Some Useful STL Functions*

Jeffrey D. Oldham

2000 Feb 12

The Standard Template Library contains many functions working on any container. We highlight a few here.

- `find`
  - `InputIterator find(InputIterator first, InputIterator last, T value)`
  - returns the leftmost occurrence of `value` if present
  - returns `last` if `value` not present
  - `#include <algorithm>`
- `copy`
  - `OutputIterator copy(InputIterator first, InputIterator last, OutputIterator result)`
  - copies everything from [`first`,`last`) to the container represented by `result`
  - output container must be large enough already!
  - returns a pointer one past the last changed position
  - `#include <algorithm>`
- `transform`
  - `OutputIterator tranform(InputIterator first, InputIterator last, OutputIterator result, UnaryFunction op)`
  - fills output container with result of `op` applied to each member of the input container
  - output container must be large enough already!
  - the input and output container can be the same
  - `#include <algorithm>`
- `for_each`

*©2000 Jeffrey D. Oldham (`oldham@cs.trinity.edu`). All rights reserved. This document may not be redistributed in any form without the express permission of the author.

  - `UnaryFunction for_each(InputIterator first, InputIterator last, UnaryFunction op)`
  - applies `op` to each element in the input container
  - the result of applying `op` is ignored. That is `op` should probably be a `void` function
  - `#include <algorithm>`

- `sort`
  - `void sort(RandomAccessIterator first, RandomAccessIterator last)` or
  - `void sort(RandomAccessIterator first, RandomAccessIterator last, ComparatorFunctio comp)`
  - sorts the specified range
  - uses the specified comparator function `comp` if present; should be a binary function acting like `<` for any two container elements
  - uses the container elements' `<` if `comp` not specified
  - `#include <algorithm>`

- `accumulate`
  - `T accumulate(InputIterator first, InputIterator last, T init)`
  - `T accumulate(InputIterator first, InputIterator last, T init, BinaryFunction op)`
  - if `op` not specified, do `+` on all the elements starting with an initial value of `init`, e.g., 0
  - if `op` specified, do `op` on all the elements starting with an initial value of `init`
  - ex: `accumulate(v.begin(), v.end(), 0)` yields sum of all vector elements
  - ex: `accumulate(v.begin(), v.end(), 1, multiplies<int>())` yields product of all vector elements
  - `#include <numeric>`

- `inner_product`
  - `T inner_product(InputIterator1 first1, InputIterator1 last1, InputIterator2 first2, T init)`
  - computes the dot product of the two specified containers starting with value `init`
  - the second container must be at least as large as the first container
  - `#include <numeric>`