Programming in C/C++
Exercises set four: containers

Christiaan Steenkist
Jaime Betancor Valado
Remco Bos

December 1, 2016

Exercise 23, vectors and shrinking

So we experimented with slicing off extra capacity with vectors and a class with a vector as a data member.

Output

```
size: 10 capacity: 16
size: 11 capacity: 16
size: 11 capacity: 11

size: 11 capacity: 16
size: 12 capacity: 16
size: 12 capacity: 12
```

Code listings

Listing 1: main.ih

```
#include "main.h"

using namespace std;
```

Listing 2: main.h

```
#ifndef MAIN_H_
#define MAIN_H_
```

```
3
4  #include <iostream>
5  #include "uwl/uniquewordlist.h"
6
7  void reader(std::istream &stream, std::vector<std::
       string> &wordList);
8  void printer(std::ostream &stream, std::vector<std::
       string> const &wordList);
9  void printer(std::ostream &stream, UniqueWordList
       const &wordList);
10
11 #endif
```

Listing 3: main.cc

```
1  #include "main.ih"
2  #include "uwl/uniquewordlist.h"
3
4  int main(int argc, char **argv)
5  {
6    vector<string> wordList;
7    reader(cin, wordList);
8    printer(cout, wordList);
9
10   wordList.push_back("test");
11   printer(cout, wordList);
12
13   wordList = vector<string>(wordList);
14   printer(cout, wordList);
15
16   UniqueWordList uwl;
17   for (auto it = wordList.begin(); it != wordList.end
       (); ++it)
18   {
19     uwl.addWord(*it);
20   }
21   cout << '\n';
22
23   printer(cout, uwl);
24
25   uwl.addWord("west");
```

```
26    printer(cout, uwl);
27
28    uwl = uwl;
29    printer(cout, uwl);
30  }
```

Listing 4: printer1.cc

```
1  #include "main.ih"
2
3  void printer(ostream &stream, vector<string> const &
       wordList)
4  {
5    stream << "size: " << wordList.size()
6      << " capacity: " << wordList.capacity() << '\n';
7  }
```

Listing 5: printer2.cc

```
1  #include "main.ih"
2
3  void printer(ostream &stream, UniqueWordList const &
       wordList)
4  {
5    stream << "size: " << wordList.size()
6      << " capacity: " << wordList.capacity() << '\n';
7  }
```

Listing 6: reader.cc

```
1  #include "main.ih"
2
3  #include <algorithm>
4
5  void reader(istream &stream, vector<string> &wordList)
6  {
7    string word;
8    while (stream >> word)
9    {
10     if (find(wordList.begin(), wordList.end(), word)
       == wordList.end())
11       wordList.push_back(word);
```

```
12    }
13  }
```

UniqueWordList

Listing 7: uniquewordlist.ih

```
1  #include "uniquewordlist.h"
2
3  using namespace std;
```

Listing 8: uniquewordlist.h

```
1  #ifndef UNIQUEWORDLIST_H_
2  #define UNIQUEWORDLIST_H_
3
4  #include <vector>
5  #include <string>
6
7  class UniqueWordList
8  {
9    std::vector<std::string> d_list;
10
11   public:
12     UniqueWordList() = default;
13     UniqueWordList(UniqueWordList const &uwl) =
      default;
14
15     UniqueWordList &operator=(UniqueWordList const &
      uwl);
16
17     void swap(UniqueWordList &uwl);
18
19     void addWord(std::string word);
20
21     std::size_t size();
22     std::size_t capacity();
23
24     std::size_t size() const;
25     std::size_t capacity() const;
26 };
27
```

```
28 #endif
```

Listing 9: addword.cc

```
1  #include "uniquewordlist.ih"
2
3  #include <algorithm>
4
5  void UniqueWordList::addWord(string word)
6  {
7    if (find(d_list.begin(), d_list.end(), word) ==
      d_list.end())
8      d_list.push_back(word);
9  }
```

Listing 10: capacity.cc

```
1  #include "uniquewordlist.ih"
2
3  size_t UniqueWordList::capacity()
4  {
5    return d_list.capacity();
6  }
```

Listing 11: capacityconst.cc

```
1  #include "uniquewordlist.ih"
2
3  size_t UniqueWordList::capacity() const
4  {
5    return d_list.capacity();
6  }
```

Listing 12: operator

```
1  #include "uniquewordlist.ih"
2
3  UniqueWordList &UniqueWordList::operator=(
     UniqueWordList const &uwl)
4  {
5    UniqueWordList copy(uwl);
6    swap(copy);
7    return *this;
8  }
```

Listing 13: size.cc

```
1  #include "uniquewordlist.ih"
2
3  size_t UniqueWordList::size()
4  {
5    return d_list.size();
6  }
```

Listing 14: sizeconst.cc

```
1  #include "uniquewordlist.ih"
2
3  size_t UniqueWordList::size() const
4  {
5    return d_list.size();
6  }
```

Listing 15: swap.cc

```
1  #include "uniquewordlist.ih"
2
3  #include <cstring>
4
5  void UniqueWordList::swap(UniqueWordList &uwl)
6  {
7    char bytes[sizeof(UniqueWordList)];
8    memcpy(bytes, this, sizeof(UniqueWordList));
9    memcpy(this, &uwl, sizeof(UniqueWordList));
10   memcpy(&uwl, bytes, sizeof(UniqueWordList));
11 }
```

Exercise 25, unique keys

We made a snippet of code to count the number of unique keys in an `unordered_multimap`. Never again.

Code listings

Listing 16: main.cc

```
1  #include "main.ih"
2  #include "uwl/uniquewordlist.h"
```

```
3
4  int main(int argc, char **argv)
5  {
6    vector<string> wordList;
7    reader(cin, wordList);
8    printer(cout, wordList);
9
10   wordList.push_back("test");
11   printer(cout, wordList);
12
13   wordList = vector<string>(wordList);
14   printer(cout, wordList);
15
16   UniqueWordList uwl;
17   for (auto it = wordList.begin(); it != wordList.end
       (); ++it)
18   {
19     uwl.addWord(*it);
20   }
21   cout << '\n';
22
23   printer(cout, uwl);
24
25   uwl.addWord("west");
26   printer(cout, uwl);
27
28   uwl = uwl;
29   printer(cout, uwl);
30 }
```