C++: `sort`

Sort `vectors` with STL `std::sort` function

`#include <algorithm>`

Modifies `vector`, arranging elements in ascending order according to < relation

- For numbers, < means less than
- For strings < means before, in ASCII order

Specify region of vector to sort by feeding in iterator to start and end

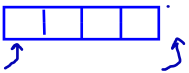
C++: sort

```
median.cpp:
#include <iostream>
#include <vector>
#include <algorithm>

using std::vector; using std::endl;
using std::cout;   using std::cin;
using std::sort;

int main() {
    vector<float> grades;
    float cur_grade;
    while(cin >> cur_grade) {
       grades.push_back(cur_grade);
    }
    sort(grades.begin(), grades.end());
    cout << "Median grade was " << grades[grades.size()/2] << endl;
    return 0;
}
```

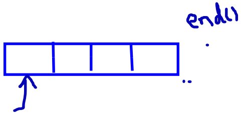
C++: sort

```
$ g++ -std=c++11 -pedantic -Wall -Wextra -c median.cpp
$ g++ -o median median.o
$ echo 49.6 48.2 84.8 3.4 33.1 | ./median
Median grade was 48.2
```

find



```
find.cpp:
#include <iostream>      // std::cout
#include <algorithm>     // std::find
#include <vector>        // std::vector

using std::cout;
using std::vector;

int main () {
    // using std::find with array and pointer:
    int arr[] = {1, 20, -2, 4};
    int * p;

    p = std::find (arr, arr + 4, 30);

    if (p != arr + 4)
        cout<<"value found in arr: "<<*p<<'\n';
```



```
  else
     cout<<"value 30 not found in arr\n";

  // using find with vector and iterator:
  vector<int> vec (arr, arr + 4);
  vector<int>::iterator it;

  it = std::find (vec.begin(), vec.end(), -2);

  if (it != vec.end())
    cout<<"value found in vec: "<<*it<<'\n';
  else
    cout<<"value -2 not found in vec\n";

  return 0;
}
```

find

```
$ g++ -std=c++11 -pedantic -Wall -Wextra -c find.cpp
$ g++ -o find find.o
$ ./find
value 30 not found in arr
value found in vec: -2
```

count

count.cpp:

```cpp
// count algorithm example
#include <iostream>     // std::cout
#include <algorithm>    // std::count
#include <vector>       // std::vector

using std::cout;
int main () {
    // counting elements in array:
    int arr[] = {10, 20, 30, 30, 20, 10, 10, 20};   // 8 elements
    int mycount = std::count (arr, arr + 8, 10);
    cout << "10 appears " << mycount << " times in arr.\n";

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

    return 0;
}
```

count

```
$ g++ -std=c++11 -pedantic -Wall -Wextra -c count.cpp
$ g++ -o count count.o
$ ./count
10 appears 3 times in arr.
20 appears 3 times in vec.
```

is_permutation

perm.cpp:

```
#include <iostream>      // std::cout
#include <algorithm>     // std::is_permutation
#include <array>         // std::array

int main () {
  std::array<int,5> foo = {1, 2, 3, 4, 5};
  std::array<int,5> bar = {3, 1, 4, 5, 2};

  if ( std::is_permutation (foo.begin(), foo.end(), bar.begin()) )
     std::cout << "foo and bar contain the same elements.\n";

  return 0;
}
```

```
$ g++ -std=c++11 -pedantic -Wall -Wextra -c perm.cpp
$ g++ -o perm perm.cpp
$ ./perm
foo and bar contain the same elements.
```

STL algorithm

List of all algorithm functions with examples:
`http://www.cplusplus.com/reference/algorithm/`