Exercises week 6

Klaas Isaac Bijlsma s2394480
David Vroom s2309939

January 10, 2018

Exercise 42

Learn to extract lines using generic algorithms into a container holding ***`string`*** *objects, although* ***`operator>>()`*** *extracts strings*

main.cc

```
1  #include <iostream>
2  #include <iterator>
3  #include <string>
4  #include <vector>
5  #include <algorithm>
6
7  using namespace std;
8
9  class Derived: public string
10 {};
11
12 istream &operator>>(istream &istr, Derived &str)
13 {
14     return getline(istr, str);
15 }
16
17 int main()
18 {
19     vector<string> vs;
20
```

```
21      copy(istream_iterator<Derived>(cin), istream_iterator<Derived>(),
22           back_inserter(vs));
23  }
```

Exercise 43

Learn to use promotion with generic algorithms and predefined function objects when manipulating basic data types.

main.cc

```
1  #include <iostream>
2  #include <algorithm>
3  #include <functional>
4
5  using namespace std;
6
7  int main(int argc, char **argv)
8  {
9      sort(argv + 1, argv + argc, greater<string>());
10     copy(argv + 1, argv + argc, ostream_iterator<string>(cout));
11     cout << '\n';
12
13     sort(argv + 1, argv + argc, less<string>());
14     copy(argv + 1, argv + argc, ostream_iterator<string>(cout));
15     cout << '\n';
16 }
```

Exercise 44

Learn to recognize a situation where lambda functions may be used

vstring/vstring.h

```
1  #ifndef EX44_VSTRING_H
2  #define EX44_VSTRING_H
3
4  #include <vector>
5  #include <string>
6  #include <map>
7  #include <istream>
8
9  class Vstring: public std::vector<std::string>
10 {
11     public:
12         typedef std::map<char, size_t> Charmap;
13
14         explicit Vstring(std::istream &in);
15
16         size_t count(Charmap &cmap, bool (*accept)(char, Charmap &));
17
18     private:
19         static size_t countChar(std::string const &str, Charmap &cmap,
20                                 bool (*accept)(char, Charmap &));
21 };
22
23 #endif
```

vstring/vstring.ih

```
1  #include "vstring.h"
2  #include <algorithm>
3  #include <iterator>
4
5  using namespace std;
```

vstring/count.cc

```
#include "vstring.ih"

size_t Vstring::count(Charmap &cmap, bool (*accept)(char, Charmap &))
{
    size_t count = 0;
    for_each(
        begin(), end(),
        [&, accept](string &str)
        {
            count += countChar(str, cmap, accept);
        }
    );
    return count;
}
```

vstring/countchar.cc

```
#include "vstring.ih"

size_t Vstring::countChar(string const &str, Charmap &cmap,
                          bool (*accept)(char, Charmap &))
{
    return count_if(
        str.begin(), str.end(),
        [&, accept](char ch)
        {
            return accept(ch, cmap);
        }
    );
}
```

main.cc

```
#include "main.ih"

int main()
{
    ifstream is("text.txt");
```

```
    Vstring vstring(is);
    Vstring::Charmap vmap;

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

    display(vmap);
}
```

main.ih

```
#include <iostream>
#include <fstream>
#include "vstring/vstring.h"
#include <algorithm>

using namespace std;

void display(Vstring::Charmap &cmap);
bool vowels(char c, Vstring::Charmap &cmap);
```

display.cc

```
#include "main.ih"

void display(Vstring::Charmap &cmap)
{
    for_each(
        cmap.begin(), cmap.end(),
        [](auto const &value)
        {
            cout << value.first << ": " << value.second << '\n';
        }
    );
}
```

vowels.cc

```
#include "main.ih"
```

```
2
3  bool vowels(char c, Vstring::Charmap &cmap)
4  {
5      if (string("aeiuoAEIUO").find(c) != string::npos)
6      {
7          ++cmap[c];
8          return true;
9      }
10     return false;
11 }
```

Exercise 45

Learn to use generic algorithms to remove elements from a vector

Exercise 46

Learn to distinguish two frequently used generic algorithms

Exercise 47

Exercise 48