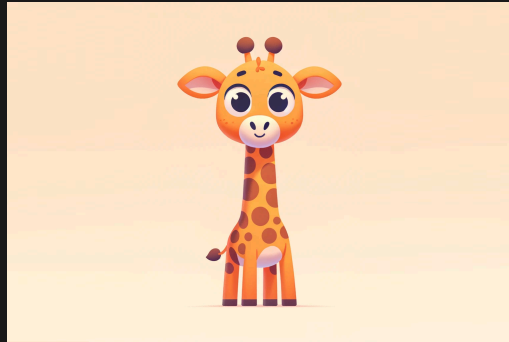


COMP6771

STL Iterators

Lecture 2.2

Author(s): Hayden Smith



[\(Download as PDF\)](#)

STL Iterators

- Iterator is an abstract notion of a pointer
- Iterators are types that abstract container data as a **sequence** of objects (i.e. linear)
- Iterators will allow us to connect a wide range of containers with a wide range of algorithms via a common interface

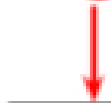


Iterator Explanation

- `a.begin()`: abstractly "points" to the first element
- `a.end()`: abstractly "points" to one past the last element
 - `a.end()` is not an invalid iterator value
- If `iter` abstractly points to the k -th element, then:
 - `*p` is the object it abstractly points to
 - `++p` abstractly points to the $(k + 1)$ -st element

- `a` is a container with all its n objects ordered

`a.begin()`



1st

2nd

...

nth

`a.end()`





Iterator Explanation

- `a.begin()`: abstractly "points" to the first element
- `a.end()`: abstractly "points" to one past the last element
 - `a.end()` is not an invalid iterator value
- If `iter` abstractly points to the k -th element, then:
 - `*p` is the object it abstractly points to
 - `++p` abstractly points to the $(k + 1)$ -st element

```
1 #include <iostream>
2 #include <string>
3 #include <vector>
4
5 int main()
6 {
7     std::vector<std::string> names;
8     for (auto iter = names.begin(); iter != names.end(); ++iter) {
9         std::cout << *iter << "\n";
10    }
11    for (std::vector<std::string>::iterator iter = names.begin();
12         iter != names.end(); ++iter) {
13        std::cout << *iter << "\n";
14    }
15 }
```

vector-iterator.cpp

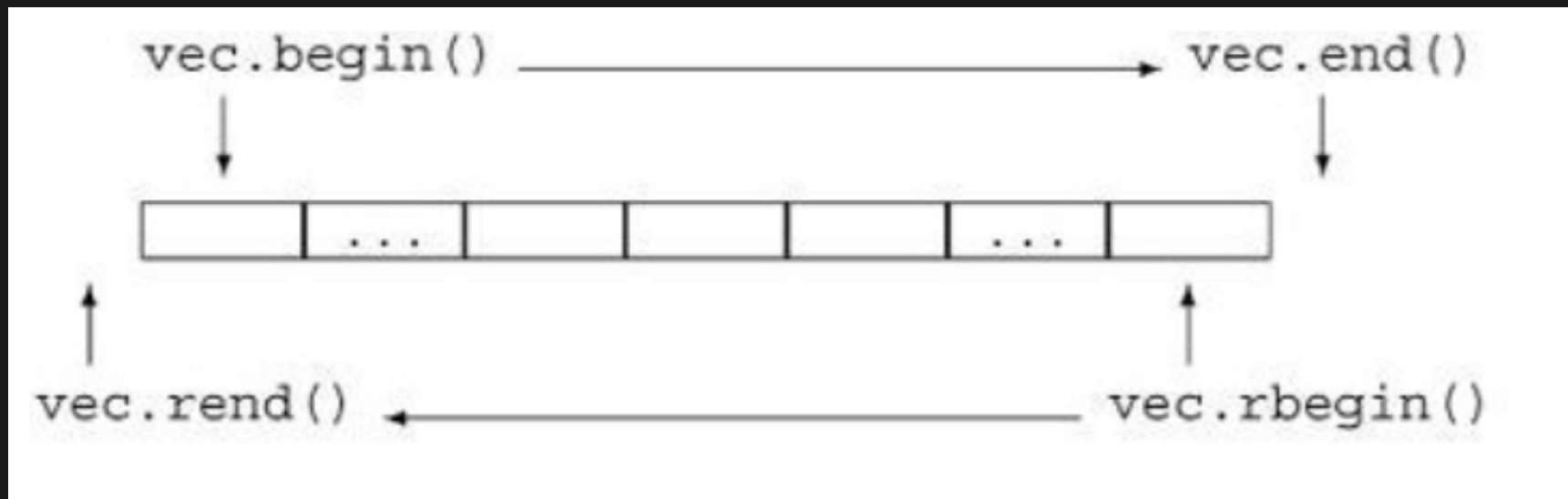


Constness & Reverse

```
1 #include <iostream>
2 #include <vector>
3
4 int main()
5 {
6     std::vector<int> ages;
7     ages.push_back(18);
8     ages.push_back(19);
9     ages.push_back(20);
10
11     // type of iter would be std::vector<int>::iterator
12     for (auto iter = ages.begin(); iter != ages.end(); ++iter) {
13         (*iter)++; // OK
14     }
15
16     // type of iter would be std::vector<int>::const_iterator
17     for (auto iter = ages.cbegin(); iter != ages.cend(); ++iter) {
18         //(*iter)++; // NOT OK
19     }
20
21     // type of iter would be std::vector<int>::reverse_iterator
22     for (auto iter = ages.rbegin(); iter != ages.rend(); ++iter) {
23         std::cout << *iter << "\n"; // prints 20, 19, 18
24     }
25
26     // Can also use crbegin and crend
27 }
```

constness-reverse.cpp

Constness & Reverse





Stream Iterators

```
1 #include <fstream>
2 #include <iostream>
3 #include <iterator>
4
5 int main()
6 {
7     std::ifstream in("data.in");
8
9     std::istream_iterator<int> begin(in);
10    std::istream_iterator<int> end;
11    std::cout << *begin++ << "\n"; // read the first int
12
13    ++begin; // skip the 2nd int
14    std::cout << *begin++ << "\n"; // read the third int
15    while (begin != end) {
16        std::cout << *begin++ << "\n"; // read and print the rest
17    }
18 }
```

stream-iterators.cpp

Feedback



Or go to the [form here](#).

