Go

Начальный курс по Go

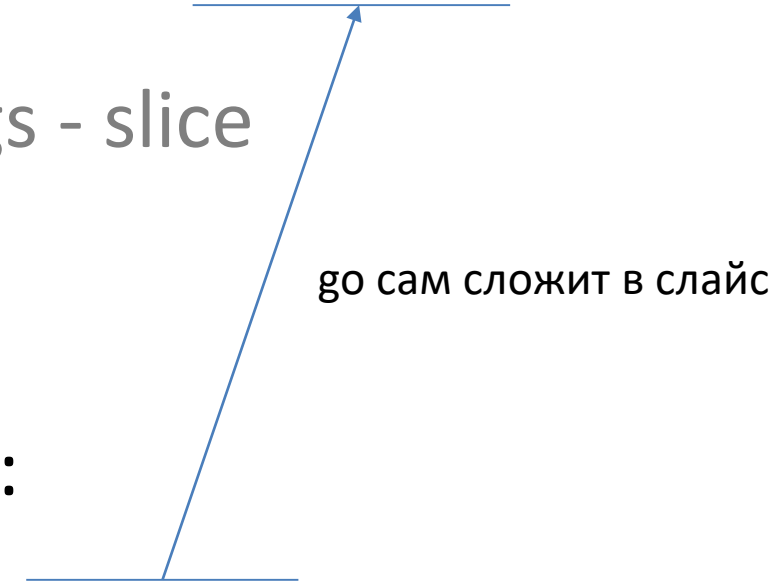
VARIADIC ARGUMENTS

Variadic arguments

```
func addAll(args ...int) int {

  // args - slice

}
```

Вызов:

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

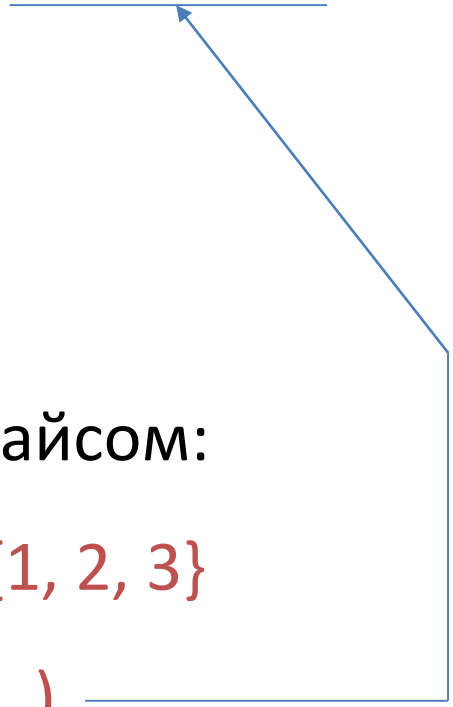


Variadic arguments

```
func addAll(args ...int) int {
  // args - slice
}
```

Пример со слайсом:

```
values := []int{1, 2, 3}
addAll(values...)
```



РАБОТА С ОШИБКАМИ

error

Интерфейс, объявленный в builtin.go:

```
type error interface {
   Error() string
}
```

errors

errors.New(message) создаёт stringError (см. errors.go)

errors

Ошибки возвращаются функцией как обычное значение (по соглашению – последним значением)

Общий формат (для main)

```
result, err := functionCall()

if err != nil {

   log.Fatal(err)

}
```

Печать ошибки + os.Exit(1)

Custom Errors

```
type queryError struct {
  query string
  err error
}
func (e *queryError) Error() string {
  return e.query + ": " + e.err.Error()
}
```

В обработке ошибок

В обработке ошибок:

```
_, err := someCall()
if err != nil {
    if typedError, ok := err.(*queryError); ok {
        ...
    }
}
```

fmt.Errorf

Форматированная ошибка, позволяющая "форматировать" текст, так же, как Printf

Начиная с Go 1.13 появился модификатор %w, который позволяет заворачивать ошибку для дальнейших тестов

Общий формат (для не main)

```
result, err := doStuff(arg)

if err != nil {

   return fmt.Errorf("can't do stuff %v, %w", arg, err)

}
```

вернёт ошибку вызывающей функции
и завернёт исходную ошибку

Общий формат (для не main)

```
result, err := doStuff(arg)

if err != nil {

   return fmt.Errorf("can't do stuff %v, %v", arg, err)

}
```

вернёт ошибку вызывающей функции
и не завернёт исходную ошибку

Но чаще всего делают

```
result, err := doStuff(arg)

if err != nil {

   return
}
```

не делайте так

Надо обработать ошибку, а не просто "молчаливо"

возвращать управление вызывающей функции

Format Strings

- %d, %x, %o, %b – integer
- %f, %g, %e - floating point number
- %t – boolean
- %c – rune
- %s – string
- %v - any value in natural format
- %T - type of value
- %w - error wrapping (only for Errorf)

Format Strings

fmt.Printf, log.Printf и fmt.Errorf используют одни и те же (за исключением %w) format strings

go 1.13+

В обработке ошибок:

```
_, err := someCall()
if err != nil {
   var typedError *queryError
   if errors.As(err, &typedError) {
      ...
   }
}
```

на соответствие какому типу проверка

делаем, если ошибка нужного типа

go 1.13+

В обработке ошибок:

```
var specificError = errors.New("specific")
_, err := someCall()
if err != nil {
   if errors.is(err, specificError) {
      ...
   }
}
```

выносят вне функций

на соответствие какой ошибке (где-то из wrapped)

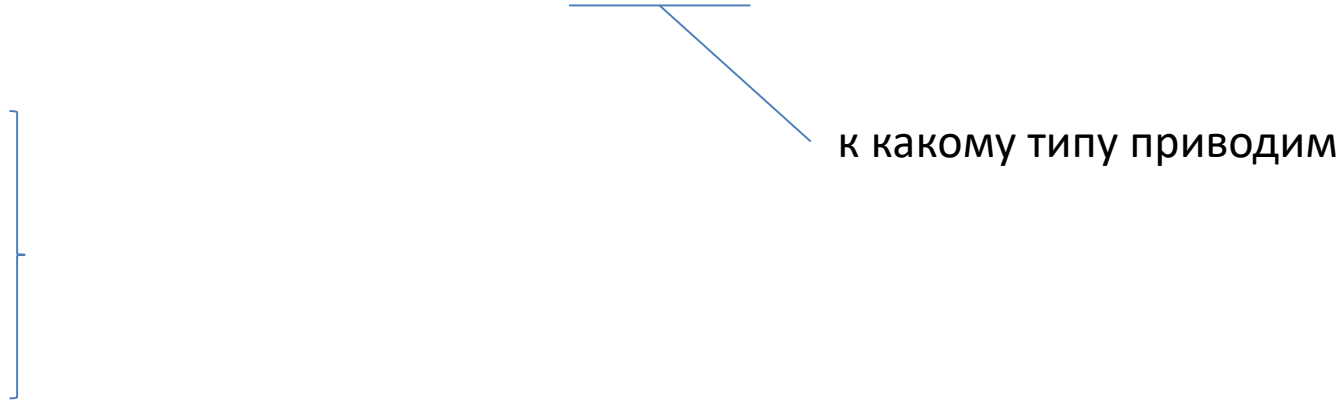
делаем, если ok

TYPE ASSERTIONS

Type Assertion

Type Assertion: попытка преобразовать "объект" из одного типа в другой:

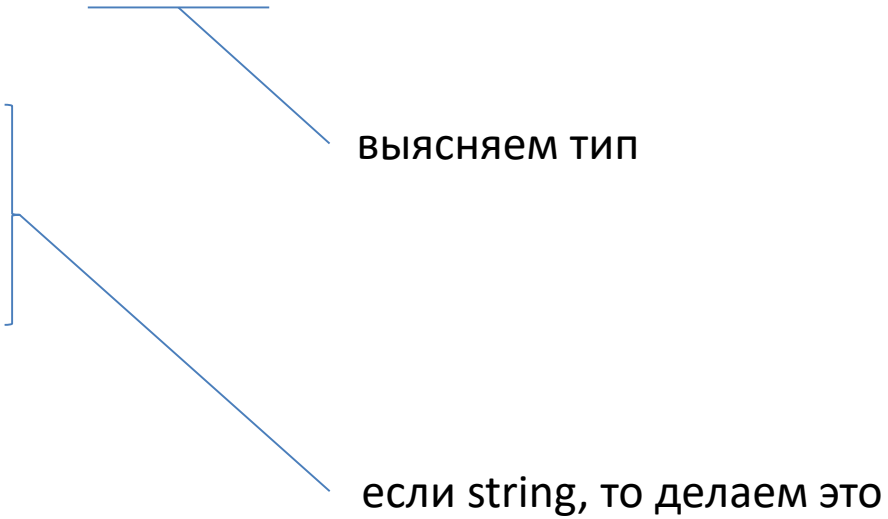
```
converted, ok := original.(string)

if ok {
    ...
}
```



Type Switch

Проверка типа

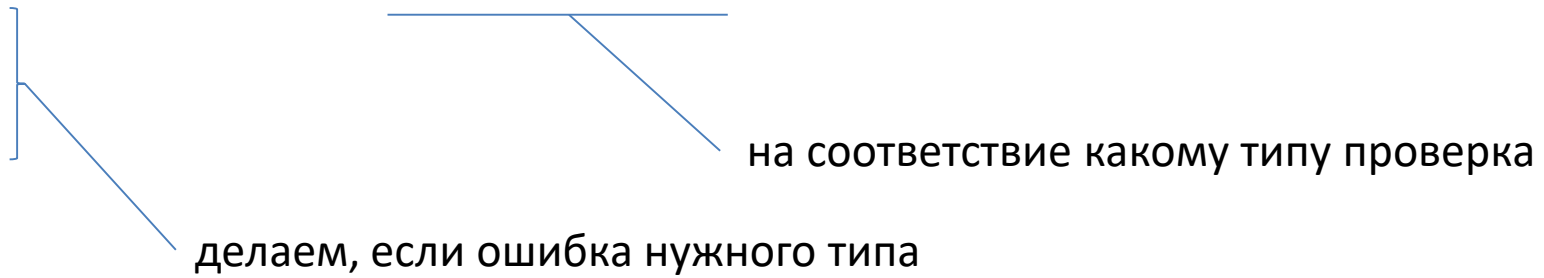
```
switch original.(type) {
case string:
  ...
case int:
  ...
```



В обработке ошибок

В обработке ошибок:

```
_, err := someCall()
if err != nil {
    if typedErr, ok := err.(*pkg.Error); ok {
        ...
    }
}
```



Дома

Посмотреть пакеты:

- errors
- strings
- fmt: Printf, Sprintf, Errorf
- sort

Спасибо за внимание

Ильназ Гильязов

2020г.