Sets, Maps, Bit Sets
Section 2.3

Dr. Mayfield and Dr. Lam

Department of Computer Science
James Madison University

Oct 02, 2015
For those unfamiliar with pointers...

```c
int i;  // integer
int *p;  // integer pointer

p = &i;  // & is the reference operator
*p = 74;  // * is the dereference operator

cout << i << endl;  // prints the value 74
```

Pointers to objects use the -> operator
- In C++, `foo->bar` is equivalent to `(*foo).bar`  
- Since Java has no pointers, it simply uses dot
Reality Check

Pointers make it possible to allocate new memory!
Stack vs heap allocation

```cpp
int main()
{
    string s1("Hello"); // constructor
    cout << s1.length();
}
```

- Local variables are allocated on the stack
- When functions return, objects are destructed

```cpp
int main() // notice use of new and ->
{
    string *s1 = new string("Hello");
    cout << s1->length();
}
```

- Pointers are simply integers (allocated on the stack)
- The objects they point to are allocated in the heap
C++ iterators

Iterators are a generalization of pointers
- They must implement * (dereference)
- They must implement ++ (i.e., get next)

// example from last week
vector<int>::iterator iter;
for (iter = v.begin(); iter != v.end(); iter++)
    cout << *iter << endl;

SPOILER ALERT!
- Not all data structures are linear (like arrays)
- But you don’t have to worry about that issue
Non-Linear Data Structures

When making an array for every possible item would be insane.
What’s a set? (C++)

Unordered collection of objects

- Set of vowels $V = \{a, e, i, o, u\}$

Hint for UVa 11849: Compact Discs

```cpp
set<int> nums;
nums.insert(74);
nums.insert(78);
nums.insert(82);
nums.insert(74); // no new element inserted

set<int>::iterator iter;
iter = nums.find(82);
// find returns nums.end() if not found
```

What’s a set? (Java)

Unordered collection of objects

- Set of vowels $V = \{a, e, i, o, u\}$

Hint for UVa 11849: Compact Discs

```java
// must use reference types like Integer
Set<Integer> nums = new HashSet<Integer>();

nums.add(74);
nums.add(78);
nums.add(82);
nums.add(74); // no new element inserted

nums.contains(82); // returns true or false
```

https://docs.oracle.com/javase/7/docs/api/java/util/Set.html
Binary search tree

Internal data structure for STL set, map (Java TreeSet, TreeMap)
What’s a map?

Data structure of (key, value) pairs
- No two entries have the same key
- Allows you to store additional data
- Implemented with trees or hashing

Applications
- Dictionaries
- Counting problems
- Lots of others!
Hash table

Internal data structure for STL `unordered_map` (Java `HashMap`)

<table>
<thead>
<tr>
<th>Keys</th>
<th>Indicies</th>
<th>Key-value pairs (records)</th>
</tr>
</thead>
<tbody>
<tr>
<td>John Smith</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lisa Smith</td>
<td>1</td>
<td>Lisa Smith: +1-555-8976</td>
</tr>
<tr>
<td>Sam Doe</td>
<td>872</td>
<td>John Smith: +1-555-1234</td>
</tr>
<tr>
<td></td>
<td>874</td>
<td>Sandra Dee: +1-555-9655</td>
</tr>
<tr>
<td></td>
<td>998</td>
<td>Sam Doe: +1-555-5030</td>
</tr>
<tr>
<td></td>
<td>999</td>
<td></td>
</tr>
</tbody>
</table>

Oct 02, 2015 Sets, Maps, Bit Sets
Maps in Java (ugly)

```java
import java.util.Iterator;
import java.util.Map;
import java.util.TreeMap;

// map each word to its anagram
TreeMap<String, String> map =
    new TreeMap<String, String>();
map.put(word, anag);

// output results in sorted order
Iterator<Map.Entry<String, String>> iter =
    map.entrySet().iterator();
while (iter.hasNext()) {
    Map.Entry<String, String> entry = iter.next();
    System.out.println(entry.getKey() + " : "+ entry.getValue());
}
```
Maps in C++ (easier)

```cpp
#include <map>
using namespace std;

// map course names to numbers
map<string, int> courses;
courses["Databases"] = 474;
courses["Networking"] = 462;
courses["Friday Fun"] = 280;

// output results
map<string, int>::iterator ii;
for (ii = courses.begin(); ii != courses.end(); ii++)
{
    cout << ii->first << ": " << ii->second << endl;
}
```
Other C++ tips

Use **typedef** to create aliases

```cpp
typedef map<string, int> map_si;
typedef map<string, int>::const_iterator itr_si;
```

```cpp
int main() // Hint for UVa 10226: Hardwood Species
{
    map_si trees;
    itr_si itr;
}
```

Use **const** whenever possible

```cpp
for (itr = trees.begin(); itr != trees.end(); itr++)
{
    const char* name = itr->first.c_str();
    double percent = 100.0 * itr->second / total;
    printf("%s %.4f\n", name, percent);
}
```
Bit sets

Sometimes linear is better :-)  
▶ Array of 1,000,000 booleans  
▶ Building a tree is too slow  
▶ Trick: use binary numbers!

```cpp
// Hint for UVa 11926: Multitasking
bitset<10> bs;
bs.set(1);
cout << bs << endl; // prints 0000000010
bs.reset(1);
cout << bs << endl; // prints 0000000000
```

Hint for UVa 11933: Splitting Numbers

```cpp
// Oct 02, 2015 Sets, Maps, Bit Sets
bitset<32> bs(74);
cout << bs << endl; // 00000000000000000000000001001010
cout << bs.to_ulong();
```