Golang基础

1. **所有基本类型（18个）**

bool byte rune int/uint int8/uint8 int16/uint16 int32/uint32 int64/uint64 float32 float64 complex64 complex128 string

rune:4字节专门用于存储Unicode字符（如中文）

complex64：64位复数类型，float32的实部和虚部联合表示

complex128：128位复数类型，float64的实部和虚部联合表示

字符串的字面量有两种表现形式：  
（1）原生：用反引号包裹，不可转义

1. 解释型：双引号包裹，可转义

两种都可以＋

1. 两种都可以互相加：a:=”abc”+`efg`
2. **所有关键字（25个）**

程序声明：import package type interface const var struct func chan map

流程控制：defer for range continue break go select if else switch case fallthrough default goto return

1. **所有内嵌函数（15个）**

append cap close complex copy

delete imag len make new

panic print println real recover

1. **所有操作符（21个）**

|| && == != < <= > >= + -

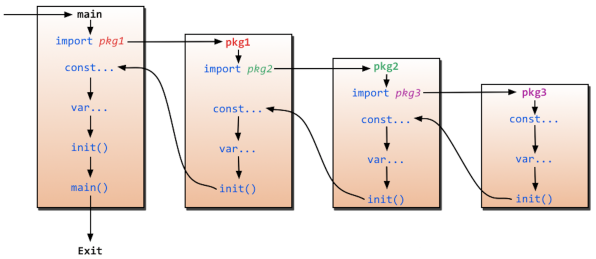
| ^ \* / % << >> & &^ ! <-

1. **高级类型（6个）：**数组 切片 map 函数和方法 接口 结构体

（详情在《Golang面试题》）

1. **可比较？？？？？？？**

**数组，切片、结构体、map、chan**

1. **函数**
   1. init函数和main函数不能有参数和返回值，init可有多个,init函数不能被其它函数调用
   2. 执行顺序：
   3. 参数：
      1. 参数必须有名称，结果则可有可无，要么全有，要么全无
      2. 参数是值传递，传入slice类型可以改变原值，但只能在函数内扩容，无法改变函数外的len（slice），但传入&slice可以
   4. 当匿名函数引用了外部的变量，那么这个匿名函数就变成了闭包函数
2. **方法**

值类型和指针类型都可以调用对应的方法，是否能对值做出改变取决于是值类型还是用指针类型定义的方法

1. **接口**

如果与myTalk关联的都是指针方法，这意味着，myTalk不是Talk接口的实现类型，\*myTalk才是。

接口没有继承，但是可以嵌入任意其它接口类型

**2.1.5 表达式**

类型断言v1.（I1）：判断一个接口的实际类型是否为某个类型

注意事项：

（1）v1必须是一个接口值（不是得转换）

（2）结果为否会产生一个恐慌，解决办法：i1,ok:=interface{}(v1).(I1)

**2.3.6 结构体**

结构体中可以有嵌入字段

值类型，因此零值不是nil而是mystruct{}

a:=struct{}{} a={}

结构体的所有字段在内存中是连续的

**2.4.3 switch**

（1）case后的switch语句都会被求值

（2）switch语句也可以包含一条子语句来初始化局部变量

（3）case后可以跟多个表达式，逗号隔开，满足一个即可

（4）break可以退出当前switch，也可以break+标签

类型sswitch语句：

（5）switch v.(type){}，case后只跟类型

（6）类型switch不允许有fallthrough

**2.4.4 for**

range字句：

1. 等号右边只有一个接收，得到的是索引
2. 迭代无元素的数组，nil切片、nil map 一开始就直接结束
3. 迭代nil通道，当前goroutine会永远阻塞在此

**2.4.5 defer语句**

注意事项：

1. 如果在延迟函数中使用外部变量应该通过参数传入
2. 同一个外围函数多个延迟函数是逆序执行的
3. 延迟函数如有参数传入，那么那些参数的值会在当前defer语句执行时求出

2.4.6 defer recover

defer func(){

if p:=recover();p!=nil{

fmt.Println(“出错”)

} //放在函数体的开始处