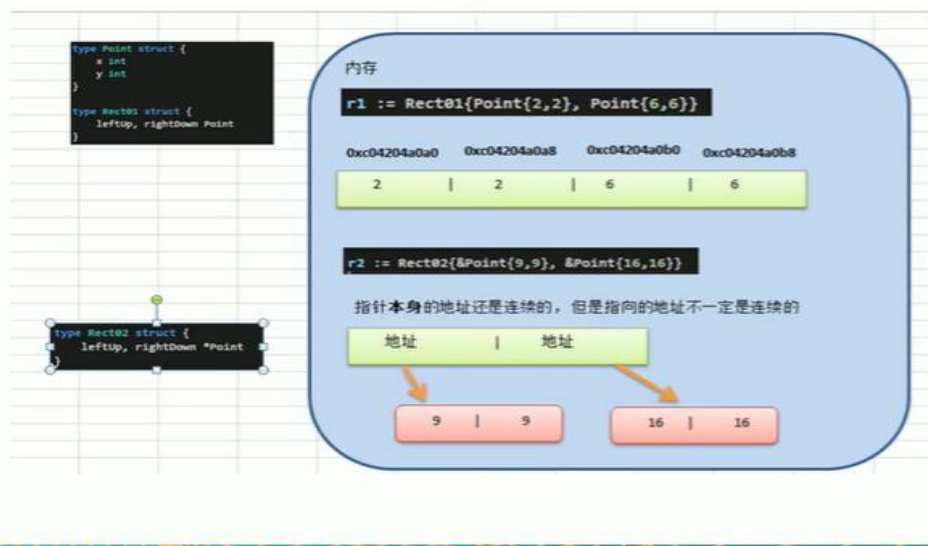


1) 结构体所有字段在内存中是连续的

1) 结构体的所有字段在内存中是连续的【举例】



2)

结构体的注意事项和使用细节

2) 结构体是用户单独定义的类型，和其它类型进行转换时需要有**完全相同的字段**(名字、个数和类型)

```
type A struct {
    Num int
}

type B struct {
    Num int
}

func main() {
    var a A
    var b B
    b = B(a) // 这里正确吗?
}
```

3)

结构体

结构体的注意事项和使用细节

3) 结构体进行type重新定义(相当于取别名)，Golang认为是新的数据类型，但是**相互间可以强转**

```
type Student struct {
    Name string
    Age int
}

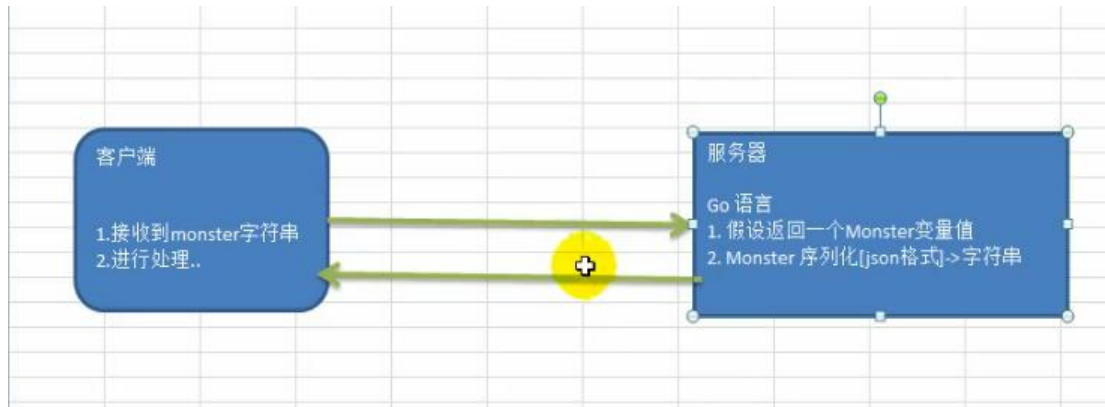
type Stu Student

func main() {
    var stu1 Student
    var stu2 Stu
    stu2 = stu1 // 正确吗?
    fmt.Println(stu1, stu2)
}
```

```
type integer int

func main() {
    var i integer = 10
    var j int = 20
    j = i // 正确吗?
    fmt.Println(i, j)
}
```

4) struct 的每一个字段上, 可以写上一个 tag, 改 tag 可以通过反射机制获取, 常用场景就是序列化和反序列化。



```
type Monster struct{
    Name string `json:"name"` // `json:"name"` 就是 struct tag
    Age int `json:"age"`
    Skill string `json:"skill"`
}
```

```
//1. 创建一个Monster变量
monster := Monster{"牛魔王", 500, "芭蕉扇~"}

//2. 将monster变量序列化为 json格式字符串
// json.Marshal 函数中使用反射, 这个讲解反射时, 我会详细介绍
jsonStr, err := json.Marshal(monster)
if err != nil {
    fmt.Println("json 处理错误 ", err)
}
fmt.Println("jsonStr", string(jsonStr))
}
```