建立存储解决大整数的存储和四则运算问题

李馨茹
曲阜师范大学　山东济宁　273165

摘　要：正常的加减乘除运算都在 long 范围内，最大的 longlong 型变量只能存储 -9223372036854775808 ~ 9223372036854775807 的整数，所以如果给一个 1000 位的整数运算就无法完成其运算了，其超出了程序设计语言整数类型的值集范围，所以设计良好的数据结构与合适的基成为大整数精确运算系统的重要基础。本文介绍了一种这样的超大整数在程序设计语言中的存储和表示方法，并对这种方法表示的超大整数的基本四则运算进行了分析，给出了实现算法。

关键词：大整数；存储；表示；四则运算；算法

The establishment of storage to solve the large integer storage and four arithmetic problem

Xinru Li
Qufu Normal University, Jining, Shandong 273165, China

Abstract: Normal addition, subtraction, multiplication, and division are in the long range. The largest longlong variable can only store integers from -9223372036854775808 to 9223372036854775807. The operation cannot be done if it is given to a 1000-bit integer, which is outside the value set of the programming language integer type. Therefore, well-designed data structure and appropriate basis become the important basis of large integer accurate operation system. This paper introduces a kind of storage and representation method of such super integer in the programming language, analyzes the basic four operations of the super integer represented by this method, and gives the realization algorithm.

Key words: large integer; Storage; express; Four operations; algorithm

一、大整数的存储与表示

1. 1 利用字符串输入大整数

字符串在存储上类似字符数组，它每一位单个元素都是能提取的，这提供给我们很多方便，例如高精度运算时每一位都能转化为数字存入数组。将大整数作为字符串输入程序时，一个数字为一个字符，占一个字节。

1. 2 利用数组存储大整数

将字符串中每一个字符依次转化为数字存入数组中，数组第一位为输入的大数字的长度，后面倒序存储，这样的存储方式便于后续的四则运算中产生的进位的存储。

转化的算法为：

```
void change_str(char *str, int A[])
{  int i,len;
   len=strlen(str);
   A[0]=len;
   char*p=str+len-1;
   for(i=1;i<=A[0];p--,i++)
   A[i]=*p-’0’;
}
```

二、大整数的四则运算

2. 1 加法

从两个数组的第二位开始对每一位对应进行加法运算，并利用加完后的数对 10 整除的结果来判断加完后的数是否大于 9，大于 9 则需要进位，即将后一位中储存的数加 1，而本位储存加完后的整数对 10 取余的结果，这便进行完了本位的加法运算，下一位中因为已经存储了前一位进行加法运算后的进位结果，所以进行这次的加法运算时依旧利用加完后的数对 10 整除的结果来判断加完后的数是否大于 9，大于 9 则进位，但本位储存的变为原本本位所存储的上一位加法运算后的进位加上本次加法运算后的整数对 10 取余的结果。后面的数依次按照这种方式进行。直到其中一个数组遍历完，则将那个没有遍历完的数组中的剩余位数对应存于存加法结果的数组中的剩余位数。

算法为：

```
void add_AB(int *A,int *B)                    if(i<=A[0])
{int i,j,len;                                 {
len=A[0]>B[0]?A[0]:B[0];
for(;i<=A[0];i++)
int C[len+2];                                 C[i]=C[i]+B[i];
for(i=0;i<=len+1;i++)                         }
```

ISSN:2661-3719(Print);2661-3727(Online)

```
C[i]=0;
for(i=1;i<=A[0]&&i<=B[0];i++)                    if(C[i]==0)
{                                                C[0]=i-1;
C[i]=C[i]+(A[i]+B[i])%10;                        else
C[i+1]=(A[i]+B[i])/10;                           C[0]=i;
}                                                }
if(i<=A[0])
{
for(;i<=A[0];i++)
C[i]=C[i]+A[i];
}
```

2.2 减法

存储好两个大整数 a 和 b，先判断两个整数的大小，使得实际上的操作永远为大数减小数，输出时则按实际大小输出结果（负数输出时先输出负号）。从数组第二位开始对应依次相减，对每一位判断大小关系，若小于往后借 1 位，然后本位加 10，后一位减 1。依次进行直到遍历完整个数组。

算法为：

```
void subtract_AB(char* str1,char* str2)
if(A[i]<B[i])
{  int len1,len2,i,k,len;                        {
   len1=strlen(str1);                            A[i]=A[i]+10;
len2=strlen(str2);
A[i+1]=A[i+1]-1;
   len=len1>=len2?len1:len2;                     }
   int A[len+1];                                 C[i]=A[i]-B[i];
   for(i=0;i<=len;i++)                           }
   A[i]=0;                                       if(i<=A[0])
   int B[len+1];                                 {for(;i<=A[0];i++)
   for(i=0;i<=len;i++)                           C[i]=A[i];}
   B[i]=0;                                       for(k=i-1;k>1;)
    if(len1<len2||((strcmp(str1,str2)<0)&&(len1==len2)))
{
   {                                             if(C[k]==0)
     change_str(str2,A);                         k--;
     change_str(str1,B);                         else
   }                                             break;}
   Else                                          C[0]=k;
   {                                             }
     change_str(str1,A);
     change_str(str2, B);
   }
   int C[A[0]+1];
   for(i=1;i<=B[0];i++)
   {
```

2.3 乘法

倒序存储好两个大整数，计算可能的结果的长度，假设乘数 a 的长度为 x，乘数 b 的长度为 y，结果长度最多为 x+y，最小为 x+y-1，因此要一个长度为 x+y 的数组，然后看实际进位情况来判断元素个数。利用两个 for 循环，第一个 for 循环从放置乘数的数组的第 2 位，依次对放置被乘数的数组的第 2 位开始的每一位进行乘法运算，利用一个新数组对结果进行存储，每次从数组的对应于乘数数组进行运算的位置开始存储，本位存前一位乘积运算的进阶与本次乘积运算对 10 取余得到的结果的加和，后面依次进行，最后用一个新数组对放乘积结果的数组进行数据处理，新数组从第 2 位开始进行存储，从放置乘积结果的数组的第 2 位开始，本位存前一位的进阶与放置乘积结果的数组的对应位置的数对 10 取余得到的结果的加和，后面依次进行处理，最终得到结果。

算法：

```
void multiply_AB(int a[],int b[])                j++;
{  int i,k,t;                                    }
   int j;                                        for(t=1;t<=k;t++)
   int C[a[0]+b[0]+1];                           {D[t]=D[t]+C[t]%10;
   int D[a[0]+b[0]+1];                           D[t+1]=C[t]/10;}
   for(k=0;k<=(a[0]+b[0]);k++)                   if(D[k]==0)
   C[k]=0;                                       D[0]=k-1;
   for(k=0;k<=(a[0]+b[0]);k++)                   else
   D[k]=0;                                       D[0]=k;
    for(i=1;i<=b[0];i++)// 乘数                  printf(“运算完结果为：\n”);
      for(k=i,j=1;j<=a[0];k++)// 被乘数
for(i=D[0];i>=1;i--)
    {                                            printf(“%d”,D[i]);
     C[k]=C[k]+(b[i]*a[j])%10;                   printf(“\n”);
     C[k+1]=C[k+1]+(b[i]*a[j])/10;               }
     j++;
```

2.4 除法

大整数的除法相对就比较复杂了，又分为大整数 / 小整数 , 大整数 / 大整数两种情况。实际上就是在模拟小学时学习除法的过程，待除数记录当前数字，一直在变化，然后商不断循环从 1 到 9，计算乘积和差值，直到计算完整个数，最后待除数的值就是余数，商即为计算的结果。

2.4.1 大整数 / 小整数

上一步的余数乘 10 加上该步的位，得到该步临时的被除数，将其与除数进行比较：如果不够除，则该位的商为 0; 如果够除，则商即为对应的商，余数即为对应的余数，最后一步要注意高位可能有多余的 0，要忽视它们但也要保证结果至少有一位数 。

2.4.2 大整数 / 大整数

大整数 / 大整数的情况，因为最终结果的商和余数都可能是大整数，所以必须转换为乘法和减法运算，最后不断循环得到结果。

三、总结

对计算机而言，无穷大的数字范围需要进行一定限制。而对系统测试、天文观测、科学计算等方面的运算来说，就有些力不从心。因此，需要通过编程语言的算法设计，提供出可行方案。而大整数算法，就是实现对大整数进行运算的途径。因此，加强基于 C++ 的大整数类型的设计与实现研究具有重要意义。综上所述，为了将 C++ 更好的应用在大整数问题的解决中，本文将基于 C++ 的大整数类型的设计与实现作为主要研究内容，在对数据结构设计进行分析的基础上，对输入输出，储存表示，四则运算等方面做出系统探究。研究结果表明，不同算法组成的 C++ 大数据的运算，不仅能够应用于大整数实际问题解决中，还能够经修改后移植到其他操作系统。在未来，还需进一步加强对 C++ 的大整数类型设计与实现研究，以此为其他语言环境下大数运算问题的解决提供思路。

参考文献：

[1] 谭浩强 .C++ 程序设计 [M].3 版 . 北京：清华大学出版社，2015.

[3] 严蔚敏，吴伟民 . 数据结构 (C 语言版)[M]. 北京：清华大学出版社，2007.