STL Examples

임종우 (Jongwoo Lim)

Example 1. Word Array

```
#include <iostream>
#include <string>
#include <vector>
using namespace std;

int main() {
  vector<string> word_arr;
  while (true) {
    // Process user input commands.
    .....
  }
  return 0;
}
```

```
list
0 word(s).
add hello
add world
add my
add book
list
0 hello
1 world
2 my
3 book
4 word(s).
find world
'world' is at 1.
erase world
'world' is erased.
find world
'world' not found.
list
0 hello
1 my
2 book
3 word(s).
clear
cleared.
list
0 word(s).
quit
```

```
.....
vector<string> word_arr;
while (true) {
  // Process user input commands.
  string cmd;
  cin >> cmd;
  if (cmd == "quit") {
    break;
  } else if (cmd == "clear") {
    word_arr.clear();
    cout << "cleared." << endl;
  } else if (cmd == "list") {
    for (int i = 0; i < word_arr.size(); ++i) {
      cout << i << " " << word_arr[i] << endl;
    }
    cout << word_arr.size() << " word(s)." << endl;
  } else if (cmd == "add") {
    string word;
    cin >> word;
    word_arr.push_back(word);
  } else if (cmd == "erase") {
    string word;
    cin >> word;
    vector<string>::iterator it = find(
        word_arr.begin(), word_arr.end(), word);
    if (it == word_arr.end()) {
      cout << "'" << word << "' not found." << endl;
    } else {
      word_arr.erase(it);
      cout << "'" << word << "' is erased." << endl;
    }
  }
  .....
```

```
.....
int main() {
  vector<string> word_arr;
  while (true) {
    // Process user input commands.
    .....
    } else if (cmd == "find") {
      // Find the given word in the array.
      string word;
      cin >> word;
      bool found = false;
      for (int i = 0; !found && i < word_arr.size(); ++i) {
        if (word_arr[i] == word) {
          cout << "'" << word << "' is at " << i << "." << endl;
          found = true;
        }
      }
      if (!found) {
        cout << "'" << word << "' not found." << endl;
      }
    }
  }
  return 0;
}
```

```
.....
int main() {
  vector<string> word_arr;
  while (true) {
    // Process user input commands.
    .....
    } else if (cmd == "find2") {
      // Another way to find the given word in the array.
      string word;
      cin >> word;
      vector<string>::const_iterator it = find(
          word_arr.begin(), word_arr.end(), word);
      if (it == word_arr.end()) {
        cout << "'" << word << "' not found." << endl;
      } else {
        cout << "'" << word << "' is in the array." << endl;
      }
    }
  }
  return 0;
}
```

Example 2. Word Set

```
#include <iostream>
#include <string>
#include <set>
using namespace std;

int main() {
  set<string> words;
  while (true) {
    // Process user input commands.
    .....
  }
  return 0;
}
```

```
list
0 word(s).
add hello
add world
add my
add book
list
'book' 'hello' 'my' 'world'
4 word(s).
find hello
'hello' is in the set.
erase hello
'hello' is erased.
list
'book' 'my' 'world'
3 word(s).
find hello
'hello' not found.
erase hello
'hello' not found.
clear
cleared.
list
0 word(s).
quit
```

```
.....
int main() {
  set<string> words;
  while (true) {
    // Process user input commands.
    string cmd;
    cin >> cmd;
    if (cmd == "quit") {
      break;
    } else if (cmd == "clear") {
      words.clear();
      cout << "cleared." << endl;
    } else if (cmd == "list") {
      for (set<string>::const_iterator it = words.begin();
           it != words.end(); ++it) {
        cout << (it == words.begin() ? "'" : " '") << *it << "'";
      }
      if (!words.empty()) cout << endl;
      cout << words.size() << " word(s)." << endl;
    } else if (cmd == "add") {
      string word;
      cin >> word;
      words.insert(word);
    }
  }
  return 0;
}
```

```
.....
int main() {
  set<string> words;
  while (true) {
    // Process user input commands.
    .....
    } else if (cmd == "find") {
      string word;
      cin >> word;
      set<string>::const_iterator it = words.find(word);
//    set<string>::const_iterator it = find(words.begin(), words.end(), word);
      if (it == words.end()) {
        cout << "'" << word << "' not found." << endl;
      } else {
        cout << "'" << word << "' is in the set." << endl;
      }
    } else if (cmd == "erase") {
      string word;
      cin >> word;
      if (words.erase(word) <= 0) {
        cout << "'" << word << "' not found." << endl;
      } else {
        cout << "'" << word << "' is erased." << endl;
      }
    }
  }
  return 0;
}
```

Example 3. Word Map

```
#include <iostream>
#include <string>
#include <map>
using namespace std;

int main() {
  map<string, string> word_map;
  while (true) {
    // Process user input commands.
    .....
  }
  return 0;
}
```

```
list
0 word mapping(s).
add fox brown
add rat gray
add mouse white
list
[fox : brown] [mouse : white] [rat : gray]
3 word mapping(s).
find fox
fox : brown found.
erase mouse
'mouse' is erased.
find mouse
'mouse' not found.
list
[fox : brown] [rat : gray]
2 word mapping(s).
clear
cleared.
quit
```

```
.....
int main() {
  map<string, string> word_map;
  while (true) {
    // Process user input commands.
    string cmd;
    cin >> cmd;
    if (cmd == "quit") {
      break;
    } else if (cmd == "clear") {
      word_map.clear();
      cout << "cleared." << endl;
    } else if (cmd == "list") {
      for (map<string, string>::const_iterator it = word_map.begin();
           it != word_map.end(); ++it) {
        cout << (it == word_map.begin() ? "[" : " [") << it->first
             << " : " << it->second << "]";
      }
      if (!word_map.empty()) cout << endl;
      cout << word_map.size() << " word mapping(s)." << endl;
    }
  }
  return 0;
}
```

```
.....
int main() {
  set<string> words;
  while (true) {
    // Process user input commands.
    .....
    } else if (cmd == "add") {
      string key, value;
      cin >> key >> value;
      word_map[key] = value;
//    word_map.insert(make_pair(key, value));
    } else if (cmd == "find") {
      string key;
      cin >> key;
      map<string, string>::const_iterator it = word_map.find(key);
      if (it == word_map.end()) {
        cout << "'" << key << "' not found." << endl;
      } else {
        cout << it->first << " : " << it->second << " found." << endl;
      }
    } else if (cmd == "erase") {
      string key;
      cin >> key;
      if (word_map.erase(key) <= 0) {
        cout << "'" << key << "' not found." << endl;
      } else {
        cout << "'" << key << "' is erased." << endl;
      }
    }
  }
  return 0;
}
```

Example 4. Map in a Class

my_dictionary.h

```
#ifndef _MY_DICTIONARY_
#define _MY_DICTIONARY_

#include <map>
#include <string>

class MyDictionary {
 public:
  MyDictionary();

  bool Lookup(const std::string& key,
              std::string* value)  const;
  bool Add(const std::string& key,
           const std::string& value);
  bool Remove(const std::string& key);

 private:
  std::map<std::string, std::string>
      word_map_;
};

#endif  // _MY_DICTIONARY_
```

main.cc

```
#include <iostream>
#include "my_dictionary.h"
using namespace std;

int main() {
  MyDictionary dict;
  while (true) {
    string cmd, key, value;
    cin >> cmd;
    if (cmd == "quit") {
      break;
    } else if (cmd == "add") {
      cin >> key >> value;
      dict.Add(key, value);
    } else if (cmd == "remove") {
      cin >> key;
      dict.Remove(key);
    } else if (cmd == "lookup") {
      cin >> key;
      if (dict.Lookup(key, &value)) {
        cout << value << endl;
      }
    }
  }
  return 0;
}
```

my_dictionary.h

```
#ifndef _MY_DICTIONARY_
#define _MY_DICTIONARY_

#include <map>
#include <string>

class MyDictionary {
 public:
  MyDictionary();

  bool Lookup(const std::string& key,
              std::string* value)  const;
  bool Add(const std::string& key,
           const std::string& value);
  bool Remove(const std::string& key);

 private:
  std::map<std::string, std::string>
      word_map_;
};

#endif  // _MY_DICTIONARY_
```

my_dictionary.cc

```
#include "my_dictionary.h"
using namespace std;

bool MyDictionary::Lookup(
    const string& key, string* value)  const {
  map<string, string>::const_iterator
      it = word_map_.find(key);
  if (it != word_map_.end()) {
    if (value) *value = it->second;
    return true;
  }
  return false;
}

bool MyDictionary::Add(
    const string& key, const string& value) {
  // insert() returns <iterator, bool> pair.
  return word_map_.insert(
      make_pair(key, value)).second;
}

bool MyDictionary::Remove(const string& key) {
  return word_map_.erase(key) > 0;
}
```

my_dictionary.h

```
#ifndef _MY_DICTIONARY_
#define _MY_DICTIONARY_

#include <map>
#include <string>

class MyDictionary {
 public:
  MyDictionary();

  bool Lookup(const std::string& key,
              std::string* value)  const;
  bool Add(const std::string& key,
           const std::string& value);
  bool Remove(const std::string& key);

  typedef std::map<std::string,
      std::string>::const_iterator
      const_iterator;

  const_iterator begin()  const {
    return word_map_.begin();
  }
  const_iterator end()  const {
    return word_map_.end();
  }

 private:
  std::map<std::string, std::string>
      word_map_;
};

#endif  // _MY_DICTIONARY_
```

main.cc

```
#include <iostream>
#include "my_dictionary.h"
using namespace std;

int main() {
  MyDictionary dict;
  while (true) {
    string cmd, key, value;
    cin >> cmd;
    if (cmd == "quit") {
      break;
    } else if (cmd == "add") {
      cin >> key >> value;
      dict.Add(key, value);
    } else if (cmd == "remove") {
      cin >> key;
      dict.Remove(key);
    } else if (cmd == "lookup") {
      cin >> key;
      if (dict.Lookup(key, &value)) {
        cout << value << endl;
      }
    } else if (cmd == "list") {
      for (MyDictionary::const_iterator
           it = dict.begin();
           it != dict.end(); ++it) {
        cout << it->first << " : "
             << it->second << endl;
      }
    }
  }
  return 0;
}
```