第37卷 第16期 计 算 机 工 程 2011年8月
Vol.37 No.16 Computer Engineering August 2011

·软件技术与数据库· 文章编号：1000—3428(2011)16—0075—02 文献标识码：A 中图分类号：TP301.5

C++中超长整数的存储与表示

杨维剑，王梅英

(四川理工学院计算机学院，四川 自贡 643000)

摘 要：提出在C++中利用32位汇编语言直接对内存操作的方法，实现对任意超长整数在计算机中的存储，给出任意超长整数的输入和输出的表示方法，以及主要程序算法框图和程序流程图。对其时间复杂度及所需空间复杂度进行分析，为直接在C++中调用提供便利条件，为实现在计算机中用超长整数运算代替浮点运算提供技术支持。

关键词：超长整数；输入；输出；时间复杂度；空间复杂度

Memory and Indication of Super Long Integer in C++

YANG Wei-jian, WANG Mei-ying

(Institute of Computer, Sichuan University of Science & Engineering, Zigong 643000, China)

【Abstract】The paper discusses the use of 32 MASM languages in C++ to directly operate to the memory of method, realizes the saving method of super long integral's garbage arbitrarily, gives the input and output meaning method of super long integral, and the main procedure calculate way frame diagram and procedure flow chart of these methods, analyzes the time complexity and the space complexity needed, for directly in the C++ in adjusted to provide a convenient condition. It makes use of a super long integral operation to replace floating point arithmetic to lay certain foundation.

【Key words】super long integer; input; output; time complexity; space complexity

DOI: 10.3969/j.issn.1000-3428.2011.16.025

1 概述

随着计算机技术应用深入，人们对计算机的精确处理能力要求越来越高。尤其是随着航天技术、卫星技术的发展，早期利用浮点数运算提供15位~16位数据精度已经不能满足实际应用的需要。

浮点数计算具有复杂性并且存在一定的累计误差。文献[1]提出超长整数的存储与表示大多是利用C/C++中的数组来存放的。数组存放的超长整数会给其存储与运算增加算法的复杂度，以及输入输出带来极大不便。研究任意超长整数的存储与表示为在计算机中整数运算代替浮点运算有着重大的应用意义。为此，本文提出任意超长整数的输入和输出的表示方法，以及主要程序算法框图和程序流程。

2 C++中任意超长整数的存储方法和表示

在C++中，常规整数的类型及表示可参阅文献[2]。在C++中提供了以上数据的输入和输出方法，这里不再累赘。所谓任意超长整数是指整数范围大于2^{32}的整数。

超长整数有2种类型：

(1)无符号整数；

(2)有符号整数。

与C++一样，对于有符号任意超长整数，最左边第1位二进制位作为符号位，其余各位作为有效数据位。对于无符号任意超长整数，所有位均作为有效数据位，在内存中的表示方法如图1所示[3]。

为了提供算法的有效性和充分利用32位计算机的特点，对任意超长整数的存储作了一些规定：即存储任意超长整数的存储空间以4 Byte为单位，并要求以规格化的存储格式加以存储，以提高访问存储器速度。

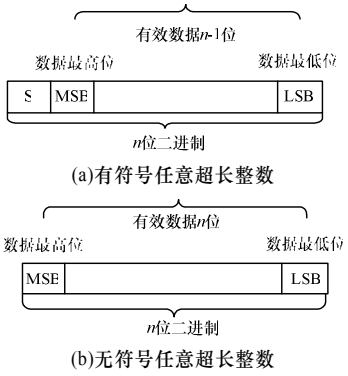

图1 任意超长整数存储格式

3 C++中任意超长整数的输入与输出

3.1 任意超长整数的输入

在C++中，对于任意超长整数的输入，利用现行C++提供的办法是无法实现的，因此，本文利用32汇编语言和API函数的方式加以解决。其算法框图如图2所示。在图2中主要难点在于：

(1)如何将输入的任意位十进制数(或0~9的ASCII)转换成二进制数存储在指定的存储器中。

(2)如何解决动态存储器分配、管理问题。

(3)如何输入。

基金项目：国家“863”计划基金资助项目“CNG储气井安全检测技术与检测设备的研发”(2008AA11A134)

作者简介：杨维剑(1968－)，男，副教授，主研方向：计算方法，数字计算，数字图像传感技术；王梅英，高级工程师

收稿日期：2011-01-31 **E-mail：**boy9boy@163.com

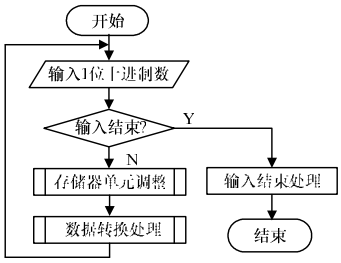

图 2 超长整数输入框图

对第(1)个问题，利用如图 3 所示的算法加以解决。

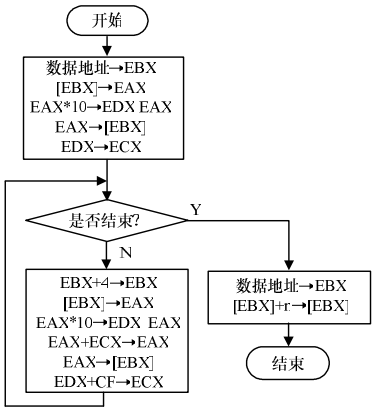

图 3 数据转换处理流程

在图 3 中，“→”表示传送；k 表示输入的十进制数(0~9)；[EBX]表示寄存器间接寻址[3]。其运算原理如下：

设存储单元个数为 n，该超长整数可以表示为：$d_1,d_2,\mathrm{L}\ ,d_n$，其中，$d_i(1\leqslant i\leqslant n)$ 为 32 位二进制数。当增加 1 位十进数 k 后，原数变为：

$$((((d_1\times 2^{32}+d_2)\times 2^{32}+d_3)\times 2^{32}+\mathrm{L}\ +d_{n-1})\times 2^{32}+d_n)\times 10+k$$

对于第(2)个问题，设存储超长整数的二进制位数 n，表示对应的十进制位数为 m，有公式：

$$2^n=10^m \tag{1}$$

成立。从而有：

$$n=m\mathrm{lb}10\approx 3.321\leqslant 3.33m \tag{2}$$

也就是说，如果任意超长整数位数为 m，则只需要 $\lceil m\mathrm{lb}10\rceil$ 二进制位就可以存放。符号“$\lceil\ \rceil$”为向上取整(下同)。在实际的算法中，判断是否需再增加一个存储单元(4 Byte)，并且转化为整数计算，将式(2)变为如下表达式：

$$(333\times m-3\,200\times n)\geqslant 3\,200 \tag{3}$$

其中，m 为输入的十进制位数；n 为存储单元个数。

在程序中判别表达式式(3)是否成立即可。若成立，则需要增加一个存储单元，否则不需要增加。

对于第(3)个问题，在 32 位汇编语言中直接调用 API 函数_scanf()加以解决即可[3]。

3.2 任意超长整数的输出

在 C++中，对于存储在存储器中任意超长整数的输出，利用现行 C++提供的办法也是无法实现的。因此，本文提出仍然利用 32 位汇编语言和 API 函数的方式加以解决。其算法框图如图 4 所示。在图 4 中，主要难点在于：

(1)根据超长整数的二进制位数如何计算存储转变后的十进制字节数。

(2)二进制超长整数如何转换成十进制对应的 ASCII。

(3)实现输出。

对于图 4 中的第(1)个问题，可以借助于式(1)的计算，做一个简单的变换可以得到：

$$m=n\lg 2\approx 0.301n\leqslant 0.31n \tag{4}$$

为了避免浮点数计算，在实际程序中，用如下公式计算：

$$m=\lceil(31\times n)/100\rceil \tag{5}$$

在式(5)中，只需利用 $31\times n$ 整除 100 加 1 即可。

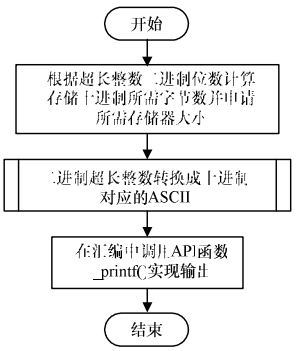

图 4 超长整数输出框图

对于图 4 中的第(2)个问题，利用图 5 所示的算法加以解决。在图 5 中，利用超长整数除法进行除 10 取余法，分离各个十进制数。利用此法分离的第 1 位十进制数是个位，第 2 是十位，……，最后分离的是最高位。这也是本输出方法中要首先动态分配一定的存储器用以存放十进制位数的原因。否则，输出会与人们的习惯不一致。

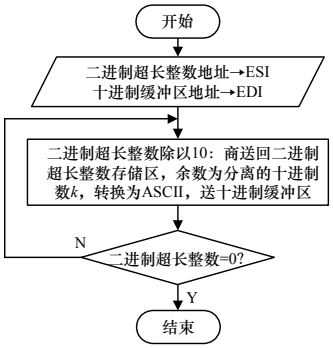

图 5 超长整数转换为十进制 ASCII 流程

对于图 4 中的第(3)个问题，在图 4 中已经说明。

4 C++中任意超长整数输入/输出复杂度分析

4.1 任意超长整数输入算法复杂度分析

在输入算法中，其空间复杂度主要是由输入的十进制位数决定的，即当输入十进制位数为 n 时，所需存储器空间复杂度为：

$$S(n)=\lceil n\mathrm{lb}10/8\rceil+c=O(n)$$

其中，c 为输入过程中的缓冲区(如键盘缓冲区等)，以字节为单位。

在输入算法中，其时间复杂度主要是由一个二重循环构成。外循环由输入的十进制位数 n 决定，内循环则随着 n 的增加而增加循环次数，其时间的平均复杂度为：

$$T(n)=n\left(\left\lceil\left\lceil n\mathrm{lb}8\right\rceil/8\right\rceil/4\right)+c=O(n^2)$$

其中，c 为循环体前后处理的时间。

4.2 任意超长整数输出算法复杂度分析

在输出算法中，其空间复杂度是由储存超长整数的二进制位数决定的。主要由以下 2 个部分构成[4]：

(1)按照十进制 ASCII 存放超长整数位数。

(2)利用超长整数计算除以 10 取余法中所需的缓冲区数。

设超长整数的二进制位数为 n，则其空间复杂度为：

$$S(n)=\lceil n\lg 2\rceil+\lceil 2n/8+4\rceil+c=O(n)$$

(下转第 89 页)