

Go

Начальный курс по Go

CONST

const

const может быть только числом, bool или строкой

iota

iota – constant generator

В объявлении констант **iota** стартует с нуля и
для каждого следующего объявления
увеличивается на 1

ТИПЫ

Типы в Go

- Базовые (числа, string, bool)
- Aggregate (array, struct)
- Reference (slice, map, pointer, function, channel)
- Interfaces

COMPOSITE TYPES

Composite types

1. `array` – массив однотипных элементов с фиксированной длиной
2. `struct` – сущность, группирующая фиксированный набор элементов

Массивы

Упорядоченный набор элементов одного типа*

```
var <name>[<length>]<type>
```

```
var data[10]int
```

- Имеет фиксированную длину (определяется `len()`)
- Элементы индексируются с нулевого индекса
- Доступ к элементу по индексу с помощью `[index]`

Массивы

Литералы:

```
items := [5]int {1, 2, 4}
```

```
items := [...]int {1, 2, 4}
```

struct

Тип, который может сгруппировать разные типы элементов:

```
student := struct {  
    name string  
    score int  
} {  
    "Oleg", 5,  
}
```

описание полей структуры

инициализированные значения

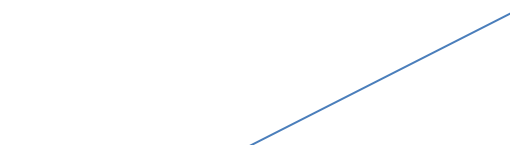
Операции

```
student.name = "Oleg Petrov"
```



запись данных в поле

```
fmt.Println(student.name)
```



чтение данных

struct

Тип, который может сгруппировать разные типы элементов:

```
students := []struct {  
    name string  
    score int  
} {  
    {"Oleg", 5},  
    {"Vasya", 4},  
}
```

описание полей структуры

инициализированные значения

ИМЕНОВАННЫЕ ТИПЫ

Именованные типы

type name definition

имя типа определение

Пример:

type accountId uint64

Именованные типы

```
id := uint64(1)
```

```
newAccountId = id // так нельзя!
```

```
newAccountId = accountId(id) // так можно
```

создалось автоматически

1. Go требует явного преобразования типов
2. Физически тип не преобразуется, мы просто говорим Go, что понимаем, что делаем

Именованные типы

```
type account struct {
```

```
    id accountId
```

```
    name string
```

```
    balance int
```

```
}
```

МЕТОДЫ

Классическое ООП

Объект – сущность, обладающая набором свойств и методов

Текущее значение всех свойств – состояние

Методы – операции, которые может выполнять объект

Методы в Go

Функции со специальным параметром, который называется `receiver` – "объект", на котором будет вызываться функция:

```
func (r Account) deposit(amount int) { ... }
```



receiver

Методы в Go

Теперь можно делать так:

`account.deposit(1000)`

receiver

метод

Методы в Go

Для Pointer Receiver'ов:

```
func (r *Account) deposit(amount int) { ... }
```

receiver

Но синтаксис вызова такой же:

```
account.deposit(1000)
```

receiver

метод

Методы в Go

На самом деле, go сам сделает:

```
(&account).deposit(1000)
```

Правило

Если хотя бы один метод с Pointer Receiver'ом,
то все должны быть объявлены с Pointer
Receiver'ом

GoLand

GoLand: meth + Tab

УКАЗАТЕЛИ

Указатели

Указатели (pointers) хранят адрес переменной в памяти (т.е. по какому адресу хранятся значения)

Через указатель можно читать и записывать значение по этому адресу

Указатели

```
count := 10
```

```
countPtr := &count
```

```
// полная запись
```

это отдельный тип



```
// var countPtr *int = &count
```

```
fmt.Println(*countPtr)
```

```
*countPtr = 11 // в count теперь тоже 11
```

Указатели

& - взятие адреса

* - dereferencing (разыменование указателя)

Debugger

В Debugger GoLand можно через **F2** менять значения переменных

nil

Нулевое значение для указателя любого типа
– **nil** (константа)

ptr **!= nil** – значит **ptr** указывает на данные

nil

`ptr1 == ptr2` – оба указывают на одну и ту же переменную или оба `nil`

nil

Важно: nil можно присвоить любой переменной типа интерфейса или reference type (ссылочного типа)

Reference Types

- slices
- maps
- channels

Важно: массивы и структуры не являются reference types (т.е. копируются при передаче)

NEW

new

Функция new создаёт переменную нужного типа, кладёт туда нулевое значение и возвращает адрес

```
var ptr *int = new(int)
```

INTERFACES

Interfaces

Интерфейс – контракт на реализацию поведения

Должны быть методы определённого типа (с нужной сигнатурой)

Interfaces

В Go интерфейсы реализуются неявно: т.е. если нужные методы реализованы, то тип уже соответствует интерфейсу

Важно: интерфейс ничего не говорит о внутренней реализации методов (они могут быть реализованы как угодно, главное, чтобы тип был нужный)

Interfaces

```
type printer interface {  
    print()  
}
```


Рассмотрение пакета sort

Практика на базе пакета sort

GoLand: **Alt + Enter -> Implement Interface**

Самостоятельно

Проанализировать следующие приложения
(посты и сообщения):

- vk.com
- Telegram
- Whatsapp

Подумать, как бы вы хранили эти данные

Спасибо за внимание

Ильназ Гильязов

2020г.