C++ Standard Library Algorithms
(A very small sample)

Administrative

- Program 3 assigned

Outline Standard library Algorithms

- Containers (odds and ends)
- Sort
  - Algorithm or member
- Count and Find
- For each
- Transform
- Min Max

Container Bits - reserve

If you have an idea how many items a container will hold then allocate that amount to start

```
container.reserve(INITIAL_SIZE)
```

Avoids reallocation, saves time

Container Bits – is it empty?

- Can do either
  - If (c.size() == 0)… //O(n) time
  - If (c.empty()) //O(1) time ... Prefer this one
- Use empty() since it is a faster

Sort – Algorithmic and member

- If using sort(…) from algorithms then containers require random access iterators
  - Vector,string,deque

```
vector<int> myVect;
sort(myVect.begin(),myVect.end());
```

- Containers with non random access iterators have built in sorting
  - list

```
mylist.sort(compare_nocase);
```

Count

- Count - Is the value there? If so how many copies? Returns int.

```
// counting elements in container:
std::vector<int> myvector (myints, myints+8);
mycount = std::count (myvector.begin(), myvector.end(), 20);
std::cout << "20 appears " << mycount  << " times.\n";
```

- Count_if

```
bool IsOdd (int i) { return ((i%2)==1); }

int main () {
  std::vector<int> myvector;
  for (int i=1; i<10; i++) myvector.push_back(i); // myvector: 1 2 3 4 5 6 7 8 9

  int mycount = count_if (myvector.begin(), myvector.end(), IsOdd);
  std::cout << "myvector contains " << mycount  << " odd values.\n";

  return 0;
}
```

Find

- Find and find_if– Is it there? If so where? Returns iterator

```
bool IsOdd (int i) {
  return ((i%2)==1);
}

int main () {
  std::vector<int> myvector;

  myvector.push_back(10);
  myvector.push_back(25);
  myvector.push_back(40);
  myvector.push_back(55);

  std::vector<int>::iterator it = std::find_if (myvector.begin(), myvector.end(), IsOdd);
  std::cout << "The first odd value is " << *it << '\n';

  return 0;
}
```

For Each

- Way to operate on each element of a container
- Usable on most containers
- Does not modify order of elements
- Can modify individual elements

```
void outputvector(studentdata i){
    std::cout<<i.grade<<' ';
}
void foreach(){
    for_each(myvect.begin(), myvect.end(), outputvector);
}
```

Transform

- Applies an operation sequentially to the elements of a range and stores the result in the range that begins at *result*.
- Make sure your result container is large enough .

```
//takes a string and applies tolower on all members in the string
std::string ConvertToLowerCase(std::string text) {
    std::transform(text.begin(), text.end(), text.begin(), tolower);
    return text;
}
```

Min and Max

- Find the minimum or maximum value in a range.

```
//MIN and MAX-----------------------------------------------------------------
bool myfn(studentdata i, studentdata j) { return i.grade<j.grade; }
void minmax() {
    myvectiter = std::min_element(myvect.begin(), myvect.end(), myfn);
    std::cout << "\nThe worst grade is " << (*myvectiter).name << " the grade is
            << (*myvectiter).grade << '\n';

    myvectiter = std::max_element(myvect.begin(), myvect.end(), myfn);
    std::cout << "\nThe best grade is " << (*myvectiter).name << " the grade is
            << (*myvectiter).grade<< '\n';
}
```

Summary

- Don’t Reinvent the wheel. The standard library is your first stop when designing a project.
  - Choose data structure (container) based on which one performs best for your needs
  - Look in Algorithms etc.. before you write anything