Chapter 2: Literals

Section 2.1: uint literals

`uint` literals are defined by using the suffix `U` or `u`, or by using an integral values within the range of `uint`:

```
uint ui = 5U;
```

Section 2.2: int literals

`int` literals are defined by simply using integral values within the range of `int`:

```
int i = 5;
```

Section 2.3: sbyte literals

`sbyte` type has no literal suffix. Integer literals are implicitly converted from `int`:

```
sbyte sb = 127;
```

Section 2.4: decimal literals

`decimal` literals are defined by using the suffix M or m on a real number:

```
decimal m = 30.5M;
```

Section 2.5: double literals

`double` literals are defined by using the suffix D or d, or by using a real number:

```
double d = 30.5D;
```

Section 2.6: float literals

`float` literals are defined by using the suffix F or f, or by using a real number:

```
float f = 30.5F;
```

Section 2.7: long literals

`long` literals are defined by using the suffix `L` or `l`, or by using an integral values within the range of `long`:

```
long l = 5L;
```

Section 2.8: ulong literal

`ulong` literals are defined by using the suffix `UL`, `ul`, `Ul`, `uL`, `LU`, `lu`, `Lu`, or `lU`, or by using an integral values within the range of `ulong`:

```
ulong ul = 5UL;
```

Section 2.9: string literals

`string` literals are defined by wrapping the value with double-quotes `"`:

```
string s = "hello, this is a string literal";
```

String literals may contain escape sequences. See String Escape Sequences

Additionally, C# supports verbatim string literals (See Verbatim Strings). These are defined by wrapping the value with double-quotes `"`, and prepending it with `@`. Escape sequences are ignored in verbatim string literals, and all whitespace characters are included:

```
string s = @"The path is:
C:\Windows\System32";
//The backslashes and newline are included in the string
```

Section 2.10: char literals

`char` literals are defined by wrapping the value with single-quotes `'`:

```
char c = 'h';
```

Character literals may contain escape sequences. See String Escape Sequences

A character literal must be exactly one character long (after all escape sequences have been evaluated). Empty character literals are not valid. The default character (returned by `default(char)` or `new char()`) is `'\0'`, or the NULL character (not to be confused with the `null` literal and null references).

Section 2.11: byte literals

`byte` type has no literal suffix. Integer literals are implicitly converted from `int`:

```
byte b = 127;
```

Section 2.12: short literal

`short` type has no literal. Integer literals are implicitly converted from `int`:

```
short s = 127;
```

Section 2.13: ushort literal

`ushort` type has no literal suffix. Integer literals are implicitly converted from `int`:

```
ushort us = 127;
```

Section 2.14: bool literals

`bool` literals are either `true` or `false`;

```
bool b = true;
```