STL Iterator Adapters

STL Iterator Adapters

- STL algorithms that copy elements are passed an iterator that marks the position within a container to begin copying
  - *e.g.*, `copy()`, `unique_copy()`, `copy_backwards()`, `remove_copy()`, & `replace_copy()`

```
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>(),
      ??v??);
```

STL Iterator Adapters

- Each copy requires the target container is of a sufficient size to hold the set of assigned elements

```
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>(),
      ??v??);
```

STL Iterator Adapters

- We can use iterator adapters to expand the containers as we perform the algorithm

```
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 Iterator Adapters

- We can use iterator adapters to expand the containers as we perform the algorithm
  - Start with an empty container, & use the inserter along with the algorithms to make the container grow only as needed

```
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 Iterator Adapters Examples

- **`back_inserter()`** causes the container's **`push_back()`** operator to be invoked in place of the assignment operator
  - The argument passed to **`back_inserter()`** is the container itself

```
std::vector<int> v;

std::vector<int>::iterator in_begin =
  std::istream_iterator<int>(std::cin);

std::vector<int>::iterator in_end =
  std::istream_iterator<int>();

std::copy (in_begin,
           in_end,
           std::back_inserter (v));
```

See github.com/douglascraigschmidt/CPlusPlus/tree/master/STL/S-06