数据类型

整数存储：二进制“补码”格式

整数以二进制“补码”格式存储

①正数的补码=正数的原码

+5 0000 0101 原码=补码

②负数的补码=负数的原码按位求反（符号位保持1不变），末位加1

-5 1000 0101 原码

1111 1010 补码

1111 1011 补码 其值为：-1*2^7+1*2^6+1*2^5+1*2^4+1*2^3+0*2^2+1*2^1+1*2^0=-5

int取值范围：-2^7 ~ 2^7 - 1

在int型为32位的硬件环境中，int正数的补码范围是0000 0001~ 0111 1111，即1~2^7-1

int 负数的补码范围是1000 0000~ 1111 1111即-2^7~-1，

因此int型整数的取值范围是-2^7~2^7-1。

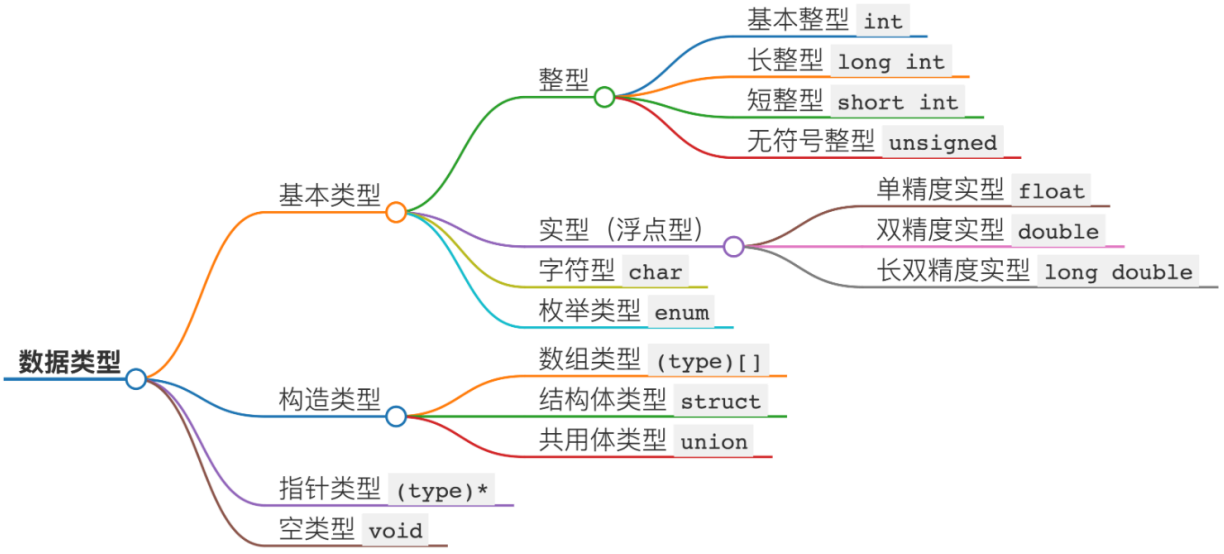
位（bit）：0 或 1，单位b

字节（Byte）：8bit，表示0~255间整数,单位B

表 1-1 衡量内存空间大小的表示单位

英文称谓	容量大小（单位字节）	换算方法
B	2 的 0 次方	1 B == 8 b
KB	2 的 10 次方	1 KB == 1,024 B
MB	2 的 20 次方	1 MB == 1,024 KB
GB	2 的 30 次方	1 GB == 1,024 MB
TB	2 的 40 次方	1 TB == 1,024 GB
PB	2 的 50 次方	1 PB == 1024 TB
EB	2 的 60 次方	1 EB == 1024 PB
ZB	2 的 70 次方	1 ZB == 1024 EB
YB	2 的 80 次方	1 YB == 1024 ZB

C语言中的数据类型：



无符号类型不会溢出，运算结果始终在 0到 2^n-1范围内（例如 255 + 1 在 uint8_t 中等于 0）
有符号类型溢出行为未定义（可能导致程序崩溃或安全漏洞）

浮点数存储：

IEEE二进制浮点数算术标准（IEEE 754）

深圳大学 SHENZHEN UNIVERSITY

IEEE 754 浮点数由三个部分组成：

- 符号位（Sign）：1 位，0 表示正数，1 表示负数。
- 指数位（Exponent）：k 位，存储经过偏置（Bias）处理的指数值。
- 尾数位（Mantissa/Fraction）：n 位，存储小数部分的有效数字。

Ascii及其扩展：

ASCII

- 最早的字符编码标准，诞生于 1963 年。它使用7 位二进制数（范围 0-127）表示 128 个字符
- 控制字符（0-31，如回车\r、换行\n）
- 可打印字符（32-126，包括英文字母、数字、标点符号）
- 删除字符（127）
- 示例
  - 'A' → 65 (0x41)
  - '0' → 48 (0x30)
  - 空格(' ') → 32 (0x20)

扩展 ASCII

- 标准 ASCII 仅覆盖 128 个字符，无法满足其他语言的需求。扩展 ASCII 使用 8 位二进制数（范围 0-255），将标准 ASCII 从 128 扩展到 256 个字符。

Unicode与UTF：

Unicode 与 UTF 编码

- 不同的国家和地区制定了不同的编码标准，互不兼容
- 解决方案：Unicode
- 它为世界上所有字符分配了唯一的数字编号（码点，Code Point），范围从 U+0000 到 U+10FFFF
- Unicode 与 UTF 的关系
- Unicode：字符集，定义字符与码点的映射关系。
- UTF：编码方式，定义如何将码点转换为二进制存储。

特殊字符：

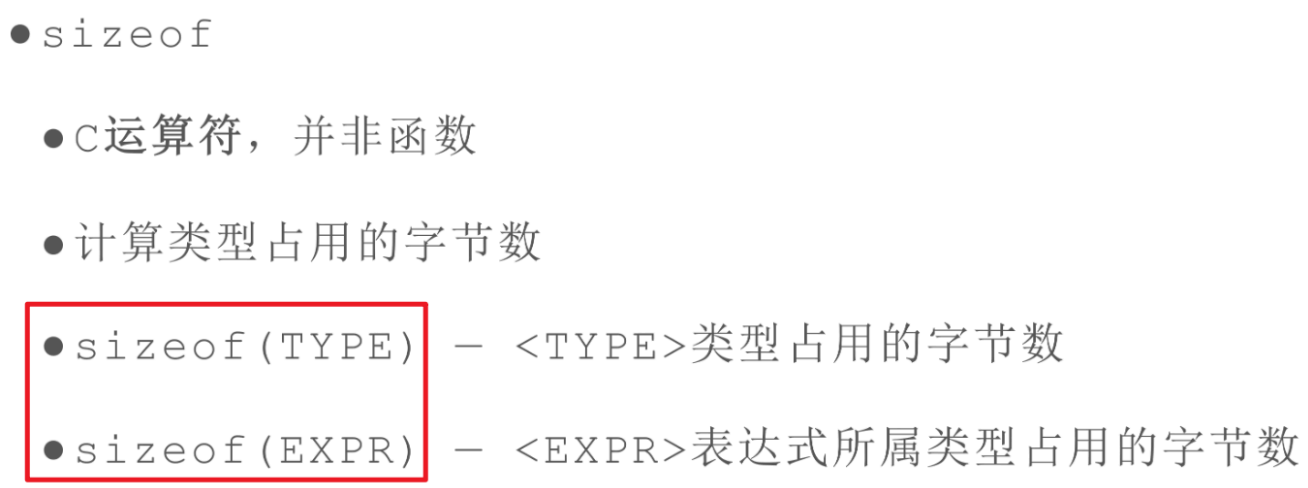
代表性字符及其编码

- 'A'和65等价，其余大写字母编码依次加1
- 'a'和97等价，其余小写字母编码依次加1
- 'a'-'A'=32，即大小写互换时，32是偏移量
- '0'和48等价，其余数字字符编码依次加1
- '7'-'0'=7，即数字字符-'0'被称为数字字符的数值化

转义字符序列

- 某些字符难以直接打出，需要通过转义字符串来表示
- '\n': 换行(LF)，相当于按下回车键
- '\r': 回车(CR)，与\n功能几乎相同，使用较少
- '\t': 制表符，相当于按下"Tab⇥"键
- '\0': 结束字符，数值等于0
- '\e': escape，具有许多功能

变量/类型的内存占用大小：



size_t：无符号，大小由系统架构决定，是sizeof()的返回类型

系统架构	通常大小	对应类型
32位系统	4字节	`unsigned int`
64位系统	8字节	`unsigned long long`