> 由 KiOii (*_EM_Cpper*)整理。 （KiOii (*_EM_Cpper*) makes this note.）

Part I: The Basics

Contents

Chapter 2. Variables and Basic Types

Contents

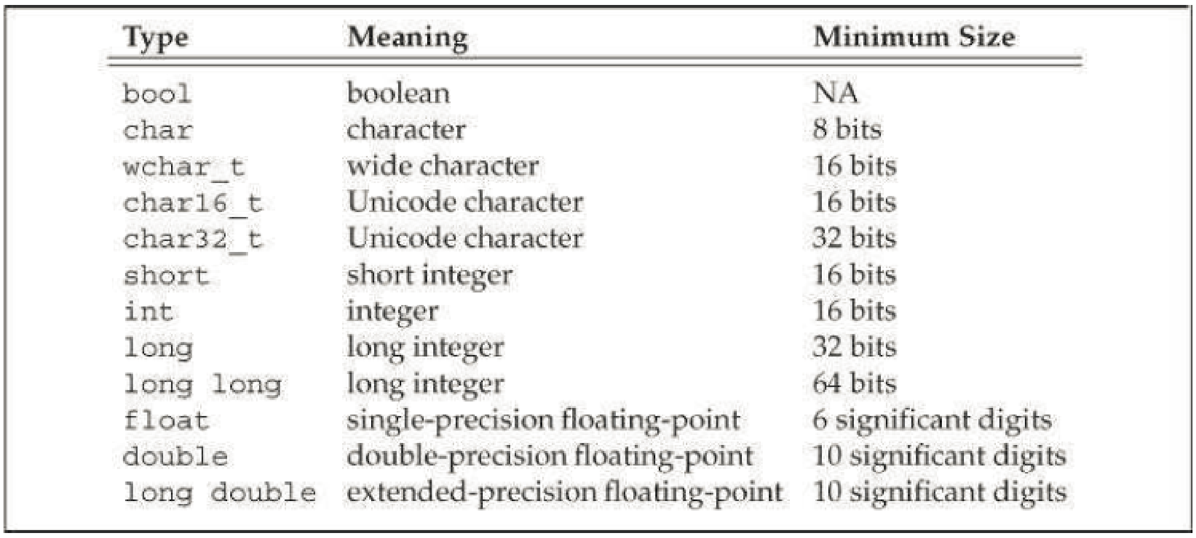
2.1 Primitive Built-in Types

- *include : **arithmetic types*** and ***void***

- *arithmetic-include:* ***characters*、*integers*、*boolean*、*float-point numbers***
- *void:* **no value,most as the return type**

2.1.1 Arithmetic Types

- *include:* ***integral types***(include ***character*** and ***boolean types***) and ***floating-point types***



Type	Meaning	Minimum Size
`bool`	boolean	NA
`char`	character	8 bits
`wchar_t`	wide character	16 bits
`char16_t`	Unicode character	16 bits
`char32_t`	Unicode character	32 bits
`short`	short integer	16 bits
`int`	integer	16 bits
`long`	long integer	32 bits
`long long`	long integer	64 bits
`float`	single-precision floating-point	6 significant digits
`double`	double-precision floating-point	10 significant digits
`long double`	extended-precision floating-point	10 significant digits

long long: introuced by **C++11** standard.

byte

- *defind:* **The smallest chunk of addressable memory**

Signed and Unsigned Types

- *include:* **expect for bool** and **the extended charactor type,**the **integeral types** may be **signed** or **unsigned**.
- *three-charactor types:* ***char*、*signed char*、*unsigned char***
  - *wraning:* ***char*** is not the same type as ***signed char***
  - *wraning:* due to compiler,***char*** is one of the ***signed char*** and ***unsigned char***[1]
  - *wraning:* **The standard does not define how signed types are represented**
- *advice:* Use ***double*** for floating-point computations

2.1.2 Type Conversions

what happens depends on the range of the values that the types permit:

- When we assign one of the nonbool arithmetic types to a bool object, theresult is false if the value is 0 and true otherwise.
- When we assign a bool to one of the other arithmetic types, the resulting value is 1 if the bool is true and 0 if the bool is false.
- When we assign a floating-point value to an object of integral type, the value is truncated. The value that is stored is the part before the decimal point.

- When we assign an integral value to an object of floating-point type, the fractional part is zero. Precision may be lost if the integer has more bits than the floating-point object can accommodate.
- If we assign an out-of-range value to an object of unsigned type, the result is the remainder of the value modulo the number of values the target type can hold. For example, an 8-bit unsigned char can hold values from 0 through 255, inclusive. If we assign a value outside this range, the compiler assigns the remainder of that value modulo 256. Therefore, assigning –1 to an 8-bit unsigned char gives that object the value 255.
- If we assign an out-of-range value to an object of signed type, the result is undefined. The program might appear to work, it might crash, or it might produce garbage values.

Expressions Involving Unsigned Types

both unsigned and int: ***int*** is converted to ***unsigned***

wraning: **unsigned never be less than 0**

2.1.3 Literals

- *wraning:* **Every literal has a type**

Integer and Floating-Point Literals

Integer Literals

- *notation:* **decimal、octal、hexadecimal**
  - *decimal:* such that **20**   **signed**
  - *octal:* begin with 0,such that **024**   **unsigned or signed**
  - *hexadecimal:* begin with 0x,such that **0x14**   **unsigned or signed**

Floating-Point Literals

- notation: 3.14159、3.14159E0、0.、0e0、.001

Character and Character String Literals

```
'a'            // character literal
"Hello World!" // string literal
```

string literal: **array of constant chars**

- *wraning:* **The compiler appends a null character ('\0') to every string literal.**

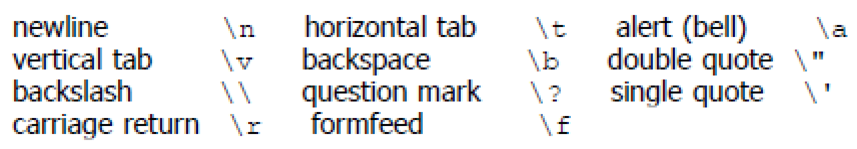
```
// multiline string literal
std::cout << "a really,really long string literal "
          << "that spans two lines" << std::endl;
```

Escape Sequences

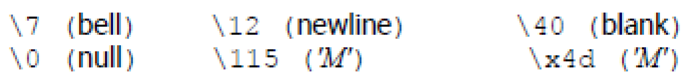
No Visible Imge Character

- *such that:* **backspace** or **control characters**

- *sort:* nonprintable、 escape sequence
- *escape sequence*
    -



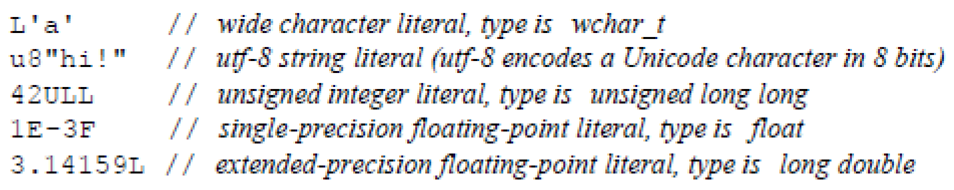
newline	`\n`	horizontal tab	`\t`	alert (bell)	`\a`
vertical tab	`\v`	backspace	`\b`	double quote	`\"`
backslash	`\\`	question mark	`\?`	single quote	`\'`
carriage return	`\r`	formfeed	`\f`		



`\7` (bell)	`\12` (newline)	`\40` (blank)
`\0` (null)	`\115` ('M')	`\x4d` ('M')

- wraning: **Note that if a \ is followed by more than three octal digits, only the first three are associated with the \.**
- wraning:**\x uses up all the hex digits following it**

Specifying the type of a Literal



```
L'a'     // wide character literal, type is wchar_t
u8"hi!"  // utf-8 string literal (utf-8 encodes a Unicode character in 8 bits)
42ULL    // unsigned integer literal, type is unsigned long long
1E-3F    // single-precision floating-point literal, type is float
3.14159L // extended-precision floating-point literal, type is long double
```

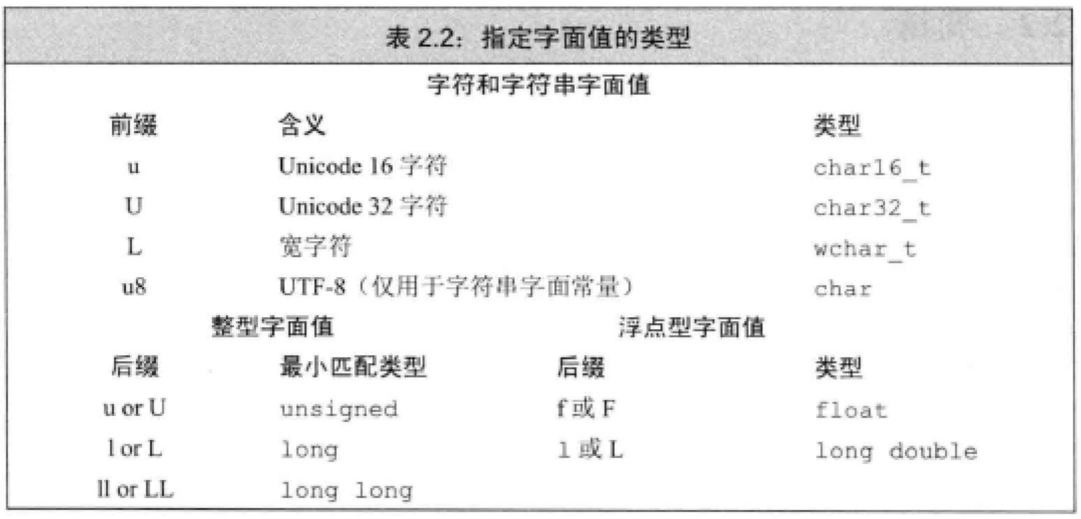


表 2.2：指定字面值的类型

字符和字符串字面值

前缀	含义	类型
u	Unicode 16 字符	`char16_t`
U	Unicode 32 字符	`char32_t`
L	宽字符	`wchar_t`
u8	UTF-8（仅用于字符串字面常量）	`char`

整型字面值		浮点型字面值	
后缀	最小匹配类型	后缀	类型
u or U	`unsigned`	f 或 F	`float`
l or L	`long`	l 或 L	`long double`
ll or LL	`long long`		

Boolean and Pointer Literals

Boolean Literals

- *two:* true and false

Pointer Literals

- nullptr

1. 不确定时，最好显式指定***signed char*** 或者 ***unsigned char***↩