**Go语言实践2**

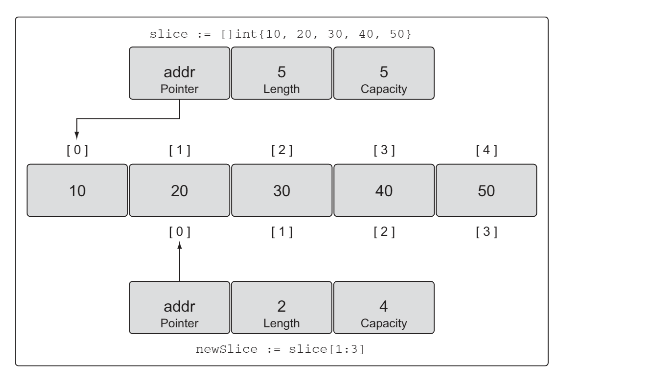
**数组**

1. 固定长度；初始化和使用上类似C++数组，注意多维数组初始化方法。
2. 赋值(=)和函数传参时会发生内存拷贝，性能较低，因此推荐用数组指针传递（没有引用）

**切片**

为了解决数组使用上的一些困难，比如无法引用传参（指针容易出错），无法增加长度等，提出了切片的概念

1. 切片其实并不包含实际数据，它由指向数组某个位置的起始指针，长度和容量三个变量组成一个类，并包含了内置函数如len append。下面这个图是理解切片的关键：数组部分是动态申请的空间大小，切片通过指针指向自己对应的起始位置，这个位置无法更改，并且确定了长度，容量则由数组的容量确定。



1. 切片初始化：几个要素，类型，长度，容量，可以类比C++中的vector
2. slice:=make([]T, length, capacity) 此时会创建一个新切片，并动态申请capacity长度的数组。
3. slice:= []T{a, b, c} 同上
4. slice\_new := slice[e:f] 左闭右开，长度为f-e，容量则参考上图进行计算。
5. slice\_new:=slice[e:f:g] 长度为f-e 容量为g-e 左闭右开。
6. var slice []T nil切片 指针指向nil 同时长度和容量都是0
7. 获取长度和容量的内置函数：len(slice) cap(slice)
8. append函数
9. slice\_new=append(slice, varT) 向slice增加一个元素。
10. 如果slice指向的底层数组capacity还没有被超过，那么等于把相应位置的值直接覆盖成varT,slice\_new和slice还会指向同一个底层数组。
11. **如果已经超过了底层数组的capacity，那么会新建一个底层数组，把slice中指针开始到capacity的值都复制一遍，并在最后追加varT，此时slice\_new和slice就指向不同的底层数组，这里一定要注意在函数里append切片时，必须把切片返回，否则外部的切片可能不会被修改**。
12. 和vector的内存动态申请机制比较像。
13. slice\_new=append(slice\_a, slice\_b...) 将slice\_b中全部元素复制追加到slice\_a后面
14. slice\_new=append(slice\_a, slice\_b[c:d]) 同理
15. range函数 用于获取下标和元素的复制，方便for循环使用.
16. for index, value := range slice { any code }
17. 当然也可以继续用 for index:= 0; index<len(slice); index++ { any code }
18. 注意value是一个副本（有内存拷贝开销） 改value不会影响slice中的底层数组，如果要改则必须要用slice[index]=xxx
19. 多维切片 也就是切片里的每个元素都是切片。
20. 函数变量尽量使用切片传递，减少传参成本 func foo(input\_slice []T) []T {any code}
21. Slice用sort排序

agelist := []int{1,2,7,3,9,5}

sort.Slice(agelist, func(i, j int), bool{

return agelist[i] < agelist[j] //从小到大

})

**映射**

1. 基本等同于C++中的std:unordered\_map 底层实现为hash
2. 声明和初始化语法
3. dict:=make(map[Type\_Key]Type\_Value)
4. dict:=map[Type\_Key]TypeValue {key1:value1, key2:value2....}
5. nil映射 var dict:=map[Type\_Key]Type\_Value
6. 类似find函数判断对应的值是否存在 value, exist := dict[key]或者value:=dict[key]默认取出来的值为value类型的零值
7. 迭代 同样适用range for key, value := range dict {any codes}
8. 删除某个key-value对 delete(dict, key)
9. **函数传参的时候为引用传参！(go默认都是值传递，复制的)**

**练习**

1. Golang中没有内置的HashSet，尝试用map实现hashset，增删改查，以及两个hashset的求并集(union)交集(join)差集(diff)等操作。
2. Leetcode中有array/set/map标签的题，以及递归回溯的题逐步尝试用golang实现一遍。
   1. Golang没有while，用for + break代替