New Features in C++14

Mochan Shrestha

Ann Arbor C++ Meetup

June 24, 2015

1. C++14 is a minor release.
2. Builds up on C++11

[[deprecated]] attribute

1. Adding `[[deprecated]]` produces a compiler warning

Binary Literals

1. We have decimal, hex and octal support. Now binary is supported
2. `int x = 0b1111101011001110`

Digit Seperator

1. Use single quotes to seperate digits for ease of reading
2. `int x = 0b1111'1010'1100'1110`
3. Location of single quote doesn't matter. Ignored by compiler
4. `int x = 3'00.0`

Variable Templates

1. Variable type can now also be templated.

```
template <typename T>
T pi = 3.14159;
```

Return Type Deduction

1. C++11 introduced the new keyword `auto`
2. C++14 exapnds it for return types as well
3. Still same compile-time type safety

Return Type Deduction

1. C++11 also introduced `decltype`
2. C++14 now allows for `decltype(auto)`
3. Just like in C++11, `decltype` manages to maintain the references.

Relaxed constexpr Restrictions

1. C++11 introduced `constexpr` which allowed compile time evaluation of the variables and functions.
2. C++14 allows using `if`, `switch` and `for` among others.

Generic Lambdas

1. C++11 introduced lambda functions

```
vector<int> v = {1, 2, 3, 4, 5, 10, 15, 20, 25, 30, 35, 40, 45};
sort(v.begin(), v.end(), [](int i, int j) -> bool {return i>j;});
```

2. Parameters in the lambda function can now also be auto

```
vector<int> v = {1, 2, 3, 4, 5, 10, 15, 20, 25, 30, 35, 40, 45};
auto cmpf = [](auto i, auto j) -> auto {return i>j;};
sort(v.begin(), v.end(), cmpf);
```

3. Generic lambda functions can now act as templates

Initalized Lambda Captures

1. C++11 lambdas had a capture section that would take any referenced variable
2. C++14 allows for any kind of initalization on the captured members
3. Useful for capture by move (`std::move`)