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

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

using namespace std;

class Line: public string
{
  public:
    friend istream &operator>>(istream &input,
      Line &line)
    {
      return getline(input, line);
    }
};

int main(int argc, char** argv)
{
  vector<string> lines;
  copy(istream_iterator<Line>(cin),
    istream_iterator<Line>(), back_inserter(lines));

  for (auto it = lines.begin(); it != lines.end();
    ++it)

    cout << *it << '\n';
}
```

Exercise 32, You get a promotion!

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

Code listing

Listing 2: main.cc

```
#include <iostream>
#include <algorithm>
#include <string>
#include <iterator>

int main(int argc, char **argv)
{
    std::sort(argv + 1, argv + argc,
        std::less<std::string>());
    copy(argv + 1, argv + argc,
        std::ostream_iterator<std::string>(std::cout,
        " "));
    std::cout << '\n';

    std::sort(argv + 1, argv + argc,
        std::greater<std::string>());
    copy(argv + 1, argv + argc,
        std::ostream_iterator<std::string>(std::cout,
        " "));
    std::cout << '\n';
}
```

Exercise 33, Lambda functions

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

Output

```
Vowels: 819
A: 7
E: 2
I: 8
O: 1
U: 3
a: 192
e: 230
i: 143
o: 148
u: 85
```

Code listings

Listing 3: vstring.ih

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

using namespace std;
```

Listing 4: 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 5: main.cc

```
#include "vstring.ih"

int main()
  {
    Vstring vstring(cin);
    Vstring::CharMap vmap;

    cout << "Vowels: " << vstring.count(vmap, vowels)
      << '\n';

    for_each(vmap.begin(), vmap.end(),
      [] (decltype(*vmap.begin()) &value)
      {
        cout << value.first << ": " << value.second
          << '\n';
      }
    );
  }
```

Listing 6: count.cc

```
#include "vstring.ih"

size_t Vstring::count(CharMap &cmap,
  bool (*accept)(char, CharMap &))
```

```
{
  size_t ret = 0;
  for_each(begin(), end(),
    [&] (string &str)
    {
      ret += countChar(str, cmap, accept);
    }
  );
  return ret;
}
```

Listing 7: countchar.cc

```
#include "vstring.ih"

size_t Vstring::countChar(std::string const &str,
  CharMap &cmap, bool (*accept)(char, CharMap &))
{
  size_t ret = 0;
  for_each(str.begin(), str.end(),
    [&] (char c)
    {
      if (accept(c, cmap))
        ++ret;
    }
  );
  return ret;
}
```

Listing 8: vowels.cc

```
#include "vstring.ih"

bool vowels(char c, Vstring::CharMap &cmap)
{
  if (string("aeiuoAEIOU").find(c)
    != string::npos)
  {
    ++cmap[c];
    return true;
  }
  return false;
}
```

Listing 9: vstring.cc

```
#include "vstring.ih"

Vstring::Vstring(std::istream &in)
{
  copy(istream_iterator<string>(in),
    istream_iterator<string>(),
    back_inserter(*this));
}
```

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

Data

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

```
ww
ww
ww
ww
```

Extra

```
waha
haha
hoohoo
```

Output

```
asd
b
d
f
g
h
hj
er
rt
ee
ww
waha
haha
hoohoo
Data size: 14; Data capacity: 14
```

Code listing

Listing 10: main.cc

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

using namespace std;

```

```cpp
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]);
  copy(istream_iterator<string>(input),
    istream_iterator<string>(),
    back_inserter(extra));
  input.close();

  // Add extra words if "extra" is found
  if (find(data.begin(), data.end(), "extra")
    != data.end())
  {
    // Remove all "extra" from data
    data.erase(remove(data.begin(), data.end(),
      "extra"), data.end());

    // Copy over the extra entries from
    // the extra vector
    copy(extra.begin(), extra.end(),
      back_inserter(data));
  }

  // Remove all copies
  data.erase(unique(data.begin(), data.end()),
    data.end());
  data.shrink_to_fit();

  // Output
  copy(data.begin(), data.end(),
    ostream_iterator<string>(cout, "\n"));

  // Data size and capacity
  cout << "Data size: " << data.size()
    << "; Data capacity: "
    << data.capacity() << '\n';
}
```

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 11: copy.cc

```
#include <iostream>
#include <algorithm>
#include <iterator>

using namespace std;

int main()
{
  int intArr[] =
  {
    3, 6, 7, 12
  };
  vector<int> intDestVector (4);

  intDestVector.push_back(15);

  cout << "Numbers in intArr: \n";
  for (size_t it = 0; it < 4; ++it)
    cout << ' ' << intArr[it] << '\n';

  cout << "Numbers in intDestVector before copy:\n";
  for (vector<int>::iterator it =
    intDestVector.begin();
    it !=intDestVector.end(); ++it)

    cout << ' ' << *it << '\n';

  copy(intArr, intArr + 4, intDestVector.begin());

  cout << "Numbers in intDestVector after copy:\n";
  for (vector<int>::iterator it =
    intDestVector.begin();
    it !=intDestVector.end(); ++it)

    cout << ' ' << *it << '\n';

}
```

Listing 12: for_each.cc

```
#include <iostream>
#include <algorithm>
#include <iterator>

using namespace std;

void squareTheInts(int &number)
{
  int numSquare = number * number;
  cout << ' ' << numSquare << '\n';
}

int main()
{
  vector<int> intVector;
  intVector.push_back(3);
  intVector.push_back(6);
  intVector.push_back(7);
  intVector.push_back(12);

  cout << "Numbers in vector squared: \n";
  for_each(intVector.begin(), intVector.end(),
    squareTheInts);
}
```