计算机
用浮点数表示小数

2进制小数的局限性：一些小数转化为2进制会无限循环

- 2进制小数转10进制小数
  - 111.111
  - 整数部分：4 + 2 + 1
  - 小数部分： 2的-1次幂 + 2的-2次幂 + 2的-3次
  - 幂 = 1/2 + 1/4 + 1/8 = 0.5 + 0.25 + 0.125
  - 其10进制为：7.875
- 10进制小数转2进制小数
  - 有限2进制小数
    - 10进制小数0.5
    - 0.5 × 2 = 1.0
    - 其2进制为：0.1
  - 无限2进制小数
    - 10进制小数0.1
    - 0.1 × 2 = 0.2
    - 0.2 × 2 = 0.4
    - 0.4 × 2 = 0.8
    - 0.8 × 2 = 1.6
    - 0.6 × 2 = 1.2
    - 0.2 × 2 = 0.4
    - 0.4 × 2 = 0.8
    - 0.8 × 2 = 1.6
    - 0.6 × 2 = 1.2
    - ...
    - loop
    - 其2进制为：
    - 0.0001100110011001...

浮点数4要素

浮点数通过以下表达式

± m × n^e

可以算出小数位的值

- 符号 ±
  - 1为负数
  - 0为正数或0
- 尾数 m
  - '将小数点前的值固定为1'的正则表达式冒号内的内容即浮点数规则
  - **小贴士：按照特定的规则来表示数据的形式即为正则表达式**
    - 例如有这样的10进制浮点数规则:
    - '小数点前为0，小数点后第1位不能为0'
    - 那么在以下3个表达式中
    - 1. 0.75 = 0.75 × 10^0
    - 2. 0.75 = 0.075 × 10^1
    - 3. 0.75 = 0.0075 × 10^2
    - 只有1式为正则表达式
  - 小贴士：对于'尾数'这个名字，单看其在浮点数表达式中m的位置，是很难理解的。但其实从表示形式来看，由于他是正则表达式'小数点后'的部分，故名为尾数
  - 实现正则表达式的方式：
    - 通过逻辑运算，将任意2进制小数左移或者右移得到一个整数位只有第1位为1，其他全是零，
    - 并且小数位用0凑齐尾数位的2进制数
    - 该2进制数即为完成的正则表达式
- 指数 e
  - 'EXCESS系统表现'
  - 通过将指数部分表示范围的中间值设为 0，使得负数不需要用符号来表示
    - 指数部分是 8 位单精度浮点数时，
    - 最大值 11111111 = 255 的 1/2，即 01111111 = 127（小数部分舍弃）表示的是 0
    - 原本00000000 ~ 11111111的范围是：
    - 0-255（共256个数）
    - EXCESS系统表现的8位2进制分为正负范围：
    - 00000000 ~ 01111111
    - -127 ~ 0（共128个负数）
    - 10000000 ~ 11111111
    - 1 ~ 128（共128个正数）
    - 指数部分是 11 位双精度浮点数时，
    - 最大值11111111111 = 2047 的 1/2，即
    - 01111111111 = 1023（小数部分舍弃）表示的是 0
    - 11位2进制范围是：
    - 0-2047（2048个数）
    - 那么他的EXCESS系统表现则会出现：
    - -1023 ~ 0（共1024个负数）
    - 1 ~ 1024（共1024个正数）
- 基数 n
  - 计算机内部使用2进制
  - 故基数n恒为2

小数0.75用单精度浮点数算出的过程

当我们用float给一个变量A赋值为0.75时，
C语言会将0.75编译成单精度浮点数：
0-01111110-10000000000000000000000

其中

- 符号部分：0 表示该小数是正数
- 指数部分：01111110
  - 原数值为126
  - EXCESS系统表现下的值为（126 - 127 = -1）
- 尾数部分：10000000000000000000000
  - ⬇ 作为2进制小数时其值为1.1
  - 转化为10进制数的结果是：1 + 1/2 = 1.5

结果

将上述的各个参数代入浮点数表达式得到：
+ 1.5 × 2^(-1) = 0.75

C语言中的浮点数数据类型（IEEE浮点数标准）

- double；双精度浮点数（64位）
  - 数值范围大于单精度浮点数
  - 符号部分（1位）+ 指数部分（11位）+ 尾数部分（52位）
- float；单精度浮点数（32位）
  - 符号部分（1位）+ 指数部分（8位）+ 尾数部分（23位）

正则表达式的例子：
2进制小数的原始数据 1011.0011

- ⬇ 右移使整数部分的第一位为1
- 1.0110011
- ⬇ 确保小数点后长度为单精度浮点数尾数的23位
- 1.01100110000000000000000
- ⬇ 仅保留小数点后的部分，得到尾数
- 01100110000000000000000
  - 小贴士：由于正则表达式的存在，该数实际上是一个个位数为1的2进制小数

用浮点数表示小数的弊端（及回避策略）

无法100%精确地用浮点数替换小数
例如：计算0.1所对应的浮点数，算出结果可能是0.99999999。

1. 忽略微小误差
2. 将小数等比例转化成整数，计算后再等比例将整数转化成小数
3. Binary Coded Decimal（BCD方法，精度更高的用2进制表示10进制的方法，此处略）