Programming in C/C++
Exercises set five: STL and GA

Christiaan Steenkist
Jaime Betancor Valado
Remco Bos

December 14, 2016

Exercise 31, Extracting lines with GAs

We extracted lines from an input stream by using `copy`, `input_iterators` and a custom Line class.

There is already an overloaded string extraction operator that we would not be overloading again. That extraction operator stops extracting at a whitespace which means it does not do what we want it to do.

Code listing

Listing 1: main.cc

```
1  #include <iostream>
2  #include <vector>
3  #include <iterator>
4  #include <algorithm>
5
6  using namespace std;
7
8  class Line: public string
9  {
10   public:
11     friend istream &operator>>(istream &input,
12       Line &line)
13     {
14       return getline(input, line);
```

```
15      }
16  };
17
18  int main(int argc, char** argv)
19  {
20    vector<string> lines;
21    copy(istream_iterator<Line>(cin),
22      istream_iterator<Line>(), back_inserter(lines));
23
24    for (auto it = lines.begin(); it != lines.end();
25      ++it)
26
27      cout << *it << '\n';
28  }
```

Exercise 32, You get a promotion!

We used sort to sort in two cool sorting ways, with promotion!

Code listing

Listing 2: main.cc

```
1   #include <iostream>
2   #include <algorithm>
3   #include <cstring>
4   #include <iterator>
5
6   class CaseInsensitiveAscending
7   {
8     public:
9       bool operator()(std::string const &left,
10        std::string const &right) const
11      {
12        return strcmp(left.c_str(),
13          right.c_str()) < 0;
14      }
15  };
16  class CaseInsensitiveDescending
17  {
18    public:
```

```
19      bool operator()(std::string const &left,
20        std::string const &right) const
21      {
22        return strcmp(left.c_str(),
23          right.c_str()) > 0;
24      }
25  };
26
27  int main(int argc, char **argv)
28  {
29    std::sort(argv + 1, argv + argc,
30      CaseInsensitiveAscending());
31    copy(argv + 1, argv + argc,
32      std::ostream_iterator<std::string>(
33      std::cout, " "));
34    std::cout << '\n';
35
36    std::sort(argv + 1, argv + argc,
37      CaseInsensitiveDescending());
38    copy(argv + 1, argv + argc,
39      std::ostream_iterator<std::string>(
40      std::cout, " "));
41    std::cout << '\n';
42  }
```

Exercise 33, Lambda functions

Lambda functions are used in this program that (currently) counts vowels.

Output

Listing 3: output

```
1  Vowels: 819
2  A: 7
3  E: 2
4  I: 8
5  O: 1
6  U: 3
7  a: 192
8  e: 230
```

```
i: 143
o: 148
u: 85
```

Code listings

Listing 4: vstring.ih

```
#include "vstring.h"
#include <iostream>
#include <algorithm>
#include <iterator>

using namespace std;
```

Listing 5: vstring.h

```
#include <map>
#include <cstring>
#include <vector>

class Vstring: public std::vector<std::string>
{
  public:
    typedef std::map<char, size_t> CharMap;

    Vstring(std::istream &in);

    size_t count(CharMap &cmap,
      bool (*accept)(char, CharMap &));

  private:
    static size_t countChar(std::string const &str,
      CharMap &cmap, bool (*accept)(char, CharMap &));
};

bool vowels(char c, Vstring::CharMap &cmap);
```

Listing 6: main.cc

```
#include "vstring.ih"

```

```
3  int main()
4    {
5      Vstring vstring(cin);
6      Vstring::CharMap vmap;
7
8      cout << "Vowels: " << vstring.count(vmap, vowels)
9        << '\n';
10
11     for_each(vmap.begin(), vmap.end(),
12       [] (decltype(*vmap.begin()) &value)
13       {
14         cout << value.first << ": " << value.second
15           << '\n';
16       }
17     );
18   }
```

Listing 7: count.cc

```
1  #include "vstring.ih"
2
3  size_t Vstring::count(CharMap &cmap,
4    bool (*accept)(char, CharMap &))
5  {
6    size_t ret = 0;
7    for_each(begin(), end(),
8      [&] (string &str)
9      {
10       ret += countChar(str, cmap, accept);
11     }
12   );
13   return ret;
14 }
```

Listing 8: countchar.cc

```
1  #include "vstring.ih"
2
3  size_t Vstring::countChar(std::string const &str,
4    CharMap &cmap, bool (*accept)(char, CharMap &))
5  {
```

```
6     size_t ret = 0;
7     for_each(str.begin(), str.end(),
8       [&] (char c)
9       {
10        if (accept(c, cmap))
11          ++ret;
12      }
13    );
14    return ret;
15  }
```

Listing 9: vowels.cc

```
1   #include "vstring.ih"
2
3   bool vowels(char c, Vstring::CharMap &cmap)
4   {
5     if (string("aeiuoAEIOU").find(c)
6       != string::npos)
7     {
8       ++cmap[c];
9       return true;
10    }
11    return false;
12  }
```

Listing 10: vstring.cc

```
1   #include "vstring.ih"
2
3   Vstring::Vstring(std::istream &in)
4   {
5     copy(istream_iterator<string>(in),
6       istream_iterator<string>(),
7       back_inserter(*this));
8   }
```

Exercise 34, GA's and removing elements

Extra items are added if "extra" is found in the input and only the unique items remain.

Inputs

Listing 11: data

```
asd
asd
b
b
d
f
g
extra
h
h
hj
hj
hj
er
rt
rt
rt
ee
ww
ww
ww
ww
```

Listing 12: extra

```
waha
haha
hoohoo
```

Output

Listing 13: output

```
asd
b
d
f
g
```

```
h
hj
er
rt
ee
ww
waha
haha
hoohoo
Data size: 14; Data capacity: 14
```

Code listing

Listing 14: main.cc

```
#include <iostream>
#include <fstream>
#include <vector>
#include <set>
#include <iterator>
#include <algorithm>

using namespace std;

int main(int argc, char **argv)
{
  // Break if there are less than 2 arguments.
  if (argc <= 2)
    return 0;

  // Open data file and read into vector data.
  vector<string> data;
  ifstream input(argv[1]);
  copy(istream_iterator<string>(input),
    istream_iterator<string>(),
    back_inserter(data));
  input.close();

  // Open extra file and read into vector extra.
  vector<string> extra;
  input.open(argv[2]);
```

```
27    copy(istream_iterator<string>(input),
28      istream_iterator<string>(),
29      back_inserter(extra));
30    input.close();
31
32    // Add extra words if "extra" is found
33    if (find(data.begin(), data.end(), "extra")
34      != data.end())
35    {
36      // Remove all "extra" from data
37      data.erase(remove(data.begin(), data.end(),
38        "extra"), data.end());
39
40      // Copy over the extra entries from
41      // the extra vector
42      copy(extra.begin(), extra.end(),
43        back_inserter(data));
44    }
45
46    // Remove all copies
47    data.erase(unique(data.begin(), data.end()),
48      data.end());
49    data.shrink_to_fit();
50
51    // Output
52    copy(data.begin(), data.end(),
53      ostream_iterator<string>(cout, "\n"));
54
55    // Data size and capacity
56    cout << "Data size: " << data.size()
57      << "; Data capacity: "
58      << data.capacity() << '\n';
59  }
```

Exercise 35, Copy and for_each

Here we explain the differences between them.

Answers

The copy generic algorithm copies a series of elements (the range of the iterator) to an output range (destination).

The for_each generic algorithm passes a series of elements (the range of the iterator) as reference to a function that may modify the series of elements.

Code listings

Listing 15: copy.cc

```
1  #include <iostream>
2  #include <algorithm>
3  #include <iterator>
4
5  using namespace std;
6
7  int main()
8  {
9    int intArr[] =
10   {
11     3, 6, 7, 12
12   };
13   vector<int> intDestVector (4);
14
15   intDestVector.push_back(15);
16
17   cout << "Numbers in intArr: \n";
18   for (size_t it = 0; it < 4; ++it)
19     cout << ' ' << intArr[it] << '\n';
20
21   cout << "Numbers in intDestVector before copy:\n";
22   for (vector<int>::iterator it =
23     intDestVector.begin();
24     it !=intDestVector.end(); ++it)
25
26     cout << ' ' << *it << '\n';
27
28   copy(intArr, intArr + 4, intDestVector.begin());
29
30   cout << "Numbers in intDestVector after copy:\n";
```

```
31    for (vector<int>::iterator it =
32      intDestVector.begin();
33      it !=intDestVector.end(); ++it)
34
35      cout << ' ' << *it << '\n';
36
37  }
```

Listing 16: for_each.cc

```
1  #include <iostream>
2  #include <algorithm>
3  #include <iterator>
4
5  using namespace std;
6
7  void squareTheInts(int &number)
8  {
9    int numSquare = number * number;
10   cout << ' ' << numSquare << '\n';
11 }
12
13 int main()
14 {
15   vector<int> intVector;
16   intVector.push_back(3);
17   intVector.push_back(6);
18   intVector.push_back(7);
19   intVector.push_back(12);
20
21   cout << "Numbers in vector squared: \n";
22   for_each(intVector.begin(), intVector.end(),
23     squareTheInts);
24 }
```