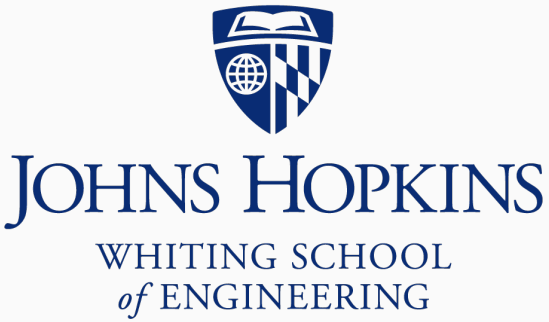
Ranged for

Ben Langmead

ben.langmead@gmail.com

www.langmead-lab.org



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

Ranged for

C++11 gives us a concise loop syntax, similar to Java for-each loops:

```
#include <iostream>

using std::cout; using std::endl;

int main() {
    for(int i : {2, 4, 6, 8}) {
        cout << i << ' ';
    }
    cout << endl;
    return 0;
}
```

Ranged for

```
$ g++ -c ranged_for1.cpp -std=c++11 -pedantic -Wall -Wextra
$ g++ -o ranged_for1 ranged_for1.o
$ ./ranged_for1
2 4 6 8
```

Ranged for

Works on our favorite STL containers:

```
#include <iostream>
#include <vector>

using std::cout; using std::endl;
using std::vector;

int main() {
    vector<int> vec = {10, 8, 6, 4, 2, 0};
    for(int i : vec) {
        cout << i << ' ';
    }
    cout << endl;
    return 0;
}
```

Ranged for

```
$ g++ -c ranged_for2.cpp -std=c++11 -pedantic -Wall -Wextra
$ g++ -o ranged_for2 ranged_for2.o
$ ./ranged_for2
10 8 6 4 2 0
```

Ranged for

Often combined with auto

```
#include <iostream>
#include <string>
#include <map>

using std::cout; using std::endl;
using std::map;  using std::string;

int main() {
    map<string, int> pres;
    pres["Washington"] = 1789;
    pres["Adams"] = 1797;
    pres["Jefferson"] = 1801;

    // compiler infers pair<string, int>
    for(auto kv : pres) {
        cout << kv.first << ':' << kv.second << endl;
    }
    return 0;
}
```

Ranged for

```
$ g++ -c ranged_for3.cpp -std=c++11 -pedantic -Wall -Wextra
$ g++ -o ranged_for3 ranged_for3.o
$ ./ranged_for3
Adams:1797
Jefferson:1801
Washington:1789
```

Avoided writing either the iterator type or the type of the dereferenced iterator

- Iterator type: `map<string, int>::iterator`
- Dereferenced iterator type: `pair<string, int>`

Ranged `for`

Works with pretty much any data structure with `begin()` & `end()` returning appropriate iterators

Including all STL containers we've discussed in class