

**Яндекс.Практикум**

Дженерики: закрепляем на практике

Яндекс Практикум

# Что нас сегодня ждет

- Разберем некоторые детали внутреннего устройства дженериков
- Напишем несколько обобщенных функций

# Дженерики в Go

## 1. Параметры типов для функций и типов

```
func GMin[T constraints.Ordered](x T, y T) T {  
    if x < y {  
        return x  
    }  
    return y  
}
```

```
import (  
    "golang.org/x/exp/constraints"  
)  
  
type Tree[T comparable] struct {  
    left, right *Tree[T]  
    value      T  
}  
  
func (t *Tree[T]) Lookup(x T) *Tree[T] {  
    if t.value == x {  
        return t  
    }  
    if t.left != nil {  
        if f := t.left.Lookup(x); f != nil {  
            return f  
        }  
    }  
    if t.right != nil {  
        if f := t.right.Lookup(x); f != nil {  
            return f  
        }  
    }  
    return nil  
}
```

# Дженерики в Go

## 2. Type sets

```
type Number interface {  
    int | int16 | int32 | int64  
}  
  
func PrintNumber[T Number](n T) string {  
    return strconv.FormatInt(int64(n), 10)  
}
```

```
type NewNumber interface {  
    ~int | ~int16 | ~int32 | ~int64  
}  
  
func PrintNewNumber[T NewNumber](n T) string {  
    return strconv.FormatInt(int64(n), 10)  
}
```



# Дженерики в Go

## 3. Type inference

```
func UseGMin() {
    x, y := 42, -42

    firstMin := GMin[int](x, y)
    fmt.Printf("first min: %d\n", firstMin)

    var x32, y32 int32 = 42, -42
    secondMin := GMin(x32, y32)
    fmt.Printf("second min: %d\n", secondMin)
}
```

```
func Scale[E constraints.Integer](s []E, c E) []E {
    r := make([]E, len(s))
    for i, v := range s {
        r[i] = v * c
    }
    return r
}
```

```
type Point []int32
```



```
func (p Point) String() string {
    elems := make([]string, len(p))
    for i, el := range p {
        elems[i] = strconv.FormatInt(int64(el), 10)
    }
    return strings.Join(elems, ", ")
}
```

```
func ScaleAndPrint(p Point) {
    r := Scale(p, 2)
    ⚡ fmt.Println(r.String())
}
```


```
func NewScale[S ~[]E, E constraints.Integer](s S, c E) S {
    r := make([]E, len(s))
    for i, v := range s {
        r[i] = v * c
    }
    return r
}
```

```
func NewScaleAndPrint(p Point) {
    r := NewScale(p, 2)
    fmt.Println(r.String())
}
```

# Дженерики – что-то новое ?

 **builtin** package standard library 

Version: [go1.22.1](#) Latest |



Documentation

Overview

Index

Constants

Variables

▸ Functions

▸ Types

Source Files

<> Documentation

## Overview

Package builtin provides documentation for Go's predeclared identifiers. The items documented here are not actually in package builtin but their descriptions here allow godoc to present documentation for the language's special identifiers.

## Index

Constants

Variables

func [append\(slice \[\]Type, elems ...Type\) \[\]Type](#)

func [cap\(v Type\) int](#)

func [clear\[T ~\[\]Type | ~map\[Type\]Type1\]\(t T\)](#)

func [close\(c chan<- Type\)](#)

func [complex\(r, i FloatType\) ComplexType](#)

func [copy\(dst, src \[\]Type\) int](#)

func [delete\(m map\[Type\]Type1, key Type\)](#)

func [imag\(c ComplexType\) FloatType](#)

func [len\(v Type\) int](#)

func [make\(t Type, size ...IntegerType\) Type](#)

func [max\[T cmp.Ordered\]\(x T, y ...T\) T](#)

func [min\[T cmp.Ordered\]\(x T, y ...T\) T](#)

func [new\(Type\) \\*Type](#)

func [panic\(v any\)](#)

func [print\(args ...Type\)](#)

func [println\(args ...Type\)](#)

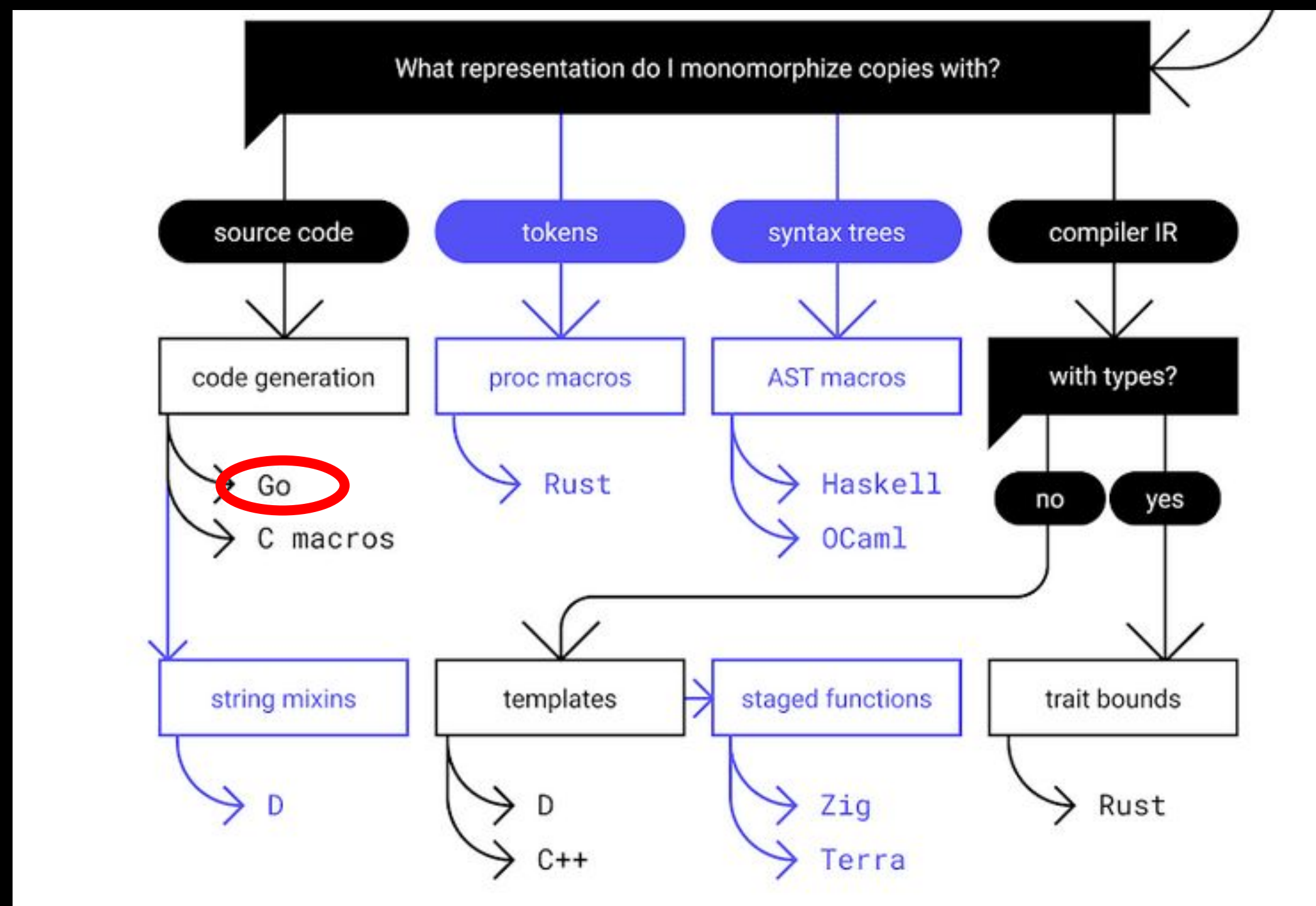
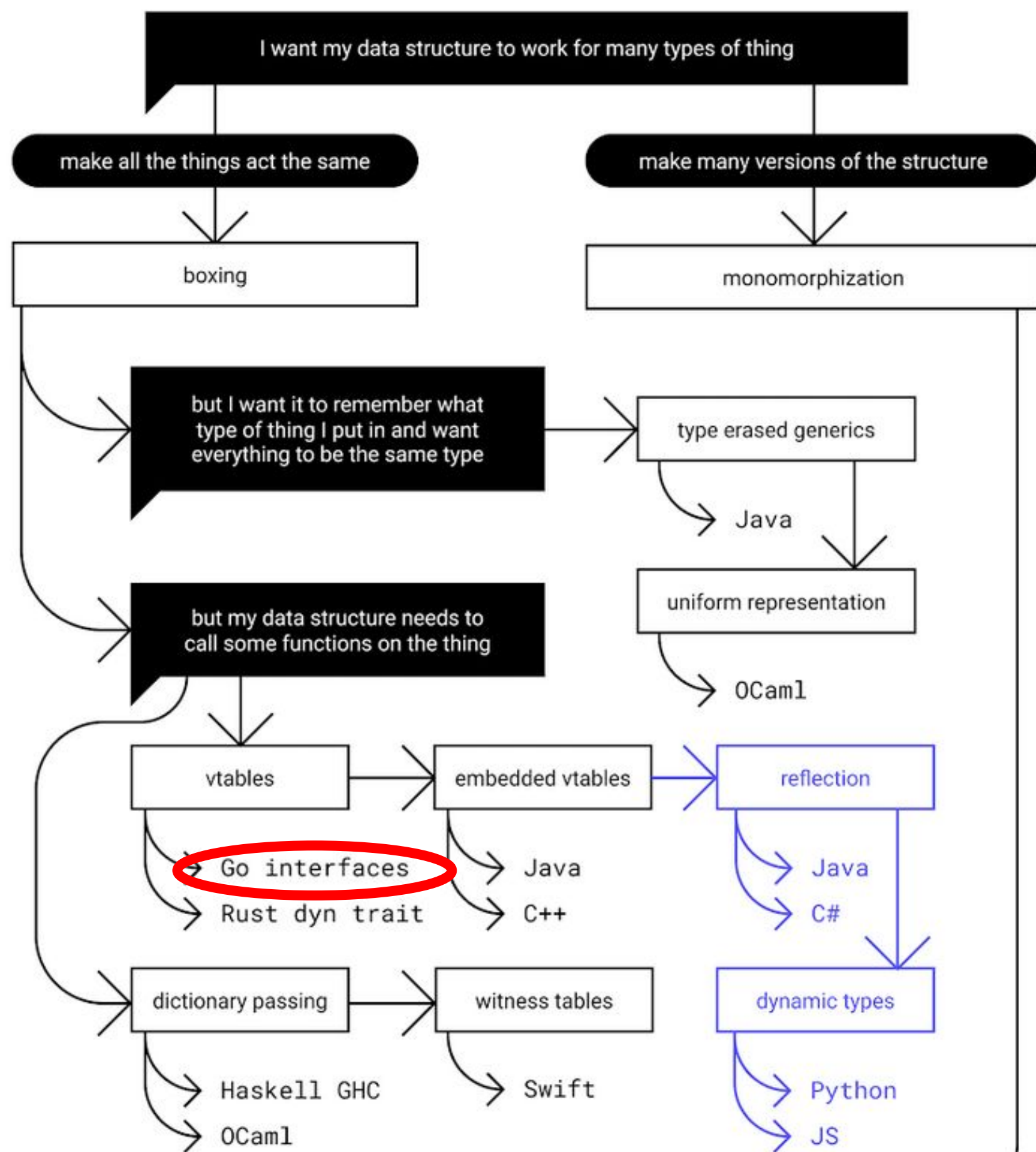
func [real\(c ComplexType\) FloatType](#)



# Реализация дженериков


How Languages Implement Generics and Extensions to Metaprogramming

<http://thume.ca/>



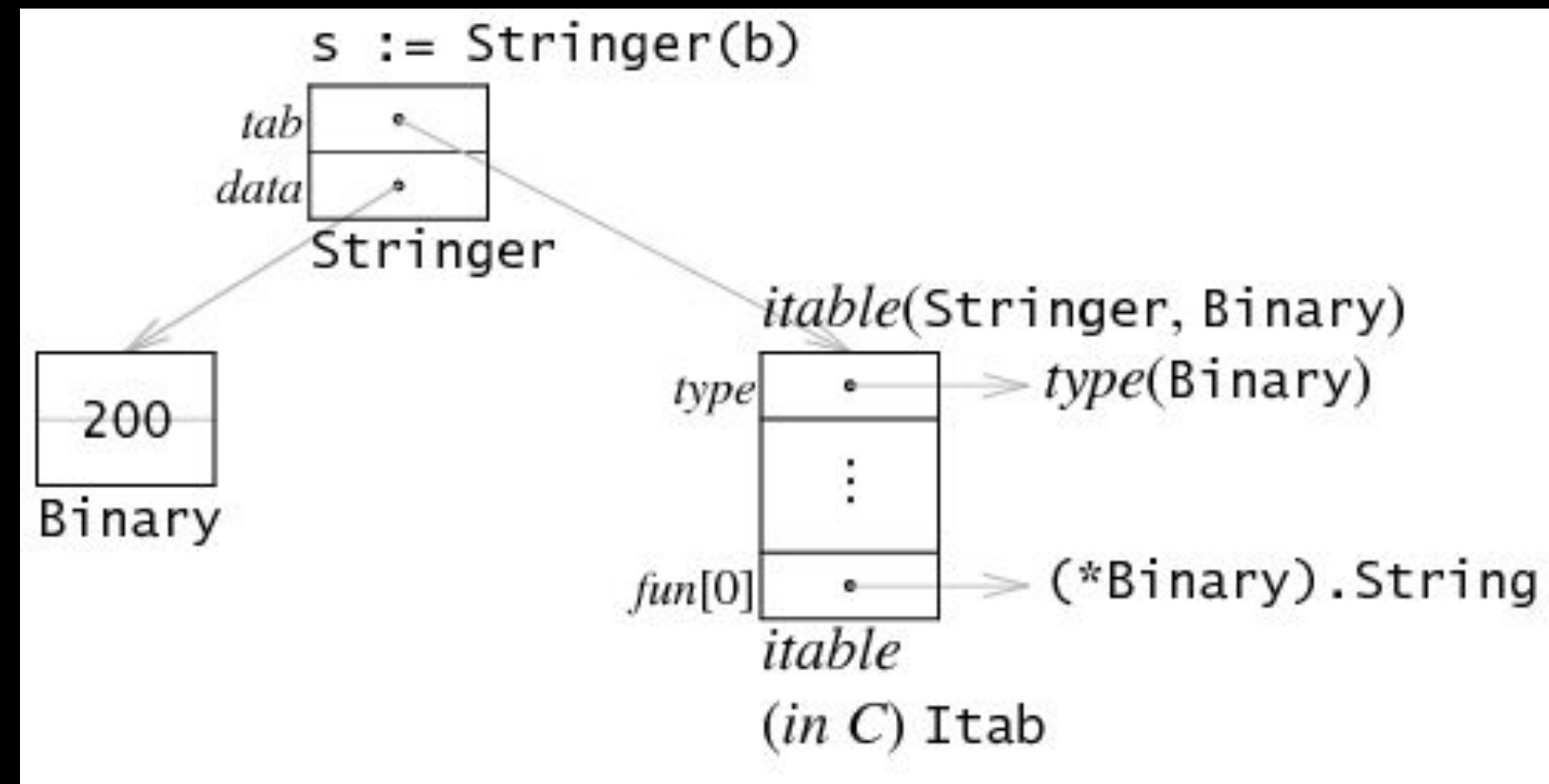
# Реализация дженериков в Go. Boxing

```
b := Binary(200)
```



A rectangular box representing a `Binary` object. It is divided horizontally by a line. The top half contains the value `200`, and the bottom half is empty.

Binary





# Реализация дженериков в Go. Частичная мономорфизация



# Практика

Спасибо за внимание!

Вопросы?