Go语言容器化编程

一、课前准备

> 说明：提前需要让学生做的课前准备，比如环境安装部署、工具安装、插件安装等，需要学生提前做的都放这，并且给出下载链接或信息源；

1. Golang开发环境环境安装就绪；
2. Golang-IDE开发环境安装就绪；
3. 练习Go语言基础语法代码；

二、课堂主题

> 说明：本堂课的总体概述，明确课堂主题和主体；

本节主要介绍Go语言当中的容器编程方法，涉及到数组、切片以及map，涉及到他们的区别以及如何遍历各个容器。

三、课堂目标

> 说明：主要是让学生了解，学了本堂课后，能达到的一个期望值，要量化；

1. 掌握Go语言多维数组的用法；
2. 掌握Go语言切片的用法；
3. 掌握Go语言切片与数组的区别；

4. 掌握Go语言map的用法；
5. 掌握容器遍历的方法；

四、知识点

1. 数组

Go语言当中，可以将一组类型相同的数据存放在一个数组当中，数组中元素的类型可以是Go语言原生类型，也可以是自定义类型（struct）。声明方式可以参考变量：

```
var variable_name [SIZE] variable_type
```

具体可以参考几个例子：

```
var sa [10] int64 //定义一个10个int64类型的数组
var ss [3] string = [3]string{"lily","lucy","lilei"}//定义3个长度的字符数组
```

Go语言的fmt包功能很强大，它可以直接打印各种变量，无论是容器，还是结构体，都可以直接打印。示例如下：

```
package main

import (
  "fmt"
)

func main()
  var a1 [5]int = [5]int{1, 2, 3, 4}
  fmt.Println(a1)
  a1[4] = 6
  fmt.Println(a1)
  s1 := [4]string{"lily","lucy","lilei"} //元素个数不能超过数组个数
  fmt.Println(s1)
}
```

执行结果如下：

```
localhost:day01 teacher$ go run 01-array.go
[1 2 3 4 0]
[1 2 3 4 6]
[lily lucy lilei ]
```

注意：数组是固定长度的，不可以越界！

在Go语言当中，同样可以定义二维数组，具体参考下面例子即可。

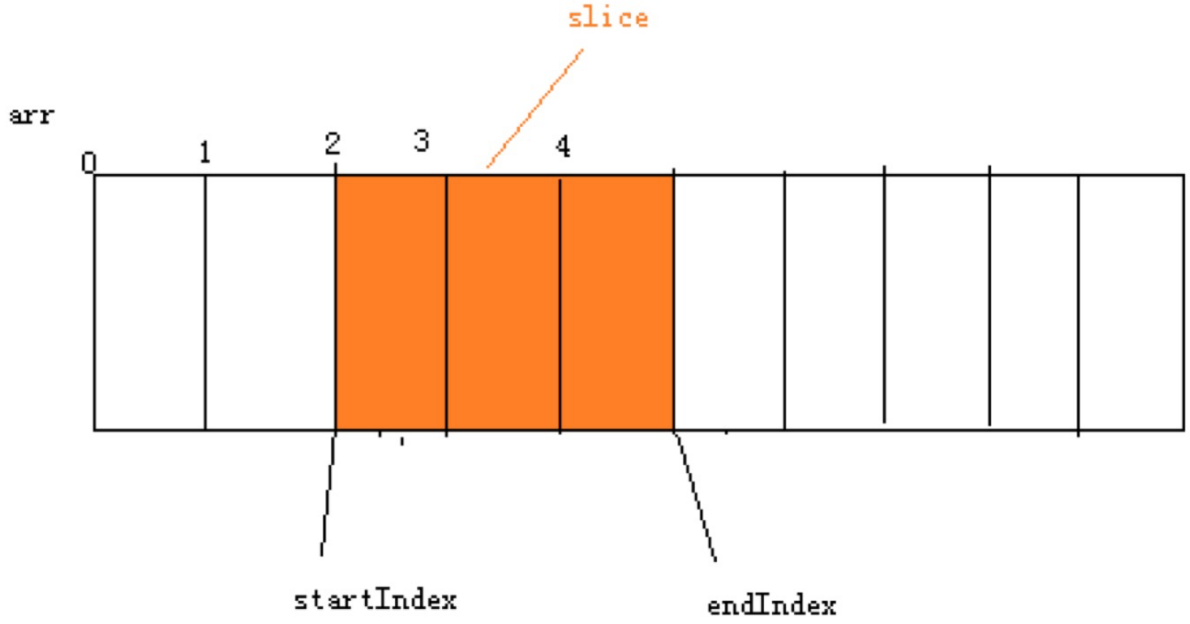
```
package main

import (
  "fmt"
)

func main() {

  //Go语言当中的二维数组,可以理解为3行4列
  a2 := [3][4]int{
    {0, 1, 2, 3},   /*  第一行索引为 0 */
    {4, 5, 6, 7},   /*  第二行索引为 1 */
    {8, 9, 10, 11}, /* 第三行索引为 2 */
  }
  //注意上述数组初始化的逗号
  fmt.Println(a2)
  //如何遍历该数组? 可以写2层for循环搞定
  for i := 0; i < 3; i++ {
    for j := 0; j < 4; j++ {
      fmt.Printf("i = %d, j = %d, val = %d\n", i, j, a2[i][j])
    }
  }
}
```

执行结果如下：

```
localhost:day01 teacher$ go run 02-array2.go
[[0 1 2 3] [4 5 6 7] [8 9 10 11]]
i = 0, j = 0, val = 0
i = 0, j = 1, val = 1
i = 0, j = 2, val = 2
i = 0, j = 3, val = 3
i = 1, j = 0, val = 4
i = 1, j = 1, val = 5
i = 1, j = 2, val = 6
i = 1, j = 3, val = 7
i = 2, j = 0, val = 8
i = 2, j = 1, val = 9
i = 2, j = 2, val = 10
i = 2, j = 3, val = 11
```

2. 切片

由于数组是固定大小的，这在使用上缺少一定便利，因此Go语言又提供了切片类型，乍一看切片与数组没有区别，只不过它的大小是可以扩充的，也就是说可以把切片理解成动态数组，至于为什么叫切片，可以脑补一下面包片（长面包想象成一个数组，切片是从其某段切下来）。



切片中有2个概念：长度和容量。

- 长度 指被赋过值的最大下标+1
- 容量 指切片能容纳的最多元素个数

切片可以用array|slice[start:end]的方式进行截取，得到新的切片，其中start和end代表下标位置，都可以省略，start省略代表从头开始，end省略代表直到末尾。

```
package main

import (
  "fmt"
)

func main() {
  a1 := [5]int{1, 2, 3, 4, 5} // a1 是一个数组
  s1 := a1[2:4]               //定义一个切片
  fmt.Println(a1)
  fmt.Println(s1)
  s1[1] = 100 //切片下标不能超过
  fmt.Println("after---------")
  fmt.Println(a1)
  fmt.Println(s1)
}
```

执行结果如下：

```
localhost:day01 teacher$ go run 03-slice.go
[1 2 3 4 5]
[3 4]
after---------
[1 2 3 100 5]
[3 100]
```

通过上述例子，我们可以得出2个知识点：

1. 切片start：end是前闭后开，也就是实际截取下标是start到end-1
2. 切片是引用类型，对切片的修改会影响对应的数组

与切片相关的内建函数：

- len 计算切片长度

```
len(s)
```

- 计算切片容量

```
cap(s)
```

- append 向切片追加元素

```
var s1 []T
append(s1,T)
```

- make 可以创建切片

```
make([]T, length, capacity) //capacity 可以省略，默认与len一致
```

- copy 复制切片

```
copy(s2,s1) //将s1内容拷贝到s2，此时s1与s2是独立的，修改互不干预
```

来看看一段具体代码:

```
package main

import (
  "fmt"
)

func main() {
  var s1 []int       //定义切片s1
  s1 = append(s1, 1) //追加，注意s1必须接收追加结果
```

```
  s1 = append(s1, 2)
  s1 = append(s1, 3, 4, 5) //可以一次追加多个
  printSlice(s1)
  s2 := make([]int, 3)
  printSlice(s2)
  s2 = append(s2, 4) //当超过容量的时候，容量会以len*2的方式自动扩大
  printSlice(s2)
}

func printSlice(s []int) {
  fmt.Printf("len = %d, cap = %d, s = %v\n", len(s), cap(s), s)
}
```

运行结果：

```
localhost:day01 teacher$ go run 04-slice2.go
len = 5, cap = 6, s = [1 2 3 4 5]
len = 3, cap = 3, s = [0 0 0]
len = 4, cap = 6, s = [0 0 0 4]
```

通过上述例子我们看到，当切片容量不够时，追加不会报错，而会扩大容量，扩大容量默认采用len*2的数据。根据多个例子编写，相信大家也发现了，Go语言默认都会对变量进行初始化，这一点很友好！

3. map（键值对）

Go语言同样提供了map这样的容器，map可以存放无序的键值对，map是一种集合，但注意它是无序的。map需要使用make来构造，否则它是一个nil-map，无法存放键值对。

```
var map_variable map[key_data_type]value_data_type
map_variable = make(map[key_data_type]value_data_type)

或

map_variable := make(map[key_data_type]value_data_type)
```

来段代码：

```
package main

import "fmt"

func main() {
  countryCapitalMap := make(map[string]string)

  // map插入key - value对,各个国家对应的首都
```

```
  countryCapitalMap["France"] = "Paris"
  countryCapitalMap["Italy"] = "Roma"
  countryCapitalMap["China"] = "BeiJing"
  countryCapitalMap["India "] = "New Delhi"

  fmt.Println(countryCapitalMap["China"])
  //当key不存在时，直接打印不太优雅，可以使用下面的方法
  val, ok := countryCapitalMap["Japan"]
  if ok {
    fmt.Println("Japan's capital is", val)
  } else {
    fmt.Println("Japan's capital not in map")
  }

}
```

在取map的值时可以用一个变量接收，也可以增加一个指示变量，来判断key值是否真实存在。下面还有一个问题，如何对map进行遍历呢？这需要用到range关键字，它可以让我们优雅的遍历Go语言各种容器，当然包括map。具体用法可以看代码：

```
package main

import "fmt"

func main() {
  countryCapitalMap := make(map[string]string)

  // map插入key - value对,各个国家对应的首都
  countryCapitalMap["France"] = "Paris"
  countryCapitalMap["Italy"] = "Roma"
  countryCapitalMap["China"] = "BeiJing"
  countryCapitalMap["India "] = "New Delhi"

  //遍历map
  for k, v := range countryCapitalMap {
    fmt.Println(k, "'s capital is", v) //k,v分别是map的key和val
  }
  //遍历数组，如果不想获得
  a := []int{10, 20, 30, 40, 50}
  for k, v := range a {
    fmt.Printf("a[%d]=%d\n", k, v) //k代表数组下标，v代表该元素值
  }

}
```

执行结果：

```
localhost:day01 teacher$ go run 06-range.go
China 's capital is BeiJing
India  's capital is New Delhi
France 's capital is Paris
Italy 's capital is Roma
a[0]=10
a[1]=20
a[2]=30
a[3]=40
a[4]=50
```

如果不想获得k或v的值时，可以用_占位，这个语法在其他多变量赋值时同样适用。

```
_, v := range countryCapitalMap //只取v的值
```

4. 小结

- make可以构造容器
- range可以优雅遍历容器
- 切片可以让编程更灵活

五、拓展点

> 说明：
>
> 1. 典型面试题、笔试题；
> 2. 新技术 or 经验分享；
> 3. 未来计划、行业趋势分享；

六、总结

本堂主要知识点如下：

- 数组、多维数组如何使用
- 切片与数组的区别
- 切片的使用
- 内建函数的使用
- map的使用
- 容器的遍历问题

七、大作业

> 说明：
>
> 1. 给出明确的作业要求，形成文字、图片或测试题等形式；

2. 给出明确的解答方式；
3. 频次低，可设计为：1次/周 或 1次/2周；

可从下面的检测题题库任意选择。

八、集中答疑

> 说明：完成本堂课所有知识点的讲解后，由学员集中提问，讲师逐一解答；

九、检测题

> 说明：
>
> 1. 针对本堂课，设计5-10个题，由学员课下完成（课下刷题）；
> 2. 出题范围可从检查点里挑选，可以是面试题或笔试题；
> 3. 题型要求是 单选、多选、判断、填空中的一种或多种；
> 4. 频次高，原则上每次课都要有检测题；

十、下节课预告