Golang不支持模版函数（类型参数化），所以看上去不得不为每一种类型都实现一个函数。但是Golang可以利用空接口interface{}和闭包/告诫函数来实现泛型函数。

空接口interface{}是指方法集为空的接口，任何类型的值都可以赋值给空接口。

|  |
| --- |
| func minimum(first interface{}, rest ...interface{}) interface{} {      minimum := first      for \_, x := range rest {          switch x := x.(type) {          case int:              if x < minimum.(int) {                  minimum = x              }          case float64:              if x < minimum.(float64) {                  minimum = float64(x)              }          case string:              if x < minimum.(string) {                  minimum = string(x)              }          }      }        return minimum  }    func main() {      i := minimum(4, 5, 1, 9, 0, -1, -5, 7)      fmt.Println(i)        j := minimum(4.5, 5.2, -0.4, 9.9, 2.1)      fmt.Println(j)        k := minimum("abc", "def", "xyz", "ctz", "{}", "#$%^&\* ", "中国")      fmt.Println(k)  } |
| wKiom1YfXduSBg53AABgUzDIuAI217.jpg |

闭包/高阶函数

所谓闭包就是一个函数捕获了和它在同一作用域的其他变量和常量，从形式上看匿名函数都是闭包。

|  |
| --- |
| funcfilter(limit int, predicate func(int) bool, appender func(int)) {      for i := 0; i < limit; i++ {          if predicate(i) {              appender(i)          }      }  }      func main() {      a := []int{4, -3, -8, 9, 0, 2, 1}      even := []int{}      filter(len(a), func(i int) bool { returna[i]%2 == 0 }, func(i int) { even = append(even, a[i]) })      fmt.Println(even)  } |
| wKiom1YfXn7xm5n6AABSp99p2RQ239.jpg |

上面均实现了泛型功能。