STL Deques

CPE 212 -- Lecture 19 continued

** Notes based on
The C++ Standard Library: A Tutorial and Reference, by Niicolai M. Josuttis

UAHuntsville

Deques

- Pronounced “deck”
- Double-ended queue
- Dynamic array that can grow in both directions
- Insertion or removal of values at either end is fast
- Middle insertions are slowed by relocation
- May free memory as elements are removed
- Provides random access
- #include <deque>

Selected Deque Operations - 1

- deque<T> someDeque;
  - Creates deque with no elements
- deque<T> someDeque (int someSize);
  - Creates deque with someSize elements, each created using the default constructor for type T
- deque<T> someDeque (int someSize, T value);
  - Creates deque with someSize elements of type T, all initialized to value
- ~deque<T>()
  - Destructor

Selected Deque Operations - 2

- size()
  - Number of elements currently stored
- max_size()
  - Maximum number of elements that can be stored without reallocation
- empty()
  - Returns true if empty, false otherwise
- front()
  - Returns first element but does not check to see if it exists
- back()
  - Returns last element but does not check to see if it exists

Selected Deque Operations - 3

- operator [ ]
  - Index into deque as if it is an array but bounds checking is not performed
- push_back(T someValue)
  - Adds someValue to back of deque
- push_front(T someValue)
  - Adds someValue to front of deque
- pop_back()
  - Removes last element from back of deque but does not return it
- pop_front()
  - Removes last element from front of deque but does not return it

Selected Deque Operations - 4

- at(int someIndex)
  - Returns value at position someIndex, throwing exception if someIndex is out of range

```cpp
//
// deque1.cpp
//
#include <iostream>
#include <deque>
#include <algorithm>
using namespace std;

void Print(deque<char> d);

int main()
{
  deque<char> d;

  cout << "Deque Size = " << d.size() << endl;

  d.push_front('a');
  d.push_back('z');
  cout << "Front element = " << d.front() << endl;
  cout << "Back element = " << d.back() << endl;
  d.pop_front();
  cout << "Front element = " << d.front() << endl;
  cout << "Back element = " << d.back() << endl;

  d.push_back('x');
  d.push_front('w');
  d.push_back('b');
  d.push_front('c');
  Print(d);
  sort(d.begin(), d.end());
  Print(d);
  cout << "Deque Size = " << d.size() << endl;

  return 0;
} // End main()

void Print(deque<char> d)
{
  for(int k = 0; k < d.size(); k++)
    cout << d.at(k) << ' ';
  cout << endl;
}
```

```
-bash-$ ./a.out
Deque Size = 0
Front element = a
Back element = z
Front element = z
Back element = z
c w z x b
b c w x z
Deque Size = 5
-bash-$
```

Container Adapters

- Why don’t we see the familiar containers (like stack and queue) in the STL?

- ***Container Adapters***
  - Combine an STL container, such as a deque, with a familiar interface to mimic other common containers

Stacks and the STL

- Traditional Stack Interface
  - push()
    - Implemented with deque push_back()
  - pop()
    - Implemented with deque pop_back()
  - top()
    - Implemented with deque back()
  - size()
    - Implemented with deque size()
  - empty()
    - Implemented with deque empty()
- #include <stack>

stack Example

```
#include <iostream>
#include <stack>
using namespace std;

int main()
{
  stack<int> s;

  s.push(5);
  s.push(10);
  s.push(15);
  s.push(20);

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

  return 0;
} // End main()
```

```
$ ./stack_example
20
15
10
5
$
```

Queues and the STL

- Traditional Queue Interface
  - push() "Enqueue"
    - Implemented with deque push_back()
  - pop() "Dequeue"
    - Implemented with deque pop_front()
  - front()
    - Implemented with deque front()
  - size()
    - Implemented with deque size()
  - empty()
    - Implemented with deque empty()
  - Etc.
- #include <queue>

queue Example

```
#include <iostream>
#include <queue>
using namespace std;

int main()
{
  queue<int> q;

  q.push(5);
  q.push(10);
  q.push(15);
  q.push(20);

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

  return 0;
} // End main()
```

```
$ ./queue_example
5
10
15
20
$
```