Range-Based For Loops and Auto

Petr Zemek

Principal Developer
Security/Engineering/VirusLab
https://petrzemek.net

C++ Developers' Guild, 2016-08-12

Introduction

```
1 std::map<std::string, int> wordCount;
```

Introduction

```
1 std::map<std::string, int> wordCount;

2 for (std::map<std::string, int>::iterator i =
3          wordCount.begin(), e = wordCount.end();
4          i != e; ++i) {
5     // ... word: i->first, count: i->second
6 }
```

Introduction

```
1 std::map<std::string, int> wordCount;

2 for (std::map<std::string, int>::iterator i =
3         wordCount.begin(), e = wordCount.end();
4         i != e; ++i) {
5     // ... word: i->first, count: i->second
6 }

7 for (std::pair<const std::string, int>& p :
8         wordCount) {
9     // ... word: p.first, count: p.second
10 }
```

Introduction

```
1 std::map<std::string, int> wordCount;

2 for (std::map<std::string, int>::iterator i =
3         wordCount.begin(), e = wordCount.end();
4         i != e; ++i) {
5     // ... word: i->first, count: i->second
6 }

7 for (std::pair<const std::string, int>& p :
8         wordCount) {
9     // ... word: p.first, count: p.second
10 }

11 for (auto& p : wordCount) {
12     // ... word: p.first, count: p.second
13 }
```

Introduction

```
1  std::map<std::string, int> wordCount;

2  for (std::map<std::string, int>::iterator i =
3          wordCount.begin(), e = wordCount.end();
4          i != e; ++i) {
5      // ... word: i->first, count: i->second
6  }

7  for (std::pair<const std::string, int>& p :
8          wordCount) {
9      // ... word: p.first, count: p.second
10 }

11 for (auto& p : wordCount) {
12     // ... word: p.first, count: p.second
13 }

14 for (auto& [word, count] : wordCount) { // C++1z
15     // ...
16 }
```

Auto Type Deduction Possibilities

- `auto`
- `const auto`
- `auto&`
- `const auto&`
- `auto&&`
- `const auto&&`
- `decltype(auto) // C++14`

auto / const auto

```
for (auto x : range)
```

- when you want to work with a copy

auto / const auto

```
for (auto x : range)
```

- when you want to work with a copy
- beware of proxy types (e.g. `std::vector<bool>`)

auto / const auto

```
for (auto x : range)
```

- when you want to work with a copy
- beware of proxy types (e.g. `std::vector<bool>`)
- doesn't work with move-only types (e.g. `std::unique_ptr`)

auto / const auto

```
for (auto x : range)
```

- when you want to work with a copy
- beware of proxy types (e.g. `std::vector<bool>`)
- doesn't work with move-only types (e.g. `std::unique_ptr`)

```
for (const auto x : range)
```

- when you want to work with an immutable copy

auto / const auto

```
for (auto x : range)
```

- when you want to work with a copy
- beware of proxy types (e.g. `std::vector<bool>`)
- doesn't work with move-only types (e.g. `std::unique_ptr`)

```
for (const auto x : range)
```

- when you want to work with an immutable copy
- may seem like you forgot to use `&`

auto / const auto

```
for (auto x : range)
```

- when you want to work with a copy
- beware of proxy types (e.g. `std::vector<bool>`)
- doesn't work with move-only types (e.g. `std::unique_ptr`)

```
for (const auto x : range)
```

- when you want to work with an immutable copy
- may seem like you forgot to use `&`
- no reason?

auto& / const auto&

```
for (auto& x : range)
```

- when you want to modify original items (non-generic code)

auto& / const auto&

```
for (auto& x : range)
```

- when you want to modify original items (non-generic code)

```
for (const auto& x : range)
```

- when you want read original items (even in generic code)

auto&& / const auto&&

```
for (auto&& x : range)
```

- when you want to modify original items in generic code

auto&& / const auto&&

```
for (auto&& x : range)
```

- when you want to modify original items in generic code
- may lead to "confuscated" code (Howard Hinnant)

auto&& / const auto&&

```
for (auto&& x : range)
```

- when you want to modify original items in generic code
- may lead to "confuscated" code (Howard Hinnant)

```
for (const auto&& x : range)
```

- no reason?

decltype(auto)

```
for (decltype(auto) x : range) // C++14
```

- no reason?

References and Further Information

Scott Meyers
Effective Modern C++
O'Reilly Media, 2014, 336 pages

Talks:

- Scott Meyers: Type Deduction and Why You Care (CppCon 2014)
  - https://www.youtube.com/watch?v=wQxj20X-tIU

SO questions:

- What is the advantage of using auto&& in range-based for loops?
  - http://stackoverflow.com/q/13130708/2580955
- What is the correct way of using C++11's range-based for?
  - http://stackoverflow.com/q/15927033/2580955
- Range-based for loop with decltype(auto)?
  - http://stackoverflow.com/q/38421392/2580955