Iterators and STL Containers

CMSC 202

Warmup

Write the class definition for the templated Bag class
- A bag has:
  - Random insertion
  - Random removal

STL

- Standard Template Library
- Why use it?
  - Good programmers know what to write.
    Great ones know what to reuse.
    - Paraphrase from "The Cathedral and the Bazaar"
      - A must-read for any computer scientist
- STL provides reusable code
  - Linked list, vector, map, multimap, pair, set, multiset, queue, stack, …
  - Don't reinvent the wheel…

List

- Linked List container
  - No random access (does not support operator[] or at())
- Essential operations
  - insert()
  - push_back()
  - push_front()
  - pop_front()
  - pop_back()
  - erase()

Set and Multiset

- Set
  - Sorted collection of unique elements
  - Cannot change value of an element
  - No random access
- Multiset
  - Allows duplicate elements
- Essential operations
  - insert()
  - erase()
  - count(element)
  - find(element)

Pair

- Pair
  - Connects two items into a single object
- Essential data
  - first
    - gets the first member of pair
  - second
    - gets the second member of pair
- Example

```
pair<int, string> hello( 5, “Hello”);
cout << hello.second << endl; // Hello
```

Map and Multimap

- Map
  - Stores key/value pairs, sorted by key
  - Value is modifiable, key is not
  - Key must be unique
- Multimap
  - Allows duplicate keys
- Essential operations
  - insert()
  - erase()
  - count(key)
  - find(key)

Iterators

Where have we seen these before???

- Problem
  - Not all STL classes provide random access
  - How do we do “for each element in X”?
- Solution
  - Iterators
    - “Special” pointers
    - “Iterate” through each item in the collection
    - Several types
      - Bidirectional
      - Const bidirectional

Why is this necessary?

Why can’t we just use a normal pointer?

What does const mean?

Iterators

- Essential operations
  - begin()
    - Returns an iterator to first item in collection
  - end()
    - Returns an iterator ONE BEYOND the last item in collection
      - How does this simplify things?
        - If the collection is empty, begin() == end()

Set Example

```
int main ( )
{
   set<int> iSet;

   iSet.insert(4);
   iSet.insert(12);
   iSet.insert(7);


   // this looping construct works for all containers

   set<int>::const_iterator position;

   for (position = iSet.begin(); position != iSet.end(); ++position)
   {
      cout << *position << endl;
   }
   return 0;
}
```

Map Example

```
int main ( )
{
   // create an empty map using strings
   // as keys and floats as values
   map<string, float> stocks;


   // insert some stock prices
   stocks.insert( make_pair("IBM",  42.50));
   stocks.insert( make_pair("XYZ",   2.50));
   stocks.insert( make_pair("WX",    0.50));

   // instantiate an iterator for the map
   map<string, float>::iterator position;

   // print all the stocks
   for (position = stocks.begin(); position != stocks.end(); ++position)
      cout << "( " << position->first << ", " << position->second << " )\n";

   return 0;
}
```

Iterators - Overloaded Operators

- * (pointer dereference)
  - Dereferences the iterator
- ++
  - Moves forward to next element
- --
  - Moves backward to previous element
- ==
  - True if two iterators point to same element
- !=
  - False if two iterators point to different elements
- =
  - Assignment, makes two iterators point to same element

Iterators and Collection Methods

- erase(iterator)
  - Parameter is an iterator
- Can have as many iterators into a collection as necessary

Practice

- Create a vector of integers
- Using an iterator and a loop
  - Change each integer to be the value of its square
- Using an iterator and a second loop
  - Print each item in reverse order

Challenge

- Using a map, create a collection of student grades
  - Key
    - Student ID
  - Value
    - Grade they want in this course
- Store 10 students and their desired grade
- Iterate through the map
  - Print each key/value pair in the map
- What sorting mechanism did the map use?
  - How would we specify that we wanted it sorted another way?