

## Week 6

### Exercise 45

../45/main.cc

```
1  #include <algorithm>
2  #include <iostream>
3  #include <iterator>
4  #include <set>
5  #include <string>
6
7  using namespace std;
8
9  int main(int argc, char const **argv)
10 {
11     set<string> iStrings;
12
13     cout << "Please enter delimited words to be sorted, end input with ^D \n";
14
15     copy(
16         istream_iterator<string>(cin),
17         istream_iterator<string>(),
18         inserter(iStrings, iStrings.end())
19     );
20
21     // Or: set<string> iStrings((istream_iterator<string>(cin)),
22     //                          istream_iterator<string>());
23
24     copy(
25         iStrings.begin(),
26         iStrings.end(),
27         ostream_iterator<string>(cout, " ")
28     );
29 }
```

## Exercise 46

../46/main.cc

```
1  #include <vector>
2  #include <algorithm>
3  #include <iostream>
4  #include <string>
5
6  using namespace std;
7
8  int main(int argc, char **argv)
9  {
10     size_t nrRvalues = stoi(argv[1]);
11     size_t maxRvalue = stoi(argv[2]);
12     size_t lookupVal = stoi(argv[3]);
13
14     vector<size_t> numbers;
15
16     for (size_t idx = 0; idx < nrRvalues; ++idx)
17         numbers.push_back(random() % (maxRvalue + 1)); //adding a random number
18                                                         //between 0 and max_rvalue
19     for (auto idx: numbers)
20         cout << idx << '\t';
21     cout << '\n';
22
23     auto it = find_if(
24         numbers.begin(),
25         numbers.end(),
26         [lookupVal](const size_t & val)
27         {
28             if (val > lookupVal)
29                 return true;
30             return false;
31         }
32     );
33
34     if (it != numbers.end())
35         cout << "The first value exceeding " << lookupVal << " is at index "
36             << distance(numbers.begin(), it) << '\n';
37     else
38         cout << "No random value exceeds " << lookupVal << '\n';
39
40 }
```

## Exercise 47

../47/main.cc

```
1  #include <algorithm>
2  #include <iostream>
3  #include <iterator>
4
5  using namespace std;
6
7  int main(int argc, char **argv)
8  {
9      sort(
10         argv, argv + argc,
11         [](char *left, char *right) // Sort ascending
12         {
13             return *left < *right;
14         }
15     );
16     copy(argv, argv + argc, ostream_iterator<string>(cout, " ")); // Print
17
18     cout << '\n'; // New line
19
20     sort(
21         argv,
22         argv + argc,
23         [](char *left, char *right) // Sort descending
24         {
25             return *left > *right;
26         }
27     );
28     copy(argv, argv + argc, ostream_iterator<string>(cout, " ")); // Print
29 }
```

## Exercise 48

The difference between `std::copy` and `std::for_each` is centered primarily around the fact that `std::copy` leaves the original intact (unless, of course, as in exercise 47 the destination range is the same as the origin). Furthermore, the `std::for_each` algorithm applies a 'transformation' (i.e. function) to the range, while `copy` does not. Hence, the latter is applicable for applications where some kind of function must be applied to all elements of a container.

../48/main.ih

```
1  #define ERR(msg) printf("%s : %d", (msg), __LINE__)
2
3  using namespace std;
4
5  #include <vector>
6  #include <algorithm>
7  #include <iterator>
8  #include <iostream>
```

../48/main.cc

```
1  #include "main.ih"
2
3  int main(int argc, char const **argv)
4  {
5      vector<size_t> numbers {1, 2, 3, 4, 5, 6, 7};
6      // Simple vector of ascending numbers
7
8      copy(numbers.begin(), numbers.end(), ostream_iterator<size_t>(cout, "\n"));
9      // Copies the vector to cout, printing them all, new-line seperated
10
11     for_each(
12         numbers.begin(),
13         numbers.end(),
14         [](size_t &n0)
15         {
16             cout << ++n0 << '\n';
17         }
18     );
19
20     // The same, but now the numbers are incremented in the vector, as well as
21     // printed
22     // This could not be accomplished with copy
23 }
```

## Exercise 49

../49E/main.ih

```
1  #define ERR(msg) printf("%s : %d", (msg), __LINE__)
2
3  using namespace std;
4
5  #include "student/student.h"
6
7  #include <vector>
8  #include <fstream>
9  #include <iostream>
10 #include <algorithm>
11 #include <numeric>
12
13 void read(string fileName, vector<Student> &vStudents);
14 string toLower(string toLowerCase);
15 void writeNames(vector<Student> const &vStudents);
16 void writeNrs(vector<Student> vStudents, vector<size_t> vIndices);
17
18 void sortByName(vector<Student> &vStudents);
19 void sortByNr(vector<Student> &vStudents, vector<size_t> &vIndices);
```

../49E/main.cc

```
1  #include "main.ih"
2
3
4  int main(int argc, char const **argv)
5  {
6      vector<Student> vStudents;
7      read(string(argv[1]), vStudents); //reading input from file
8
9      sortByName(vStudents);             //sorting by name
10
11     writeNames(vStudents);              //printing sorted vector with names
12
13     vector<size_t> vIndices(vStudents.size());
14     iota(vIndices.begin(), vIndices.end(), 0); //filling indices with values
15                                           //0,1,...,n where n is the last
16                                           //element
17
18     sortByNr(vStudents, vIndices);      //sorting by student number
19
20     writeNrs(vStudents, vIndices);      //printing student number and
21                                           //grade
22 }
```

../49E/student/student.h

```
1  #ifndef INCLUDED_STUDENT_
2  #define INCLUDED_STUDENT_
3
4  #include <string>
5
6  class Student
7  {
8
9      std::string d_name;
10     std::string d_lastName;
11     size_t      d_sNo;
12     double      d_grade;
13
14 public:
```

```
15     Student(std::string firstName, std::string lastName, size_t sNo, double grade);
16     size_t sNo() const;
17     std::string lastName();
18     void printName();
19     void printNr();
20
21 };
22
23 #endif
24
25 inline size_t Student::sNo() const
26 {
27     return d_sNo;
28 };
29
30 inline std::string Student::lastName()
31 {
32     return d_lastName;
33 };
```

../49E/student/student.ih

```
1 #include "student.h"
2
3 #include <iostream>
4 #include <iomanip>
5
6 using namespace std;
```

../49E/student/c\_studentInfo.cc

```
1 #include "student.ih"
2
3 Student::Student(string firstName, string lastName, size_t sNo, double grade)
4 : d_name(firstName + ' ' + lastName), d_lastName(lastName), d_sNo(sNo), d_grade(
5     grade)
6 {
7 }
```

../49E/student/printname.cc

```
1 #include "student.ih"
2
3
4 void Student::printName()
5 {
6     cout << left << setw(25) << d_name << setw(25) << d_lastName << setw(25)
7         << d_sNo << setw(25) << d_grade << '\n';
8 }
```

../49E/student/printnr.cc

```
1 #include "student.ih"
2
3 void Student::printNr()
4 {
5     cout << d_sNo << '\t'
6         << d_grade << '\n';
7 }
```

../49E/read.cc

```
1 #include "main.ih"
2
```

```
3 void read(string fileName, vector<Student> &vStudents)
4 {
5     ifstream textFile(fileName);
6
7     while (true)
8     {
9         string firstName;
10        string lastName;
11        string sNo;
12        string grade;
13
14        getline(textFile, firstName, '\t');
15
16        if (textFile.eof())
17            break;
18
19        getline(textFile, lastName, '\t');
20        getline(textFile, sNo, '\t');
21        getline(textFile, grade);
22
23        vStudents.push_back(Student(firstName, lastName, stoi(sNo), stod(grade)));
24    }
25 }
```

../49E/sortbyname.cc

```
1 #include "main.ih"
2
3 void sortByName(vector<Student> &vStudents)
4 {
5     sort(
6         vStudents.begin(),
7         vStudents.end(),
8         [](Student left, Student right)
9         {
10            return toLower(left.lastName()) < toLower(right.lastName());
11        }
12    );
13 }
```

../49E/sortbynr.cc

```
1 #include "main.ih"
2
3 void sortByNr(vector<Student> &vStudents, vector<size_t> &vIndices)
4 {
5     sort(
6         vIndices.begin(),
7         vIndices.end(),
8         [vStudents](int left, int right)
9         {
10            return vStudents[left].sNo() < vStudents[right].sNo();
11        }
12    );
13 }
```

../49E/strCaseCmp.cc

```
1 #include "main.ih"
2
3 string toLower(string toLowerCase)
4 {
5     transform(toLowerCase.begin(), toLowerCase.end(), toLowerCase.begin(),
6         ::toupper);
7     return toLowerCase;
8 }
```

../49E/writeNames.cc

```
1  #include "main.ih"
2
3  void writeNames(vector<Student> const &vStudents)
4  {
5      for(auto idx: vStudents)
6      {
7          idx.printName();
8      }
9      cout << '\n';
10 }
```

../49E/writeNrs.cc

```
1  #include "main.ih"
2
3  void writeNrs(vector<Student> vStudents, vector<size_t> vIndices)
4  {
5
6      for (size_t idx = 0; idx < vIndices.size(); ++idx)
7          vStudents[vIndices[idx]].printNr();
8
9      cout << '\n';
10 }
```



## Exercise 50

../50-2/main.ih

```
1  #include <iostream>
2  #include <string>
3  #include "line/line.h"
4  #include "vector"
5  #include "iterator"
6
7  namespace Icmbuild
8  {
9      extern char version[];
10     extern char years[];
11     extern char author[];
12 };
13
14 void usage(std::string const &programe);
15
16 using namespace std;
```

../50-2/main.cc

```
1  #include "main.ih"
2
3  void operator>>(istream &istr, vector<string> &dest)
4  {
5      std::copy(std::istream_iterator<Line>(istr), std::istream_iterator<Line>(),
6              inserter(dest, dest.begin()));
7  }
8
9  int main(int argc, char **argv)
10 {
11     vector<string> vs;
12
13     cin >> vs;
14
15     for (auto it: vs)
16         cout << it << '\n';
17 }
18
19 /*
20 If we had used to use istream &operator>>(std::istream &istr, std::string &str)
21 rather than std::istream &operator>>(std::istream &is, Line &line)
22 (in line.h) Then we would have extracted individual words rather than lines.
23 This is due to how istream_iterator<string> iterating works. It would use the
24 extraction operator to a string which is coded this way. By instead using
25 a istream_iterator<Line> we can create our own extraction operator which
26 returns lines instead of words.
27 */
```

../50-2/line/line.h

```
1  #ifndef INCLUDED_LINE_
2  #define INCLUDED_LINE_
3
4  class Line : public std::string
5  {
6      friend std::istream &operator>>(std::istream &is, Line &line)
7      {
8          return std::getline(is, line);
9      }
10 };
11
12 #endif
```

## Exercise 52

../52/main.ih

```
1 #define ERR(msg) printf("%s : %d", (msg), __LINE__)
2
3 #include <vector>
4 #include <string>
5 #include <algorithm>
6 #include <iostream>
7 #include <fstream>
8 #include <iterator>
9
10 using namespace std;
```

../52/main.cc

```
1 #include "main.ih"
2
3 int main(int argc, char const **argv)
4 {
5     ifstream textFile(argv[1]); // Open file1
6
7     vector<string> data(
8         (istream_iterator<string>(textFile)),
9         istream_iterator<string>());
10
11     // Construct vector data using istream iterator that goes through file1
12
13     textFile.close(); // Close file1
14     textFile.clear(); // Clear flags
15     textFile.open(argv[2]); // Open file2
16
17     vector<string> data2(
18         (istream_iterator<string>(textFile)),
19         istream_iterator<string>());
20
21     // Construct vector data using istream iterator that goes through file2
22
23     remove_if(
24         data.begin(),
25         data.end(),
26         [](string findMe)
27         {
28             return findMe == "extra";
29         });
30
31     // Go through vector, remove instances of 'extra'
32
33     data.insert( data.end(), data2.begin(), data2.end() );
34     // Insert data2 at the end of data
35
36     data.erase( unique(data.begin(), data.end()), data.end() );
37     // Erase all non unique entries in data
38
39     vector<string>(data).swap(data);
40     // Shrink to fit
41
42     for (auto el: data)
43         cout << el << '\n';
44     // Print entries, line-separated
45
46     cout << data.size() << '\t' << data.capacity();
47     // Checking space
48 }
```