CS 162, Lecture 23: Standard Template Library

23 May 2018

Standard Template Library

- Collection of common abstract data types
- When found in the STL, abstract data type == container classes
- Data Structure: a particular way of storing and organizing data
  - Array, Linked List, Trees
- Abstract Data Type: the rules that govern how data is accessed
  - Stack – Last in, First Out
  - Queue – First in, First Out
  - Priority Queue – Highest priority out first

Types of Containers

- Sequential: arrange data items into a list such that there is a first, next, and so on up to last
- Adapters: implemented on top of other classes (abstract data types)
- Associative: very simple databases (set, map)

Iterators

- A construct that allow you to cycle through the data items stored in a data structure
- Generalization of a pointer: typically implemented as a pointer but is abstracted so we don’t have to deal with it
- Every container class in STL has its own iterator

access.engr.orst.edu - PuTTY

```
1  #include <iostream>
2  #include <vector>
3
4  using namespace std;
5
6  int main() {
7
8          vector<int> v;
9
10         for(int i=0; i<5; i++) {
11                 v.push_back(i);
12         }
13         cout << "Print v" << endl;
14
15         vector<int>::iterator itr;
16
17         for(itr = v.begin(); itr != v.end(); itr++) {
18                 cout << *itr << " ";
19         }
20         cout << endl;
21
22         return 0;
23 }
~
"iter.cpp" 23L, 305C                                    23,1          All
```

Overload Operators

- ++ move to next item
- -- move to previous item
- == equality test
- != not equal test

Types of Iterators

- Forward: ++ works on the iterator
- Bidirectional: both ++ and -- work on the iterator
- Random Access: ++, --, and random access all work
- Constant: doesn’t allow changes to be made to element at its location
- Mutable: can change the element at its location
- Reverse: can be used to cycle through all elements of a container with bidirectional iterators
- Input: forward iterator that can be used with input stream
- Output: forward iterator to be used with output stream

Linked Lists

- A list constructed using pointers
- Can grow and shrink while the programming is running
- Not stored contiguously in memory
- Use structs to create

```
struct Node {
        int val;
        node* next;
};
```

Singly Linked List