STL Cheat Sheet

Creation

- Make an empty vector of integers.

```
vector< int > iseq1;
```

- Make a 10-element vector of doubles, each initialized to -1.

```
vector< double > iseq2( 10, -1 );
```

- A value that is a 10-element vector of ints, each initialized to 50.

```
vector< double >( 10, 50 );
```

- Make a string, integer pair initialized to "up", 15.

```
pair< string, int > myPair( "up", 15 );
```

- A value that is a double, integer pair containing 3.14, 7.

```
pair< double, int >( 3.14, 7 );  // Types given explicitly
make_pair( 3.14, 7 );              // Types inferred by the compiler.
```

- Make a 100-element vector of string, double pairs, each initialized to "height", -1.

```
vector< pair< string, double > > pseq( 100, make_pair( string("height"), -1.0 ) );
```

- Make an empty vector of vectors of ints.

```
vector< vector< int > > matrix1;
```

- Make a 10 $\times$ 20 vector of vectors of ints, each element initialized to 3.

```
vector< vector< int > > matrix2( 10, vector< int >( 20, 3 ) );
```

- Make an iterator that can point to an element of a vector of ints.

```
vector< int >::iterator pos;
```

Access and Modification

- Number of items in a vector (typically unsigned int)

```
iseq1.size()
```

- Number of rows in a vector of vectors.

```
matrix2.size()
```

- Number of elements in the first row of a vector of vectors.

  ```
  matrix2[ 0 ].size()
  ```

- Access first item in a vector (modifiable).

  ```
  iseq2.front()
  ```

- Access last item in a vector (modifiable).

  ```
  iseq2.back()
  ```

- Return an iterator pointing to the first element of the vector.

  ```
  iseq1.begin()
  ```

- Return an iterator pointing to the imaginary position one past the end of the vector.

  ```
  iseq1.end()
  ```

- Return the value of the element at index 5 in the vector (modifiable)

  ```
  iseq2[ 5 ]
  ```

- Value of row 7 in a vector of vectors (modifiable)

  ```
  matrix2[ 7 ]
  ```

- Value at row 7, column 3 in a vector of vectors (modifiable)

  ```
  matrix2[ 7 ][ 3 ]
  ```

- Compute the value of an iterator pointing to the element at index 5 of the vector.

  ```
  iseq1.begin() + 5
  ```

- Access first field of a pair (modifiable)

  ```
  myPair.first
  ```

- Access second field of a pair (modifiable)

  ```
  myPair.second
  ```

Insertion and Removal

- Add a integer to the end of a vector.

```
iseq1.push_back( 20 );
```

- Add a pair to the end of a vector.

```
pseq.push_back( make_pair( string("weight"), 175.5 ) );
```

- Remove last element in a vector.

```
iseq1.pop_back();
```

- Insert a value at the start of a vector (linear time).

```
pseq.insert( pseq.begin(), make_pair( string("weight"), 175.5 ) );
```

- Insert a value at position 5 in the vector (linear time).

```
iseq2.insert( pseq.begin() + 5, 99 );
```

- Append a new row of 100 elements (each set to zero) to the end of this vector of vectors.

```
matrix1.push_back( vector< int >( 100, 0 ) );
```

- Insert a new row of 55 elements (each initialized to 75) at the start of this vector of vectors.

```
matrix1.insert( matrix1.begin(), vector< int >( 55, 75 ) );
```

- Remove first element from a vector (linear time)

```
iseq2.erase( pseq.begin() );
```

- Remove element at index 7 element from a vector (linear time)

```
pseq.erase( pseq.begin() + 7 );
```

- Clear contents of the vector.

```
iseq2.clear();
```

- Empty the last row of a vector of vectors, but don't remove it.

```
matrix2.back().clear();
```

Supporting Algorithms

- Print out every element of a vector.

```
// Using integer index
for ( unsigned int i = 0; i < iseq1.size(); i++ )
  cout << iseq1[ i ] << endl;

// Using iterators
for ( vector< int > :: iterator pos = iseq1.begin(); pos != iseq1.end(); pos++ )
  cout << *pos << endl;
```

- Sort contents of vector based on the < operator.

```
sort( iseq2.begin(), iseq2.end() );
```

- Sort contents of vector of pairs, ordering by < for the first fields and using the second fields if first fields are identical.

```
sort( pseq.begin(), pseq.end() );
```

- Sort based on our own sorting function.

```
// Return true if a should come before b
bool myComparison( pair< string, double > const &a,
                   pair< string, double > const &b ) {
  if ( a.first.length() < b.first.length() )
    return true;
  if ( b.first.length() < a.first.length() )
    return false;

  return a.second < b.second;
}

sort( pseq.begin(), pseq.end(), myComparison );
```

- Return an iterator pointing to the first occurrence of the value 5 in a vector. If not found, return the given end iterator.

```
find( iseq1.begin(), iseq1.end(), 5 )
```

- Reverse sequence of values in the given vector.

```
reverse( iseq2.begin(), iseq2.end() );
```