Ошибки в Go. Кастинг типов Шпаргалка

Возвращение ошибки из функции

Функции, которые могут вернуть ошибку, возвращают пару значений: результат и описание ошибки.

```
func OpenFile(name string) (*File, error) {
    file, err := os.Open(name)
    if err != nil {
        return nil, err
    }
    return file, nil
}
```

Проверка ошибки

Ошибки в Go проверяются сразу после их возникновения.

```
file, err := OpenFile("file.txt")
if err != nil {
    log.Fatal(err)
}
```

Создание ошибки

Кастомные ошибки можно создавать с помощью функции `errors.New(text string) error`.

```
func MyFunc(value int) (int, error) {
    if value < 0 {
        return 0, errors.New("Value cannot be negative")
    }
    return value, nil
}
```

Кастомные типы-ошибки

Go позволяет создавать собственные типы ошибок, реализуя интерфейс `error`.

```
type MyError struct {
    Msg string
}

func (e *MyError) Error() string {
    return e.Msg
}

func MyFunc(value int) (int, error) {
    if value < 0 {
        return 0, &MyError{"Value cannot be negative"}
    }
    return value, nil
}
```

Defer

Конструкция `defer` позволяет отложить выполнение функции до тех пор, пока не будет выполнена текущая функция или не произойдет паника.

```
file, err := OpenFile("file.txt")
if err != nil {
    log.Fatal(err)
}
defer file.Close()
```

Panic

Оператор `panic` используется для генерации паники. Паника означает, что произошла неожиданная ошибка и выполнение программы не может быть продолжено. При возникновении паники, выполнение программы останавливается.

```
func main() {
    fmt.Println("Начало")
    panic("Что-то пошло не так")
    fmt.Println("конец")
}


//

panic: runtime error: integer divide by zero

goroutine 1 [running]:
main.main()
  /Users/user/main.go:4 +0x3e
```

Recover

Оператор `recover` используется для обработки паники. Функция, которая вызывает `recover`, должна быть вызвана внутри отложенной функции. Если в момент вызова defer произошла паника, то `recover` вернет значение типа `interface{}` (возможно, `nil`), которое было передано в `panic`. Вызов `recover` останавливает раскручивание стека вызовов функций и позволяет обработать панику.

```
func MyFunc() {
    defer func() {
        if err := recover(); err != nil {
            log.Println("Recovered:", err)
        }
    }()
    panic("Something went wrong")
}
```