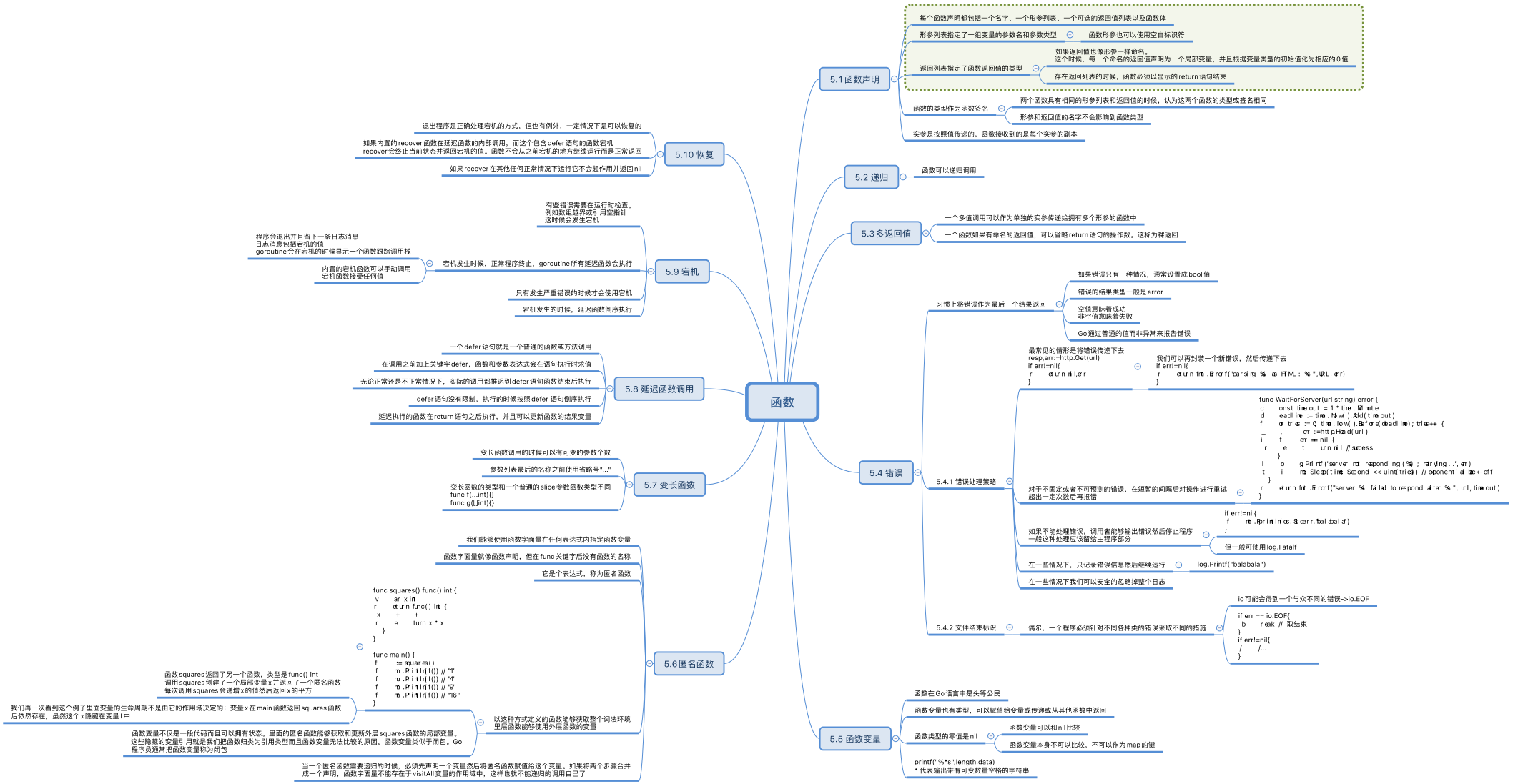
函数
5.1 函数声明
每个函数声明都包括一个名字、一个形参列表、一个可选的返回值列表以及函数体
形参列表指定了一组变量的参数名和参数类型
函数形参也可以使用空白标识符
返回列表指定了函数返回值的类型
如果返回值也像形参一样命名。
这个时候，每一个命名的返回值声明为一个局部变量，并且根据变量类型的初始值化为相应的0值
存在返回列表的时候，函数必须以显示的return语句结束
函数的类型作为函数签名
两个函数具有相同的形参列表和返回值的时候，认为这两个函数的类型或签名相同
形参和返回值的名字不会影响到函数类型
实参是按照值传递的，函数接收到的是每个实参的副本
5.2 递归
函数可以递归调用
5.3 多返回值
一个多值调用可以作为单独的实参传递给拥有多个形参的函数中
一个函数如果有命名的返回值，可以省略return语句的操作数。这称为裸返回
5.4 错误
习惯上将错误作为最后一个结果返回
如果错误只有一种情况，通常设置成bool值
错误的结果类型一般是error
空值意味着成功
非空值意味着失败
Go通过普通的值而非异常来报告错误
5.4.1 错误处理策略
最常见的情形是将错误传递下去
resp,err:=http.Get(url)
if err!=nil{
return nil,err
}
我们可以再封装一个新错误，然后传递下去
if err!=nil{
return fmt.Errorf("parsing %s as HTML: %v",URL, err)
}
func WaitForServer(url string) error {
const timeout = 1 * time.Minute
deadline := time.Now().Add(timeout)
for tries := 0; time.Now().Before(deadline); tries++ {
_, err := http.Head(url)
if err == nil {
return nil // success
}
log.Printf("server not responding (%s); retrying...", err)
time.Sleep(time.Second << uint(tries)) // exponential back-off
}
return fmt.Errorf("server %s failed to respond after %s", url, timeout)
}
对于不固定或者不可预测的错误，在短暂的间隔后对操作进行重试
超出一定次数后再报错
如果不能处理错误，调用者能够输出错误然后停止程序
一般这种处理应该留给主程序部分
if err!=nil{
fmt.Fprintln(os.Stderr,"balabala")
}
但一般可使用log.Fatalf
在一些情况下，只记录错误信息然后继续运行
log.Printf("balabala")
在一些情况下我们可以安全的忽略掉整个日志
5.4.2 文件结束标识
偶尔，一个程序必须针对不同各种类的错误采取不同的措施
io可能会得到一个与众不同的错误->io.EOF
if err == io.EOF{
break // 取结束
}
if err!=nil{
//...
}
5.5 函数变量
函数在Go语言中是头等公民
函数变量也有类型，可以赋值给变量或传递或从其他函数中返回
函数类型的零值是nil
函数变量可以和nil比较
函数变量本身不可以比较，不可以作为map的键
printf("%*s",length,data)
* 代表输出带有可变数量空格的字符串
5.6匿名函数
我们能够使用函数字面量在任何表达式内指定函数变量
函数字面量就像函数声明，但在func关键字后没有函数的名称
它是个表达式，称为匿名函数
func squares() func() int {
var x int
return func() int {
x++
return x * x
}
}
func main() {
f := squares()
fmt.Println(f()) // "1"
fmt.Println(f()) // "4"
fmt.Println(f()) // "9"
fmt.Println(f()) // "16"
}
以这种方式定义的函数能够获取整个词法环境里层函数能够使用外层函数的变量
函数squares返回了另一个函数，类型是func() int
调用squares创建了一个局部变量x并返回了一个匿名函数
每次调用squares会递增x的值然后返回x的平方
我们再一次看到这个例子里面变量的生命周期不是由它的作用域决定的：变量x在main函数返回squares函数后依然存在，虽然这个x隐藏在变量f中
函数变量不仅是一段代码而且可以拥有状态。里面的匿名函数能够获取和更新外层squares函数的局部变量。
这些隐藏的变量引用就是我们把函数归类为引用类型而且函数变量无法比较的原因。函数变量类似于闭包。Go程序员通常把函数变量称为闭包
当一个匿名函数需要递归的时候，必须先声明一个变量然后将匿名函数赋值给这个变量。如果将两个步骤合并成一个声明，函数字面量不能存在于visitAll变量的作用域中，这样也就不能递归的调用自己了
5.7 变长函数
变长函数调用的时候可以有可变的参数个数
参数列表最后的名称之前使用省略号"..."
变长函数的类型和一个普通的slice参数函数类型不同
func f(...int){}
func g([]int){}
5.8 延迟函数调用
一个defer语句就是一个普通的函数或方法调用
在调用之前加上关键字defer，函数和参数表达式会在语句执行时求值
无论正常还是不正常情况下，实际的调用都推迟到defer语句函数结束后执行
defer语句没有限制，执行的时候按照defer语句倒序执行
延迟执行的函数在return语句之后执行，并且可以更新函数的结果变量
5.9 宕机
有些错误需要在运行时检查，
例如数组越界或引用空指针
这时候会发生宕机
宕机发生时候，正常程序终止，goroutine所有延迟函数会执行
程序会退出并且留下一条日志消息
日志消息包括宕机的值
goroutine会在宕机的时候显示一个函数跟踪调用栈
内置的宕机函数可以手动调用
宕机函数接受任何值
只有发生严重错误的时候才会使用宕机
宕机发生的时候，延迟函数倒序执行
5.10 恢复
退出程序是正确处理宕机的方式，但也有例外，一定情况下是可以恢复的
如果内置的recover函数在延迟函数的内部调用，而这个包含defer语句的函数宕机
recover会终止当前状态并返回宕机的值。函数不会从之前宕机的地方继续运行而是正常返回
如果recover在其他任何正常情况下运行它不会起作用并返回nil