· 工程与实践 ·

文章编号：1006－7329(2000)05－0107－04

高有效位数的加、减、乘、除、乘方和阶乘运算法*

(22)
107－110
袁政强
(重庆建筑大学 建筑工程学院，重庆 400045)
TP311.12
O242

摘要：采用面向对象的 C＋＋语言，给出了高有效位数值的类定义，并给出了该数值的加、减、乘、除、乘方和阶乘的运算程序。

关 键 词：面向对象；有效位数；数值运算

中图法分类号：TP311.12；O242 **文献标识码**：A

用计算机作数值运算时，一般的 32 位计算机的计算数值范围是±3.4×(10－38～10＋38)，有效位是 6 位。采用双精度的 64 位数值计算，数值的范围是±1.7×(10－308～10＋308)，有效位数是 16 位。一般计算器的计算精度就更低。而随着计算要求的提高，(如：动力分析中的精细积分算法[1][2])需要对高有效位的数值进行计算。本文给出的计算方法，解决了任意有效位数的数值计算。算例给出了 30 位有效数的计算算例。

用 C＋＋的类将一般的高有效位数数值定义为数的小数部分、符号部分和阶数，小数部分的每一位用字符串的一位表示，即 char f[NF]，NF 为计算的有效位数，在后面的算例中取为 30。如果要求更高的精度时，可加大 NF 的取值。用宏：

#define NF 30 和 #define NF2 60 给出 NF 和 NF2 的值。阶数用整数表示 int Exp，和并定义了相应的多精度数值类和运算方法。

```
class fdouble{
  protected:
  char fsigned; // 小数部分的符号 0 ：＋，1：－
  int Exp;      // 阶部分
  char f[NF]; // 存放小数部分的 NF 位有效数
  public:
  fdouble(char *fp,int ep,char s);
  char operator [] (int ii){return(f[ii]);}
  int E(){return(Exp);}
  char S(){return(fsigned);}
  void ch(){if(fsigned) fsigned=0;else fsigned=1;}
  void setfs(char fs){fsigned=fs;}
  void set(int ii,char cc){f[ii]=cc;}
```

* 收稿日期：2000－01－10
作者简介：袁政强(1962－)，男，贵阳人，讲师，博士生，主要从事计算数学和钢筋混凝土结构分析研究。

```
void printf();}   //类定义结束
```

重载"+"、"-"、"*"、"/"运算符号。

```
fdouble operator + (fdouble a, fdouble b);
fdouble operator - (fdouble a, fdouble b);
fdouble operator * (fdouble a, fdouble b);
fdouble operator / (fdouble a, fdouble b);
```

1 高有效位数的运算

在每一种运算中，用 X_1 和 X_2 表示参与运算的数，其中可用科学计数法表示为

$$X_1 = 10^{b_1} * a_1, \quad X_2 = 10^{b_2} * a_2 \tag{1}$$

$$\frac{1}{10} \leqslant |a_1| < 1, \quad \frac{1}{10} \leqslant |a_2| < 1$$

a_1、a_2 是 X_1 和 X_2 的小数部分，有效位数设为 t 位。

1.1 加法运算

1) 如果 $b_1-b_2>t$，那么因为 X_2 太小，所以它对和的前 t 位有效数字没有任何影响，如：0.234 54+0.000 002，有效位是 5 时，结果是前一数，有

$$fl(X_1 + X_2) \equiv X_1 \tag{2}$$

2) 如果 $b_1-b_2\leqslant t$，那么将 a_2 向右移动 $t_1=(b_1-b_2)$ 位以实现 a_2 除以 10^{t_1}。然后精确的计算和 $a_1+10^{t_1}a_2$，显然此和要求的位数小于 $2t+1$。然后移动位数使小数位达到第一位非零。由于 a_2 数位数的移动，和数的有效位大于 t。在作和运算时，先将和数放入 $2t$ 位的数中。计算完成后至取前 t 位即可。这部分计算程序如下：

```
c=0;      // 是进位数，初始进位为零
for(i=NF2-1;i>=0;i--)   // 从 2t 位的最后一位开始运算
{w=a[i]+b[i]+c;   // 将按位进行相加，加上进位数
if(w>=10) {c=1;w=w-10;}   // 大于等于 10 时，要进位，保留个位
else c=0;      // 小于 10 时，进位为零
temp[i]=w; }
```

1.2 减法运算

1) 如果 $b_1-b_2>t$，那么因为 X_2 太小，所以它对和的前 t 位有效数字没有任何影响，如：0.234 54-0.000 002，有效位是 5 时，结果是前一数，有

$$fl(X_1 - X_2) \equiv X_1 \tag{3}$$

2) 如果 $b_1-b_2\leqslant t$，那么将 a_2 向右移动 $t_1=(b_1-b_2)$ 位以实现 a_2 除以 10^{t_1}。然后精确的计算和 $a_1-10^{t_1}a_2$，显然此和要求的位数小于 $2t+1$。然后移动位数使小数位达到第一位非零。由于 a_2 数位数的移动，和数的有效位大于 t。在作和运算时，先将和数放入 $2t$ 位的数中。计算完成后至取前 t 位即可。这部分计算程序如下：

```
for(i=NF2-1;i>=0;i--) // 对 2t 位数进行相减
 {
  if(a[i]>=b[i]) {temp[i]=a[i]-b[i];} // 第 i 位的 a 大于 b，直接相减
  else
  {temp[i]=10+a[i]-b[i]; // 第 i 位的 a 小于 b，借上位的 1 当本位的 10，相减
   for(j=i-1;j>=0;j--)
   if(a[j]==0) a[j]=9; // 如果上一位是零时，要向再上位相借，借来的 10 被下位借一后变
```

```
    为 9
    else {a[j]=a[j]-1; break;} // 上一位非零时，减一
  }
}
```

1.3 乘法运算

指数 b_1 和 b_2 相加得 b_3，计算 a_1 与 a_2 的精度是 $2t$ 位的积。这是积满足

$$\frac{1}{100} \leqslant |a_1 \cdot a_2| < 1 \tag{4}$$

在积小于 1/10 时，b_3 加 1，积向右移动一位。乘积运算转化为加法运算，将 a_2 写为如下形式：

$$a_2 = a_{21}10^{-1} + a_{22}10^{-2} + \cdots + a_{2t}10^{-t} \tag{5}$$

a_{2l}是 0 1 2 3 4 5 6 7 8 9 中的一数字(其中 $l=1,2,\ldots t$)。a_1 与 a_2 的积可以用下式表示：

$$a_1a_2 = a_1a_{21}10^{-1} + a_1a_{22}10^{-2} + \cdots + a_1a_{2t}10^{-t} \tag{6}$$

每一项是将 a_1 向左移动 l 位后，自加 a_{2l}次(其中 $l=1,2,\ldots t$)。然后求和。把乘法运算变为了加法运算。

1.4 除法运算

指数 b_1 减 b_2 去得 b_3，将 a_1 转化为精度是 $2t$ 位的数，即在后 t 位放零。将除法转化为减法运算，方法是：

1) 如果 $|a_1| > |a_2|$，那么 b_3 要加 1。这时用 a_1 减去 a_2，一直减到 $|a_1| < |a_2|$。将减的次数记为 a_{31}；

2) 如果 $|a_1| < |a_2|$，将 a_2 缩小 10 倍；如果 $|a_1| > |a_2|$，采用方法 1)求 a_3 的下一位数；

3) 如果经过 2)后，还是有 $|a_1| < |a_2|$，则 a_3 的这一位为零；将 a_2 缩小 10 倍后再判断；

4) 如果 a_3 的有校位达到 t 后，结束运算。

1.5 任意数的整数次方运算

次方数可以是正负整数和零。

1) 次方数为零时，定义结果值为 1.0；

2) 次方大于零，采用连乘；

3) 次方小于零，采用连除。

1.6 任意数的整数阶乘运算

X 是大于 1 的任意实数，X 的阶乘为：

$$X(X-1)(X-2)(X-3)\cdots(X-n) \tag{7}$$

其中 $0 \leqslant (X-n) < 1$

2 算例

$a=0.823\ 564\ 330\ 000\ 000\ 000\ 000\ 000\ 000\ 000 \times 10^4$；

$b=-0.123\ 374\ 430\ 000\ 000\ 000\ 000\ 000\ 000\ 000 \times 10^4$；

$a+b=0.700\ 190\ 100\ 000\ 000\ 000\ 000\ 000\ 000\ 000 \times 10^4$；

$a-b=0.946\ 938\ 560\ 000\ 000\ 000\ 000\ 000\ 000\ 000 \times 10^4$；

$a*b=-0.101\ 606\ 615\ 069\ 215\ 900\ 000\ 000\ 000\ 000 \times 108$；

$a/b=-0.667\ 533\ 511\ 657\ 985\ 626\ 333\ 797\ 584\ 795 \times 10^1$；

$a^{40}=0.424\ 522\ 909\ 690\ 592\ 911\ 064\ 964\ 866\ 407 \times 10^{157}$；

$a^{-40}=0.235\ 558\ 547\ 530\ 175\ 661\ 108\ 788\ 596\ 061 \times 10^{-157}$；

$a!\ =0.638\ 575\ 996\ 254\ 692\ 299\ 450\ 850\ 002\ 577 \times 10^{28\ 674}$。

3 小结

1）小数部分的位用的是一个字符表示，一个字符是8个二进制位。4个二进制位就能表示0～9的数，可以采用联合体结构将一个字符表示二个十进制的位。这样可以节省一半的存储空间。为了书写清楚，给出的是现在的存储法。

2）任意数的整数次方可以计算后，采用Taylor级数展开，可以计算任意的初等函数和特殊函数。下面就 ab 的计算加以说明。令 $Y=a^b$，$\ln(Y)=b\times\ln(a)$，$\ln(a)$的Taylor展开为[4]：

$$\ln x = 2\left[a + \frac{1}{3}a^3 + \cdots + \frac{1}{2n+1} + \cdots\right]，其中 a = \frac{x-1}{x+1}, x > 0 \tag{8}$$

在计算好 $b\ln(a)$后，最后结果 $Y=\exp(b\ln(a))$，而 $\exp(x)$的Taylor展开式为(4)：

$$\exp(x) = \frac{x^0}{0!} + \frac{x^1}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots + \frac{x^n}{n!} + \cdots = \sum_{i=0}^{\infty}\frac{x^i}{i!} \quad x > 0 \tag{9}$$

另外，需要以上软件源程序的读者请与作者联系，E _ mail：zqyuan@public. cta. cq. cn

参考文献：

[1] 钟万勰.结构动力方程的精细时程积分法[J].大连理工大学学报，1994，34(2)：131～136

[2] 钟万勰.暂态历程的精细计算方法[J].计算结构力学及其应用，1995，12(1)：1～6

[3] [英]J. H. Wilkinson 代数过程的舍入误差[M].北京：人民教育出版社，1982

[4] 《数学手册》编写组.数学手册[M].北京：人民教育出版社，1979

Addition, Subtraction, Multiplication, Division, Involution and Factorial of High Effective Locations

YUAN Zheng-qiang

(Faculty of Civil Engineering, Chongqing Jianzhu University, Chongqing 400045, China)

Abstract: This paper presents the class of high effective locations and its addition, subtraction, multiplication, division, involution and factorial by the object—oriented programming.

Key words: object—oriented; effective location; digit operation