C++超长浮点类的设计及π的精确计算

汤　韬，瞿　洋
(空军工程大学 工程学院，陕西 西安 710038)

摘　要：文中利用C++面向对象的编程技术，设计并实现了不受计算机位数和软件数据类型限制的，可以完成任意位数的浮点数存储、运算和输入输出的超长浮点类。并据此提出了基于级数的、对圆周率π和自然对数的底e精确到小数点后任意位的计算方法。

关键词：面向对象编程(OOP)；运算符重载；级数
中图分类号：TP301.6　　**文献标识码**：A　　**文章编号**：1009-3516(2001)05-0039-03

一般情况下，计算机对浮点数的处理精度是有限的，为了实现更高精度的计算，通常需要更长字宽的巨型机来完成。而在近期的有关参考文献中，还没有发现由软件实现的、可以处理任意位浮点数的实用、有效的算法。因此，我们利用面向对象的编程方法[1~2]，在普通计算机上方便地定义并实现了用户自定义数据类型——超长浮点类，使之在不考虑计算机存储容量和运算时间限制时可以实现任意位数的浮点数据的存储、计算、输入输出等功能。

在C++中[3~5]，不仅可以方便地定义用户需要的数据类型并通过运算符重载等方法实现其运算，而且可以充分利用面向对象程序设计方法的诸多优点开发出高效、实用的应用程序[6]。因此，我们以VC++为开发工具，定义了一个超长浮点类(ChugeFloat)，由其构造函数将超长浮点数的整数和小数部分分别按位存放在两个数组中，通过重载类操作运算符实现该类数据的运算，从而实现任意位数的浮点数的存储、计算、输入/输出等算法。为了说明该数据类实现的正确性及其应用，我们采用级数有限项求和法[7]及误差分析[8]等理论实现了圆周率π、自然对数的底e的值可以精确到任意位数的算法。

1　类ChugeFloat的实现

1.1　比较运算符(>)的重载

比较运算符(>)的重载时，对于异号数直接由符号位可以判定，而同号数从最高位开始按位判定，如判定f1>f2?，只要一出现f1的某一位不等于f2的对应位时，即可得出结论. 如果该位f1的值大于f2的值且f1、f2皆为正数或该位f1的值小于f2的值且f1、f2皆为负数，即可知f1>f2，否则f1<=f2。

1.2　加法运算符(+)的重载实现

加法运算符是最基本的运算符之一，其重载算法的基本思想为：

设$D_1=h_0h_y\cdots h_1.fh_1\cdots fh_x$，$D_2=d_0d_k\cdots d_1.fd_1\cdots fd_j$为加数和被加数，$D_s=s_0s_m\cdots s_1.f_1\cdots f_n$为$D_1$，$D_2$相加的和($x,y,j,k,m,n\in N$)。其中，$h_i$，$d_i$，$s_i$为整数位；$fh_i$，$fd_i$，$f_i$为小数位($i\in N$)。则$D_1$，$D_2$相加可描述为：

1)若h_0与d_0同号，则$s_0=h_0$(或$=d_0$)。不考虑符号位，然后从D_1、D_2最低位开始(即，若$x>j$，则最低位为x，反之为j)，按对应位带进位标志逐位相加(直到整数最高位)，并在每次相加并进行十进制调整(即如果该位结果大于9，则减9，并将进位标志置位)后赋给D_s对应位。结果需要判定是否溢出(定义的缓冲区最高位是否大于9)；

2)若h_0与d_0异号，对D_1，D_2取绝对值，如果$abs(D_1)>abs(D_2)$，则$s_0=h_0$；否则$s_0=d_0$。同样，在不考

收稿日期：2001-03-30
作者简介：汤　韬(1971-)，男，重庆人，硕士生，主要从事计算机应用研究.

虑符号位的情况下从 D_1、D_2 最低位开始,按对应位带进位标志逐位相减(直到整数最高位),并在每次相减并进行十进制调整后赋给 D_1 对应位。显然结果不会溢出。

1.3 乘法运算符(*)的重载

乘法运算符的重载比较复杂。将乘数和被乘数的整数部分和小数部分各自分开相乘,再求四个乘积项之和。而四个乘积项在形式上完全相同,只是积的小数点位置有变化,因此定义友元函数 friend CHugeFloat Multiply(CHugeFloat &f1,CHugeFloat &f2,int type)来计算乘积项。其中 f1、f2 分别对应乘数和被乘数,由参数 type 取值 0,1,2,3 来对应四个乘积项。然后通过四次调用该函数(只变 type 参数)并求其和就可以方便地实现 * 运算。在此主要介绍 Multiply(CHugeFloat &f1,CHugeFloat &f2,int type)函数的实现:

该函数实现思想与手工乘法相似,即乘数与被乘数均按位相乘,通过移位调整和求和运算来实现。只是此处乘法并非传统的从最低位开始,而是从最高位开始,每计算一次的结果以调用函数 Multiply10()来调整数位。其实现代码略。

需要说明的是,为减少前(后)导零计算所占用的时间,同时便于调整计算结果的数据位,需要确定缓冲区前后非零位位置。显然,只有被乘数和乘数的整数部分相乘才可能产生溢出,而其小数部分相乘才可能产生精度损失。

2 π、e 的高精度算法

π 的计算方法相当多,从古至今有不少科学家提出过许多算法,计算方式也从手工计算发展到计算机编程计算。在此我们借助前面已经实现的类和级数有限项求和法来计算。考虑到不同的求和公式对运算速度、运算精度等的影响,因此必须进行必要的误差分析和速度估算。

设定精确到小数点后 100 位的 π 值。如果采用最常见的求和公式 $\pi=\sum_{n=0}^{\infty}\left((-1)^n\frac{4}{2n+1}\right)$,则要使截断误差不大于 10^{-100},即 $\frac{4}{2n+1}\leqslant 10^{-100}$,必须 $n\geqslant 2\times 10^{100}$。显然这一算法是不现实的。因此,我们选用收敛速度快的马信(J. Machin)公式计算:

$$\pi=\sum_{n=0}^{\infty}\left\{(-1)^n\frac{16}{(2n+1)\cdot 5^{(2n+1)}}-(-1)^n\frac{4}{(2n+1)\cdot 239^{(2n+1)}}\right\} \tag{1}$$

由于该公式第一部分的收敛速度远小于第二部分,因此将其分为两部分计算。如果要使截断误差不大于 10^{-100},则有 $\frac{16}{(2n+1)\cdot 5^{(2n+1)}}\leqslant 10^{-100}$,即可以估算 $n>70$ 时就可以满足要求,而且各项的计算误差引起的精度损失之和最多只影响到小数点后末两位(因为每一项只有作除法运算时才有一次精度损失)。对于第二部分,同理可以分析得出 $n>20$ 即可。

为了判定 π 的计算机结果的精确性,我们另选择级数来计算 π 值,以便于对照检测。

$$\pi=\sum_{n=0}^{\infty}\left\{(-1)^n\frac{4}{(2n+1)\cdot 2^{(2n+1)}}+(-1)^n\frac{4}{(2n+1)\cdot 3^{(2n+1)}}\right\} \tag{2}$$

通过上述两种方法得出的计算结果见图 1。其中 $n=70,21$ 分别为式(1)满足精度要求时第一、第二部

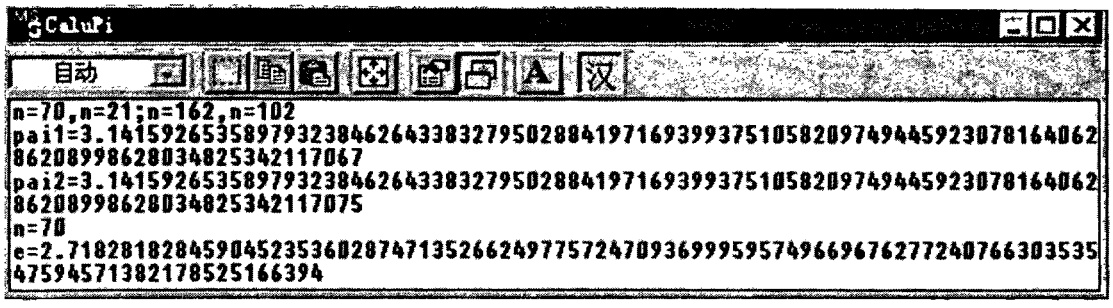

图 1 π、e 的输出结果

分所取项数,n=162,102 为式(2)满足精度要求时第一、第二部分所取项数。可以看出,两种算法只有最后

两位有差异,因此,计算结果满足分析要求,这也说明了类的定义和实现是正确的。而完成式(1)算法的运算在PⅡ300上大约费时15 s左右,式(2)由于级数收敛速度较慢而费时1min(由此也可看出在相同精度要求下级数收敛速度对运算速度的影响)。

同样,可以采用级数 $e=\sum_{n=0}^{\infty}\left\{\frac{1}{n!}\right\}$ 计算出e的值(见图1)

3　结论

1)面向对象编程的方法是一种优越的编程方法,本文充分展现出C++类型扩展能力;

2)本文实现的可表示任意精度的超长浮点类,其算术运算与系统已有数据类型使用方法无太大差别,可以作为一种通用的数据类型使用;

3)本文提供了一种精确计算如圆周率π、自然对数的底e等特殊值的算法思想,可以据此实现诸如$\sqrt{x}$、$\sin x$、$\log_a^b$ 等等形式的精确求值。

由于本文重点在于介绍面向对象的类扩展及π、e精确求值的算法和思路,因此在代码介绍中省略了许多具体细节和一些诸如溢出、错误告警等的处理。同时也未采用快速算法进行优化以提高运算速度。

参考文献:

[1] 王斌君,卢安国. 面向对象的方法学与C++语言[M]. 西安:陕西师范大学出版社,1995.
[2] Bing Swen . Extended Object - orientation[R]. Bejing:Peking University,1999.
[3] Leinecker R C,Archer Tom . Visual C++60Bible[M]. 北京:电子工业出版社,1999.
[4] 吕凤翥. C++语言基础教程[M]. 北京:清华大学出版社,1999.
[5] ISO/IEC:98 4882. Programming Languages - C++[S].
[6] Bing Swen. Object - oriented programming with induction[J]. ACM SIGPLAN Notices,2000,35(2):61 - 67.
[7] 吉米多维奇著 B Π. 数学分析习题集题解[M]. 济南:山东科学技术出版社,1983.
[8] 邓建中,葛仁杰,程正兴. 计算方法[M]. 西安:西安交通大学出版社,2000.

The Design of a C++ Over - long Floating - point Data Type and Accurate Calculation of π

TANG Tao,　QU Yang
(The Engineering Institute,Air force Engineering University,Xi'an 710038,China)

Abstract: An over - long floating - point data type based on the C++ Object Oriented Programming(OOP) is designed and implemented in this paper. The data type is beyond the limitation of software data type and computer digit length and can accomplish storage and arithmetic operation and input/output among the over - long floating/point numbers given at discration. On the basis of the data type,we advance a series - based method by which we can calculate values of circumference ratio π and the natural logarithm's fundus **e** up to any digit after the radix point.

Key words: Object Oriented Programming (OOP) ;Operator - over - loading;Series