C++ STL Vector Container

And C++ STL Array container,

A few STL algorithms

STL Vector Container

- Header file <vector>
- Important member functions
  - begin() and end(): return iterators
  - clear(): delete all elements
  - push_back(): insert at end
  - pop_back(): delete last element
  - size(): number of elements
  - empty(): if vector is empty
  - resize(): change vector size

Maximum and Average Values

- See max_avg.txt and max_avg.cpp
  - To compile: make max_avg.x
  - In general, in order to compile a program with name.cpp, you can type: make name.x, due to the way we write our makefile

Word Puzzle

- Problem description
  - word_puzzle.pdf
  - Section 1.1 in textbook
- Key question
  - How to maintain word list (dictionary)
  - Throughout the semester, we will show a number of different solutions using different containers

- Find the words in the puzzle, given a 2-D array and a word-list.

	1	2	3	4
1	t	h	i	s
2	w	a	t	s
3	o	a	h	g
4	f	g	d	t

Word Puzzle

- Explaining the source code
  - word_puzzle_vector.h, word_puzzle_vector.cpp
  - Pay attention to the use of I/O streams, stringstreams, strings, and vector
  - To compile: make

C++ STL Array Container

- New in C++11
- Similar to array in C/C++ in that Array has fixed size memory
  - Will not grow or shrink like other containers such as Vector
- Support similar interfaces as other containers
  - But with notable difference (see next slide)
- Likely more efficient than Vector
  - Use vector if you need to grow or shrink container
  - Use vector if you are not sure

STL Array Container

- Header file <array>
  - template < class T, size_t N > class array;
- Important member functions
  - begin(), end(), for iterator support
  - size(), max_size(), size of array
  - empty(), test if size of array is zero
  - Index operator[], at(), access element at specified position
  - front(), back(), refer to first and last element, respectively
  - data(), return a pointer to internal data (C/C++ pointer)
  - fill(), set all elements in array to the specified value

STL Array Container

- See array_sort.cpp
  - To compile: make array_sort.x
  - Copy is an STL algorithm in <algorithm>
  - http://www.cplusplus.com/reference/algorithm/copy/

  - Similarly, sort is an STL algorithm
  - http://www.cplusplus.com/reference/algorithm/sort/

  - Note also the behavior of size(), max_size(), and fill()

STL Algorithm sort()

- Function signature

  void sort(RandomAccessIterator first, RandomAccessIterator last);
  void sort(RandomAccessIterator first, RandomAccessIterator last, Compare comp);

  - Sort elements in the range [first, last)
  - Note that containers must be random access containers
  - The first version using operator<() overloaded for the corresponding data type
  - The second version using function object of type Compare
  - You can also use function or lambda function
    - Function objects and lambda functions discussed in next few slides
  - http://www.cplusplus.com/reference/algorithm/sort/

Using Regular Function with Sort()

- You can use regular function to compare elements in container to be sorted
  - See r3/example5.c

Function Objects (Functors)

- Objects whose primary purpose is to define a function
  - To be passed into another function, such as sort()
  - Read Section 1.6.4 in the textbook
  - Review lecture notes ch1_cpptemplate.pptx
  - http://www.cplusplus.com/reference/functional/

  - Compared to function (and lambda function), functors have the advantages that you can “configure” the comparison, for example, you only want to compare elements within certain range, which can be specified by the member variables of functors

Function Objects

- Using function objects with the sort() function
  - See r3/example2.cpp

Lambda or Unnamed Functions

- Since C++2011
- Similar to regular function, but you do not pre-define it, instead, you specify it on the function call (for example, sort())
- It does not have a function name
- https://msdn.microsoft.com/en-us/library/dd293608.aspx

- See r3/example4.cpp

STL Algorithm find()

- Function signature
  - **InputIterator find (InputIterator first, InputIterator last, const T& val);**

  - Search val in the range [first, last)
  - Return iterator to first appearance of val in the range
  - Return last if not found in the range
  - http://www.cplusplus.com/reference/algorithm/find/

- See examples/r3/example1.cpp

STL Algorithm max_element()

- Function signature

  ForwardIterator max_element(ForwardIterator first, ForwardIterator last);
  ForwardIterator max_element(ForwardIterator first, ForwardIterator last, Compare comp);

  - Return largest value in the range [first, last)
  - The first version using operator<() overloaded for the corresponding data type
  - The second version using function object of type Compare
  - You can also use lambda function for the comparison
  - http://www.cplusplus.com/reference/algorithm/max_element/

- See r3/example3.cpp