Some C++ STL examples

- We’ll look at just a few examples of the C++ STL
- The list class (basic linked list)
- The stack class (typical LIFO stack)
- The queue class (typical FIFO queue)

Lists and methods

- Create a list of items, e.g. *list<int> L;*
- Insert at either end by pushing *L.push_front(i); // or back*
- Remove from either end by popping *L.pop_front(); // or back*
- Remove all content *L.clear();*
- Look up front/back element *e = L.front(); // or back*
- Look up size *x = L.size();*

Stacks and methods

- Based off same underlying code as lists
- Create, e.g. *stack<string> S; // stack of strings*
- Push new element *S.push_back("foo")*;
- Pop top element *S.pop_back();*
- Look up top element *e = S.back();*
- Look up size *x = S.size();*
- Check if empty *if (S.empty()) {* ...

Queues and methods

- Based off same underlying code as lists
- Create, e.g. *queue<float> Q; // queue of floats*
- Push new back element *Q.push_back(3.14)*;
- Pop front element *Q.pop_front();*
- Look up back/front element *e = Q.back(); // or front*
- Look up size *x = Q.size();*
- Check if empty *if (Q.empty()) { …*

Iterators

- For ADTs that are data collections, we often want to use a loop to walk (iterate) through each element in sequence
- STL has a standardized iterator syntax
- We declare an iterator of desired type
- We set it to refer to first element
- We can extract/use the element it refers to currently
- We can use ++ or -- to go forward/backward
- We test when we reach the end

List of ints example

- Assumes we’ve #included <list>

```
list<int> L;   // create the list
... do a bunch of L.push_back(x)’s to fill with data ...
std::list<int>::iterator i;   // declares an iterator for list of ints
i = L.begin();   // set it to refer to first one
while (i != L.end()) {  // keep going to the end
    int current = (*i);   // get the actual data value through the iterator
    ... do something with current ...
    i++;   // move on to next one
}
```