STL Output Iterators

STL Output Iterators

- *Output* iterator is a type that provides a mechanism for storing (but not accessing) a sequence of values

```
template<typename InputIterator,
            typename OutputIterator>
OutputIterator copy
   (InputIterator first,
    InputIterator last,
    OutputIterator result) {
   for (; first != last;
          ++first, ++result)
     *result = *first;
     return result;
}
vector<int> v;

copy (istream_iterator<int> (cin),
        istream_iterator<int>(),
        back_inserter(v));
```

See www.cplusplus.com/reference/iterator/OutputIterator

STL Output Iterators

- *Output* iterators are in some sense the converse of input iterators, but have a more restrictive interface:
  - Must support non-const operator *
    - *e.g., *iter = 3*
  - Operators = & == & != need not be defined (but could be)

```
template<typename InputIterator,
         typename OutputIterator>
OutputIterator copy
  (InputIterator first,
   InputIterator last,
   OutputIterator result) {
  for (; first != last;
       ++first, ++result)
    *result = *first;
    return result;
}
vector<int> v;

copy (istream_iterator<int> (cin),
      istream_iterator<int>(),
      back_inserter(v));
```

STL Output Iterators

- Intuitively, an *output* iterator is like a tape where you can write a value to the current location & you can advance to the next location
  - However, but you cannot read values & you cannot back up or rewind

STL Output Iterator Example

```
int main () {
  // An initially empty vector.
  vector<int> v;

  // copy contents of cin as "int" and store at the end of vector v.
  for (istream_iterator<int> i (cin);
       i != istream_iterator<int> ();
       ++i)
    // Add int to the end of the vector.
    v.push_back (*i);


  // Use STL copy() algorithm along with back_inserter()!
  copy (istream_iterator<int> (cin),
        istream_iterator<int>(),
        back_inserter(v));
}
```

See github.com/douglascraigschmidt/CPlusPlus/tree/master/STL/S-04/4.4/4.4a