CS 162
Intro to Programming II
Standard Template Library
Standard Template Library

• Data structure has a structure to hold data and standard operations on that data
• STL provides generic implementation of many standard data structures
  – Uses templates to allow use of different data types with little or no additional programming
• A **container** is a class that holds and organizes data in the STL
• **Iterator** is an object that provides access to elements stored in the container
Lists

- **Sequential containers**: maintain a sequence of data items such that there is a first element, a second element, etc.
- Examples of sequential contains: vector, list
- More accurately, STL uses doubly-linked lists
What

- STL has a `list<T>` template class. Examples of functions for a list:
  - `size()`: returns the size of the list
  - `begin()`: returns an iterator at the first element of the list
  - `end()`: returns an iterator beyond the last element of the list
  - `push_front()`: puts an element at the front of the sequence
  - `push_back()`: adds an element to the end of the list
  - `list<T>::iterator` type for iterators
What

- Example using lists:

```cpp
int main(int argc, char** argv) {
    std::list<std::string> l;
    l.push_back("Alice");
    l.push_back("Bob");
    l.push_back("Chuck");
    std::list<std::string>::iterator i;
    for( i = l.begin(); i != l.end(); i++ ) {
        std::cout << *i << std::endl;
    }
}
```
What

• What are the differences between a list and a vector?
• A vector has random access iterators.
• A list does not have random access iterators (only bi-directional iterators)
• A list has `push_front()` function, vector does not
Container Adapters

• Container adapters are template classes built on top of other classes
• You can use the default underlying container (recommended) or use another one
• eg. the default underlying container for a stack is a deque (double ended queue)
• We will talk about stacks and queues
Stacks

• Can only add/remove items from one end
• Last In First Out (LIFO) order
• Think of a stack of books on your table:

- How to Lie With Statistics
- War and Peace
- Goedel Escher Bach
Stacks

• Common operations:
  – `top()`: Returns the top element of the stack
  – `pop()`: Removes the top element of the stack. Does NOT red
  – `push(Element)`: inserts a copy of `Element` at the top of the stack
  – `size()`: returns the number of elements in the stack
  – `empty()`: returns true if the stack is empty
Stacks

#include <iostream>
#include <stack>

int main(int argc, char** argv) {
  std::stack<int> s;
  s.push(0);
  s.push(1);
  s.push(1);
  s.push(2);
  while (s.empty() == false) {
    int i = s.top();
    std::cout << i << std::endl;
    s.pop();
  }
}

Prints out:
2
1
0
Stacks

• Where are stacks used?
  – Call stack for functions
  – Converting a base 10 number to binary
  – HP calculators (reverse Polish notation)
Queues

• First-In First Out
• Add items to the back, remove from the front
• Think of the queue in real life
• “Queue” is British for line as in:
  – “Waiting in line” or
  – “Standing on line”
#include <iostream>
#include <queue>

int main(int argc, char** argv) {
    std::queue<int> q;
    q.push(0);
    q.push(1);
    q.push(2);
    while(!q.empty()) {
        int i = q.front();
        std::cout << i << std::endl;
        q.pop();
    }
}

Prints out:
0
1
2
Queue

• Where are Queues used in Computer Science?
  – Print queue
  – Email queue
  – Sending messages on a network (buffers)
  – Event queues for GUIs