# OC_6

## 集合遍历和数组排序

### 集合遍历

用allObjects属性取集合的所有元素
存储到数组,在用for循环

**集合遍历**

```
NSSet *set = [NSSet setWithObjects:@"element1",
@"element2", @"element3", @"element4", nil];
NSArray *objects = set.allObjects;
for (NSUInteger i = 0; i < objects.count; i++) {
    NSString *value = [objects objectAtIndex:i];
    NSLog(@"value:%@", value);
}
```

### for循环遍历

通过循环获取不同的下标元素
循环次数就是数组元素的个数

**数组遍历**

```
NSArray *array = @[@"element1", @"element2",
@"element3", @"element4"];

for (NSUInteger i = 0; i < array.count; i++) {
    NSString *value = [array objectAtIndex:i];
    NSLog(@"value:%@", string);
}
```

### 字典遍历

通过key取出对应的value
循环次数等于字典元素个数

**字典遍历**

```
NSDictionary *dictionary = @{@"key1":@"value1",
@"key2":@"value2", @"key3":@"value3",
@"key4":@"value4"};

NSArray *keys = [dictionary allKeys];

for (NSUInteger i = 0; i < keys.count; i++) {
    NSString *key = [keys objectAtIndex:i];
    NSString *value = [dictionary objectForKey:key];
    NSLog(@"value:%@", value);
}
```

### NSEnumerator遍历

枚举器,遍历集合中的元素
对可变集合枚举时,不能通过添加删
除对象来改变集合容器的元素个数

#### 数组正序枚举

```
NSArray *array = @[@"element1", @"element2",
@"element3", @"element4"];
NSEnumerator *enumerator = [array
objectEnumerator]; //正序的枚举器对象
id value = nil; //存储数组中的元素
while ((value = [enumerator nextObject])) {
    NSLog(@"%@", value);
}
```

注：枚举器的nextObject方法只能取出一个对象，所以需要和while循环结合把所有元素依次取出。

#### 数组倒序枚举

```
NSArray *array = @[@"element1", @"element2",
@"element3", @"element4"];
NSEnumerator *reverse = [array
reverseObjectEnumerator]; //逆序的枚举器对象
id value = nil; //存储数组中的元素
while ((value = [reverse nextObject])) {
    NSLog(@"%@", value1);
}
```

#### 字典枚举遍历

```
NSDictionary *dictionary = @{@"key1":@"value1",
@"key2":@"value2", @"key3":@"value3",
@"key4":@"value4"};
NSEnumerator *enumerator = [dictionary
objectEnumerator]; //获取枚举器对象
id value = nil; //存储字典中的value
while ((value = [enumerator nextObject])) {
    NSLog(@"%@", value);
}
```

#### 集合枚举遍历

```
NSSet *set = [NSSet setWithObjects:@"element1",
@"element2", @"element3", @"element4", nil];
NSEnumerator *enumerator = [set
objectEnumerator]; //获取枚举器对象
id value = nil; //存储集合中的元素
while ((value = [enumerator nextObject])) {
    NSLog(@"%@", value);
}
```

注：由于集合中存储的元素是无序的，因此没有反向枚举的概念！

### for-in遍历

快速枚举器
集合枚举时不能添加删除对象来改变
集合容器的元素个数

#### 数组for..in遍历

- 数组快速遍历，遍历得到的是数组中的每个元素。
- 例:

```
NSArray *array = @[@"element1", @"element2",
@"element3", @"element4"];
for (NSString *string in array) {
    NSLog(@"%@", string);
}
```

#### 字典for..in遍历

- 字典快速遍历，遍历得到的是字典中的Key。
- 通过key从字典中获取对应value对象。
- 例:

```
NSDictionary *dictionary =
@{@"key1":@"value1", @"key2":@"value2",
@"key3":@"value3", @"key4":@"value4"};
for (NSString *key in dictionary) {
    NSString *value = [dic valueforKey:key];
    NSLog(@"%@", value);
}
```

#### 集合for..in遍历

- 集合快速遍历，遍历得到的是集合中的每个元素。
- 例:

```
NSSet *set = [NSSet setWithObjects:@"element1"
@"element2", @"element3", @"element4", nil];
for (NSString *string in set) {
    NSLog(@"%@", string);
}
```

### 数组排序

数组是有序容器,所以集合中只有数
组才能排序

#### 数组排序一字典

把元素属性作为key记下升序或降序

NSSortDescriptor

- 初始化方法

```
- (instancetype)initWithKey:(NSString *)key
ascending:(BOOL)ascending;
```

key:按照数组中对象的哪个属性进行排序，如果数组中存放的是能够直接排序的对象（比如：字符串），直接使用@"self"或者nil即可；如果存放的是自定义类的对象，使用想要进行排序的属性名即可（比如：想按照Person类的name进行排序，使用@"name"作为参数。

ascending:排序的标志，是升序还是降序，YES－升序，NO－降序。

- 数组根据排序条件进行排序，得到排好序的新的数组对象。

```
- (NSArray *)sortedArrayUsingDescriptors:(NSArray
*)sortDescriptors;
```

sortDescriptors：数组类型的对象，数组中可以存放多个排序描述对象，做终端实现控制多个要求进行排序的目的。
比如：如果该参数中存储了由name和age两个Person属性参与创建的NSSortDescriptor对象，在按name进行排序的同时，如果遇到姓名相同的就会按照age进行排序。

#### 数组排序

- 数组中存放直接可进行排序的对象
- 创建数组对象

```
NSArray *array = @[@"zhengzhou", @"beijing",
@"shanghai", @"guangzhou", @"xian", @"dalian"];
```

- 创建排序条件

```
NSSortDescriptor *descriptor = [[NSSortDescriptor
alloc] initWithKey:@"self" ascending:YES];
```

##### 实例

- 数组中存放自定义的类的对象
- 创建数组对象,数组中存储多个Person对象

```
NSArray *array = @[per1, per2, per3, per4, per5];
```

- 创建排序条件，按照Person对象的姓名降序排序

```
NSSortDescriptor *descriptor = [[NSSortDescriptor
alloc] initWithKey:@"name" ascending:NO];
```

- 数组根据排序条件进行排序

```
NSArray *resultArray = [array
sortedArrayUsingDescriptors:@[descriptor]];
```

- resultArray得到的就是将数组中Person对象按照姓名降序排序之后的数组。

#### 其他数组排序

排序结果生成新数组,原数组无改变

##### 不可变数组排序

```
- (NSArray *)sortedArrayUsingSelector:
(SEL)comparator;
```

注：SEL类型的参数comparator：需要传入一个返回结果是NSComparisonResult的方法名。

```
NSArray *array = @[@"lanou", @"zhengzhou", @"henan",
@"huimian"];

NSArray *newArray = [array
sortedArrayUsingSelector:@selector(compare:)];
```

- 数组array中的元素都是字符串。
- 字符串比较的方法compare:返回值类型正好满足SEL参数的需求。

##### 可变数组

```
- (void)sortUsingSelector:(SEL)comparator;
```

注：SEL类型的参数comparator:需要传入一个返回结果是NSComparisonResult的函数

注意：可变数组排序方法与不可变数组排序方法的区别！！！

```
NSMutableArray *array = [@[@"lanou", @"zhengzhou",
@"henan", @"huimian"] mutableCopy];

[array sortUsingSelector:@selector(compare:)];
```

- 数组array中的元素都是字符串。
- 字符串比较的方法compare:返回值类型正好满足SEL参数的需求。

#### 数组排序

数组中存放自定义的类对象

- 数组中存放自定义的类的对象
- 创建数组对象,数组中存储多个Person对象

```
NSArray *array = @[per1, per2, per3, per4, per5];
```

- 将数组按照Person对象的姓名升序排序

```
NSArray *newArray = [array
sortedArrayUsingSelector:@selector(compareByName:)];
```

注：需要Person类提供两个对象按照姓名进行升序排序比较的方法。

##### Person比较方法

- 比较方法的声明

```
- (NSComparisonResult)compareByName:(Person
*)anotherPerson;
```

- 比较方法的实现

```
- (NSComparisonResult)compareByName:(Person
*)anotherPerson {
    return [self.name compare:anotherPerson.name];
}
```

### 总结

集合遍历的几种方法,其各自特点
数组排序的几种方法,其各自特点