Outline

- STL overview
- Prolog examples
Standard Template Library (STL)

- a C++ library of:
  - container classes
  - algorithms
  - iterators
- it is a *generic* library
iterators are a generalization of pointers
they are an interface between containers and algorithms
several sorts of iterators:
- **Input Iterators** only guarantee read access
- **Output Iterators** only guarantee write access
- **Bidirectional, Random Access** etc.
Concepts: Motivating example

Prototype:

```cpp
template <class InputIterator, class EqualityComparable>
InputIterator find (InputIterator first,
                   InputIterator last,
                   const EqualityComparable& value);
```

Usage:

```cpp
list<int> L;
L.push_back(3);
L.push_back(6);
L.push_back(7);
list<int>::iterator result = find(L.begin(), L.end(), 3);
assert (result == L.end() || *result == 3);
```
A concept is a set of requirements on a type. We say that a type \textit{conforms to a concept} or that it \textit{is a model of a concept} if it satisfies all those requirements. Concepts are not part of the C++ programming language. Concepts make it easy to separate implementation from interface and reuse and combine components.
We say that a concept X is a refinement of a concept Y if all the types that model Y also model X.

- very similar to inheritance of C++ classes
- both iterators and containers are organized into hierarchies of concepts
Adaptors

- sometimes a class will have the functionality you seek but not the right interface

- an **adaptor class** is a class that acts like a “translator” by “adapting” the messages you want to send to into messages the other class wants to receive.

- examples:
  - container adaptors: stack, queue, priority queue
  - iterator adaptors: stream iterators, insertion iterators
main ()
{
    vector<int> v; // create an empty vector of integers
    int input;
    while (cin >> input) // while not end of file
        v.push_back (input); // append to vector

    sort(v.begin(), v.end());

    int n = v.size();
    for (int i = 0; i < n; i++)
        cout << v[i] << "\n";
}
main ()
{
    vector<int> v(10);
    istream_iterator<int> start (cin);
    istream_iterator<int> end;
    vector<int>::iterator dest = v.begin();

    copy (start, end, dest);
    sort(v.begin(), v.end());
    copy (v.begin(), v.end(),
      ostream_iterator<int>(cout, "\n"));
}
permit algorithms to operate in insert mode rather than overwrite mode (default mode)

examples:

- `back_inserter()`
- `front_inserter()`
- `inserter`
Demo.