struct {
    firstName string
    lastName  string
    email     string
    age       int
}
```

```go
func NewUser(
    firstName string,
    lastName string,
    email string,
    age int,
) *User {
    return &User{
        firstName: firstName,
        lastName:  lastName,
        email:     email,
        age:       age,
    }
}
```

```go
func main() {
    user1 := NewUser("Misha", "A", "example@g.com", 20)
    user2 := User{firstName: "Boris", lastName: "B", email: "exampple@g.com", age: 22}
```

Значения по умолчанию

- Вариант попроще

```
50  func NewUser(
51      lastName string,
52      email string,
53      age int,
54  ) *User {
55      user := User{
56          lastName: lastName,
57          email:    email,
58          age:      age,
59      }
60      user.firstName = "Anton"
61      return &user
62  }
```

Значения по умолчанию

- Вариант посложнее

```go
import (
    "fmt"
    "reflect"
)

type User struct {
    firstName string `default:"Anton"`
    lastName  string
    email     string
    age       int
}
```

```go
func NewUser(
    lastName string,
    email string,
    age int,
) *User {
    user := User{
        lastName: lastName,
        email:    email,
        age:      age,
    }
    typ := reflect.TypeOf(user)
    if user.firstName == "" {
        f, _ := typ.FieldByName("firstName")
        user.firstName = f.Tag.Get("default")
    }
    return &user
}
```

```go
func main() {
    user := NewUser("A", "exampple@g.com", 20)
    fmt.Println(user.firstName)
```

```
ПРОБЛЕМЫ  ВЫХОДНЫЕ ДАННЫЕ  КОНСОЛЬ ОТЛАДКИ  ТЕРМИНАЛ  ПОРТЫ

PS D:\NewProgrammingLanguages\New-programming-languages\classes> go run .\main.go
Anton
```

Методы. Мы же все-таки в ООП лезем

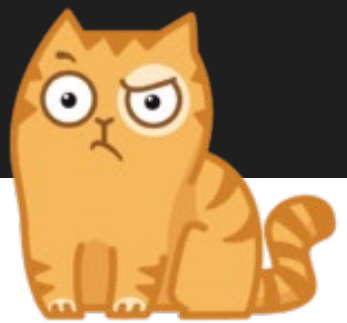
- Пишем их снаружи, все просто

```go
func (u *User) SetName(name string) {
    u.firstName = name
}
```

Методы. Это как функции, но...

- Не путать с функциями!

```
34  func print_user(u *User) {
35      fmt.Printf("name: %s %s\n", u.firstName, u.lastName)
36      fmt.Printf("email: %s\n", u.email)
37      fmt.Printf("age: %d\n", u.age)
38  }
39
40  func (u *User) print_me() {
41      fmt.Printf("name: %s %s\n", u.firstName, u.lastName)
42      fmt.Printf("email: %s\n", u.email)
43      fmt.Printf("age: %d\n", u.age)
44  }
```



Композиция?

- Присутствует
  - Объект одной структуры будет полем другой

```go
type Adress struct {
    country string
    city    string
}

type User struct {
    firstName string
    lastName  string
    email     string
    age       int
    Adress    Adress
}
```

```go
func main() {
    user := NewUser("Misha", "A", "exampple@g.com", 20, Adress{"Russia", "SPb"})
    fmt.Printf("user's adress is %s, %s", user.Adress.country, user.Adress.city)
```

Наследование?

- Встраивание!
  - Дочерняя структура получает те же поля, что и родительская
  - Множественное встраивание тоже доступно

```
type Admin struct {
    IsAdmin bool
    User
    department string
}
```

```
113  func main() {
114      user := NewUser("Misha", "A", "exampple@g.com", 20, Adress{"Russia", "SPb"})
115      admin := Admin{true, *user, "articles"}
116      fmt.Printf("Hello! I'm %s %s and I administer the %s section",
117          admin.firstName, admin.lastName, admin.department)
118
```

```
ПРОБЛЕМЫ  ВЫХОДНЫЕ ДАННЫЕ  КОНСОЛЬ ОТЛАДКИ  ТЕРМИНАЛ  ПОРТЫ

PS D:\NewProgrammingLanguages\New-programming-languages\classes> go run .\main.go
Hello! I'm Misha A and I administer the articles section
PS D:\NewProgrammingLanguages\New-programming-languages\classes>
```

Доступ к полям и методам

- Публичные
  - Названия начинаются с большой буквы
  - Видны везде
- Приватные
  - Названия начинаются с маленькой буквы
  - Видны в пределах пакета

Интерфейсы

- Описывают поведение
- Не указываются напрямую в структуре
- Часто создаются лишь для одного метода

Интерфейсы

```go
type Stringer interface {
    String() string
}

func (u *User) String() string {
    return u.firstName + " " + u.lastName + " " + u.email + " " + strconv.Itoa(u.age)
}

func (a *Adress) String() string {
    return a.country + " " + a.city
}

func print_smth(s Stringer) {
    fmt.Println(s.String())
}
```

```go
func main() {
    adress := NewAdress("Russia", "Saint-P")
    user := NewUser("A", "exampple@g.com", 20)

    print_smth(adress)
    print_smth(user)
```

```
ПРОБЛЕМЫ  ВЫХОДНЫЕ ДАННЫЕ  КОНСОЛЬ ОТЛАДКИ  ТЕРМИНАЛ  ПОРТЫ

PS D:\NewProgrammingLanguages\New-programming-languages\classes> go run .\main.go
Russia Saint-P
Anton A exampple@g.com 20
```

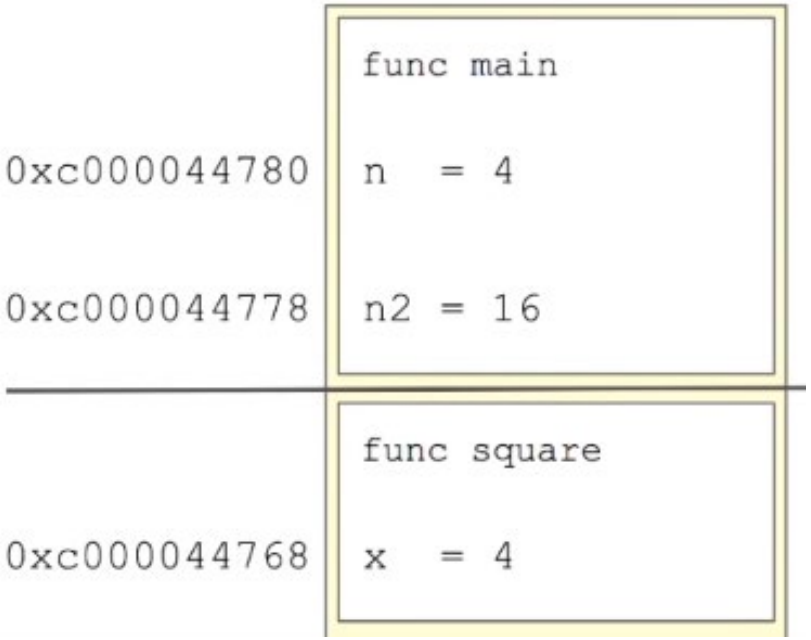
Стек или куча?

- Golang всегда будет стараться выделить значение переменных на стеке, кроме некоторых исключительных ситуаций.
- Escape Analysis — это механизм, который решает, будет ли храниться значение на стеке или на куче.

Простой пример стека

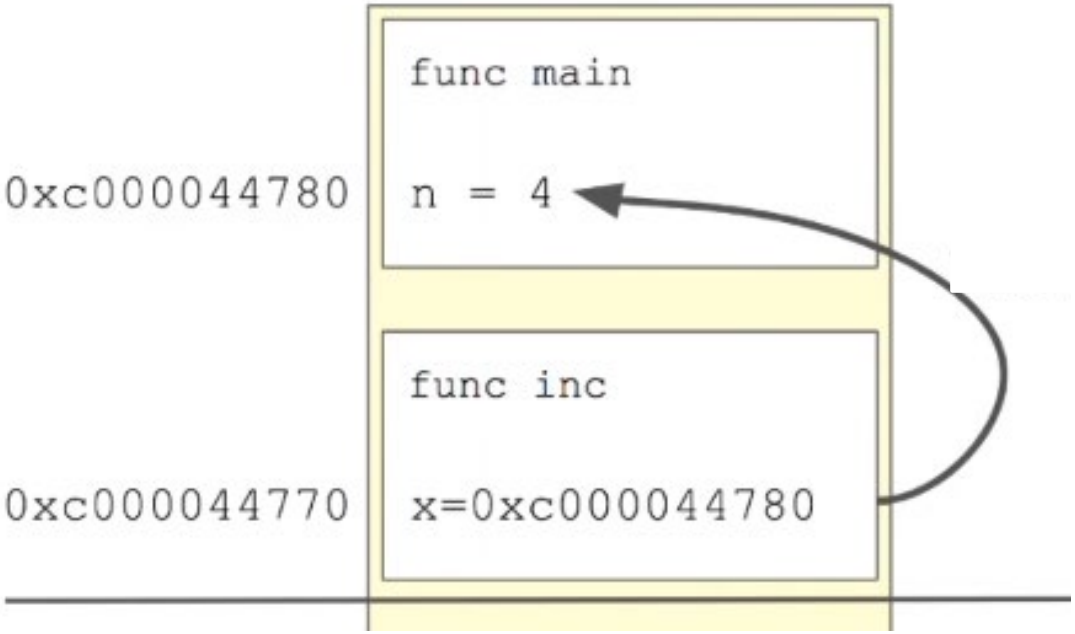
```
6  func main(){
7      n := 4
8      n2 := square(n)
9      fmt.Println(n2)
10 }
11
12 func square(x int) int {
13     return x * x
14 }
```

- Схема стека



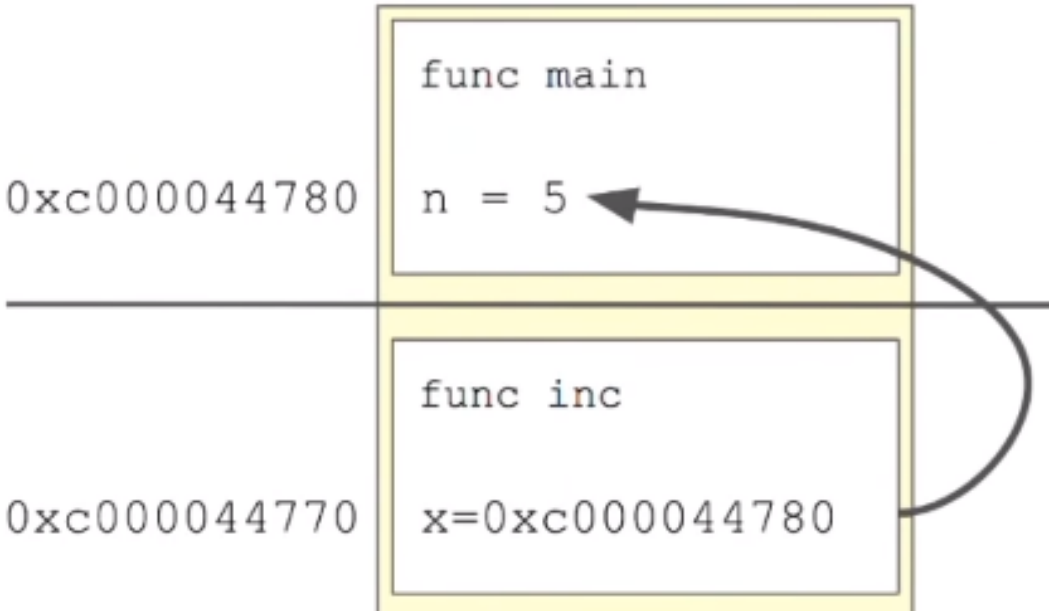
Пример стека с передачей указателя 1/2

```
 6  func main(){
 7      n := 4
 8      inc(&n)
 9      fmt.Println(n2)
10  }
11
12  func inc(x *int) {
13      *x++
14  }
```



Пример стека с передачей указателя 2/2

```
 6  func main(){
 7      n := 4
 8      inc(&n)
 9      fmt.Println(n2)
10  }
11
12  func inc(x *int) {
13      *x++
14  }
```

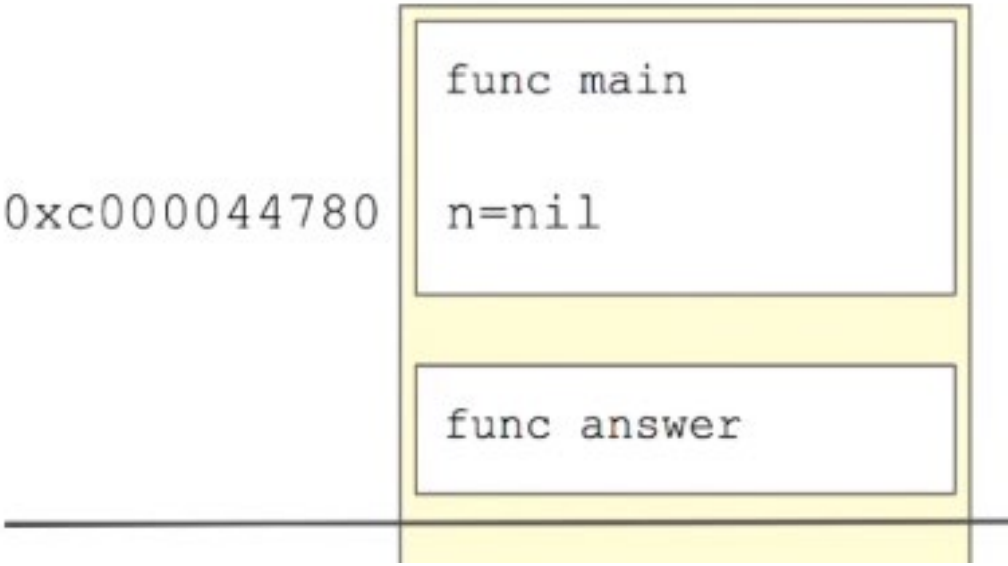


Так когда на стеке?

- Когда переменная не будет использоваться после выхода из локальной области, например, когда она не передается в другие функции или не сохраняется для использования после возврата из текущей функции.
- В примере: передача по ссылке остается на стеке.

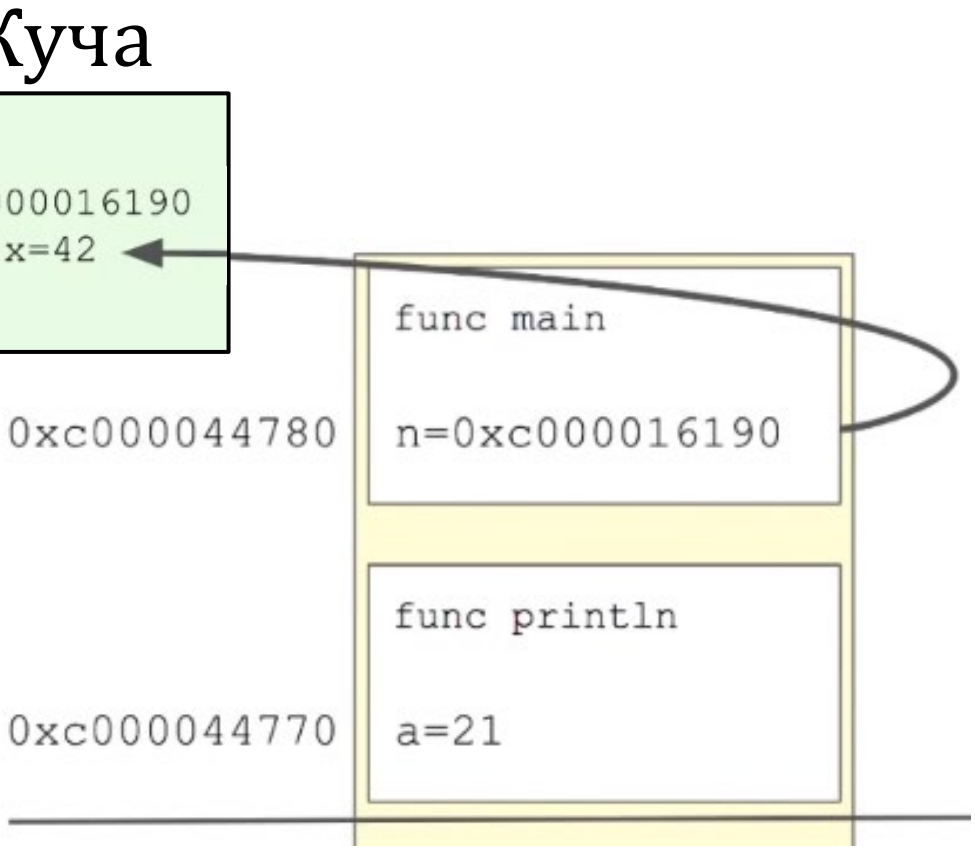
А если возвращать указатель? 1/2

```
 6   func main(){
 7       n := answer()
 8       fmt.Println(*n/2)
 9   }
10
11   func answer() *int {
12       x := 42
13       return &x
14   }
```



А если возвращать указатель? 2/2

```
6   func main(){
7       n := answer()
8       fmt.Println(*n/2)
9   }
10
11  func answer() *int {
12      x := 42
13      return &x
14  }
```



Когда создается на куче

- Если Escape analysis обнаруживает, что переменная будет использоваться за пределами локальной области, то объект или переменная будет выделена на куче.
- В примере: когда функция возвращает значение по ссылке.

Как узнать где лежит переменная?

- go build -gcflags=" -m "

```
5   func main() {
6       n := answer()
7       fmt.Println(*n/2)
8   }
9
10  func answer() *int {
11      x := 42
12      return &x
13  }
```

```
./main.go:10:6: can inline answer
./main.go:6:13: inlining call to answer
./main.go:7:16: inlining call to fmt.Println
./main.go:7:16: ... argument does not escape
./main.go:7:19: *n / 2 escapes to heap
./main.go:11:2: moved to heap: x
```