

CS 368 Announcements

Wednesday, December 3, 2013

Program p5 – due Wednesday, 12/11, at 10:00 pm – 10%

Last Time

- Ch. 7 (Templates)
- templated functions
- templated classes
- compiling with templates
- more template features

Today

- start Ch. 10 (Collections: The STL)
- STL overview
- containers
- iterators

Next Time

- finish Ch. 10
- function objects (functors)
- algorithms

Standard Template Library (STL)

reference & download: <http://www.sgi.com/tech/stl/>

STL is a library of:

- containers
- algorithms
- iterators

Find Algorithm:

```
template <class InputIterator, class T>
InputIterator find(InputIterator first, InputIterator last,
                  const T& value) {
    while (first != last && *first != value) ++first;
    return first;
}
```

STL makes heavy use of:

- templates
- operator overloading
- iterators

Some STL Containers

Sequence Containers

- vector
- list
- deque

Associative Containers

- set
- multiset

- map
- multimap

- + hash versions

Container Adaptors

- stack
- queue
- priority_queue

Container Operations

Operations all containers support:

- `int size() const`
- `void clear()`
- `bool empty() const`
- some kind of add op

Sequence container operations:

Iterators

Each container defines these iterator member functions:

```
iterator begin( );  
const_iterator begin( ) const;  
iterator end( );  
const_iterator end( ) const;
```

Using Iterators

```
list<double> L;  
L.push_back(1.2);  
L.push_front(3.4);  
L.insert(L.begin(), 5.6);  
L.insert(L.end(), 7.8);
```

```
list<double>::const_iterator iter;  
for (iter = L.begin(); iter != L.end(); ++iter)  
    cout << *iter << " ";  
cout << endl;
```

Iterator “Concepts”

Input Iterator

Output Iterator

Forward Iterator

Bidirectional Iterator

Random-access Iterator

Iterator Operations

Input Iterator

`*iter`

`iter1 == iter2`

`iter1 != iter2`

Forward/Input/Output Iterator

`++iter` and `iter++`

Bidirectional Iterator

`--iter` and `iter--`

Random-access Iterator

`iter+=k`

`iter+k`