# The Design and Implementation of Conversion Between Decimal Representation and N-scale Notation

# 十进制与任意进制转换的设计与实现

**摘要**：本文从项目应用中实际遇到的问题出发，从理论上解决十进制与任意进制转换或者任意进制之间转化的算法，设计思路清晰简捷并提供了实现算法的程序源码。

**关键词**：任意进制　算法　转换　位权

刘建波（河北医科大学流行病学与统计学教研室　050017）
蔡文水、吕铁山（国家邮政局石家庄培训中心　050021）

## 1　引言

当前常见的数值计算，在编程工具中均已提供，如二进制、八进制、十六进制，它们都很方便地与十进制进行转换和计算。然而笔者在使用Delphi中的Excel自动化对某一单元格赋值为公式的时候遇到了困难，在Excel中公式的列是用从A至IV的大写字母表示的，如何将列号码（数字）转换为相应的字母表达是需要解决的问题。

从表象看，这是一个十进制与26（26个英文大写字母）进制的转换。然而，需要乃创造之母，每一个数学进制的发明都是在人们的日常生活、生产活动中搞出来的，都有它的来龙去脉、自然用处和特定意义。例如，我们将某一事物进行编码，编码要求固定3位字符长度，并且最大限度地包含元素数目，如果使用0-9的十进制字符，它最大包含1000（103）个元素；如果使用10个阿拉伯数字与52个英文字母（包括大小写）构成62进制，包含238328（623）个元素；如果使用10个阿拉伯数字与24个大写英文字母（去掉了O、I，因为容易和数字0、1混淆），这就构成了34进制。

更进一步情况是，在上述的62进制中，3Ar与TfQ的四则运算的结果是多少?

## 2　通用位权展开式

3Ar与TfQ的加减法采用逐位算法尚可实现，但乘除法运算再采用逐位方式其复杂度大得惊人，使程序员显得非常茫然，必须思考一种简便易行的办法，充分利用现成的研究成果，即十进制的各种算法。首先将任意进制表示转化为十进制，利用十进制的计算方法得到结果，再将十进制的结果转化为任意进制。在这一思路指引下，得到了满意的通用算法并引进基数、位权等概念。

### 2.1　任意进制转化为十进制

回顾十六进制的转化方法，将十六进制的2BF转化为十进制的展开过程如下：

$2BF(16)=2\times162+B\times161+F\times160$
$=2\times256+11\times16+15\times1=703$

十六进制用0～9与A～F共16个字符表示数的大小，这16个符号我们称为“数码集”。全部数码的个数称之为“基数”，十六进制的基数是16，计数原则为“逢十六进一”。进位以后的数字，根据所在位置的前后，将代表不同的数值，表示各位有不同的“位权”。按照从低位到高位的次序，第一位代表1（160），第二位代表16（161），第三位代表256（162），以此类推。位权与基数的关系是：位权的数值等于基数的N-1次幂，N代表位次。由此，任何进制都可以写成按照位权展开的多项式之和，它的通用形式如下：

$N=dn\text{-}1\times bn\text{-}1+dn\text{-}2\times bn\text{-}2\cdots\cdots+dn\text{-}3\times bn\text{-}3$ 或者 $N=\sum dk\text{-}1\times bk\text{-}1$

式中：n=数字的总位数；d下标=该位的数码；b=基数（二进制b=2；十进制b=10；十六进制b=16；62进制b=62；……）；b上标=位权；k表示从右向左数码在数字中所在的位数。

**2.2　按照上述的位权通用展开式，很方便的将任意进制转化为人们所熟悉的十进制，其四则运算、乘方、开方、对数等操作在各种开发工具软件中均有现成的函数，这大大减轻了计算的复杂程度。**

**2.3　将任意进制转化为十进制，这是十进制转化为任意进制的逆过程，思路相同。**具体步骤是：将十进制除以基数，所得余数作为对应任意进制低位的值；继续对商除以基数，所得的各次余数就是任意进制的各位值；如此进行直到商等于0为止。我们称这种方法为除基取余法，这里需要注意两个问题：(1)最后一项余数为任意进制数最高位的值；(2)用各位的值在数码集中查找对应的数码（字符）。

## 3 举例

设定34进制的字符集为(0123456789ABCDEFGHIJKLMNPQRSTUVWXYZ)，求MPZ(34)与7HM(34)的差值。

### 3.1 按照通用展开式将34进制表达转化为10进制

MPZ(34)=M×342+P×341+Z×340=21×1156+23×34+33×1=25091(10)

7HM(34)=7×342+H×341+M×340=7×1156+17×34+21×1=8691(10)

### 3.2 执行十进制运算

25091(10)-8691(10)=16400(10)

### 3.3 按照2.3所述原则将十进制的结果转化为34进制，列表如下：

| 除数(基数) | 被除数 | 商 | 余数 | 34进制数码 |
|---|---|---|---|---|
| 34 | 16400 | 482 | 12 | C |
| 34 | 482 | 14 | 6 | 6 |
| 34 | 14 | 0 | 14 | E |

表1 十进制16400转化为34进制过程

表1中“34进制数码”列是由“余数”列在34进制字符集中对应查找而得。MPZ(34)与7HM(34)的差值结果为E6C(34)

## 4 程序实现

### 4.1 将任意进制转化为十进制的Pascal函数

```
function NScaleNotation2Dec(
  const NumericChars :Array of char;//某一个任意进制的数码字符集
  const NumericStr :String        //某一个进制的数字表达字符串
  ) :Int64; //返回值为十进制的signed 64-bit 范围-2^63..2^63-1
var
  iTmp :Int64;
  Divisor, //除数(基数)
  I, J :Integer;
  ANoStr :String;

  function GetPos(const AChar :Char) :Word;
  var B :Boolean;
  begin
    B :=false;
    for result :=0 to Length(NumericChars)-1 do
    begin
      if NumericChars[result] = AChar then
      begin
        B :=true;
        System.Break;
      end;
    end;
    if not B then raise Exception.Create(format(" "%s" 不是有效的字符', [AChar]));
  end;
begin
  Divisor :=Length(NumericChars);//多少进制(基数)
  result :=0;

  ANoStr :=NumericStr;
  if ANoStr[1] = '-' then Delete(ANoStr,1,1);
  J :=Length(ANoStr);
  for I :=1 to J do
  begin
    iTmp :=GetPos(ANoStr[I]) * trunc(Math.Power(Divisor, (J-I)));
    result :=result + iTmp;
  end;
  if NumericStr[1] = '-' then
  begin
    Result :=0-Result;
    if result > 0 then
      raise Exception.Create(format('不是有效的64位整数范围(从%d至%d)', [low(Int64), High(Int64)]));
  end else
  begin
    if result < 0 then
      raise Exception.Create(format('不是有效的64位整数范围(从%d至%d)', [low(Int64), High(Int64)]));
  end;
end;
```

### 4.2 将十进制转化为任意进制的Pascal函数

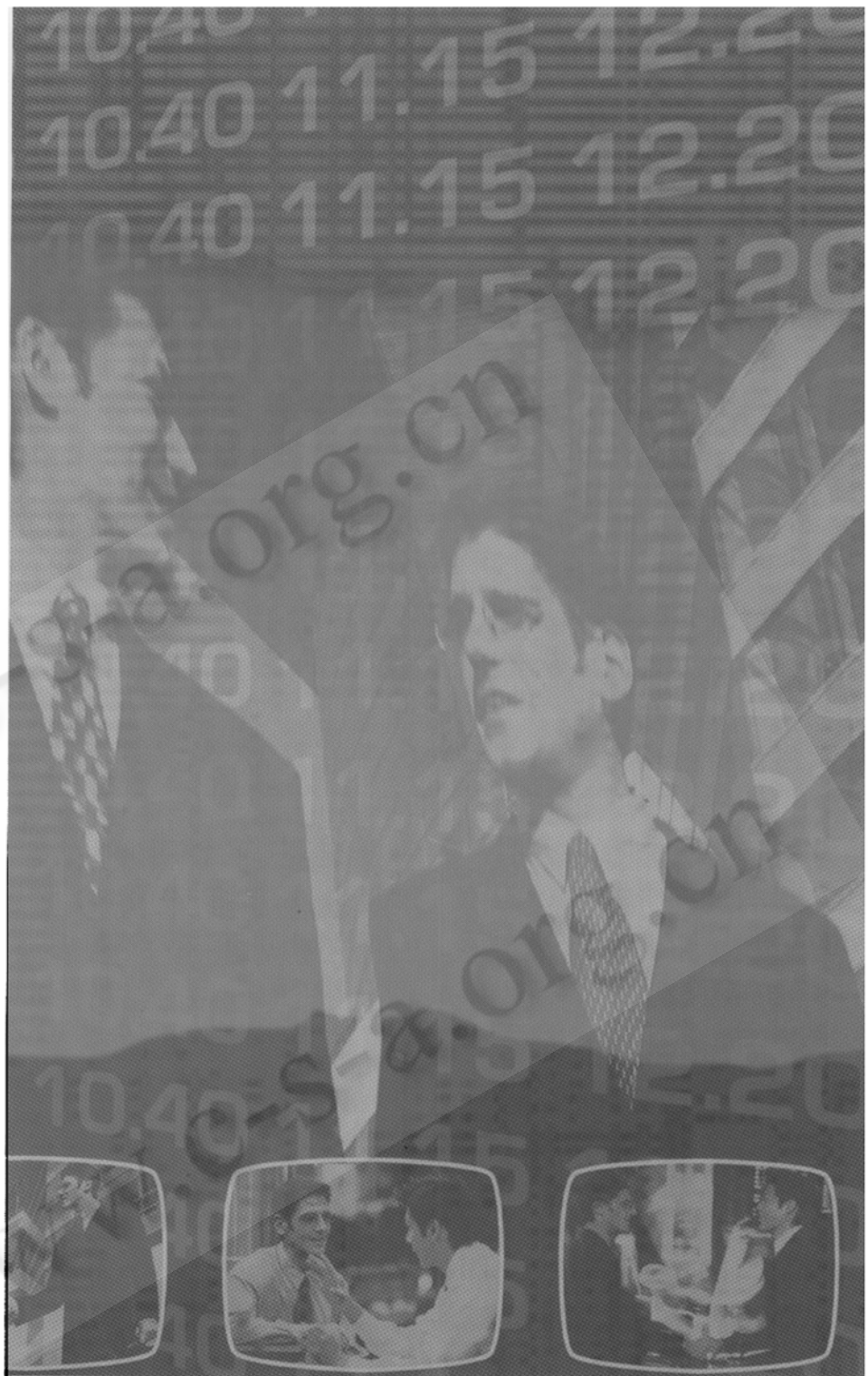
```
function Dec2NScaleNotation(
  const NumericChars :Array of char; //某一个任意进制的数码字符集
  const DecimalInt :Int64//十进制数字
  ) :String; //返回值为任意进制的字符串表达
var Dividend, //被除数
    Quotient :Int64; //商
    Divisor, //除数(基数)
  I, J :Integer;
  ModArray :Array of Int64;//余数
    ModInt :Int64;//当前的余数
begin
  //因Int64取值范围在
  -9223372036854775808 .. 9223372036854775807
  //所以-9223372036854775808转化为正数的时候会出现不正确的结果
  Dividend :=DecimalInt;//因上所述 不能使用abs函数转化为正数
  Divisor :=Length(NumericChars);//进制=基数
  if DecimalInt < 0 then Divisor :=0-Divisor;
```

```
    Quotient :=Dividend div Divisor; //Quotient一定是正数  并且一定小于9223372036854775808
    ModInt :=Dividend mod Divisor;  //ModInt不一定是正数-15 mod -16 = -15 = -15 mod 16
    ModInt :=Abs(ModInt); //转化为正数
    SetLength(ModArray, 1);
    ModArray [0] :=ModInt;
    Dividend :=Quotient;          //在这个时候Divident一定是正数
    Divisor :=Abs(Divisor); //转化为正数
    while Quotient > 0 do
    begin
      Quotient :=Dividend div Divisor;
      ModInt :=Dividend mod Divisor;
      I :=Length(ModArray) + 1;
      SetLength(ModArray, I);
      ModArray [I-1] :=ModInt;
      Dividend :=Quotient;
    end;
    result :='';
    for I :=Length(ModArray) - 1 downto 0 do
      result :=result + NumericChars [ModArray [I] ];
    if DecimalInt < 0 then result := '-' + result;//转化为负数

  end;
```

## 5 结束语

随着社会的进步和计算机的广泛应用，程序员在实际工作中会越来越多地遇到各种各样的进制及其计算。该文采用位权思想成功地处理了十进制与任意进制之间转化的问题，思路清晰简捷，算法实用有效。程序代码提供了负数的处理方法以及适用于较大取值范围，具有通用性和实用性。



## 参考文献


1 张斌荣、曹少彰，数学进制的发明与应用 [J]，革新与发明，1999,(2)，18。

2 李凤 等，计算机应用基础教程 [M]，中国统计出版社，2000.10-16。

3 陈国章、王剑君、蔡兆海，汇编语言百日通，天津科学技术出版社 [M]，1996，12-18。

4 叶向前，超长十六进制整数到十进制整数的转换 [J]，延安大学学报（自然科学版），1997，16(4). 38-41。