Урок Спринт 1.11

Сортировка в Go

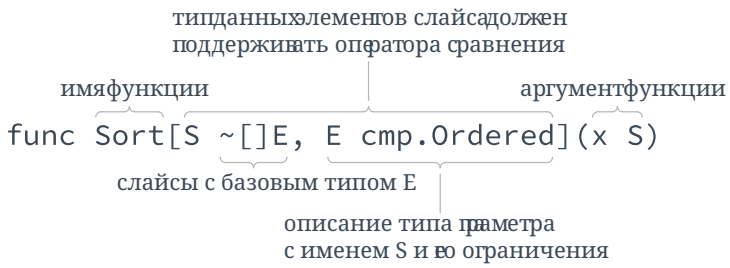
В стандартной библиотеке Go есть широкий набор средств для сортировки. В версии Go 1.21 используется одна из последних модификаций алгоритма быстрой сортировки (сложность `O(n log n)`), поэтому разработчику редко приходится реализовывать сортировку самому.

package slices

В той же версии Go в стандартную библиотеку добавили пакет `slices`. Он существенно облегчает работу со слайсами. Чтобы выполнить сортировку, достаточно воспользоваться вот такой функцией:

```
func Sort[S ~[]E, E cmp.Ordered](x S)
```

Остановимся на ней поподробнее. Здесь применяются дженерики (помните прошлое занятие?):



Так можно отсортировать слайс с элементами любых типов данных, которые поддерживают операторы сравнения.

Вот как можно использовать эту функцию:

```
smallInts := []int8{0, 42, -10, 8}
slices.Sort(smallInts)
fmt.Println(smallInts) // [-10 0 8 42]
```

Чтобы настроить пользовательскую функцию сравнения, нужна функция `SortFunc`:

```
func SortFunc[S ~[]E, E any](x S, cmp func(a, b E) int)
```

Здесь мы видим, что ограничения `cmp.Ordered` на элементы данных не накладываются. Мы должны передать на вход функцию `cmp func(a, b E) int`, которая будет сравнивать значения из слайса. Она должна возвращать:

- отрицательное значение, если a < b
- положительное число, если a > b
- ноль, если a == b

Допустим, нам нужно отсортировать слайс из примера выше по убыванию. Для этого нам пригодится `SortFunc`:

```
smallInts := []int8{0, 42, -10, 8}
slices.SortFunc(smallInts, func(a, b int8) int {
        // Здесь мы сравниваем сами
        switch {
        case a < b:
            return 1
        case a > b:
```

```
            return -1
        default:
            return 0
        }
})
fmt.Println(smallInts) // [42 8 0 -10]
```

А если надо отсортировать слайс с объектами посложнее? Например, слайс с пользователями по возрасту:

```
type User struct {
    Name string
    Age  int
}
users := []User{
    {
        Name: "Ivan",
        Age:  56,
    },
    {
        Name: "Tim",
        Age:  33,
    },
    {
        Name: "Bob",
        Age:  89,
    },
}
slices.SortFunc(users, func(a, b User) int {
        // Здесь мы сравниваем сами
        switch a.Age < b.Age {
        case true:
            return -1
        case false:
            return 1
        default:
            return 0
        }
})
fmt.Println(users) // [{Tim 33} {Ivan 56} {Bob 89}]
```

package sort

Примитивные типы данных

Пакет `sort` появился в стандартной библиотеке гораздо раньше пакета `slices` у них во многом похожие функции, но дженериков в `sort` нет. Иногда их отстутствие даёт небольшой выигрыш во времени. Мы рекомендуем пользоваться пакетом `slices`, а `sort` использовать по необходимости.

Чтобы сортировать коллекции примитивных типов данных (`string`, `int`, `float64`), есть специальные методы:

```
intSlice := []int{4, 5, 2, 1, 3, 9, 7, 8, 6}
fmt.Println(sort.IntsAreSorted(intSlice)) // Проверим, отсортирован ли слайс
sort.Ints(intSlice) // Сама сортировка
fmt.Println(intSlice) // [1 2 3 4 5 6 7 8 9]
```