Integer Literals

Explicit values like `235` or `-75` are called literals. An integer literal is a sequence of decimal digits, with no spaces or commas allowed, preceded by an optional (`+/-`) sign. It is stored as a **signed** `int`.

- Change the representation from `signed` to `unsigned` by add a `U` to the end.
- Change the storage from `int` to `long`, or to `long long` by adding an `L` or an `LL`.

Here are some examples:

```
auto a = 15;      // a is stored as a signed decimal int
auto b = 15L;     // b is stored as a signed decimal long
auto c = 15LL;    // c is stored as a signed decimal long long
auto d = 15UL;    // d is an unsigned decimal long
```

*Using `auto` instead of an explicit type to create the variables `a`, `b`, `c`, and `d`, allows the compiler to **infer** or **deduce** their types from their initializers. This **type inference** is a new feature of C++11.*

You can also write literals in base 8 (**octal**), base 16 (**hexadecimal**) and base 2 (**binary**).

```
auto oct32 = 040;           // 4 8s and no 0s
auto hex32 = 0x20;          // 2 16s and no 0s
auto bin32 = 0b10'0000;     // 1 32 and no 16s, 8s, 4s, 2s or 1s
```

Starting in C++14 you can use the apostrophe as a visual separator, as I've done here to separate the digits in `bin32` into groups of 4.