Using Iterators

Learning Objectives

1. Utilize Iterators in your own code

stlList.cpp

```
1  #include <list>
2  #include <string>
3  #include <iostream>
4
5  struct Animal {
6    std::string name, food;
7    bool big;
8    Animal(std::string name = "blob", std::string food = "you", bool big = true) :
9      name(name), food(food), big(big) { /* nothing */ }
10 };
11
12 int main() {
13   Animal g("giraffe", "leaves", true), p("penguin", "fish", false), b("bear");
14   std::vector<Animal> zoo;
15
16   zoo.push_back(g);
17   zoo.push_back(p);   // std::vector's insertAtEnd
18   zoo.push_back(b);
19
20   for (std::vector<Animal>::iterator it = zoo.begin(); it != zoo.end(); ++it) {
21     std::cout << (*it).name << " eats " << (*it).food << std::endl;
22   }
23
24   return 0;
25 }
```

stlList.cpp

```
 1 #include <list>
 2 #include <string>
 3 #include <iostream>
 4
 5 struct Animal {
 6   std::string name, food;
 7   bool big;
 8   Animal(std::string name = "blob", std::string food = "you", bool big = true) :
 9     name(name), food(food), big(big) { /* nothing */ }
10 };
11
12 int main() {
13   Animal g("giraffe", "leaves", true), p("penguin", "fish", false), b("bear");
14   std::vector<Animal> zoo;
15
16   zoo.push_back(g);
17   zoo.push_back(p);   // std::vector's insertAtEnd
18   zoo.push_back(b);
19
20   for ( auto it = zoo.begin(); it != zoo.end(); ++it ) {
21     std::cout << (*it).name << " eats " << (*it).food << std::endl;
22   }
23
24   return 0;
25 }
```

stlList.cpp

```
 1 #include <list>
 2 #include <string>
 3 #include <iostream>
 4
 5 struct Animal {
 6   std::string name, food;
 7   bool big;
 8   Animal(std::string name = "blob", std::string food = "you", bool big = true) :
 9     name(name), food(food), big(big) { /* none */ }
10 };
11
12 int main() {
13   Animal g("giraffe", "leaves", true), p("penguin", "fish", false), b("bear");
14   std::vector<Animal> zoo;
15
16   zoo.push_back(g);
17   zoo.push_back(p);   // std::vector's insertAtEnd
18   zoo.push_back(b);
19
20   for ( const Animal & animal : zoo ) {
21     std::cout << animal.name << " eats " << animal.food << std::endl;
22   }
23
24   return 0;
25 }
```

For Each and Iterators

```
    for ( const TYPE & variable : collection ) {
      // ...
    }
```

```
14  std::vector<Animal> zoo;
 …  …
20  for ( const Animal & animal : zoo ) {
21      std::cout << animal.name << " " << animal.food << std::endl;
22  }
```

For Each and Iterators

```
for ( const TYPE & variable : collection ) {
  // ...
}
```

```
14  std::vector<Animal> zoo;
…   …
20  for ( const Animal & animal : zoo ) {
21      std::cout << animal.name << " " << animal.food << std::endl;
22  }
```

```
…   std::unordered_set<std::string, Animal> zoo;
…   …
20  for ( const Animal & animal : zoo ) {
21      std::cout << animal.name << " " << animal.food << std::endl;
22  }
```

Other Things to do with iterators

- insert `vec.insert(it, 100);`
- find `std::find(vec.begin(), vec.end(), 42);`
- sort `std::sort(vec.begin(), vec.end());`
- advance `std::advance(it, 3);`