Iterators

Victor Eijkhout, Susan Lindsey

COE 322 Fall 2021

Begin/end iterator

1. Begin and end iterator

Use independent of looping:

Code:

```
vector<int> v{1,3,5,7};
auto pointer = v.begin();
cout << "we start at "
     << *pointer << endl;
pointer++;
cout << "after increment: "
     << *pointer << endl;

pointer = v.end();
cout << "end is not a valid
 element: "
     << *pointer << endl;
pointer--;
cout << "last element: "
     << *pointer << endl;
```

Output

```
[stl] iter:

we start at 1
after increment: 3
end is not a valid
    element: 0
last element: 7
```

(Note: the auto actually stands for `vector::iterator`)

2. About that star

This is not a C-style pointer dereference,
but rather an overloaded oeprator.

3. Erase at/between iterators

Erase from start to before-end:

Code:

```
vector<int> counts{1,2,3,4,5,6};
vector<int>::iterator second = counts.
    begin()+1;
auto fourth = second+2; // easier than
    'iterator'
counts.erase(second,fourth);
cout << counts[0] << "," << counts[1]
    << "\n";
```

Output

```
[iter] erase2:

1,4
```

(Also single element without end iterator.)

4. Insert at iterator

Insert at iterator: value, single iterator, or range:

Code:

```
vector<int> counts{1,2,3,4,5,6},zeros
    {0,0};
auto after_one = zeros.begin()+1;
zeros.insert( after_one,counts.begin()
    +1,counts.begin()+3 );
//vector<int>::insert( after_one,
    counts.begin()+1,counts.begin()+3
     );
cout << zeros[0] << "," << zeros[1] <<
     ","
    << zeros[2] << "," << zeros[3]
    << "\n";
```

Output

```
[iter] insert2:

0,2,3,0
```

5. Reconstruct index

Code:

```
vector<int> numbers{1,3,5,7,9};
auto it=numbers.begin();
while ( it!=numbers.end() ) {
  auto d = distance(numbers.begin(),it
     );
  cout << "At distance " << d << " we
     find " << *it << endl;
  it++;
}
```

Output

```
[loop] distance:

At distance 0 we find
     1
At distance 1 we find
     3
At distance 2 we find
     5
At distance 3 we find
     7
At distance 4 we find
     9
```

Algorithms

6. Reduction operation

Default is sum reduction:

Code:

```
vector<int> v{1,3,5,7};
auto first = v.begin();
auto last  = v.end();
auto sum = accumulate(first,last,0);
cout << "sum: " << sum << endl;
```

Output

```
[stl] accumulate:

sum: 16
```

7. Reduction with supplied operator

Supply multiply operator:

Code:

```
vector<int> v{1,3,5,7};
auto first = v.begin();
auto last  = v.end();
first++; last--;
auto product =
  accumulate
    (first,last,2,multiplies<>());
cout << "product: " << product << endl
     ;
```

Output

```
[stl] product:

product: 30
```

8. Use lambda to find any of

Here is an example using any_of to find whether there is any even element in a vector:

Code:

```
vector<int> integers{1,2,3,5,7,10};
auto any_even = any_of
  ( integers.begin(),integers.end(),
    [=] (int i) -> bool { return i
     %2==0; }
    );
if (any_even)
  cout << "there was an even" << endl;
else
  cout << "none were even" << endl;
```

Output

```
[range] anyof:

there was an even
```

Exercise 1

Use `for_each` to sum the elements of a vector.

Hint: the problem is how to treat the sum variable. Do not use a global variable!