Containers

• Standard C++ Library provides classes that implement containers of objects
• Can store any object that can be copied
• Types of containers:
  – *Sequence containers*, for ordered collections of elements
  – *Associative containers*, for unordered collections
    • elements identified by other means
    • examples: sets and maps
Sequence Containers

- **vector<T>** stores elements of type $T$
  - elements stored contiguously in memory
  - elements can be relocated if vector grows, reducing efficiency
- **deque<T>** is a double-ended queue
  - fast addition at both ends; middle slower
  - doesn't relocate elements
  - access slightly slower than vector
- **list<T>** is a doubly-linked list
  - element access slower, no random access
  - insertion/deletion always fast

Iterator Member Functions

- Used to iterate over all elements within a container
- **begin()** points to first element
- **end()** points to location beyond last element, to know when to end iteration
- **rbegin()** points to last element, for reverse iteration
- **rend()** points to location before first element, for concluding reverse iteration
Iterator Types

- For defining your own iterators
- Iterator type names preceded by container type and ::
- Types: iterator for forward iteration, reverse_iterator for backward
- For const containers, precede iterator type name with const_
- Can initialize iterators from container member functions like begin()
- Then, can use ++, *, etc. on iterator

Modifying Containers

- push_back(t) adds t to end,
  push_front(t) to beginning (not for vectors)
- insert(iter, t) adds t before position pointed to by iter
- erase(iter) removes element at iter
- Can also use ranges of iterators with insert and erase
- To remove all elements: clear()
Sizing Containers

- Can declare sequential containers with an `unsigned int` initializer to pre-allocate that number of elements
- Can use `resize(unsigned int)` member function to change size later
- In both cases, can add argument of value type to initialize pre-allocated elements
- *Always* pre-allocate if possible, for efficiency

More **string** Operations

- Strings are containers too
- Additional string-specific overloads of `insert` and `erase` available
- Use `assign` to replace a string with another or `replace` to replace a portion of a string
- `substr` extracts a portion of a string
- Use `find`, `rfind` to search within a string, compare to compare strings
- Many more! See cppreference.com
Stacks and Queues

• A stack is a sequence in which elements are added/removed in a LIFO (last in/first out) manner
• A queue uses FIFO (first in/first out)
• A priority_queue orders elements by a given priority
• Use `push(item)` to add element, `pop()` to remove first element
• Use `top()` for first element (for queue, use `front()`, `back()` to get first, last)

Next Time

• All about Associative Containers