Vector, Stack, Queue
Section 2.1–2.2

Dr. Mayfield and Dr. Lam

Department of Computer Science
James Madison University

Sep 25, 2015
Announcement #1

Your portfolio must be done in LaTeX

- See portfolio.tex on Course Website
  - Read through the LaTeX 101 slides
  - In Texmaker, press F1 twice to build
  - The rest should be self-explanatory :)

For each problem you should:

- Write 1–2 paragraphs of reflection
- Document any complex algorithms
- Format your code neatly for printing
  - There should be no text wrapping!
  - Double check the page boundaries!
Announcement #2

Qualification contest: Oct 3rd, 3:00–8:00 PM, here

- You should already be registered
  - But you will need to complete your registration
- We’ll have pizza around 5:00 or 6:00 PM
- Extra credit / make-up opportunity

Regional contest sign-up

- Need to finalize teams and register
Environment Information

Official environment reference:

- OS: Ubuntu 14.04.1 LTS w/ GNOME desktop
- Editors: vi/vim/gvim, emacs, gedit, geany, Eclipse 4.4.1
- C++11 (g++ 4.8.2)
  - Build: "g++ -g -O2 -std=gnu++11 -static $*
- Java 1.7.0 (compile with "-source 7" on other platforms)
  - Build: "javac -encoding UTF-8 -sourcepath . -d . $*
  - Run: "java -client -Xss8m -Xmx1024m $*"
Today’s Problems: Linear Data Structures

Containers: vector, stack, queue
C++ standard template library

See STL Guide on course website!

**Sorting example**

```cpp
#include <algorithm>

int A[] = {1, 4, 2, 8, 5, 7};
const int N = sizeof(A) / sizeof(int);
sort(A, A + N);

// output is "1 2 4 5 7 8"
```
Java Collections Framework

See Java Collections Framework Guide

Sorting example

```java
import java.util.*;

List<Integer> A = Arrays.asList(1, 4, 2, 8, 5, 7);
Collections.sort(A);
for (int a : A) {
    System.out.printf(" %d", a);
}

// output is " 1 2 4 5 7 8"
```
#include <vector>

- Dynamic array that grows as needed (like Java ArrayList)
- Contiguous pre-allocated memory; efficient random access
- Inserts/deletes are only efficient at the end of the array

**UVa 10038: Jolly Jumpers**
- Don’t use a vector when you can just use an array

**UVa 10107: The Median**
- Read integers into a vector
- Use `nth_element` algorithm
Vector example (C++)

// read in a bunch of numbers
int i;
vector<int> v;
while (cin >> i)
    v.push_back(i);

// print them all back again
for (i = 0; i < v.size(); i++)
    cout << v[i] << endl;

// same thing with iterators
vector<int>::iterator iter;
for (iter = v.begin(); iter != v.end(); iter++)
    cout << *iter << endl;
Vector example (Java)

// read in a bunch of numbers
Scanner in = new Scanner(System.in);
Vector<Integer> v = new Vector<Integer>();
while (in.hasNextInt())
    v.add(in.nextInt());

// print them all back again
for (int i = 0; i < v.size(); i++)
    System.out.println(v.get(i));

// same thing with for-each
for (int a : v)
    System.out.println(a);
Common functions

- `v.size()`: number of items
- `v.empty()`: true if size is 0 (Java: `isEmpty`)
- `v.capacity()`: allocated storage
- `v.reserve(n)`: reallocate storage (Java: `ensureCapacity`)
- `v.push_back(x)`: append element `x` (Java: `add`) (may expand)
- `v.pop_back()`: erase last element (no Java equivalent)
- `v.insert(pos, x)`: inserts `x` before `pos` (Java: `add`)
- `v.erase(pos)`: erases element at `pos` (Java: `remove`)
- `v.clear()`: erases all elements
- `v.front()`: reference to first element (Java: `firstElement`)
- `v.back()`: reference to last element (Java: `lastElement`)

Sep 25, 2015 Vector, Stack, Queue 11 of 16
Common operations

\[ v[i] \quad (\text{C++}) \text{ access element at index (no bounds checking)} \]
\[ v.\text{at}(i) \quad (\text{C++}) \text{ access element at index (may throw exception)} \]
\[ v.\text{get}(i) \quad (\text{Java}) \text{ access element at index (may throw exception)} \]
\[ v1 = v2 \quad \text{copy v2 into v1 (Java: addAll or copyInto)} \]
\[ v1 == v2 \quad \text{pairwise comparison (Java: equals)} \]
\[ v1 < v2 \quad \text{lexicographic comparison (no Java equivalent)} \]

Don't forget: http://www.cplusplus.com/reference/
STL stack

Last In, First Out (LIFO)

```cpp
stack<int> s;
for (int i = 0; i < 5; i++)
    s.push(i);

while (!s.empty())
{
    cout << ' ' << s.top();
    s.pop();
}
cout << endl;
```

// output is " 4 3 2 1 0"

UVa 514: Rails

#include <stack>
Java stack

Last In, First Out (LIFO)

```java
Stack<Integer> s = new Stack<Integer>();
for (int i = 0; i < 5; i++)
    s.push(i);
while (!s.isEmpty())
    System.out.print(" "+s.pop());
System.out.println();
// output is " 4 3 2 1 0"
```
STL queue

First In, First Out (FIFO)

```cpp
queue<int> q;
for (int i = 0; i < 5; i++)
    q.push(i);

while (!q.empty())
{
    cout << ' ' << q.front();
    q.pop();
}
cout << endl;
```

// output is " 0 1 2 3 4"

UVa 10901: Ferry Loading

```cpp
#include <queue>
```
Java queue

First In, First Out (FIFO)

```java
Queue<Integer> q = new LinkedList<Integer>();
for (int i = 0; i < 5; i++)
    q.add(i);

while (!q.isEmpty())
    System.out.print(" "+ q.remove());

System.out.println();

// output is " 0 1 2 3 4"
```