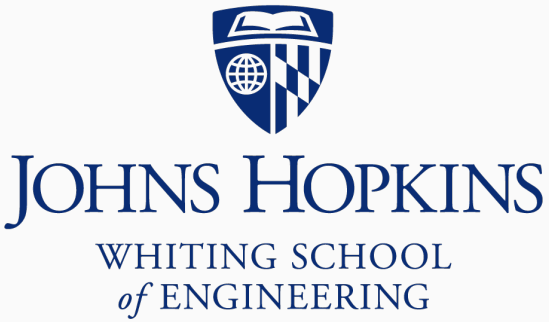
C++: More on iterators

Ben Langmead

ben.langmead@gmail.com

www.langmead-lab.org



Source markdown available at github.com/BenLangmead/c-cpp-notes

C++: More on iterators

Consider these map iterator types:

- `map<int, string>::iterator` – iterator over a `map`
- `map<string, map<string, int> >::iterator` – iterator over a `map` *where the values are themselves* `map`s

`typedef` can help by:

- Reducing clutter
- Bringing related type declarations closer together in your code:

```
typedef map<int, string> TMap;    // map type
typedef TMap::iterator TMapItr;   // map iterator type
```

C++: Iterators

With `iterator` (or `reverse_iterator`) you can modify the data structure via the dereferenced iterator:

```
typedef vector<int>::iterator TItr;

void prefix_sum(TItr begin, TItr end) {
    int sum = 0;
    for(TItr it = begin; it != end; ++it) {
        *it += sum;
        sum += *it;
    }
}
```

```
$ g++ -c prefix_sum_iter.cpp -std=c++11 -pedantic -Wall -Wextra
$ g++ -o prefix_sum_iter prefix_sum_iter.o
$ ./prefix_sum_iter
Before: 1 1 1 1
After: 1 2 4 8
```

C++: Iterators

const_iterator *does not* allow modifications

```
typedef vector<int>::const_iterator TItr;
//                  ^^^^^

void prefix_sum(TItr begin, TItr end) {
    int sum = 0;
    for(TItr it = begin; it != end; ++it) {
        *it += sum;
        sum += *it;
    }
}
```

```
$ g++ -c prefix_sum_iter.cpp -std=c++11 -pedantic -Wall -Wextra
prefix_sum_iter.cpp: In function 'void prefix_sum(TItr, TItr)':
prefix_sum_iter.cpp:13:16: error: assignment of read-only location
'it.__gnu_cxx::__normal_iterator<const int*, std::vector<int> >::operator*()'
        *it += sum;
               ^~~
```

C++: Iterators

Type	++it	--it	Get with	*it type
`iterator`	forward	back	`.begin()/end()`	-
`const_iterator`	forward	back	`.cbegin()/cend()`	const
`reverse_iterator`	back	forward	`.rbegin()/rend()`	-
`const_reverse_iterator`	back	forward	`.crbegin()/crend()`	const

Reminder: When using a variant of `.begin()`/`.end()`, always initialize the the iterator with `.begin()` and stop as soon as it == `.end()`. Don't switch the two, even if you're *trying* to go backwards. Use the `reverse_` version instead.