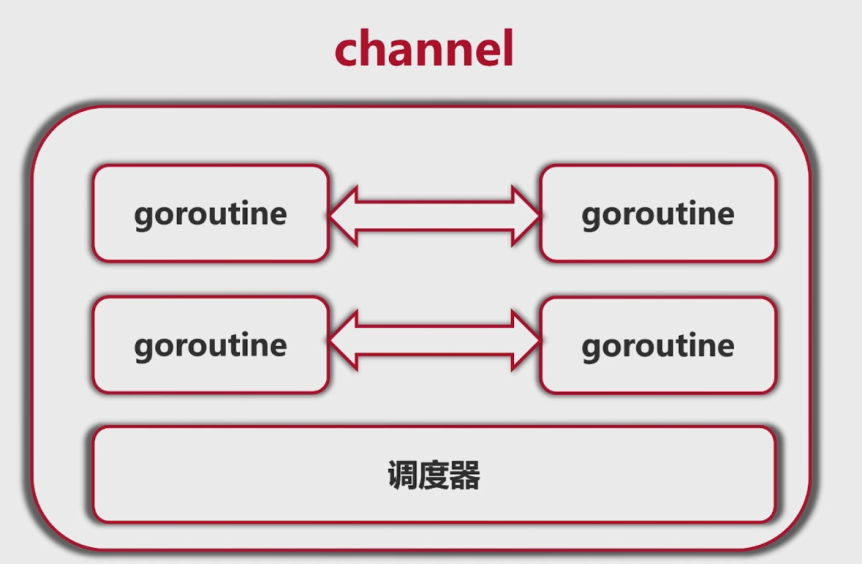
# 什么是channel

goroutine与goroutine之间的双向通道就是channel



Channel也是一等公民，也能作为参数、返回值



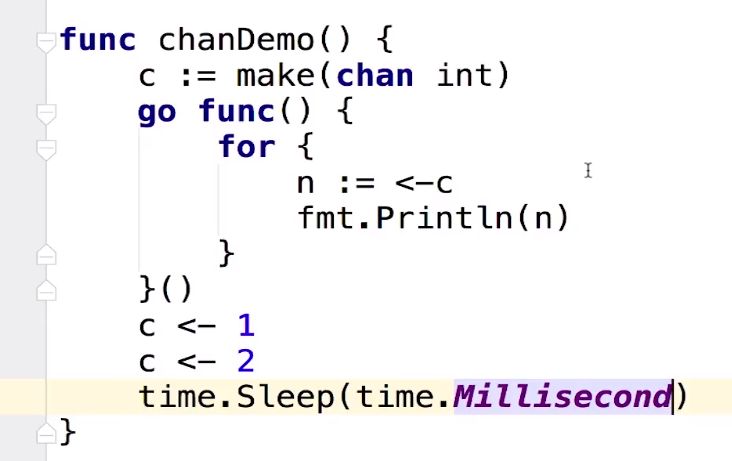
Go语言的并发是基于CSP模型，所以channel的设计也是基于CSP

**发送（chan<-）或接收（<-chan）都会阻塞当前goroutine，必须在其它goroutine里面接收或发送chan才能解除阻塞**

# 逐步讲解channel

## 一

**channel是goroutine之间的交互**



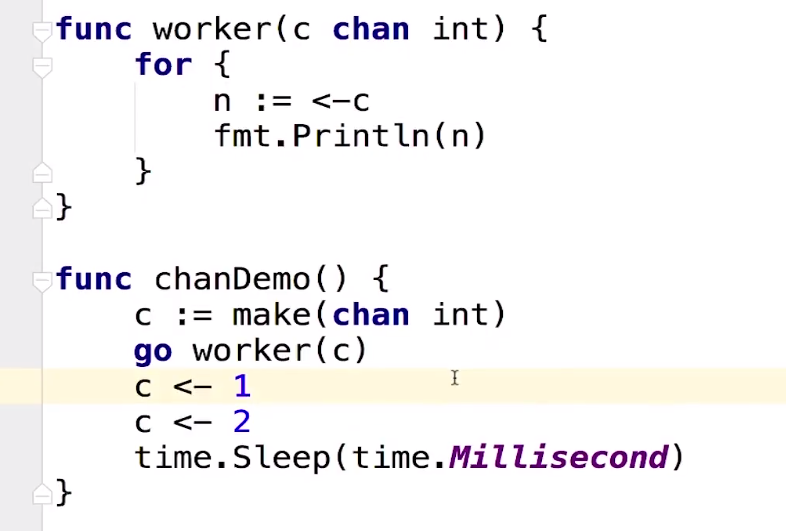
必须开一个goroutine接收数据，因为channel就是goroutine之间的交互，只有一个goroutine是会死锁的

接收方开一个goroutine死循环接收

发送方延迟一微秒结束程序

## 二

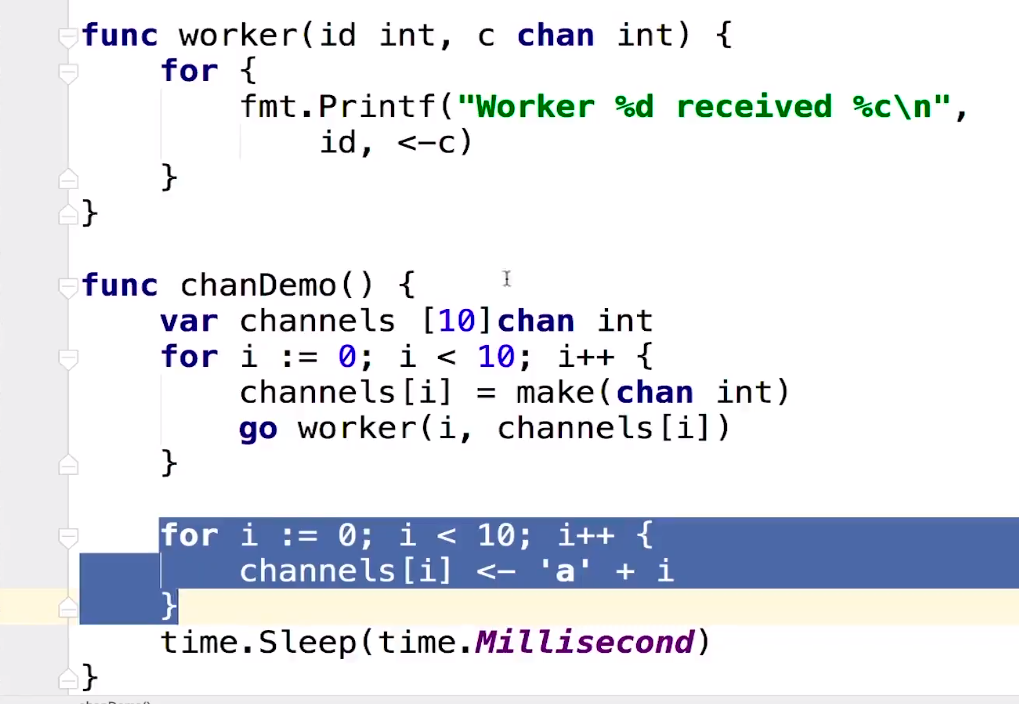
**Chan作为参数**



把匿名函数提出去，把c作为参数传递

## 三

**Chan作为数组**



定义一个大小为10的chan数组

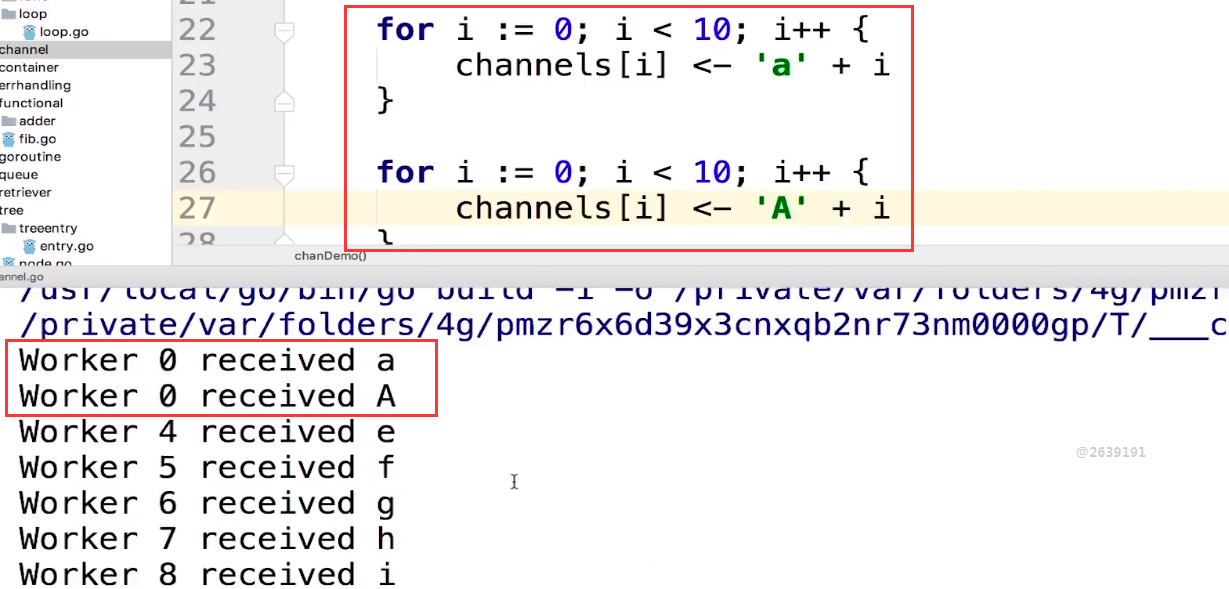
用for循环一次性开10个goroutine，把10个chan传给函数。

10个goroutine都在等待主程序发送数据

用for循环发送10个chan数据

## 四

**goroutine调度的随机性**

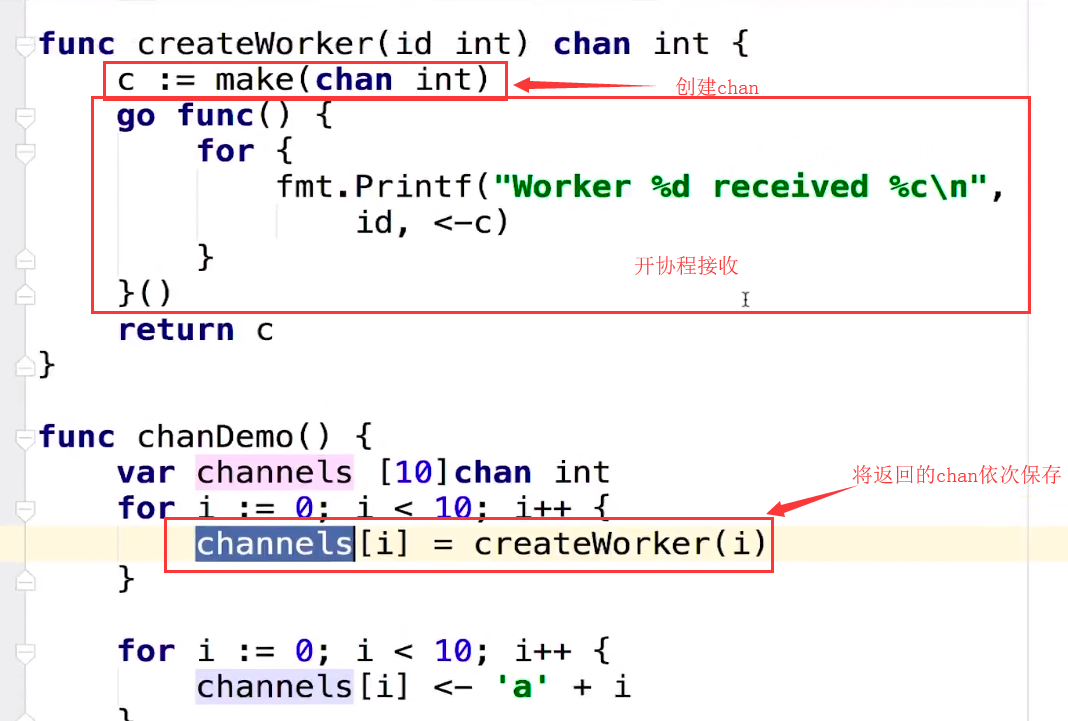


在三的基础上，再用一个for循环依次给10个chan发送数据

可以看到虽然是按顺序发送的，也是按顺序接收的。但是接收的时候有fmt（I/O）操作，导致会切换goroutine，由于goroutine调度的随机性，所以会产生随机打印的结果

## 五

**Chan作为返回值**



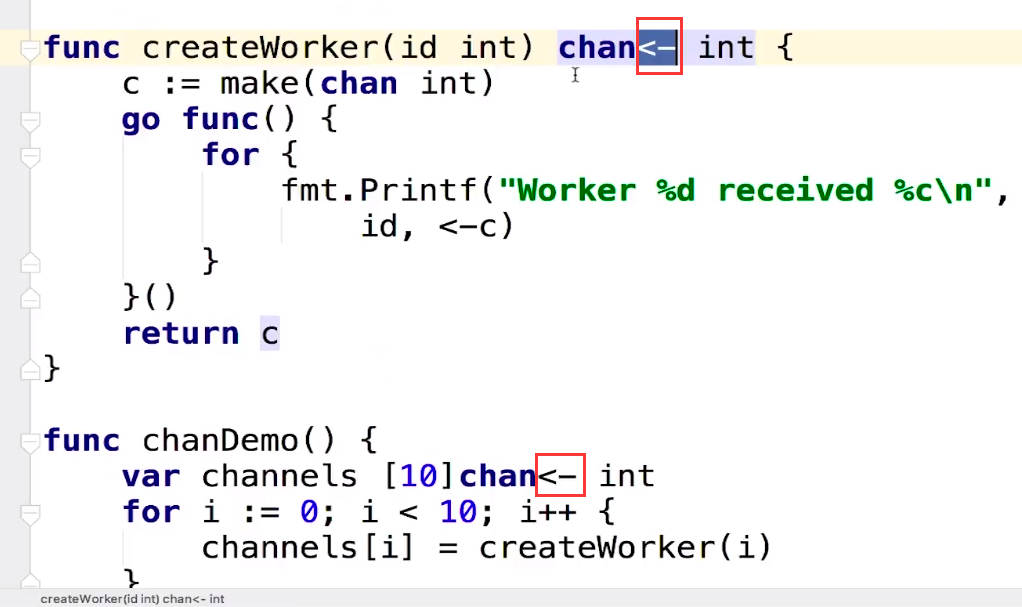
用一个专门的函数创建chan并返回，goroutine放到创建chan的函数里面去了

主程序用for循环接收chan变量

## 六

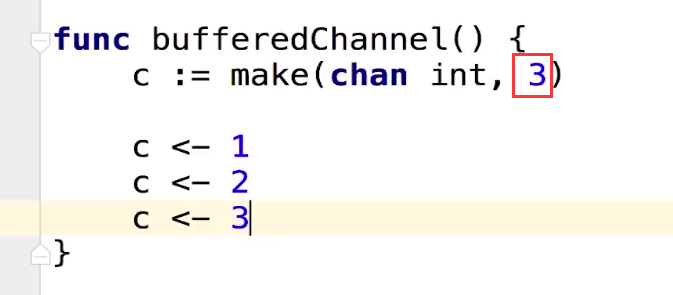
**chan<- 定义chan只能发送数据**

**<-chan 定义chan只能接收数据**



在五的基础上定义了createWorker的返回值chan只能发送数据

# Buffered Channel

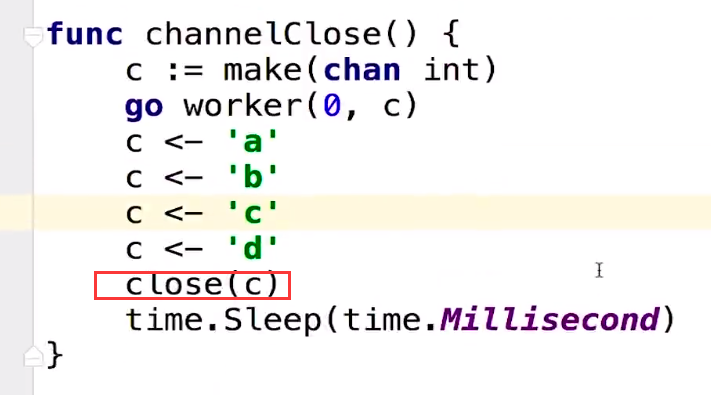


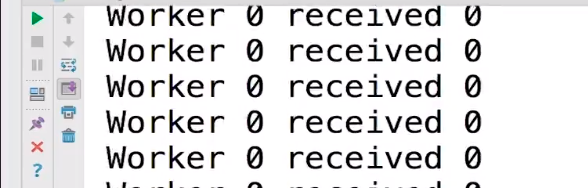
**可以给chan加一个缓存，在提升性能上有一定的优势**

这里可以先缓存3个数据不需要接收

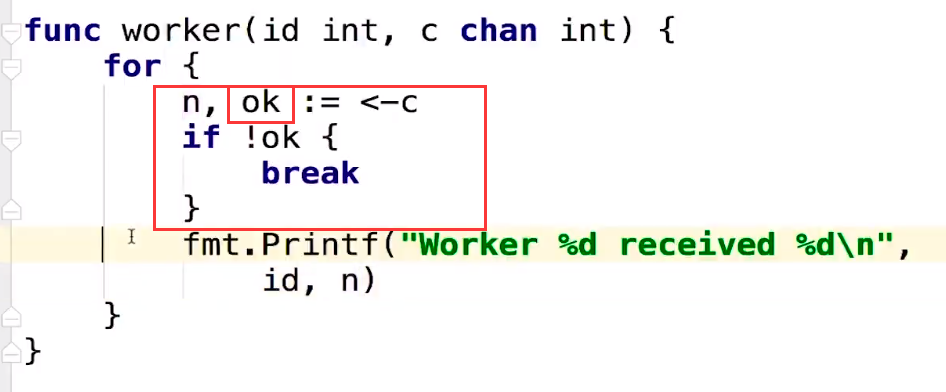
# Channel Close

**永远是由发送方close**





Close之后接收方还会继续收到空数据，需要接收方判断是否关闭



或

