

## **CS 368 Announcements**

### **Wednesday, December 11, 2013**

**Program p5** – due tonight, 12/11, at 10:00 pm – 10%

#### **Last Time**

- start Ch. 10 (Collections: The STL)
- STL overview
- containers
- iterators

#### **Today**

- finish Ch. 10
- iterators
- function objects (functors)
- algorithms
- course evals (Skrentny, CS 368 section 1)

## **Iterator “Concepts”**

**input iterator**

**output iterator**

**forward iterator**

**bidirectional iterator**

**random-access iterator**

## Iterator Operations

### Input Iterator

`*iter`

`iter1 == iter2`

`iter1 != iter2`

### Forward/Input/Output Iterator

`++iter` and `iter++`

### Bidirectional Iterator

`--iter` and `iter--`

### Random-access Iterator

`iter+=k`

`iter+k`

## Function Objects (functors)

### Defining a functor:

```
class IsPositive {
public:
    bool operator() (int n) const {
        return n > 0;
    }
};
```

### Using a functor:

```
IsPositive test;
int x;
cout << "Enter an integer: ";
cin >> x;
if (test(x))
    cout << x << " is positive" << endl;
else
    cout << x << " is not positive" << endl;
```

## Generic Algorithms

**Need to #include <algorithm>**

### Sorting

```
void sort( RandomAccessIterator begin,
          RandomAccessIterator end );

void stable_sort( RandomAccessIterator begin,
                 RandomAccessIterator end );
```

- iterators must be non-constant random-access iterators
- optional comparator argument
- example:

```
vector<int> V;
V.push_back(4);
V.push_back(8);
V.insert(V.begin(), 12);
V.insert(V.end(), 6);
print(V); // prints: 12 4 8 6
sort(V.begin(), V.end());
print(V); // prints: 4 6 8 12
```

### Searching

```
InputIterator find( InputIterator begin,
                   InputIterator end,
                   const EqualityComparable & x );
```

```
InputIterator find_if( InputIterator begin,
                      InputIterator end,
                      Predicate pred );
```

## Searching Example

```
vector<int>::iterator found, found1, found2, found3;

found = find(V.begin(), V.end(), 0);

if (found != V.end())
    cout << *found << endl;

found1 = find_if(V.begin(), V.end(), IsPositive());

if (found1 == V.end())
    cout << "no positive items" << endl;

else {
    cout << *found1 << endl;

    found2 = find_if(++found1, V.end(), IsPositive());

    if (found2 == V.end())
        cout << "no more positive items" << endl;
    else {
        cout << *found2 << endl;

        found3 = find_if(++found2, V.end(), IsPositive());

        if (found3 == V.end())
            cout << " no more positive items" << endl;
        else
            cout << *found3 << endl;
    }
}
```