GO语言学习笔记

变量：一段或多段用来存储数据的内存，类型决定了变量内存的长度和存储格式。简短模式并不总是重新定义变量，可能是部分退化的赋值操作（P19），前提：最少有一个新变量被定义且必须是同一作用域，err变量可以重复使用。

常量：不使用也不会引起编译错误，在常量组中如不指定类型和初始化值，则与上一行非空常量右值相同（P25）。

枚举：可借助iota标识符实现一组自增常量来实现枚举类型（P25）。

常量与变量的区别：变量在运行期分配存储内存，常量通常会被编译器在预处理阶段直接展开，作为指令数据使用。

引用类型：slice、map、channel，使用make函数创建。

未命名类型：数组、切片、字典、通道等类型与具体元素类型或长度属性有关，可用type为其提供具体名称（P35）。

25个关键字：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| break | default | func | interface | select |
| case | defer | go | map | struct |
| chan | else | goto | package | switch |
| const | fallthrough | if | range | type |
| continue | for | import | return | var |

指针：是一个实体，用来保存地址的整型变量（P42）。在 Go 语言中，通过指向结构体的指针获取结构体中的值时，可以使用点操作符直接访问字段，而无需显式地使用 \* 运算符进行解引用。这是因为 Go 语言在访问指针指向的结构体字段时，会自动解引用指针并访问字段。

函数：结构化变成的最小模块单元，函数只能判断其是否为nil（P59）。形参是指函数定义中的参数，实参则是函数调用时所传递的参数。

变参：本质上是一个切片。只能接受一个到多个同类型参数，且必须放在列表尾部（P66）。

闭包：在其词法上下文中引用了自由变量的函数，或者是函数和其引用的环境的组合体（P71）。

字符串：不可变字节序列，本身是一个复合结构，可以使用len(str)得到字符串的长度。其默认值为””，允许以索引号访问字节数组，但不能获取元素地址。以切片语法[起始:结束]返回字符串时，其内部依旧指向原字节数组（P87）。使用for循环遍历字符串分为byte和rune两种，其中byte是for i:=0;i<len(s);i++ s[i]。rune则是for I,j := range s。byte是uint8别名通常表示ASCII字符，rune是int32别名，表示Unicode包含ASCII，可以表示中文，使用单引号默认为rune。三者可以直接转换，string(s) byte(s) rune(s)。

数组：长度是类型组成部分，元素类型相同，但长度不同的数组不属于同一类型。c:=[...]int{5,3:10}结果为[5 0 0 10]。内置len和cap都返回第一维度长度，持== ！=操作。

指针：指针数组指元素为指针类型的数组，数组指针是取数组变量的地址。a为数组，p:=&a获得数组a的指针，但p[1]是获取数组第二个元素的值而不是地址。Go数组是值类型，复制恶化传参操作都会复制整个数组数据，可以使用指针嚯切片避免数据复制。

切片：内部通过指针引用底层数组，设定相关属性将数据读写操作限定在指定区域内（P100）。不支持比较操作，可获取元素地址，但不能像数组那样直接用指针访问元素内容，想要获取则需要\*p[0]。切片只是很小的机构体对象，用来代替数组传参，避免复制开销。make函数允许在运行期动态指定数组长度，绕开了数组类型必须使用编译期常量的限制。

reslice:将切片看做[cap]slice数据源，据此创建新切片对象。不能超出cap，但不受len限制（P105）。新建切片对象依旧指向原底层数组，修改对所有关联切片可见。P106利用reslice实现栈。

append：向切片尾部添加数据，返回新的切片对象。append并未向原数组写入部分数据（P108）。

copy：在两个切片对象间复制数据，允许指向同一底层数组，允许目标区间重叠，最终所复制长度以较短的切片长度为准（P109）。

字典：（哈希表）在创建字典时，不需要指定初始容量，因为字典会根据实际的键值对动态地进行内存分配和管理，是引用类型，使用make函数或初始化表达语句来创建。作为无序键值对集合，字典要求key必须是支持相等运算符（==、！=）的数据，比如数字、字符串、指针、数组、结构体，以及对应接口类型。（P110）delete删除键值对时，不存在的键值返回零值不出错。对字典进行迭代，每次返回的键值次序都不相同。len返回当前键值对数量，不能直接修改字典value成员。即当m为字典，不能使用m[1].age+=1。正确做法为u:=m[1] u.age+=1 m[1]=u。当value是指针类型时可以使用m[1].age++.

安全：在迭代期间删除或新增键值是安全的。

性能：字典对象本身就是指针包装，传参时无须再次取地址。在创建时预先准备足够空间有助于提升性能，减少扩张时的内存分配和重新哈希操作。