大整数代数计算

一、问题描述 C/C++语言中的int类型能表示的整数范围是-231~231–1，unsigned int能表示的整数范围是0 ~ 232–1，即4294967295，所以，int 和unsigned int类型都不能存储超过10位的整数。但有些问题需要处理的整数远不止10位，这种大整数用C/C++语言的基本数据类型无法直接表示。请编写算法完成两个大整数的加、减、乘、除等基本的代数运算。

二、基本要求

①大整数的长度在100位以下；

②设计存储结构表示大整数；

③设计算法实现两个大整数的加、减、乘、除等基本的代数运算；

④分析算法的时间、空间复杂度。

1. 概要设计
2. 数据结构设计

因为大整数需要分几个存储单元来存，我就使每个数据元素存一位数。加法和减法我是用最简单的线性表来完成的。但乘法除法太过复杂，用了最简单的一维数组（整型指针）来存放各位数，不过我输入数据是用的是字符类型的数组，通过每位减48再转化为整型数据（如字符’1’的ASCII码为49，减去48，再转化为整数，变为整数1）。

1. 算法设计

分八个模块

第一个、调用加法和减法的函数，完成两个大数的线性表的准备工作， 通过swtch语句调用加法函数或加法函数。

#define List 100

#define Last 10

Sqlist suan()

{

printf("请输入将要计算的两数(减法时第一个数要大于等于第二个数)的长度：");

scanf("%d %d",&a.length,&b.length);//a第一个线性表，代表第一个大数，b第一个线性表，代表第二个大数

n=a.length>b.length? a.length:b.length;

n=n+1;//代表两大数长度最大者的长度加一

a.elem=(Elemtype\*)malloc(n\*sizeof(Elemtype));

b.elem=(Elemtype\*)malloc(n\*sizeof(Elemtype));

for(i=0;i<n;i++)

{

b.elem[i]=a.elem[i]=0;

}

printf("从低位到高位输入第一个数，位与位之间用空格：");

for(i=0;i<a.length;i++)

{

scanf("%d",&a.elem[i]);

}

printf("从低位到高位输入第二个数，位与位之间用空格：");

for(i=0;i<b.length;i++)

{

scanf("%d",&b.elem[i]);

}

printf("输入1,2代表加，减：");

scanf(&m);//1就执行加法，2就执行减法

switch(m)

{

case 1:{b=jia(a,b,n);print(b);}break;

case 2:{b=jian(a,b,n);print(b);}break;

default:break;

}

return b;

}

第二个、加法函数，通过各位相加进位得到结果（放入第二个大数的线性表中以节省空间）以及结果的长度：

Sqlist jia(Sqlist a,Sqlist b,int n)

{

for(i=0;i<n-1;i++)

{ if((a.elem[i]+b.elem[i])/10>0)b.elem[i+1]+=(a.elem[i]+b.elem[i])/10;

b.elem[i]=(a.elem[i]+b.elem[i])%10;

}

if(b.elem[n-1]!=0)

b.length=n;

return b;

}

第三个、加法函数，通过各位相加进位得到结果（放入第二个大数的线性表中以节省空间）以及结果的长度：

Sqlist jian(Sqlist a,Sqlist b,int n)

{

int i;

for(i=0;i<n-1;i++)

{

if(i<b.length)

{

if(a.elem[i]<b.elem[i])

{

b.elem[i]=a.elem[i]+10-b.elem[i];

if(a.elem[i+1]==0)a.elem[i+1]=10;

a.elem[i+1]=a.elem[i+1]-1;

}

else b.elem[i]=a.elem[i]-b.elem[i];

}

else if(i<a.length)b.elem[i]=a.elem[i];

}

for(i=n-2;i>=0&&b.elem[i]==0;i--);

b.length=i+1;

return b;

}

第四个、输出函数（输出加法减法的结果），从高位（一开始位上的数不能为0）到低位输出结果；

void print(Sqlist b)

{

int i;

for(i=b.length;i>=0;i--)

printf("%d",b.elem[i]);

}

第五个，乘法函数，利用一个大数的每一位与另一个大数相乘，然后做加法得出结果：

void cheng()

{

char mult1[List];//存储输入的大数的两个字符型数组

char mult2[List];

printf("输入两个数:\n");

scanf(mult1,mult2);

int len1 = strlen(mult1);

int len2 = strlen(mult2);

int \*c1 = (int \*)malloc(sizeof(int)\*len1);

int \*c2 = (int \*)malloc(sizeof(int)\*len2);

for(i=0; i<len1; i++)

c1[i]=(int)mult1[i]-48;

for(i=0; i<len2; i++)

c2[i]=(int)mult2[i]-48;

int \*result = (int \*)malloc(sizeof(int)\*(len1+len2));//result指向的空间用来存储结果

n=len1+len2-1;

for(i=0; i<=n; i++)

result[i]=0;

temp=0;

for(i=len2-1; i>-1; i--)

{

t=c2[i];

for(j=len1-1; j>-1 ;j--)

{

if((temp=result[n]+t\*c1[j])>=10)//temp临时存放相加后每位的数

{

result[n]=(temp%10);

result[n-1]+=temp/10;

}

else

result[n]=temp;

n--;

}

n=n+len1-1;

}

if(result[0]!=0)

printf(result[0]);

for(i=1;i<=len1+len2-1;i++)

printf(result[i]);

printf("\n");//太惭愧了http://blog.sina.com.cn/hexw100

}

第六个、除法自带的减法函数，先做一些判断，使得只有被减数大于等于减数时才能执行加法，返回减法之后的被减数的位数或零，否返回-1。

int SubStract( int \*p1, int \*p2, int len1, int len2 )//p1指向第一个大数（被除数），p2指向第二个大数（除数）

{

if( len1 < len2 )//len1，len2大数位数

return -1;

if( len1 == len2 )

{ //判断p1 > p2

for( i=len1-1; i>=0; i-- )

{

if( p1[i] > p2[i] ) //若大，则满足条件，可做减法

break;

else if( p1[i] < p2[i] ) //否则返回-1

return -1;

}

}

for( i=0; i<=len1-1; i++ ) //从低位开始做减法

{

p1[i] -= p2[i];

if( p1[i] < 0 ) //若p1<0，则需要借位

{

p1[i] += 10; //借1当10

p1[i+1]--; //高位减1

}

}

for( i=len1-1; i>=0; i-- ) //查找结果的最高位

if( p1[i] ) //最高位第一个不为0

return (i+1); //得到位数并返回

return 0; //两数相等的时候返回0

}

第七个、除法函数，利用做竖式除法常用的减数补0的方法来做减法，以得到除法的结果，输出商：

void chu()

{

printf("输入相除的两个数：\n");

scanf(str1); //以字符串形式读入大数

scanf(str2);

for ( i=0; i<List; i++ ) //初始化清零操作

{

num\_a[i] = 0;//num\_a被除数

num\_b[i] = 0;//num\_b除数

num\_c[i] = 0;//num\_c商

}

len1 = strlen(str1); //获得大数的位数

len2 = strlen(str2);

for( j=0, i=len1-1; i>=0; j++, i-- )

num\_a[j] = str1[i] - '0'; //将字符串转换成对应的整数,颠倒存储

for( j=0, i=len2-1; i>=0; j++, i-- )

num\_b[j] = str2[i] - '0';

nTimes = len1 - len2; //相差位数

for ( i=len1-1; i>=0; i-- ) //将除数扩大，使得除数和被除数位数相等

{

if ( i>=nTimes )

num\_b[i] = num\_b[i-nTimes];

else //低位置0

num\_b[i] = 0;

}

len2 = len1;

for( j=0; j<=nTimes; j++ ) //重复调用，同时记录减成功的次数，即为商

{

while((nTemp = SubStract(num\_a,num\_b + j,len1,len2 - j)) >= 0)//nTemp存放SubStract函数返回值

{

len1 = nTemp; //结果长度

num\_c[nTimes-j]++;//每成功减一次，将商的相应位加1

}

}

printf("商为：");

for( i=List-1; num\_c[i]==0 && i>=0; i-- );//跳过高位0

if( i>=0 )

for( ; i>=0; i-- )

printf("%d", num\_c[i]);

else

printf("0");

printf("\n");

}//感觉我太废了，这么个程序除了加法减法是自己的思想，乘法除法却要抄网上的大数,高精度计算---大数除法 - javawebsoa - 博客园

第八个、主函数，通过switch语句调用算加减的函数或乘法函数或除法函数，return0；结束：

int main()

{

printf("输入1，2,3代表算加减或算乘或算除:");

scanf("&a);

switch(a)//a为1，2，3相应执行算加减的函数或乘法函数或除法函数

{

case 1:b=suan();break;////b为线性表

case 2:cheng();break;

case 3:chu();break;

}

printf("\n");

return 0;

}

1. 抽象数据类型设计

加法减法

typedef int Elemtype;//Elemtype是自定义的类型

typedef struct

{

Elemtype \*elem;//数据元素

int length;//数据长度

int listsize;//当前分配的存储容量

}Sqlist;

乘法

char mult1[List];//输入第一个大数

char mult2[List];//输入第二个大数

int len1 = strlen(mult1);//第一个大数的长度

int len2 = strlen(mult2);//第二个大数的长度

int \*c1 = (int \*)malloc(sizeof(int)\*len1);//指向第一个大数的长度的临时空间的指针

int \*c2 = (int \*)malloc(sizeof(int)\*len2);//指向第一个大数的长度的临时空间的指针

除法

int num\_a[List]; //被除大数

int num\_b[List]; //除大数

int num\_c[List]; //商

char str1[List + 1]; //读入的第一个大数

char str2[List + 1]; //读入的第二个大数

1. 详细设计

1、主函数

用一个switch语句来判断调用算加减或乘或除的函数，return 0；结束。

2、子函数

算加减的函数

因为加减用的是同一种存储结构，所以把它们放进一个函数，进行同一种初始化准备工作，再通过switch语句判断执行加法函数还是减法函数，得到结果的返回值，调用输出函数，输出结果。先输入两个大数的长度，减法的第一个大数要长于或等于第二个大数的长度，然后通过两个大数的长度的最大者加一，给两个大数所代表的线性表分配临时空间。然后从低位到高位将两个大数的每一位赋给线性表的每一个数据元素。

加法函数

从低位做加法，如果当前相加大于10，就除10，通过默认类型转换转换为int型，加到下一位。而当前位取余作为结果。执行两大数的最大长度数作为次数。最后得到的大数的长度要么是相加的两大数的长度最大的那个数的长度，要么是前者加一，做判断，如前者10，后者11，看第11位是否为零，然后将位数赋给结果（第二个大数所代表的线性表，以节省空间）。

减法函数

采用的是与加法不同的思想，加法进位，减法借位。由于事先要求输入的第一个大数要比第二个数大的长度大或相等，所以做第一个大数的长度次数的循环，如果当前位数小于等于第二个大数的长度，就各位相减，但如果第一个大数的当前位小于对应的第二个大数的当前位，就第一个大数此位加10再相减，要到第一个大数下一位借1，否则就直接相减。如果当前位已超过第二个大数的长度，就直接把第一个大数赋给结果（第二个大数所代表的线性表，以节省空间）。然后就找结果对应的线性表第一个大数原有的最大位找不为零的位，往低位找，直至找到从第一个不为零的高位向低位输出，或者找不到就输出零，最后将找到的位数赋给结果的线性表的长度变量。

输出函数

这是给加法函数和加法函数用的输出函数，以结果的长度作为循环次数，从高位到低位输出各位。

乘法函数

其实用的是跟加法一样的思想，进位。我输入数据是用的是字符类型的数组，这样通过算字符串的长度函数便可得知两大数的长度通过每位减48再转化为整型数据（如字符’1’的ASCII码为49，减去48，再转化为整数，变为整数1），这是一个小技巧。当然这次输入就是从高位到低位了。经过研究，两大数相乘所得的长度要么是原来两大数的长度之和要么是前者减一，所以就建了一个两大数长度之和大小的临时空间，用结果result指针指向它，先全都初始化为零。然后从低位到高位做乘法，将两数对应位的数相乘，加到结果（从低位到高位）上，如果大于10，此位取余，通过原位除10，加到下一位，，否则就不变，依此类推，但每次循环，比如上次从结果的第1位开始加，这次从第2位开始加。得到结果后开始输出，判断result[0]是否为空（结果长度是否是原两大数之和），不为空则输出，否则不输出，再从剩下的高位向低位输出。

除法函数

为是算法简洁一些，我只求商，不求余数。建了三个整型数组代表被除数、除数、商，两个字符型数组来输入数据。我输入数据是用的是字符类型的数组，这样通过算字符串的长度函数便可得知两大数的长度通过每位减48再转化为整型数据（如字符’1’的ASCII码为49，减去48，再转化为整数，变为整数1），这是一个小技巧。输入也是从高位到低位。但与乘法不同我颠倒了整型数组里的存储顺序。

for( j=0, i=len1-1; i>=0; j++, i-- )

num\_a[j] = str1[i] - '0'; //将字符串转换成对应的整数,颠倒存储

for( j=0, i=len2-1; i>=0; j++, i-- )

num\_b[j] = str2[i] - '0';

nTimes = len1 - len2; //相差位数

为下一步扩大除数的位数和被除数一样准备足够的空间。比如121/11，扩大11的位数变为

1. 然后做减法，调用除法自带的减法函数，121减110，再11-110，一直从被减数上减去除数，每减一次，将商的相应位加一，直到最后被减数（11）小于110，无法再做减法，然后在除数上消去添的0，得11，用已得的被减数减11，一直减，每减一次将商的对应位加1，，直至被减数小于除数，就再消去添的0，直至消完为止（若原来被除数在低位有几个0，千万不能消去）。然后输出，从空间最大位向低位开始找不为零的位，循环，直至找到便从第一个不为0的高位向低位输出，找不到便输出0。

除法自带的减法函数

先比较两大数的长度，如果被减数（第一个大数）的长度小于除数（第二个大数）的长度就不做这函数，如果相等，就从最高位比较谁大谁小，如果第一个大数大于第二大数就跳出循环做加法；相等就函数返回值为零，结束函数；第二个大数比第一个大数大，就函数返回值为代表错误的-1，结束函数。如果第一个大数的长度大于第二个大数的长度，就也做减法。但如果当前位的第一个数小于对应的第二个数，就第一个大数此位加10再相减，要到第一个大数下一位借1，否则就直接相减。然后就找结果对应的数组从第一个数原有的最大位向低位开始找不为零的位，往低位找，直至找到从第一个不为零的位，函数返回位数，或者找不到就返回0。

1. 运行与测试
2. 加法没问题，减法本来想添一些其他功能，如较小的大数减较大的大数，但还是想

洁些，就去掉了这一想法。

乘法除法我很惭愧，那是我从网上借鉴的，本来乘法想出来了，可是运行失败，怕时间不够，就上网搜了。但网上的乘法缺了一个取余符号，我把那算法弄懂了，因此看出了错误，给添上了。

除法我去掉了取余这一功能，只求商，以保证程序的简单直接性。

1. 我使用各种数据测试。

减法我测试了一下两数相等的情况，可以正确输出0。

乘法由于网上缺了取余符号，123\*123是中间多了两个一，弄懂程序后添上了取余符号，成功输出。

除法测试了能整除的数和不能整出的数，确实如预料那样。

1. 总结与心得

我应当在每次编程之前仔细设计算法，将各类细节研究明白，画流程图或写伪代码是不错的选择，再编程。这样会避免出现无法预知的情况，导致不好解决，忙中出错。不能太过心急，有个模糊的大纲就编。应该把编大程序的思想用到编程中。

我也要更加努力的学习，这次没能想出乘法除法，还要从网上借鉴让我很惭愧。应多提升自己的能力。

时间复杂度：乘法函数里的

for(i=len2-1; i>-1; i--)

t=c2[i];

for(j=len1-1; j>-1 ;j--)

{

if((temp=result[n]+t\*c1[j])>=10)

{

result[n]=(temp%10);

result[n-1]+=temp/10;

}

else

result[n]=temp;

n--;

}

O（len1\*len2）（len1和 len2是要计算的两个大数的长度）

空间复杂度：int \*result = (int \*)malloc(sizeof(int)\*(len1+len2));O（len1+len2）