Chapter 7: Literals for numbers, characters and strings

Section 7.1: Floating point literals

Floating point literals are used to represent signed real numbers. The following suffixes can be used to specify type of a literal:

Suffix	Type	Examples
none	`double`	`3.1415926` `-3E6`
`f`, `F`	`float`	`3.1415926f` `2.1E-6F`
`l`, `L`	`long double`	`3.1415926L` `1E126L`

In order to use these suffixes, the literal *must* be a floating point literal. For example, `3f` is an error, since `3` is an integer literal, while `3.f` or `3.0f` are correct. For `long double`, the recommendation is to always use capital `L` for the sake of readability.

Section 7.2: String literals

String literals are used to specify arrays of characters. They are sequences of characters enclosed within double quotes (e.g. `"abcd"` and have the type `char*`).

The `L` prefix makes the literal a wide character array, of type `wchar_t*`. For example, `L"abcd"`.

Since C11, there are other encoding prefixes, similar to `L`:

prefix	base type	encoding
none	`char`	platform dependent
`L`	`wchar_t`	platform dependent
`u8`	`char`	UTF-8
`u`	`char16_t`	usually UTF-16
`U`	`char32_t`	usually UTF-32

For the latter two, it can be queried with feature test macros if the encoding is effectively the corresponding UTF encoding.

Section 7.3: Character literals

Character literals are a special type of integer literals that are used to represent one character. They are enclosed in single quotes, e.g. `'a'` and have the type `int`. The value of the literal is an integer value according to the machine's character set. They do not allow suffixes.

The `L` prefix before a character literal makes it a wide character of type `wchar_t`. Likewise since C11 `u` and `U` prefixes make it wide characters of type `char16_t` and `char32_t`, respectively.

When intending to represent certain special characters, such as a character that is non-printing, escape sequences are used. Escape sequences use a sequence of characters that are translated into another character. All escape sequences consist of two or more characters, the first of which is a backslash `\`. The characters immediately following the backslash determine what character literal the sequence is interpreted as.

Escape Sequence Represented Character

`\b`	Backspace
`\f`	Form feed
`\n`	Line feed (new line)
`\r`	Carriage return
`\t`	Horizontal tab
`\v`	Vertical tab
`\\`	Backslash
`\'`	Single quotation mark
`\"`	Double quotation mark
`\?`	Question mark
`\nnn`	Octal value
`\xnn...`	Hexadecimal value

`Version ≥ C89`

Escape Sequence	Represented Character
`\a`	Alert (beep, bell)

`Version ≥ C99`

Escape Sequence	Represented Character
`\unnnn`	Universal character name
`\Unnnnnnnn`	Universal character name

A universal character name is a Unicode code point. A universal character name may map to more than one character. The digits `n` are interpreted as hexadecimal digits. Depending on the UTF encoding in use, a universal character name sequence may result in a code point that consists of multiple characters, instead of a single normal `char` character.

When using the line feed escape sequence in text mode I/O, it is converted to the OS-specific newline byte or byte sequence.

The question mark escape sequence is used to avoid trigraphs. For example, `??/` is compiled as the trigraph representing a backslash character `'\'`, but using `?\?/` would result in the *string* `"??/"`.

There may be one, two or three octal numerals `n` in the octal value escape sequence.

Section 7.4: Integer literals

Integer literals are used to provide integral values. Three numerical bases are supported, indicated by prefixes:

Base	Prefix	Example
Decimal	None	`5`
Octal	`0`	`0345`
Hexadecimal	`0x` or `0X`	`0x12AB`, `0X12AB`, `0x12ab`, `0x12Ab`

Note that this writing doesn't include any sign, so integer literals are always positive. Something like `-1` is treated as an expression that has one integer literal (1) that is negated with a `-`

The type of a decimal integer literal is the first data type that can fit the value from `int` and `long`. Since C99, `long long` is also supported for very large literals.

The type of an octal or hexadecimal integer literal is the first data type that can fit the value from `int`, `unsigned`, `long`, and `unsigned long`. Since C99, `long long` and `unsigned long long` are also supported for very large literals.

Using various suffixes, the default type of a literal can be changed.

Suffix	Explanation
`L`, `l`	`long int`
`LL`, `ll` (since C99)	`long long int`
`U`, `u`	`unsigned`

The U and L/LL suffixes can be combined in any order and case. It is an error to duplicate suffixes (e.g. provide two `U` suffixes) even if they have different cases.