Practical C++14 and C++17 Features

CONVENIENT SYNTACTIC SUGAR

Giovanni Dicanio
AUTHOR, SOFTWARE ENGINEER

https://blogs.msmvps.com/gdicanio

Prerequisites and Learning Objectives

Practical C++14 and C++17 Features

C++11

C++14

C++17

Tools for writing *simpler, clearer, higher-quality* C++ code

Prerequisites and Learning Objectives

C++11 from Scratch

Basic C++11 knowledge

C++ Fundamentals

C++11 Language Features

...

Practical C++14 and C++17 Features

Tools for writing *simpler, clearer, higher-quality* C++ code

Prerequisites and Learning Objectives

Already know C++14

Go to module

Practical Convenient C++17 Language Improvements

Tools for writing *simpler, clearer, higher-quality* C++ code

Standard C++

Clang/LLVM logo by http://llvm.org/Logo.html

«C++11 from Scratch»
Building C++ Programs

Compiling from the Command Line

bit.ly/CppCompile

Overview

Digit separators

Binary literals

Automatic return type deduction

10000000

10,000,000

Digit Separators Improve Readability

10,000,000

```
long x = 10'000'000;
double EarthDiameterKm = 12'742;
```

C++14 Digit Separator

APOSTROPHE (U+0027): '

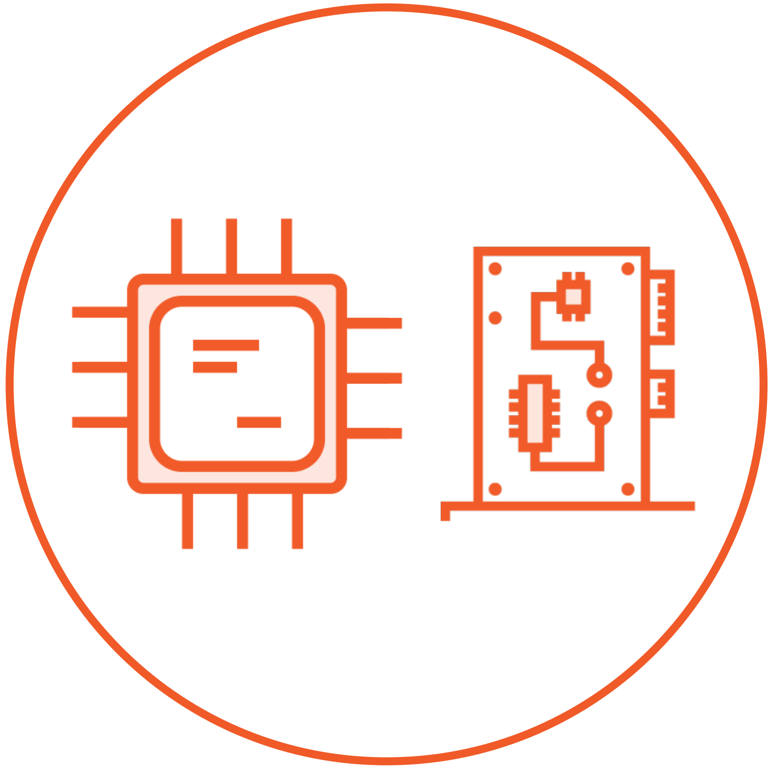
```
long x = 10'000'000;
long y = 10'00'00'00; // == x !!
```

Clear, readable

C++14 Digit Separator

Note: Position is *arbitrary*

Binary Values



01000111

```
auto d = 0x47; // Binary data:  01000111
```

Binary Data Written in Comments

```
0b
0B
```

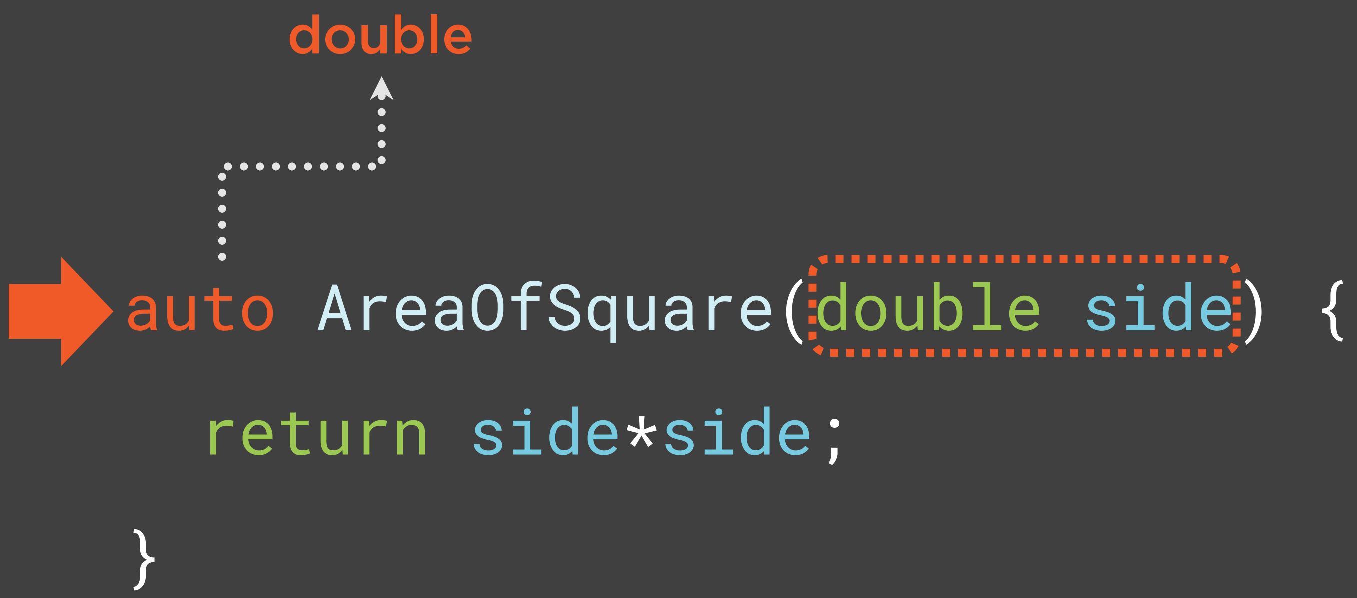
```cpp
auto d = 0b 01000111;
```

C++14 Binary Literals

Binary data written *in code*

```
auto word = 0b 01000111'01000011 ;
```

C++14 Binary Literals + Digit Separators



```
auto AreaOfSquare(double side) {
   return side*side;
}
```

Automatic Return Type Deduction

Complex type *automatically* deduced

```
auto ComplexFunctionTemplate(...) {
  // Complex template code…
  return result;
}
```

Automatic Return Type Deduction

Comes in handy for templates and cumbersome/noisy types

```cpp
auto BuildCoolMap() {
  std::map<std::string, SomeLongValueType> result;
  // Fill the result map object…
  return result;
}
```

Complex *return* type *automatically* deduced

Automatic Return Type Deduction

Don't want to bother mentioning the return type *twice*

Balance for Using auto Return Type Deduction

Looking *inside* the function implementation

***Not* having *repeated* type information**

Summary

Digit separators

Binary literals

Automatic return type deduction