

词典

散列：循对象访问

We are shaped by our thoughts; we become what we think.

- Buddha

Man's thought is shaped by his tongue.

- Anonymous

于是在熟人中，我们话也少了，我们“眉目传情”，我们“指石相证”，  
我们抛开了比较间接的象征原料，而求更直接的会意了

邓俊辉

deng@tsinghua.edu.cn

# 联合数组：更直接、更有效的访问

## ❖ 数组？再常见不过，比如：

fib[0] = 0

fib[1] = 1

fib[2] = 1

fib[3] = 2

fib[4] = 3

fib[5] = 5

fib[6] = 8

...

## ❖ Associative Array

——与常规的数组有何区别？

## ❖ 根据数据元素的取值，直接访问！

style["关羽"] = "云长"

style["张飞"] = "翼德"

style["赵云"] = "子龙"

style["马超"] = "孟起"

## ❖ 下标不再是整数，甚至没有大小次序

——更为直观、便捷

## ❖ 支持的语言：

Snobol4、MUMPS、SETL、Rexx、AWK、  
Java、Python、Perl、Ruby、PHP、...

## Java: HashMap + Hashtable

```
import java.util.*;

public class Hash {
    public static void main(String[] args) {

        HashMap HM = new HashMap(); //Map
        HM.put("东岳", "泰山"); HM.put("西岳", "华山"); HM.put("南岳", "衡山");
        HM.put("北岳", "恒山"); HM.put("中岳", "嵩山"); System.out.println(HM);

        Hashtable HT = new Hashtable(); //Dictionary
        HT.put("东岳", "泰山"); HT.put("西岳", "华山"); HT.put("南岳", "衡山");
        HT.put("北岳", "恒山"); HT.put("中岳", "嵩山"); System.out.println(HT);

    }
}
```

## Perl: %Hash Type

❖ 由字符串 (string) 标识的一组无序标量 (scalar) //亦即MAP

---

❖ `my %hero = ( "云长"=>"关羽", "翼德"=>"张飞", "子龙"=>"赵云", "孟起"=>"马超") ;`

---

`foreach $style (keys %hero) # Hash类型的变量由%引导`

`{ print "$style => $hero{$style}\n"; }`

❖ `$hero{"汉升"} = "黄忠";`

`foreach $style (keys %hero)`

`{ print "$style => $hero{$style}\n"; }`

`foreach $style (reverse sort keys %hero)`

`{ print "$style => $hero{$style}\n"; }`

## Python: Dictionary Class

❖ `beauty = dict( { "沉鱼":"西施", "落雁":"昭君", "闭月":"貂蝉", "羞花":"玉环" } )`

---

`print( beauty )`

❖ `beauty["红颜"] = "圆圆"`

`print( beauty )`

❖ `for (alias, name) in beauty.items() :`

`print( alias, ":", name )`

❖ `for alias, name in sorted( beauty.items() ) :`

`print( alias, ":", name )`

❖ `for alias in sorted( beauty.keys(), reverse = True ) :`

`print( alias, ":", beauty[alias] )`

## Ruby: Hash Table

```
scarborough = # declare and initialize a hashtable  
  { "P"=>"parsley", "S"=>"sage", "R"=>"rosemary", "T"=>"thyme" }
```

---

```
puts scarborough # output the hash table
```

```
for k in scarborough.keys # output hash table items
```

```
  puts k + "=>" + scarborough[k] # 1-by-1
```

```
end
```

```
for k in scarborough.keys.sort # output hash table items
```

```
  puts k + "=>" + scarborough[k] # 1-by-1 in order
```

```
end
```

# 词条 ~ 映射/词典

❖ entry = (key, value)

❖ Map/Dictionary: 词条的集合

- 关键码禁止/允许相等

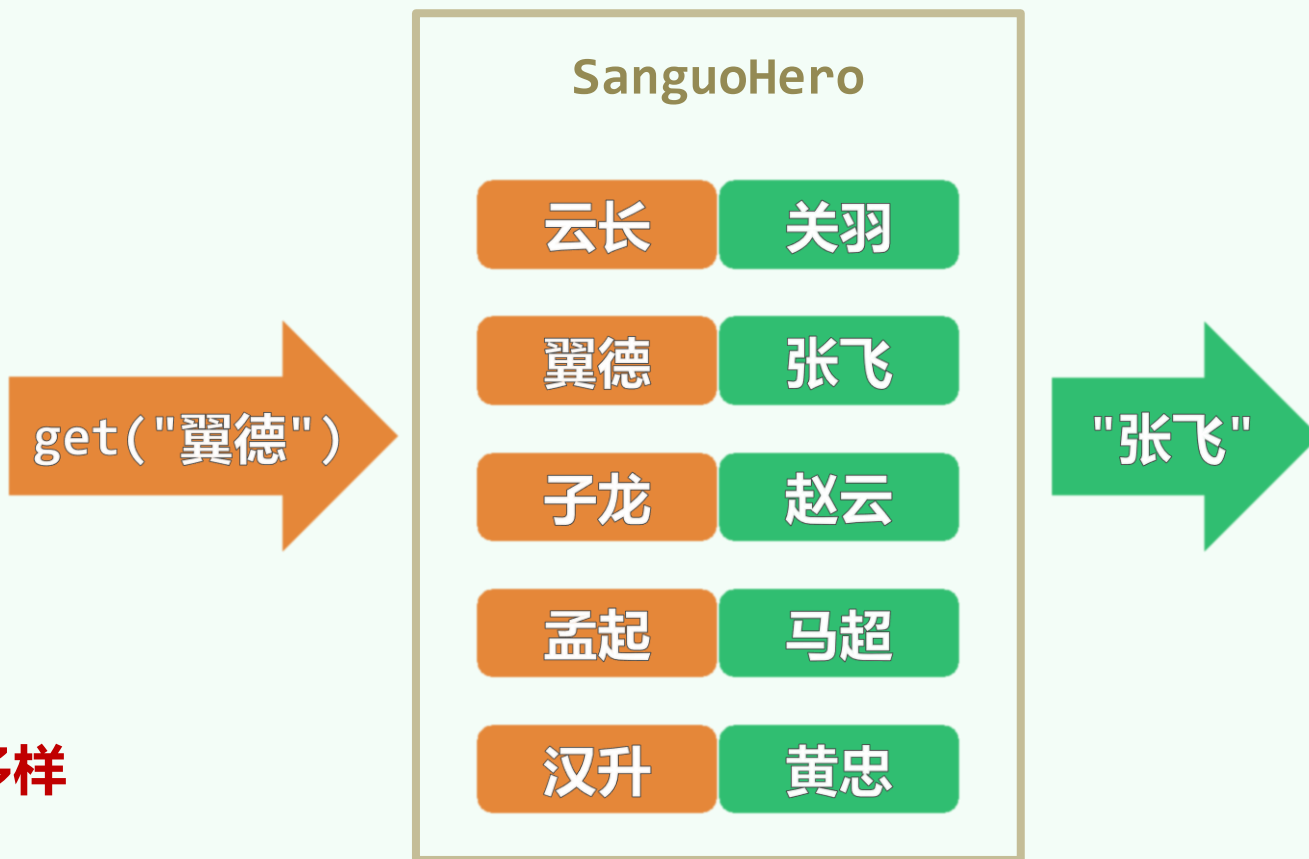
- get(key)

- put(key, value)

- remove(key)

❖ 关键码未必可定义大小, 元素类型较BST更多样

查找对象不限于最大/最小词条, 接口功能较PQ更强大



# Dictionary ADT

❖ `template <typename K, typename V> //key、value`

```
struct Dictionary {  
    virtual Rank size() = 0;  
    virtual bool put( K, V ) = 0;  
    virtual V* get( K ) = 0;  
    virtual bool remove( K ) = 0;  
};
```

❖ 词典中的词条只需支持**判等/比对**操作，尽管

诸如 `Java::TreeMap` 等实现仍支持**大小/比较器**

