第 2 章数制预编码（总结+考点）

这部分是计算机世界的基础

这部分涉及一些规范或标准的定义，学习的思路是首先理解该知识点是为了解决什么问题？这个问题让你解决你有什么方案？再看看目前业界是怎么解决的？补码、移码和 IEEE754 都是这个思路。

1、 计算机中数据分类：**数值型**和**非数值型**，数值型要掌握十进制和二进制数值间的转换，二进制和十六进制之间的转换，掌握常用 BCD 码和十进制 0-9 之间的关系；非数值型数据又分逻辑数据和字符数据等。。

2、 **整数表示**：n 位无符号整数的数据表示范围？有符号数表示范围？**掌握有符号整数的补码**（重点）表示，给出真值能得到对应补码，给了补码能得到对应的真值。

3、 有符号数采用补码表示有哪 3 个好处？说明一下：书上有小数补码的例题，这部分可以不用看，因为小数在计算机里是以浮点数形式存放的，补码都是针对有符号整数而言，不讨论小数的补码。

4、 什么是**零扩展**，什么是**符号位扩展**？

5、 为什么引入移码（增码）？它和补码的关系？

例 2.29 以下是一个 C 语言程序，用来计算一个数组 a 中每个元素的和。当参数 len 为 0 时，返回值应该是 0，但是在机器上执行时，却发生了存储器访问异常。请问这是什么原因造成的，并说明程序应该如何修改。

```
1  float sum_array(int a[], unsigned len)
2  {
3      int i, sum = 0;
4
5      for(i = 0; i <= len-1; i++)
6          sum += a[i];
7
8      return sum;
9  }
```

6、 实数表示中，什么是定点表示？什么是浮点表示？体会引入浮点表示的优点，将**数据表示范围（阶码）**和**精度（尾数）**分开表示的好处。在字长一定的情况下如何进行取舍？（难两全，不同的浮点数标准的本质差别就在这里）

阶码和尾数常用补码或移码表示，具体看上下文约定，这里没有对错，只要统一即可。例如现在大多数计算机采用的 **IEEE754 标准**中，尾数就采用的是**原码**，阶码采用的是**移码（偏置是 127 或 1023（单双精度的区别））**，一个浮点数其实是由两个定点数来描述的。

7、 掌握 IEEE754 浮点表示标准（大多数计算机浮点表示都采用该标准），其中尾数部分巧妙的省了一位（怎么省的？），单精度偏置为什么选择 127，选择其他值可以吗（其实可以，只是数据表示范围的取舍）？

（假定 int 为 32 位）。

① 从 int 转换为 float 时，不会发生溢出，但有效数字可能被舍去。

② 从 int 或 float 转换为 double 时，因为 double 的有效位数更多，故能保留精确值。

③ 从 double 转换为 float 时，因为 float 表示范围更小，故可能发生溢出，此外，由于有效位数变少，故数据可能被舍入。

④ 从 float 或 double 转换为 int 时，因为 int 没有小数部分，所以数据可能会向 0 方向被截断。例如，1.9999 被转换为 1，-1.9999 被转换为 -1。此外，因为 int 的表示范围更小，故可能发生溢出。将大的浮点数转换为整数可能会导致程序错误，这在历史上曾经有过惨痛的教训。

1996 年 6 月 4 日，Ariane 5 火箭初次航行，在发射仅仅 37 秒后，偏离了飞行路线，然后解体爆炸，火箭上载有价值 5 亿美元的通信卫星。调查发现，原因是控制惯性导航系统的计算机向控制引擎喷嘴的计算机发送了一个无效数据。它没有发送飞行控制信息，而是发送了一个异常诊断位模式数据，表明在将一个 64 位浮点数转换为 16 位带符号整数时，产生了溢出异常。溢出的值是火箭的水平速率，这比原来的 Ariane 4 火箭所能达到的速率高出了 5 倍。在设计 Ariane 4 火箭软件时，设计者确认水平速率决不会超出一个 16 位的整数，但在设计Ariane 5 时，他们没有重新检查这部分，而是直接使用了原来的设计。

在不同数据类型之间转换时，往往隐藏着一些不容易被察觉的错误，这种错误有时会带来重大损失，因此，编程时要非常小心。

例 2.25 假定变量 i、f、d 的类型分别是 int、float 和 double，它们可以取除 $+\infty$、$-\infty$ 和 NaN 以外的任意值。请判断下列每个 C 语言关系表达式在 32 位机器上运行时是否永真。

① i == (int)(float) i

② f == (float)(int) f

③ i == (int)(double) i

④ f == (float)(double) f

⑤ d == (float) d

⑥ f == -(-f)

⑦ (d + f) - d == f

解 ① 不是，int 有效位数比 float 多，i 从 int 型转换为 float 型时有效位数可能丢失。

② 不是，float 有小数部分，f 从 float 型转换为 int 型时小数部分可能会丢失。

③ 是，double 比 int 有更大的精度和范围，i 从 int 型转换为 double 型时数值不变。

④ 是，double 比 float 有更大的精度和范围，f 从 float 型转换为 double 型时数值不变。

⑤ 不是，double 比 float 有更大的精度和范围，当 d 从 double 型转换为 float 型时可能丢失有效数字或发生溢出。

⑥ 是，浮点数取负就是简单地将数符取反。

⑦ 不是，例如，当 $d = 1.79 \times 10^{308}$、$f = 1.0$ 时，左边为 0（因为 $d + f$ 时 f 需向 d 对阶，对阶后 f 的尾数有效数位被舍去而变为 0，故 $d + f$ 仍然等于 d，再减去 d 后结果为 0），而右边为 1。 ■

第二章的考点：

1) 会用基数乘除法或降幂法完成十进制与 2（8，16）进制间的转换（注意小数部分可能存在非完全转换问题，如没办法用二进制精确表述十进制 0.32）；
2) 会用直接替换法完成 2 进制与 8，16 进制间的转换（注意首、尾补零位置）；

3) 掌握压缩 BCD 码（主要是 8421 和余三）的数据表示；
4) 掌握原码、反码、补码和移码的表示方式，有符号数的补码表示有什么优点（0 的问题；数据扩展；加法的一次运行不同解释等）；
5) 会使用 IEEE754 浮点表示方式；
6) 可靠性编码中只要求掌握奇偶校验码。