G205 Fundamentals of Computer Engineering

CLASS 4, Mon Sept. 20 2004

Stefano Basagni

Fall 2004

M-W, 1:30pm-3:10pm

C++ Libraries

- Concept of programming language library
- C++ Standard library
- C++ Standard Template Library = powerful, template-based, reusable software components

C++ Standard Template Library (STL)

- Implements common data structures and related algorithms
- Three key components:
  - Containers: Popular templatized data structures
  - Iterators: like pointers, used to manipulate STL-container elements
  - Algorithms: Functions that performs data popular data manipulation

STL Containers

- An STL container implements a (popular) data structure
- Three major categories:
  - Sequence containers, for linear data structures (lists, arrays, etc.)
  - Associative containers, non-linear data structures (sets, maps, etc.)
  - Container adapters: Sequential containers in a constrained manner (stacks, queues, etc.)

STL Container Classes

- Sequence containers:
  - vector: rapid direct access anywhere, dynamic size
  - deque: rapid ins and del at front and back, rapid direct access
  - list: doubly-linked list, rapid ins and del anywhere
- Associative containers:
  - set, multiset: rapid lookup without/with duplicates
  - map, multimap: one-to-one/many mapping
- Container adapters:
  - stack: LIFO
  - queue: FIFO
  - priority_queue: highest priority element is the first out

Member Functions for All STL Containers

- Default constructor, copy constructor, destructor
- empty
- max_size, size
- =, <, <=, >, >=, ==, !=
- swap

Member Functions for FIRST CLASS Containers Only

- First class containers = sequential and associative containers only
- Member functions:
  - begin
  - end
  - rbegin
  - rend
  - erase
  - clear

STL Header Files

- <vector>
- <deque>
- <list>
- <set> (for set and multiset)
- <map> (for map and multimap)
- <queue> (for queue and priority_queue)
- <stack>

(All in namespace std)

STL Iterators

- Used to point to elements of first class containers
- Common features with pointers:
  - Dereferencing operator * allows to use the element the iterator is pointing to
  - The ++ operation on an iterator moves the iterator to the next element in the container
  - Iterators can be constant (const_iterator for non modifiable elements) or not (iterator)

Categories of Iterators

- Input: Read an element from a container, only forward
- Output: Write an element to a container, only forward
- Forward = input + output
- Bidirectional: All of the above, multi-pass algorithms
- Random access: Can directly access any element, jumping an arbitrary number of elements

Categories Supported by STL Containers

- Sequence containers:
  - vector: random access
  - deque: random access
  - list: bidirectional
- Associative containers:
  - set, multiset: bidirectional
  - map, multimap: bidirectional
- Container adapters: No iterators supported

STL Algorithms

- Over 70 standard algorithms for manipulating containers:
  - Inserting
  - Deleting
  - Searching
  - Sorting
- Almost all for any type of container
- Often they use iterators and return iterators

STL Algorithms, examples

- find() locates an element and return an iterator to that element or the end() iterator if the element is not present
- remove() removes all occurrences of a given element from the corresponding container

Vector Container

- Class vector is based on array
- It can change size dynamically
- Vectors can be assigned to one another (impossible with pointer-based C-like arrays)
- Insertion at the back of a vector is efficient and the vector resizes if needed
- Vector subscripting DOES not perform range checking, but class vector allows it via the member function at

Vector Operations

- Common to all sequence containers:
  - front
  - back
  - push_back
  - pop_back
- All STL algorithms can operate on a vector

Vector, the Use, 1

```
#include <iostream>

using std::cout;
using std::cin;
using std::endl;

#include <vector>

template <class T>
void print_vector(const std::vector< T > &integers2);

int main() {
  std::vector<int> integers;
```

Vector, the Use, 2

```
    cout << integers.size() << ` ` << integers.capacity() << endl;
    integers.push_back(2);
    integers.push_back(3);
    integers.push_back(4);
    cout << integers.size() << " " << integers.capacity() << endl;
    print_vector( integers );
    std::vector<int>::reverse_iterator ri;
    for (ri = integers.rbegin(); ri != integers.rend(); ++ri)
      cout << *ri << ` `;
    cout << endl;
    return 0;
}
```

Vector, the Use, 3

```
template<class T>
void printVector(const std::vector<T> &integers2) {
  std::vector<T>::const_iterator ci;
  for (ci = integers2.begin(); ci != integers2.end(); ci++)
    cout << *ci << ' ';
}
```

List Container

- Class list is based on pointers (doubly linked list)
- Supports bidirectional iterators
- Insertion and deletion at any location is implemented efficiently
- All member functions of STL containers are provided

List Operations

- Eight new member functions:
  - splice
  - push
  - push_front
  - pop_front
  - remove
  - unique
  - merge
  - reverse
  - sort

STL Used for Defining Other Data Structures

- The class Matrix, for bidimensional matrices
- #include <matrix>
- Matrix< int > M(10,15,0): defines a 10x15 matrix. Entries are all 0
- Defined as a vector of vectors

Assignments

- Deitel & Deitel book, chapter 21
- Updated information on the class web page:

  www.ece.neu.edu/courses/eceg205/2004fa