小马加编信息学教案(二十八)

进制转换

一. 课程内容

1. 进制的定义
2. 进制转化

二. 知识讲解

1. 进制的定义

1. 1 十进制

> 实际生活中，我们会遇到很多需要表示数字的问题。

> 比如说，桌子上有很多个苹果，那么我们怎么表示有多少个苹果呢？那么我们可以采取最笨的方法，有多少个苹果画多少条竖线，如果有9个，就画9条，如果有15个，就画15条。但是假如有32个苹果，那就需要画32条竖线来表示，会显得很麻烦，那怎么办呢？
> 那优化一下我们的表示方法，如果累计有10个苹果，我们就用1条横线来代表10条竖线，对于32个苹果我们只需要用3条横线和2条竖线来表示苹果数量。

> 不难发现，这种满十进一的方式就是我们日常生活中常用的表达方式，竖线表示个位，横线表示十位，着就是我们常说的十进制。

> 日常生活中，我们通常使用十进制表示一个数字，即个位表示有多少个1，十位表示有多少个10，百位表示有多少个100，即满十进一的思想

1. 2 其它进制

> 那么跟十进制同理，1条横线可以不表示10条竖线，我们令1条横线表示2条竖线，那么相当于是满二进一，我们把这种计数方式称为二进制。
> 同样若1条横线表示16条竖线，那么就是满十六进一，称作十六进制。

> 我们来观察一下十进制和二进制数的区别。
>
> - 十进制数：11表示了1个10和1个1，总共是11。
> - 二进制数：由于是满二进一的原则，11表示了1个2和1个1，总共是3。
> - 十六进制数：与二进制数同理，11表示了1个16和1个1，总共是17
>
> 可以发现虽然都是11，但在不同进制下表示的数却是不一样的，这是因为每个数位代表的数是不一样的。

> 而有这么多种进制，为什么我们偏偏将十进制定为常用的表达方式呢?
> 可能是因为我们有10根手指吧。

1. 3 表示不同进制的字符

> 我们都知道十进制由于是满十进一，所以每一位我们都用$0 \sim 9$来表示。
> 那么二进制数由于是满二进一，即大于等于2的数字都会往高位进位，不需要表示，所以只需要0和1两个数字。
> 而十六进制数由于是满十六进一，不仅要有$0 \sim 9$，还需要有字符表示$10 \sim 15$，通常我们使用A, B, C, D, E, F分别表示$10, 11, 12, 13, 14, 15$。

> 另外定义**基数**表示一种进制是是满多少进一的，比如说十进制的基数就是十，二进制数的基数就是二。

> 为了区分不同进制的数，我们可以在每个数字的右下角处标上其表示的进制，或者说基数。如十进制的11，我们可写成$(11)_{10}$，二进制的11我们可以写作$(11)_2$
> 另外对与二进制，十进制，十六进制，我们还可以通过在数字后面分别加上大写字母D、B、H来表示，如$10010B$，$96D$，$AF013H$

1. 4 不同进制数表示的数值

> - 十进制
>   十进制数我们都知道，比如42523表示的就是:
> $3 * 1 + 2 * 10 + 5 * 100 + 2 * 1000 + 4 * 10000 = 42523$
>   我们也可以写成
> $3 * 10^0 + 2 * 10^1 + 5 * 10^2 + 2 * 10^3 + 4 * 10^4 = 42523$
>   那么假设从低到高为将每个数存在数组$a[0..bit-1]$中，一个bit位的十进制数就可以表示为:
>
> $$\sum_{i=0}^{bit-1} a_i * 10^i$$

> - 二进制
>   二进制也一样，比如10100表示的就是:
> $0 * 1 + 0 * 2 + 1 * 4 + 0 * 8 + 1 * 16 = 20$
>   我们也可以写成
> $0 * 2^0 + 0 * 2^1 + 1 * 2^2 + 0 * 2^3 + 1 * 2^4 = 20$
>   同理，即
>
> $$\sum_{i=0}^{bit-1} a_i * 2^i$$

> - X进制
>   不难发现，一个X进制的数，从低到高为将每个数存在数组$a[0..bit-1]$中，其表示的数就是

$$\sum_{i=0}^{bit-1} a_i * X^i$$

2. 进制转化

2. 1 X进制转十进制

X进制转成十进制其实我们已经在上面提到过，实际就是求出当前这个X进制数的数值是多少。那么就模拟上面的过程。

假如读入了一个X进制数n，我们要将其转化为十进制数m后输出。
分两个步骤：

1. 那么首先我们要处理出数组a，存下n的每一位是多少。（如果X大于10的话还要特殊处理一下大写字母的问题）

```
bit = 0;
for (; n > 0; bit ++) {
    a[bit] = n % 10;
    n = n / 10;
}
```

2. 然后再通过公式

$$\sum_{i=0}^{bit-1} a_i * X^i$$

将$a[0..bit-1]$转化位十进制数m。

```
t = 1;
for (int i = 0; i < bit; i ++) {
    m = m + a[bit] * t;
    t = t * X;
}
```

这样我们就成功将X进制数n转化成十进制数m。

2. 2 十进制转X进制

假如读入了一个十进制数m，我们要将其转化为X进制数n后输出。

类比将十进制的每一位拆出来的过程：

```
bit = 0;
for (; m > 0; bit ++) {
    a[bit] = m % 10;
    m = m / 10;
}
```

实际上就是每次求出最低位的数，然后再把最低位去掉。比如十进制，最低位的数就是满十进一后剩下的数即`m % 10`，提取完最后一位后，再通过`m = m / 10`把最后一位去掉。
比如234，最低位是4，得到后我们再把234除10，得到23，继续找最低位。

同理，假如将当前数字转成二进制形式，最低位的数就是不足满二进一的数，即`m % 2`，提取完最后一位后，通过`m = m / 2`把最后一位去掉。

> 那么将m拆分成X进制的方法与拆分成10进制的很像。
> 每次对X取模得到最低位，再去通过除X来去掉最低位，直到$m=0$为止。

```
bit = 0;
for (; m > 0; bit ++) {
    a[bit] = m % X;
    m = m / X;
}
```

> 注意，由于该方法是从低位往高位确定的，而输出时要从高位往低位输出，所以要倒着输出。（X大于10时要特殊处理）

```
for (int i = bit - 1; i >= 0; i --)
    printf("%d", a[i]);
```

2. 3 任意进制之间的转化

> 假如我们需要将一个十六进制数转化成九进制数，我们很难直接处理。但是我们知道十六进制转十进制的方法和十进制转九进制的方法，所以我们可以同过十进制在它们之间建立桥梁。

> 即对于任意进制间的转化，如X进制数a转化成Y进制数b，我们都可以分两步进行：
>
> 1. 将X进制数a转化成十进制数m
> 2. 将十进制数m转化成Y进制数b

三. 经典例题

1. n进制转十进制

> 读入一个n进制数，转化位十进制后输出。
> $1<n<10$

> 输入格式：
> 第一行一个整数n
> 第二行一个n进制数x。
> $x \leq 10^6$

> 输出格式：
> 一行一个整数表示x的十进制表示。

样例输入	样例输出
8 17	15

2. 十进制转n进制

> 读入一个十进制数x，转化为n进制之后输出。
> $1<n<17$

> 输入格式：
> 第一行一个整数n
> 第二行一个十进制数x。
> $x \leq 10^7$

> 输出格式:
> 一行一个整数表示x的n进制表示。

样例输入	样例输出
8 15	17

3. 二进制转十六进制

> 读入一个二进制数x，转化为十六进制之后输出。

> 输入格式:
> 第二行一个二进制数x。
> x的长度小于1000

> 输出格式:
> 一行一个整数表示x的十六进制表示。

样例输入	样例输出
10001100	8C

四. 提高巩固

1. 进制转化

> 读入一个n进制数x，转化为m进制之后输出。
> $1 < n, m < 17$

> 输入格式:
> 第一行两个整数n, m
> 第二行一个n进制数x。
> $x \leq 10^6$

> 输出格式:
> 一行一个整数表示x的m进制表示。

样例输入	样例输出
8 9 15	16

2. 汽车牌照

> 小Y最近发现街上的汽车越来越多了，作为汽车的重要标志——汽车牌照也是越来越不够用了，已经从以前的十进制发展到三十六进制了，以前的一个汽车牌照“苏 D88888”，现在的牌照“苏 D0YY11”。
> 小Y突发其想，想知道他看到的大量汽车牌照中最近的两个汽车牌照相差多少?

> 输入格式:
> 第一行一个整数n表示总共有n个车牌。
> 接下来n行，每行为一个汽车牌照，每个汽车牌照为一个7位的字符串，格式为 SD×××××，其中一个 × 表示一个 0~9 或 A~Z，所涉及的字母均为大写。
> $n \leq 100$

> 输出格式:
> 一行一个数，表示最接近的两个汽车牌照之间的差值，要求为十进制数。

样例输入	样例输出
3 SD12345 SD88888 SD99999	1678245

3. 数列

> 给定一个正整数k，把所有k的幂及所有有限个互不相等的k的幂之和构成一个递增的序列。例如，当 $k=3$ 时，这个序列是：1，3，4，9，10，12，13，...
> $1=3^0$
> $3=3^1$
> $4=3^0+3^1$
> $9=3^2$
> $10=3^0+3^2$
> ...
> 请求出这个序列的第n项的值(用十进制数表示)。

> 输入格式:
> 一行两个正整数k和n，之间用一个空格隔开。
> $3 \leq k \leq 15$
> $10 \leq n \leq 1000$

> 输出格式:
> 一个正整数，表示第n项。

样例输入	样例输出
3 100	981