Sorting: Natural and Custom Orders

Every data type can have a *natural ordering*, which is used to determine in which order objects of that type should be sorted. Data types can also have any number of *custom orderings*, which can be explicitly used to sort objects in a different order. Functions are available in the standard libraries to efficiently sort arrays and vectors (quicksort).

C++

```
#include <set>
#include <map>
#include <vector>
#include <algorithm>
#include <string>
using namespace std;
...

// Define a custom type:
class mytype {
  public:
    int foo;
    string bar;
};

// Natural ordering:
// Returns true if x < y,
// false if x >= y.
bool operator<(const mytype &x,
    const mytype &y) {
  if (x.foo != y.foo)
    return x.foo < y.foo;
  else return x.bar < y.bar;
}

// Custom ordering:
bool otherorder(const mytype &x,
    const mytype &y) {
  if (x.bar != y.bar)
    return x.bar < y.bar;
  else return x.foo < y.foo;
}

// Custom order set/map:
set<mytype, typeof(&otherorder)>
  s(&otherorder);
map<mytype, string,
  typeof(&otherorder)> m(&otherorder);

// Sorting:
vector<mytype> vec;
mytype ary[27];
sort(v.begin(), v.end());
sort(v.begin(), v.end(), &otherorder);
sort(ary, ary + 27);
sort(ary, ary + 27, &otherorder);
```

Java

```
import java.util.*;
...
// Define a custom type:
class MyType implements Comparable {
  public int foo;
  public String bar;

  // Natural ordering: returns >0 for >,
  // 0 for =, <0 for <
  public int compareTo(MyType other) {
    if (foo != other.foo)
      return foo - other.foo;
    else
      return bar.compareTo(other.bar);
  }
}

// Custom ordering:
class OtherOrder implements
    Comparator<MyType> {
  public int compare(MyType x, MyType y)
  {
    if (!x.bar.equals(y.bar))
      return x.bar.compareTo(y.bar);
    else
      return x.foo - y.foo;
  }
}

// Custom order set/map:
new TreeSet<MyType>(new OtherOrder());
new TreeMap<MyType, String>(
  new OtherOrder());

// Sorting:
List<MyType> vec;
MyType[] ary;
Collections.sort(vec);
Collections.sort(vec, new OtherOrder());
Arrays.sort(ary);
Arrays.sort(ary, new OtherOrder());
```