We are shaped by our thoughts;

we become what we think.

- Buddha

Man's thought is shaped by his tongue.

- Anonymous

The diversity of languages is not

a diversity of signs and sounds, but

a diversity of views of the world.

- Wilhelm von Humboldt, 1820

9.词典

(a) 散列：循值访问

邓俊辉

deng@tsinghua.edu.cn

联合数组

❖ 数组？再常见不过，比如：fib[0] = 0, fib[1] = 1, fib[2] = 1, fib[3] = 2, ...

❖ associative array：与此前的数组有何区别？

❖ 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、...

映射 + 词典 = 符号表

- 词条：entry = (key, value)
- 映射：Map = 词条的集合，其中各词条（的关键码）互异

  ADT：get(key)，put(key, value)，remove(key)
- 较之BST，关键码之间未必可比较 call-by-value

  较之PQ，查找对象更广泛，不限于最大、最小词条
- 词典：Dictionary：与映射基本相同，但允许词条（的关键码）雷同

  Sorted dictionary：关键码之间可定义全序关系的词典
- 映射和词典都是动态的，统称符号表 symbol table

Dictionary

```
template <typename K, typename V> //key、value
struct Dictionary { //Dictionary模板类
    virtual int size() = 0; //查询当前的词条总数
    virtual bool put( K ) = 0; //插入词条(key, value)
    virtual V* get( K ) = 0; //查找以key为关键码的词条
    virtual bool remove( K ) = 0; //删除以key为关键码的词条
};
```

- 尽管诸如Java::TreeMap等实现仍然要求支持比较器，但实际上词典中的词条，只需支持比对判等操作，而不必支持大小比较

Dictionary

❖ 在这里，无论对外的访问方式，还是内部的存储方式

都更直接地依据数据对象自身的取值

key与value地位等同，不必区分

❖ 循值访问 call-by-value：方式更为自然，适用范围也更广泛

❖ 回忆一下，初次接触程序设计时，你首先想到的应该就是这种方式

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(HM);
    }
}
```

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 # Python dictionary (hashtable)
    ( { "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 hash table
    "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
```

课后

- 了解Java中HashMap与Hashtable的异同
- 安装JDK（http://www.java.com）

  尝试HashMap和Hashtable类
- 安装Perl（http://www.perl.org）

  尝试%Hash类型
- 安装Python（http://www.python.org）

  尝试Dictionary类
- 安装Ruby（http://www.ruby-lang.org）

  尝试Hash Table