Go 语言从入门到项目实战

第6章 结构体

6.1 类型

- 通过type或struct关键字定义新的类型，这种新的类型被称为自定义类型。
- 自定义类型格式：

  type new_type_name origin_type_name

- 自定义别名格式：

  type new_type_name = origin_type_name

6.1 类型

- 自定义类型示例：

```
func main() {
        type NewInt int
        var intNum NewInt=10
        fmt.Println("intNum的值为：",intNum,"，类型为：",reflect.TypeOf(intNum))
}
```

6.1 类型

- 自定义别名示例：

```
func main() {
        type NewInt = int
        var intNum NewInt=10
        fmt.Println("intNum的值为：",intNum,"，类型为：
",reflect.TypeOf(intNum))
}
```

6.2 结构体

- 结构体的声明格式

```
type struct_name struct {
        field_name definition
        field_name definition
        ...
}
```

6.2 结构体

- 结构体的声明示例

```
type Book struct {
    title string
    author string
    subject string
}
```

6.2 结构体

- 结构体的声明示例（省略字段名）

```
type AnonymousStruct struct {
        int
        string
}
```

6.2 结构体

- 示例化结构体格式

  var instance_name struct_type

6.2 结构体

- 示例化结构体示例

```
var bookOne Book
bookOne.title="书籍名称"
bookOne.author="作者名称"
bookOne.subject="书籍主题"
fmt.Println(bookOne)
fmt.Println(reflect.TypeOf(bookOne))
```

6.2 结构体

- 示例化结构体示例（使用键值对进行字段赋值时，键值对的赋值顺序和结构体中定义的顺序无需保持一致）

```
bookOne:=Book{
        subject: "书籍主题",
        title:"书籍名称",
        author: "作者名称"}
fmt.Println(bookOne)
fmt.Println(reflect.TypeOf(bookOne))
```

6.2 结构体

- 示例化结构体示例（使用键值对赋值时，允许省略某个字段）

```
bookOne:=Book{
        title: "书籍名称"}
fmt.Println(bookOne)
fmt.Println(reflect.TypeOf(bookOne))
```

6.2 结构体

- 示例化结构体示例（省略“键”，直接使用“值列表”的方式实现赋值）

```
bookOne:=Book{
        "书籍名称",
        "作者名称",
        "书籍主题"}
fmt.Println(bookOne)
fmt.Println(reflect.TypeOf(bookOne))
```

6.2 结构体

- 匿名结构体声明格式

```
variable_name:=struct {
        field_name definition
        field_name definition
        ...
}{
        field_name:value
        field_name:value
        ...
}
```

6.2 结构体

- 匿名结构体声明示例

```
bookOne:=struct{
        //结构体的声明
        title string
        author string
        subject string}{
        //结构体变量的赋值
        title:"图书名称",
        author:"作者名称",
        subject:"图书主题",
}
fmt.Println(bookOne)
fmt.Println(reflect.TypeOf(bookOne))
```

6.2 结构体

- 结构体的内存分配规律

```
func main() {
        testStruct :=struct{
                intA int8
                intB int8
                intC int8
                intD int8}{
                1,
                2,
                3,
                4}
        fmt.Println(&testStruct.intA)
        fmt.Println(&testStruct.intB)
        fmt.Println(&testStruct.intC)
        fmt.Println(&testStruct.intD)
}
```

6.2 结构体

- 指针类型的结构体变量

```
type Book struct {
        title string
        author string
        subject string
}
func main() {
        var bookTwo=new(Book)
        fmt.Println(&bookTwo)
        fmt.Println(bookTwo)
        fmt.Println(reflect.TypeOf(bookTwo))
}
```

6.2 结构体

- 指针类型的结构体变量

```
func main() {
        bookTwo:=&Book{}
        fmt.Println(&bookTwo)
        fmt.Println(bookTwo)
        fmt.Println(reflect.TypeOf(bookTwo))
        bookTwo.author="作者名称"
        bookTwo.title="图书名称"
        bookTwo.subject="图书主题"
        fmt.Println(bookTwo)
}
```

6.3 构造函数与方法

- 构造函数示例

```
type Cat struct {
        gender int
        color string
        name  string
}
func newCat(gender int,color string,name string) *Cat {
        return &Cat{
                gender: gender,
                color: color,
                name: name}
}
```

```
func main(){
        //通过newCat()函数生成Cat结构
        catOne:=newCat(0,"white","三酷猫")
        fmt.Println(catOne)
        fmt.Println(reflect.TypeOf(catOne))
}
```

6.3 构造函数与方法

- 方法的声明格式

```
func (receiver_name receiver_type) function_name(params)(return_types){
        //要执行的代码块
}
```

6.3 构造函数与方法

- 方法示例

```
type Cat struct {
        gender int
        color string
        name  string
}
func newCat(gender int,color string,name string) *Cat {
        return &Cat{
                gender: gender,
                color: color,
                name: name}
}
```

```
//吃饭
func (catInstance *Cat) eat(food string){
        fmt.Println(catInstance.name,"正在吃：",food)
}
//睡觉
func (catInstance *Cat) dream(){
        fmt.Println(catInstance.name,"睡得正香")
}
//喵喵叫
func (catInstance *Cat) mewing(){
        fmt.Println(catInstance.name,"喵喵喵")
}
```

6.3 构造函数与方法

```
func main(){
        //通过newCat()函数生成Cat结构
        catOne:=newCat(0,"white","三酷猫")
        fmt.Println(catOne)
        fmt.Println(reflect.TypeOf(catOne))
        //catOne执行吃饭动作
        catOne.eat("鱼")
        //catOne执行睡觉动作
        catOne.dream()
        //catOne执行喵喵叫动作
        catOne.mewing()
}
```

6.4 结构体的嵌套

- 结构体嵌套示例

```
//猫结构体
type Cat struct {
        name  string          //名字
        bodyInfo BodyInfo     //身体数据
}
//身体数据结构体
type BodyInfo struct{
        weight float64        //体重
        color string          //颜色
}
```

```
func main(){
        catOne:=Cat{
                name: "三酷猫",
                bodyInfo: BodyInfo{
                        weight: 10.5,
                        color: "白色"}}
        ffmt.Println("我的名字是：",catOne.name,"，体重：
",catOne.bodyInfo.weight,"，毛色：",
catOne.bodyInfo.color)
}
```

6.4 结构体的嵌套

- 匿名结构体嵌套示例

```
//猫结构体
type Cat struct {
        name  string          //名字
        bodyInfo BodyInfo     //身体数据
}
//身体数据结构体
type BodyInfo struct{
        weight float64        //体重
        color string          //颜色
}
```

```
func main(){
        var catOne Cat
        catOne.name="三酷猫"
        catOne.weight=10.5
        catOne.color="白色"
        fmt.Println("我的名字是：",catOne.name,"，体重：",catOne.weight,"
，毛色：",catOne.color)
}
```

6.4 结构体的嵌套

- 使用结构体实现继承

```
//猫结构体
type Cat struct {
        //眼睛的颜色
        eyeColor string
        //动物结构体
        animal *Animal
}
//猫-喵喵叫
func (catInstance Cat) mewing(){
        fmt.Println(catInstance.animal.name,"喵喵喵")
}
```

```
//狗结构体
type Dog struct {
        //身体的颜色
        bodyColor string
        //动物结构体
        animal *Animal
}
//狗-汪汪叫
func (dogInstance Dog) bowwow(){
        fmt.Println(dogInstance.animal.name,"汪汪汪")
}
//动物结构体
type Animal struct{
        //名字
        name  string
}
```

6.4 结构体的嵌套

- 使用结构体实现继承

```
func main(){
        dogOne:=&Dog{bodyColor: “黑色”,animal: &Animal{name: "贝贝"}}
        fmt.Println(dogOne.animal.name,"身体的颜色是",dogOne.bodyColor)
        dogOne.bowwow()
        catOne:=&Cat{eyeColor: "蓝色",animal: &Animal{name: "三酷猫"}}
        fmt.Println(catOne.animal.name,"眼睛的颜色是",catOne.eyeColor)
        catOne.mewing()
}
```