枚举类型常被用于实现特定类或结构体的功能。同样地，也能够在有多种变量类型的环境中方便地定义通用类或结构体。为了实现这种功能，Swift允许你定义类型嵌套，可以在枚举类型、类和结构体中定义支持嵌套的类型。

要在一个类型中嵌套另一个类型，将需要嵌套的类型的定义写在被嵌套类型的区域{}内，而且可以根据需要定义多级嵌套。

**类型嵌套实例**

下面这个例子定义了一个结构体BlackjackCard，用来模拟BlackjackCard(游戏：二十一点)中的扑克牌点数。BlackjackCard结构体包含2个嵌套定义的枚举类型 Suit 和 Rank。

在BlackjackCard规则中，Ace牌可以表示1或者11，Ace牌的这一特征用一个嵌套在枚举型Rank中的结构体Values来表示。

1. struct BlackjackCard {
3. // 嵌套定义枚举型Suit
4. **enum** Suit: Character {
5. **case** Spades = "♠", Hearts = "?", Diamonds = "?", Clubs = "♣"
6. }
8. // 嵌套定义枚举型Rank
9. **enum** Rank: Int {
10. **case** Two = 2, Three, Four, Five, Six, Seven, Eight, Nine, Ten
11. **case** Jack, Queen, King, Ace
12. struct Values {
14. let first: Int, second: Int?
15. }
16. **var** values: Values {
17. **switch** self {
18. **case** .Ace:
19. **return** Values(first: 1, second: 11)
20. **case** .Jack, .Queen, .King:
21. **return** Values(first: 10, second: nil)
22. **default**:
23. **return** Values(first: self.toRaw(), second: nil)
24. }
25. }
26. }
28. // BlackjackCard 的属性和方法
29. let rank: Rank, suit: Suit
30. **var** description: String {
31. **var** output = "suit is \(suit.toRaw()),"
32. output += " value is \(rank.values.first)"
33. **if** let second = rank.values.second {
34. output += " or \(second)"
35. }
36. **return** output
37. }
38. }

枚举型的Suit用来描述扑克牌的四种花色，并分别用一个Character类型的值代表花色符号。

枚举型的Rank用来描述扑克牌从Ace~10,J,Q,K,13张牌，并分别用一个Int类型的值表示牌的面值(这个Int类型的值不适用于Ace,J,Q,K的牌)。

如上文所提到的，枚举型Rank在自己内部定义了一个嵌套结构体Values。这个结构体包含两个变量，只有Ace有两个数值，其余牌都只有一个数值。结构体Values中定义了两个属性：

first, 为Int ；

second, 为 Int?, 或 “optional Int”；

Rank还定义了一个计算属性values，这个计算属性会根据牌的面值，用适当的数值去初始化Values实例，并赋值给values。对于J,Q,K,Ace会使用特殊数值，对于数字面值的牌使用Int类型的值。

BlackjackCard结构体自身有两个属性—rank与suit，它还定义了一个计算属性description，description属性使用rank和suit中的内容来构建对这张扑克牌名字和数值的描述，并且使用可选类型来检查是否存在第二个值，若存在，则在原有的描述中增加对第二数值的描述。

因为BlackjackCard是一个没有自定义构造函数的结构体，正如《[Memberwise Initializers for Structure Types](https://github.com/CocoaChina-editors/Welcome-to-Swift/blob/master/The%20Swift%20Programming%20Language/02Language%20Guide/14Initialization.md)》中所描述的，BlackjackCard结构体有默认的成员构造函数，所以你可以使用默认的initializer去初始化新的常量theAceOfSpades:

1. let theAceOfSpades = BlackjackCard(rank: .Ace, suit: .Spades)
2. println("theAceOfSpades: \(theAceOfSpades.description)")
3. // 打印出 "theAceOfSpades: suit is ♠, value is 1 or 11"

尽管Rank和Suit嵌套在BlackjackCard中，但仍可被引用，所以在初始化实例时能够通过枚举类型中的成员名称（.Ace 和 .Spades）单独引用。在上面的例子中，description属性能正确地输出theAceOfSpades有1和11两个值。

**类型嵌套的引用**

在外部对嵌套类型的引用，是以被嵌套类型的名字为前缀，加上所要引用的属性名：

1. let heartsSymbol = BlackjackCard.Suit.Hearts.toRaw()
3. // 红心的符号 为 "?"

对于上面这个例子，这样做可以使Suit, Rank, 和 Values的名字尽可能的简短，因为它们的名字会自然地由被定义的上下文来限定。

原文：[Nested Types](https://developer.apple.com/library/prerelease/ios/documentation/Swift/Conceptual/Swift_Programming_Language/NestedTypes.html#//apple_ref/doc/uid/TP40014097-CH23-XID_309)