Урок Спринт 1.11

sort.Interface

Иногда бывает нужно отсортировать что-то посложнее слайса. Предположим, на складе магазина техники каждый товар лежит в определённой коробке. Мы можем описать склад вот такой структурой:

```
// Опишем склад с товарами
type Goods struct {
    Names []string  // Название товара
    Boxes []int     // Коробка, в которой лежит товар
}
```

В третью коробку положим телефон, в первую — планшет, а во вторую — наушники:

```
goods := Goods{
        Names: []string{"phone", "tablet", "earphones"},
        Boxes: []int{3, 1, 2},
}
```

Чтобы быстрее находить товары, нужно отсортировать их. При этом нам важно сохранить позиции товаров: товар с индексом `i` должен лежать в коробке с номером, который хранится в `Boxes[i]`. Для решения этой задачи реализуем специальный интерфейс:

```
type Interface interface {
    Len() int
    Less(i, j int) bool
    Swap(i, j int)
}
```

Так мы сможем использовать алгоритм сортировки из стандартной библиотеки:

```
func (g Goods) Len() int {
    return len(g.Names)
}
func (g Goods) Swap(i, j int) {
    g.Names[i], g.Names[j] = g.Names[j], g.Names[i] // Меняем местами как название товара,
    g.Boxes[i], g.Boxes[j] = g.Boxes[j], g.Boxes[i]  // так и коробки, где товар хранится
}
func (g Goods) Less(i, j int) bool {
    return g.Boxes[i] < g.Boxes[j]
}
```

Теперь воспользуемся функцией `Sort`:

```
sort.Sort(goods)
fmt.Println(goods) // {[tablet earphones phone] [1 2 3]}
```

Вуаля! Наши товары отсортированы.