多属性对象的表示问题

张华

64174234@qq.com

内容

1.1 多属性对象的一体化描述问题
1.2 多属性对象类型的定义
1.3 多属性对象的定义与使用
1.4 对象的继承机制——匿名组合
1.5 总结
1.6 思考

1.1多属性对象的一体化描述问题

- 想在程序里表示一个人（姓名、年龄、性别、身高、体重……），怎么表示？
  - var name string
  - var age uint
  - var sex byte
  - …
- 想表示多个人呢？
  - 定义多个数组？(有些搞笑了)

- 应该采用面向对象的方法

1.2多属性对象类型的定义

- Go通过结构体来刻画一个多属性对象
  - 首先定义对象的类型，这里定义了一个Person类型，有三个属性，分别为Name、Age和Sex。

```
type Person struct { //定义了一个名为Person的结构体类型
    Name string
    Age  uint
    Sex  byte
}
```

  - 对象类型定义的形式为：

```
type <name> struct {
    ...
}
```

1.3多属性对象的定义及使用

```go
package main
import "fmt"
type Person struct { //定义了一个名为Person的结构体类型
    Name string
    Age  uint
    Sex  byte
}
/**Stringer是一个可用字符描述自己的类型
是一个普遍存在的接口，定义在fmt包中：
type Stringer struct {
    String() string
}
*/
func (p Person) String() string { //具体实现fmt包中的Stringer接口
    /*Sprintf 将参数列表a填写到格式控制串format的占位符中*/
    /*func Sprintf(format string, a ...interface{}) string*/
    return fmt.Sprintf("\n%v(%c,%v years)\n", p.Name, p.Sex, p.Age)
}
func main() {
    F1 := Person{"言承旭", 37, 'm'}
    F2 := Person{"周渝民", 33, 'm'}
    F3 := Person{"吴建豪", 36, 'm'}
    F4 := Person{"朱孝天", 35, 'm'}
    fmt.Println(F1, F2, F3, F4)
}
```

```
D:\go-dev\src>go run struct1.go
言承旭(m,37 years)
周渝民(m,33 years)
吴建豪(m,36 years)
朱孝天(m,35 years)
```

1.3多属性对象的定义及使用(续)

- 结构体对象使用点号来访问其成员
  - 对象名.成员名

```
1  package main
2
3  import "fmt"
4
5  type Person struct {
6      Name string
7      Age  uint
8      Sex  byte //0代表男，1代表女
9  } //定义了一个名为Person的结构体类型
10 func main() {
11     F1 := Person{}
12     F1.Name = "言承旭"
13     F1.Age = 37
14     F1.Sex = 0
15     fmt.Println(F1)
16 }
```

1.3多属性对象的定义及使用(续)

- 通过结构体指针定义对象及访问对象成员

```
D:\go-dev\src>go run struct3-1.go
{言承旭 37 0}
&{言承旭 37 0}
```

```go
package main

import "fmt"

//Go中的结构体struct地位相当于其他语言的类class
type Person struct {
    Name string
    Age  uint
    Sex  byte //0代表男，1代表女
} //定义了一个名为Person的结构体类型

func main() {
    F1 := &Person{}
    F1.Name = "言承旭" //没有C中的→，依旧用.号通过指针实现间接访问
    F1.Age = 37
    F1.Sex = 0
    fmt.Println(*F1)
    fmt.Println(F1)
}
```

1.3多属性对象的定义及使用(续)

- 通过’构造函数’初始化对象

```
1  package main
2  import "fmt"
3  //Go中的结构体struct地位相当于其他语言的类class
4  type Person struct {
5      Name string
6      Age  uint
7      Sex  byte //0代表男，1代表女
8  } //定义了一个名为Person的结构体类型
9  /**
10 Go中没有构造函数的概念，
11 对象的创建和初始化可交由一个全局的创建函数来完成，
12 以NewXXX来命名，表示'构造函数'
13 **/
14 func NewPerson(name string, age uint, sex byte) *Person {
15     return &Person{name, age, sex}
16 }
17 func main() {
18     F1 := NewPerson("言承旭", 37, 0)
19     fmt.Println(*F1)
20 }
```

```
D:\go-dev\src>go run struct3-2.go
{言承旭 37 0}
```

1.3多属性对象的定义及使用(续)

- 通过字面值初始化一个对象

```go
package main

import "fmt"

//Go中的结构体struct地位相当于其他语言的类class
type Person struct {
    Name string
    Age  uint
    Sex  byte //0代表男，1代表女
} //定义了一个名为Person的结构体类型
func main() {
    F1 := &Person{Name: "jow", Age: 37} //指定初始化哪些成员
    F1.Sex = 0
    fmt.Println(*F1)
}
```

```
D:\go-dev\src>go run struct4.go
{jow 37 0}
```

1.3多属性对象的定义及使用(续)

- 通过new关键字建立并初始化一个对象

```go
package main

import "fmt"

//Go中的结构体struct地位相当于其他语言的类class
type Person struct {
    Name string
    Age  uint
    Sex  byte //0代表男，1代表女
} //定义了一个名为Person的结构体类型
func main() {
    F1 := new(Person) //新建一个Person指针对象，内容空
    F1.Name = "言承旭"
    F1.Age = 37
    F1.Sex = 0
    fmt.Println(*F1)
}
```

```
D:\go-dev\src>go run struct5.go
{言承旭 37 0}
```

1.3多属性对象的定义及使用(续)

- 匿名结构与匿名属性
  - 匿名结构——没有名称的对象结构
  - 匿名属性——属性没有名称,只给出属性类型
    - 具有匿名属性的对象的初始化，必须严格按照顺序进行，否则会报告类型不匹配的错误

```
1  package main D:\go-dev\src>go run struct7.go
2                {言承旭 37 0}
3  import "fmt"
4
5  func main() {
6      F1 := struct { //定义了一个匿名结构，属性也是匿名的
7          string //姓名
8          uint   //年龄
9          byte   //0代表男，1代表女
10     }{"言承旭", 37, 0} //按照顺序初始化各个匿名字段
11     fmt.Println(F1)
12 }
```

1.3多属性对象的定义及使用(续)

- 具体匿名嵌套结构的对象定义及初始化

```go
package main
import "fmt"
type Person struct {
    Name     string
    Age      uint
    Sex      byte      //0代表男，1代表女
    Birthday struct { //成员Birthday是一个匿名结构
        Year, month, day int
    }
} //定义了一个名为Person的结构体类型
func main() {
    F1 := &Person{Name: "言承旭", Age: 37, Sex: 0} //用字面值初始化非匿名部分
    F1.Birthday.Year = 1977                        //匿名结构成员的一一赋值
    F1.Birthday.month = 8
    F1.Birthday.day = 1
    fmt.Println(*F1)
}
```

```
D:\go-dev\src>go run struct8.go
{言承旭 37 0 {1977 8 1}}
```

1.4对象的继承机制——匿名组合

- 匿名组合:声明的一个类(结构体)中包含了已经定义的其他类(结构体)或其他Go基本类型作为内置字段的情况.
  - 即其他类的类名作为一个类的成员,如Person中的Birth

```
1  package main
2
3  import "fmt"
4
5  type Birth struct {
6      Year, month, day int
7  }
8  type Person struct {
9      Name   string
10     Age    uint
11     Sex    byte  //0代表男，1代表女
12     Birth        //匿名组合字段
13 } //定义了一个名为Person的结构体类型
14 func main() {
15     F1 := &Person{Name: "言承旭", Age: 37, Sex: 0, Birth: Birth{1977, 8, 1}}
16     fmt.Println(*F1)
17 }
```

当嵌入结构作为匿名字段的时候，在初始化的时候将结构类型当做一个属性名称来处理，如Birth

```
D:\go-dev\src>go run struct9.go
{言承旭 37 0 {1977 8 1}}
```

1.4对象的继承机制——匿名组合(续)

- “重载”
  - 相同字段采用最外层优先访问，类似于重载
  - em1.sex 访问的是 Employee 中最外层的 sex
  - em1.Person.sex 访问的是 Employee 中 Person 中的 sex

```
D:\go-dev\src>go run struct6.go
{{rain 23 0} 5000 100 0}
```

```go
package main
import "fmt"
//Go中的结构体struct地位相当于其他语言的类class
type Person struct {
    Name string
    Age  uint
    Sex  byte //0代表男，1代表女
} //定义了一个名为Person的结构体类型
type Employee struct {
    Person //匿名字段
    salary int
    int          //用内置类型作为匿名字段
    Sex    byte //类似于重载
}
func main() {
    em1 := Employee{Person{"rain", 23, 0}, 5000, 100, 0}
    fmt.Println(em1)
}
```

1.5总结

- 对象类型的定义方式
  - type typeName struct {
    //...
    }
- 声明、初始化及访问
  - var varName typeName //①
  - varName := new(typeName) //②
  - varName := typeName{[初始化值]} //③
  - varName := &typeName{[初始化值]} //④
    - ①③返回tpeyName类型变量
    - ②④返回*tpeyName类型变量
    - ③④的[]可省略，若无初始化，默认为零值
  - 初始化可分为两种
    - 有序: typeName{value1, value2, ...} 必须一一对应
    - 无序: typeName{field1:value1, field2:value2, ...} 无需对应
  - 通过.来访问成员： varName.field

1.5总结

- 具有匿名组合的对象对其成员的控制方式有两种
  - 对象名.匿名类型名.成员名——比较绕
  - 对象名.成员名——符合面向对象语言的继承模式

```
D:\go-dev\src>go run struct10.go
{言承旭 37 0 {1977 8 1}}
```

```go
package main
import "fmt"
type Birth struct {
    Year, month, day int
}
type Person struct {
    Name  string
    Age   uint
    Sex   byte //0代表男，1代表女
    Birth      //匿名组合字段
} //定义了一个名为Person的结构体类型
func main() {
    F1 := &Person{}
    F1.Name = "言承旭"
    F1.Age = 37
    F1.Sex = 0
    F1.Year = 1977 //继承了Birth的Year属性，也可F1.Birth.Year = 1977
    F1.month = 8   //继承了Birth的Month属性，也可F1.Birth.month = 8
    F1.day = 1     //继承了Birth的Day属性，也可F1.Birth.day = 1
    fmt.Println(*F1)
}
```

1.6思考

- 编写程序，实现从键盘录入一个对象的各属性值
- 如果匿名字段和外层结构有相同字段该如何进行操作?请思考并尝试
- 请尝试如何将一个对象作为函数参数。
  - 传值/值语义的时候，能否改变对象自身的属性值？
  - 传地/引用语义的时候，能否改变对象自身的属性值？

Thank you very much

Any comments and suggestions are beyond welcome