The Standard Template Library (STL) Containers

Curtis Larsen

DSU-CIT

Spring 2014
1 Containers
   Using containers
     string
     pair
     map
     set

2 Iterators
   Using iterators
     string
     map
     set

3 Challenges
   string
   set
   map
Using the string

Common operations

- `std::string s;` // create a string
- `std::string s("hello");` // create a string
- `std::string s = "world";` // create a string
- `y = s.size();` // get number of chars in string
- `y = s[i]; s[i] = y;` // get/set char at position
- `s += t;` // concatenate
- `s.push_back('x');` // append 1 char
- `const char *z = s.c_str();` // extract c-string
- `s.clear();` // remove contents
- `MANY OTHERS;` // several useful string operations
Using the pair

Common operations

- `std::pair<type1, type2> p;` // create a pair
- `std::pair<type1, type2> p(x1, x2);` // create a pair
- `y1 = p.first;` // read value of first item
- `y2 = p.second;` // read value of second item
Using the map

Common operations

• `std::map<key_type, data_type> m;`  // create a map
• `y = m[k];`  // read item value, if exists, else default object
• `m[k] = y;`  // write item value, create if doesn’t exist
• `m.insert(std::pair<key_type, data_type>(k,x));`  // add item to map
• `m.erase(k);`  // remove item from map
• `m.clear();`  // remove all items from map
Using the set

Common operations

- `std::set<type> s;` // create a set
- `s.size();` // number of elements in the set
- `s.clear();` // remove all items from set
- `s.erase(x);` // remove item from set
- `s.insert(x);` // add item to set
- `iter = s.find(x);` // find item in set
- `y = s.count(x);` // count item in set
Element pointers

- Strings can be resized, and we can traverse using index-like notation:

  ```cpp
  std::string s; s.resize(5); int i;
  for(i = 0; i < s.size(); i++) {
      s[i] = '0' + i;
  }
  ```

- With strings, we can also traverse using a pointer-like interface:

  ```cpp
  std::string::iterator sit; i = 0;
  for(sit = s.begin(); sit != s.end(); sit++) {
      *sit = 'a' + i; i++;
  }
  ```
Element pointers

- Maps can not be resized, except for adding or removing items:
  ```cpp
  std::map<char, int> m; char c; int i;
  for(i = 0; i < 5; i++) {
    c = 'a' + i; m[c] = i;
  }
  ```

- With maps, given indexes, we can traverse using an index-like interface:
  ```cpp
  for(c = 'a', i = 0; c < 'a' + 5; c++, i++) {
    m[c] = i*i;
  }
  ```

- With maps, we can also traverse using a pointer-like interface:
  ```cpp
  std::map<char, int>::iterator mitr; i = 0;
  for(mitr = m.begin(); mitr != m.end(); mitr++) {
    mitr->second = i*i; i++;
  }
  ```
Element pointers

- Sets can not be resized, except for adding or removing items:

```cpp
std::set<char> s; char c; int i;
for(i = 0; i < 5; i++) {
    c = 'a' + std::rand() % 26; s.insert(c);
}
```

- With sets, we can also traverse using a pointer-like interface:

```cpp
std::set<char>::iterator sitr;
for(sitr = s.begin(); sitr != s.end(); sitr++) {
    c = *sitr - 'a' + 'A';
    s.erase(*sitr); s.insert(c);
}
```
string Challenges

Write programs that:

• Reads words (white space separated text) from standard input, until end of input. Display each word to standard output.

• Reads words from standard input, until end of input. Add each word to a string. After end of input, display the accumulated string to standard output.

• Reads words from standard input, until end of input. As each word is read, reverse the text, and display to standard output.

• Reads words from standard input, until end of input. Insert each word into a vector. After end of input, display the contents of the vector.

• Other string manipulations you can think of.
Write programs that:

- Reads words (white space separated text) from standard input, until end of input. Store each word in a set. After end of input display the members of the set to standard output.
- Read numbers from standard input. After end of input, display the unique numbers that were read from input.
- Other set manipulations you can think of.
map Challenges

Write programs that:

- Count the number of times each unique word appears in standard input. After end of input, display each word and the number of times it appeared.
- Same as above, but don’t display all words. Just randomly pick 100 words and display them to standard output.
- Same as above, but translate each word by moving to lower case and removing all non-alpha characters.
- Same as above, display the probability of each word appearing, instead of it’s count.
- After collecting information from previous program, randomly pick 100 words based on the probabilities, and display them to the output.
- Other map manipulations you can think of.