



**数据结构课程设计报告**

**课程名称 数据结构**

**题目名称 长整数四则运算**

**学生学院 计算机学院**

**专业班级 22软件工程卓越班**

**学 号 3222004465**

**学生姓名 陈婉瑜**

**指导教师 苏 庆**

**2023 年 12 月 31 日**

目录

[一、需求分析 2](#_Toc154530385)

[二、 概要设计 4](#_Toc154530386)

[三、详细设计 6](#_Toc154530387)

[四、调试分析 24](#_Toc154530388)

[五、用户使用说明 25](#_Toc154530389)

[六、测试结果 26](#_Toc154530390)

[七、附录 29](#_Toc154530391)

**题目1 长整数四则运算**（难度系数：1.0）

**[问题描述]**

一个实现任意长的整数进行加法运算的演示程序。

**[基本要求]**

利用双向循环链表实现长整数的存储，每个结点含一个整型变量。任何整型变量的范围是-(215-1)～(215-1)。输入和输出形式：按中国对于长整数的表示习惯，每四位一组，组间用逗号隔开。

**[测试数据]**

(1) 0；0；应输出“0”。

(2) -2345,6789；-7654,3211；应输出“-1,0000,0000”。

(3) -9999,9999；1,0000,0000,0000；应输出“9999,0000,0001”。

(4) 1,0001,0001；-1,0001,0001；应输出“0”。

(5) 1,0001,0001；-1,0001,0000；应输出“1”。

(6) -9999,9999,9999；-9999,9999,9999；应输出“-1,9999,9999,9998”。

(7) 1,0000,9999,9999；1；应输出“1,0001,0000,0000”。

**[实现提示]**

(1) 每个结点中可以存放的最大整数为215-1=32767，才能保证两数相加不会溢出。但若这样存，即相当于按32768进制数存，在十进制数与32768进制数之间的转换十分不方便。故可以在每个结点中仅存十进制数的4位，即不超过9999的非负整数，整个链表视为万进制数。

(2) 可以利用头结点数据域的符号代表长整数的符号。用其绝对值表示元素结点数目。相加过程中不要破坏两个操作数链表。两操作数的头指针存于指针数组中是简化程序结构的一种方法。不能给长整数位数规定上限。

**[选做内容]**

（1）实现长整数的四则运算；

（2）实现长整数的乘方和阶乘运算；

（3）整型量范围是-(2n-1)～(2n-1)，其中，n是由程序读入的参量。输入数据的分组方法可以另行规定。

# 一、需求分析

本程序是一个实现任意长的整数四则运算和乘方阶乘运算的演示程序，由用户选择相应的运算方式，然后输入数据，程序计算得出结果并显示在屏幕上。

1. 本程序中，利用双向循环链表现实长整数的存储，每个结点含一个整形变量，整型变量的范围是-(215-1)～(215-1)。可以直接输入正数或负数，输入值集合的元素限定为数字字符[0~9]。每次输入完成后按回车键结束。
2. 输出运算结果以每四位一组，组间用逗号隔开。
3. 该程序可以计算两个任意长的整数的四则运算结果，包括正负数相加减乘除，正数加减乘除，负数加减乘除。还扩展了阶乘，乘方运算。
4. 测试数据：

一、加法功能测试：

1： 0；0；应输出“0”。

2： -2345,6789；-7654,3211；应输出“-1,0000,0000”。

3： -9999,9999；1,0000,0000,0000；应输出“9999,0000,0001”。

4： 1,0001,0001；-1,0001,0001；应输出“0”。

5： 1,0001,0001；-1,0001,0000；应输出“1”。

6： -9999,9999,9999；-9999,9999,9999；应输出“-1,9999,9999,9998”。

7： 1,0000,9999,9999；1；应输出“1,0001,0000,0000”。

二、减法功能测试：

9： 12,3456,0000；12,3456,0000；应输出“0”。

10：12,3456,0000；-12,3456,0000；应输出“24,6912,0000”

11：0；-12,3456；应输出“12,3456”

12：-12,3456；-12,3456；应输出“0”

三、乘法功能测试：

13：12,3456；1000；应输出“1,2345,6000”

14：-12,3456；1000；应输出“-1,2345,6000”

15：0；4,0000,000；应输出“0”

16：-12,3456；-1000；应输出“1,2345,6000”

四、除法功能测试：

17：9999,9999；400；应输出“24,9999”

18：-9999,9999；-400；应输出“24,9999”

19：-9999,9999；400；应输出“-24,9999”

20：0；-4,0000,0000；应输出“0”

21：-9999,9999；0；应输出“ERROR!公式错误，0不可以做除数”

22：1,2300,4500; 12300,4500; 应输出“1”

五、乘方功能测试：

23: 0^4；应输出“0的任何次乘方没有意义”

24：23^23；应输出“2088,0467,9998,4791,2034,3550,3291,0567”

25：1^3434；应输出“1”

26：78,7878^1；应输出“78,7878”

27:12,3456,7890^0；应输出“1”

28：-45^4；应输出“410,0625”

29：-233^3:；应输出“-1264,9337”

六、阶乘功能测试：

30：-34；应输出“[ERROR]I!I负数没有阶乘”

31：0；应输出“1”

32：17；应输出“355,6874,2809,6000”

33：1；应输出“1”

七、出错测试：

23：输入数据错误测试

123ss21; 应输出“ERROR!数字输入格式错误”

24：选择操作时输入错误数据

按下数字8 ；应输出“您选择退出程序”并退出程序

# 二、 概要设计

为实现上述程序功能，采用双向循环链表来储存长整数。

利用头结点数据域的符号代表长整数的符号，用其绝对值表示元素结点数目。

**双向循环链表的储存结构:**

typedef struct Node{

int data;

struct Node \*prior,\*next;

}Node,\*LinkList;

//表头结点数据域的符号代表长整数的符号

//其绝对值表示数据存储在链表的结点数目

为实现该程序四则运算并显示结果功能，主程序调用了以下函数

**void InitList(LinkList &L);**

操作结果：初始化一个双向循环链表，建立空的链表

**void ListInsert(LinkList &L,int x);**

操作结果：在链表L尾部插入一个存储数据为x的结点

**void destroyList(LinkList &head)**

操作结果：销毁双向循环链表的函数

**int getnum(LinkList &L);**

操作结果：读取用户输入的数据并判断输入数据是否符合格式，输入正确并成功插入链表的返回1，否则返回0

**void IN(LinkList &L);**

操作结果：对链表L每个结点数据检查是否大于9999产生进位，若产生进位则调整链表进位后的结点数据

**int Compare(LinkList a,LinkList b);**

操作结果：比较链表a、b存储数据的绝对值大小，a>b返回1 a=b返回2 a<b返回0

**void Printdlist(LinkList &L);**

操作结果：按中国对于长整数的表示习惯，每四位一组，组间用逗号隔开打印输出链表L存储数据

**void Add(LinkList a,LinkList b,LinkList &c);**

操作结果：将链表a、b数据相加a+b，求和存入c链表

**void subtract(LinkList a,LinkList b,LinkList &c)**;

操作结果：链表a、b数据相减a-b，求差存入c链表

**void multiply(LinkList a,LinkList b,LinkList &c);**

操作结果：链表a、b数据相乘a\*b，求积存入c链表

**void divide(LinkList a,LinkList b,LinkList &c);**

操作结果：链表a、b数据相除a/b，求商存入c链表

**void Impl(LinkList a,LinkList &c,int x)**

操作结果：乘方运算a^x=c

**void Factorial(LinkList a,LinkList &c)**

操作结果：阶乘运算 a!=c

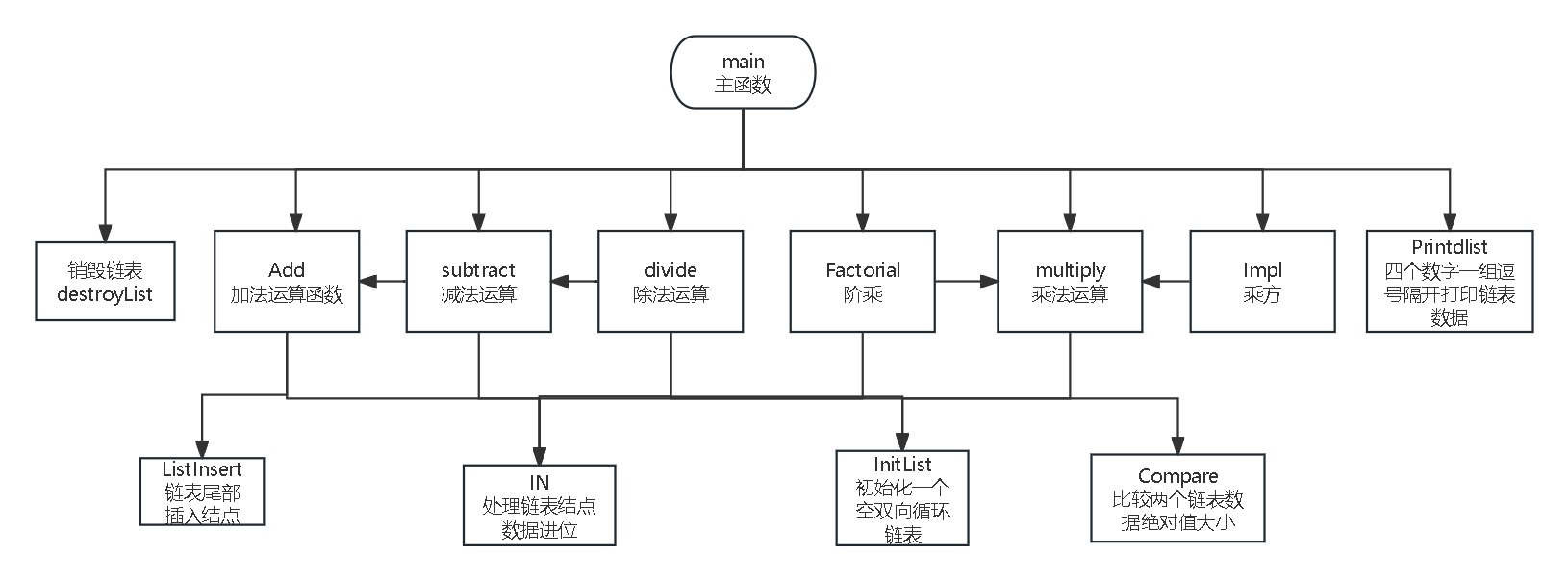


图1：各函数间的调用关系图

**主程序流程:**

（1）菜单页面提示用户选择计算方式

（2）用户输入数据

（3）程序判断用户输入数据是否格式正确并成功构建链表，输入错误将重新提示输入数据直至输入正确，输入正确则调用对应运算函数

（4）屏幕打印显示计算结果。重复循环上述步骤

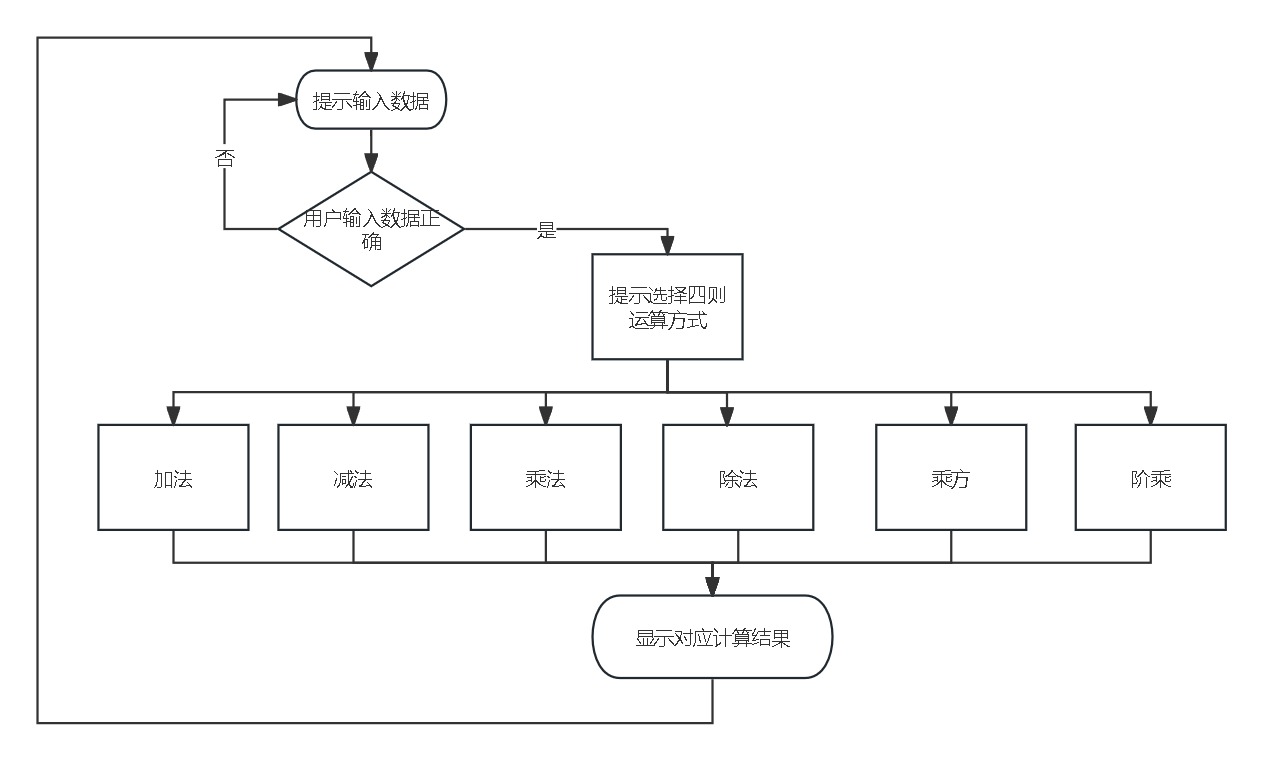


图2：主程序运行流程图

# 三、详细设计

头文件的定义部分：

#include<stdio.h>

#include<stdlib.h>

#include<cstring>

#include<conio.h>

#include <iostream>

1. **初始化一个双向循环链表，建立空的双向循环链表：**

**实现代码：**

void InitList(LinkList &L)

{ L=(LinkList)malloc(sizeof(Node));

L->data=0;

L->next=L;

L->prior=L;

}

1. **在链表L尾部插入一个存储数据为x的结点**

**实现代码：**

void ListInsert(LinkList &L,int x)

{ LinkList p=(LinkList)malloc(sizeof(Node));

LinkList q=L->prior;

p->data=x;

p->next=L;

p->prior=q;

q->next=p;

L->prior=p;

if(L->data>=0)L->data++;

else L->data--;}

1. **按中国对于长整数的表示习惯，每四位一组，组间用逗号隔开打印输出链表L存储数据**

**实现代码：**

void Printdlist(LinkList &L)//输出打印链表数据

{printf("（");

LinkList p=L->prior;

if(L->data<0)printf("-");

while(p!=L)

{if(p->next!=L)printf("%04d",p->data);

else printf("%d",p->data);

if(p->prior!=L)printf(",");

p=p->prior;}

printf("）");}

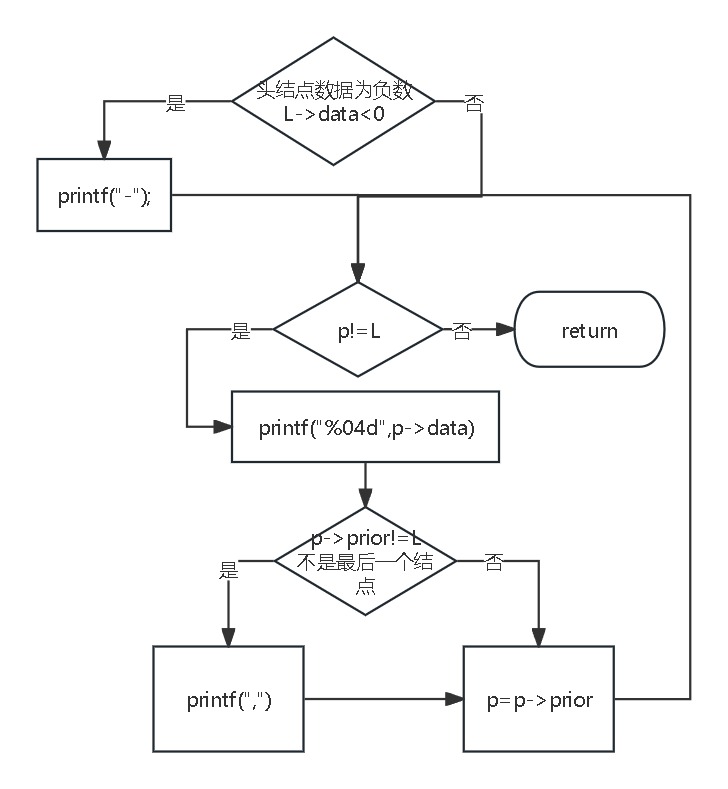


图3：打印链表数据Printdlist流程图

**4.对链表L每个结点数据检查是否大于9999产生进位，若产生进位则调整链表进位后的结点数据**

**实现思路**：

因为链表结点存储的数据是从低位开始，即最高位在尾结点，因此从第二个结点开始开始遍历链表，当p->data>9999发生进位时，用if(p->next==L)判断若是尾结点最高位发生进位 ，则多一步调用ListInsert函数在尾部插入新结点，存储进位后的值。

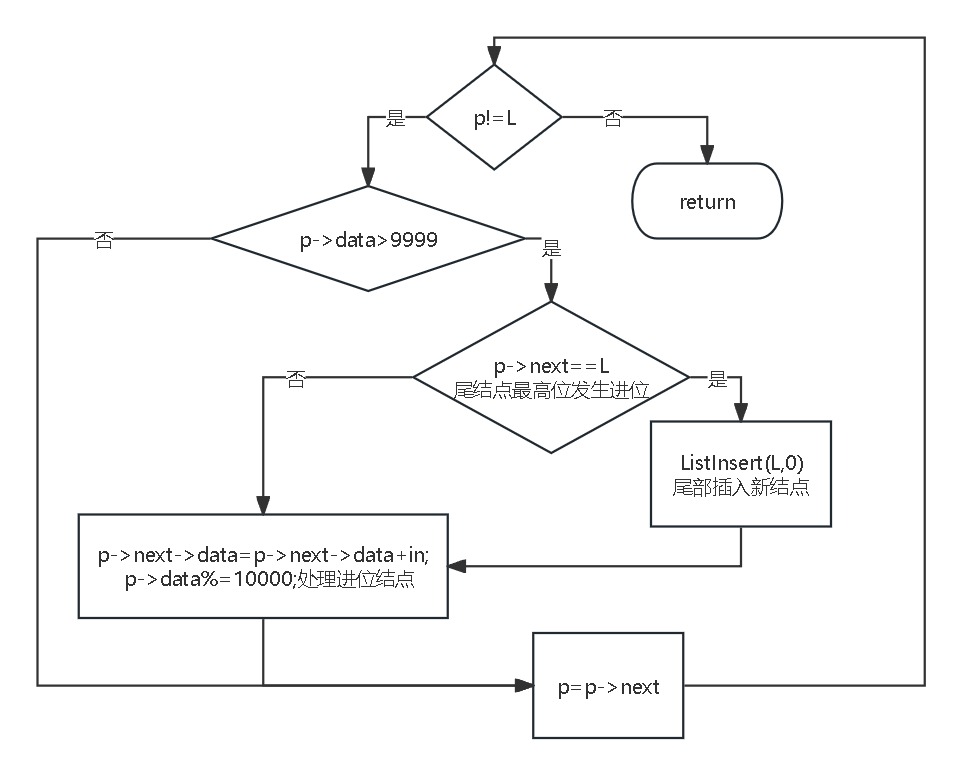


图4：处理链表结点进位IN函数流程图

**实现代码**：

void IN(LinkList &L)

{LinkList p=L->next;

while(p!=L)

{while(p->data>9999)//产生进位

{int in=p->data/10000;//进位值

if(p->next==L)ListInsert(L,0);//尾结点最高位发生进位 ，尾部插入新结点

p->next->data=p->next->data+in;//高位加上进位数值

p->data%=10000;//进位完成后减去进位值 }

p=p->next;}

while(L->prior->data==0&&abs(L->data)>1){

// 删除链表尾部高位多余的0

if(L->data==0)return;

LinkList a=L->prior;

LinkList b=a->prior;

b->next=L;

L->prior=b;

free(a);

// 更新头结点数据域存储的结点个数值

if(L->data>=0)L->data--;

else L->data++;}}

**5.读取用户输入的数据并判断输入数据是否符合格式，输入正确并成功插入链表的返回1，否则返回0**

**实现思路：**

先创建一个一维数组保存用户输入的数字，接着通过for循环遍历数组判断数据是否合法（是数字0~9除了第一位可能为“-”、“+”号），若输入数据合法，则按照四位数字一组存入链表L，尾结点数据为最高位，表头结点数据域的符号代表长整数的符号，其绝对值表示数据存储在链表的结点数目，创建链表成功返回1，否则返回0

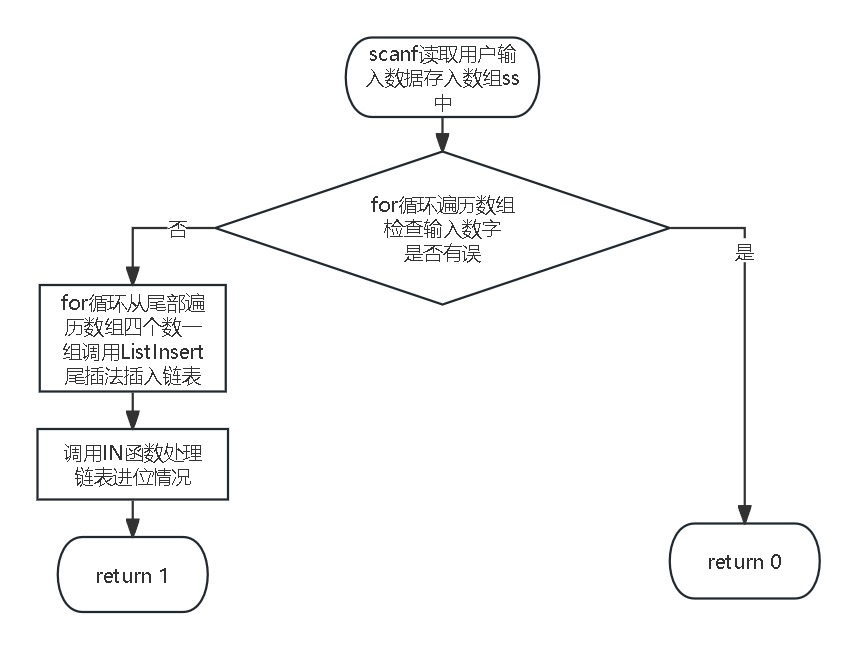


图5：读取输入数据getnum函数流程图

**实现代码：**

int getnum(LinkList &L)//读取一个数

{ char ss[100000];

scanf("%[^\n]",ss);

getchar();

//检查输入数字，不是数字且不是第一位的正负号

for(int i=0;i<strlen(ss);i++)

if(!(i==0&&(ss[0]=='-'||ss[0]=='+'))&&(!(ss[i]>='0'&&ss[i]<='9')))

return 0;

// 链表每个结点存放四位数字不超过9999

int j=0;

for(int i=strlen(ss)-1;i>=0;i--)

{if(ss[i]>='0'&&ss[i]<='9')

{j++;

if(j==4||i==0||(i==1)&&ss[0]=='-')

{int sum=0;

for(int k=i;k<i+j;k++)

sum=10\*sum+(ss[k]-'0');

ListInsert(L,sum);

j=0; } }}

if(ss[0]=='-')L->data=-L->data; // 输入负数时

if(L->data==0)return 0;

IN(L);

return 1;}

**6. 比较链表a、b存储数据的绝对值大小，a<b返回0、a>b返回1、a=b返回2**

**实现思路：**

先比较ab链表头结点绝对值，存储数据长度大的绝对值一定大，若长度相同，则从尾结点最高位开始遍历各个结点直至比较出大小。

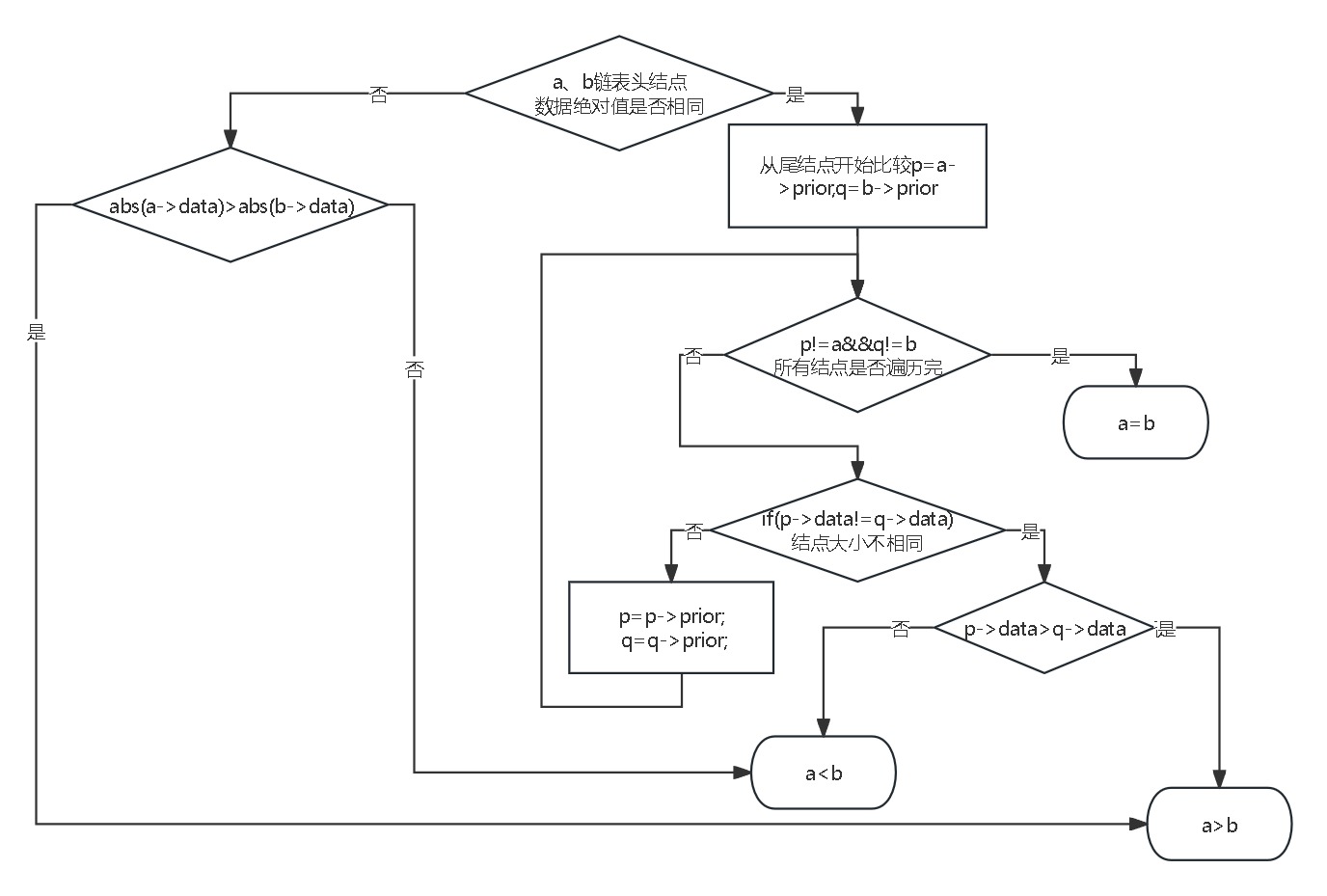


图6：比较链表ab数值绝对值大小函数Compare流程图

**实现代码：**

int Compare(LinkList a,LinkList b)

{if(abs(a->data)!=abs(b->data))//存储结点个数不同时

{if(abs(a->data)>abs(b->data))return 1;

else return 0;}

LinkList p=a->prior,q=b->prior;//从尾结点，最高位开始比较大小

while(p!=a&&q!=b)

{if(p->data!=q->data)

{if(p->data>q->data)return 1;

else return 0;}

p=p->prior;

q=q->prior;}

return 2;}

7. **链表a、b数据相加a+b，求和存入c链表**

**实现思路：**

分同号和异号两种情况，同号相加只需要遍历两个链表所有结点数据并相加存入c结点，异号需要先比较两个数绝对值大小，绝对值大的作为被减数，遍历两个链表所有结点做减法，存入c结点。最后调用IN函数处理c链表进位情况

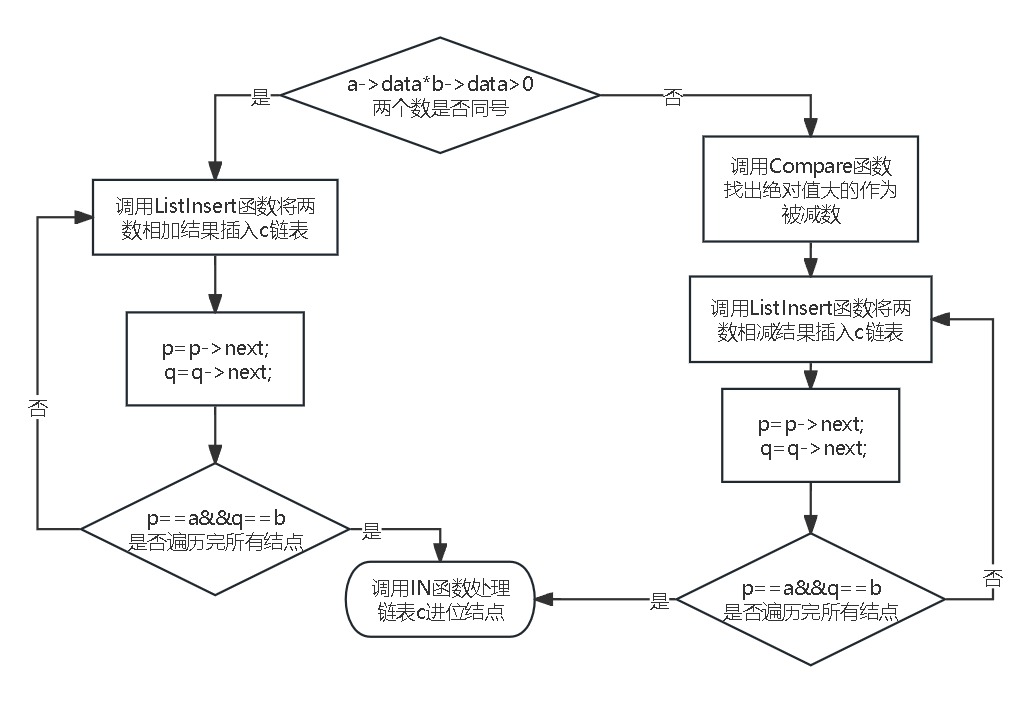


图7：加法函数Add流程图

**实现代码：**

void Add(LinkList a,LinkList b,LinkList &c)//a+b=c

{//同号相加

if(a->data\*b->data>0)

{LinkList p=a->next,q=b->next;

//两个链表都未加完所有结点数

while(p!=a&&q!=b)

{ListInsert(c,p->data+q->data);

p=p->next;

q=q->next;}

// 两个数结点数不同，其中一个所有结点数已遍历相加完

while(p!=a)

{ListInsert(c,p->data);

p=p->next;}

while(q!=b)

{ListInsert(c,q->data);

q=q->next;}

if(a->data<0||b->data<0) c->data=-1\*(c->data);

//负数相加更新得数的头结点数据为负数 }

//异号相加 else{

// 判断绝对值较大的数作为被减数，得数符号与被减数同号

switch(Compare(a,b))

{ case 0://绝对值a<b

{LinkList p=a->next,q=b->next;

// 两个数位数都未遍历完，两数相减

while(p!=a&&q!=b)

{ if(q->data<p->data)//不够减则向上借位

{q->next->data--;//高位-1

q->data+=10000;}

ListInsert(c,q->data-p->data);

p=p->next;

q=q->next;}

// 被减数未遍历完，减数遍历完

while(q!=b)

{if(q->data<0)//不够减则向上借位

{q->next->data--;//高位-1

q->data+=10000;}

ListInsert(c,q->data);

q=q->next;}

if(b->data<0)c->data=-c->data;

break; }

case 1://a>b

{LinkList p=a->next,q=b->next;

while(p!=a&&q!=b)

{if(p->data<q->data)//不够减则向上借位

{p->next->data--;

p->data+=10000;}

ListInsert(c,p->data-q->data);

p=p->next;

q=q->next;}

while(p!=a) {

if(p->data<0)//不够减则向上借位

{ p->next->data--;//高位-1

p->data+=10000; }

ListInsert(c,p->data);

p=p->next;}

if(a->data<0)c->data=-c->data;//绝对值较大者确定符号

break; }

case 2://a=b

ListInsert(c,0);

return;}}

IN(c);//处理进位}

**8. 链表a、b数据相减a-b，求差存入c链表**

**实现思路：**

减法相当于被减数加上负的减数，因此可以调用加法函数来实现减法函数a-b=a+(-b)=c

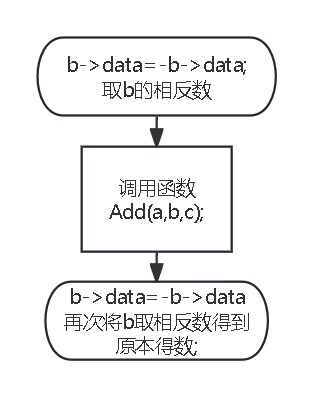


图8：减法函数subtract流程图

**实现代码：**

void subtract(LinkList a,LinkList b,LinkList &c)

{

b->data=-b->data;

Add(a,b,c);

b->data=-b->data;}

**9. 链表a、b数据相乘a\*b，求积存入c链表**

**实现思路：**

先判断ab两个数是否有为0的，若有则调用ListInsert函数插入0，得出计算结果。若没有则用两层嵌套的for循环，遍历链表a和 b，使a各个结点依次与b的所有结点相乘，同时乘完后所得结果加到对应位数的c链表结点上，最后再调用In函数处理c链表进位结点

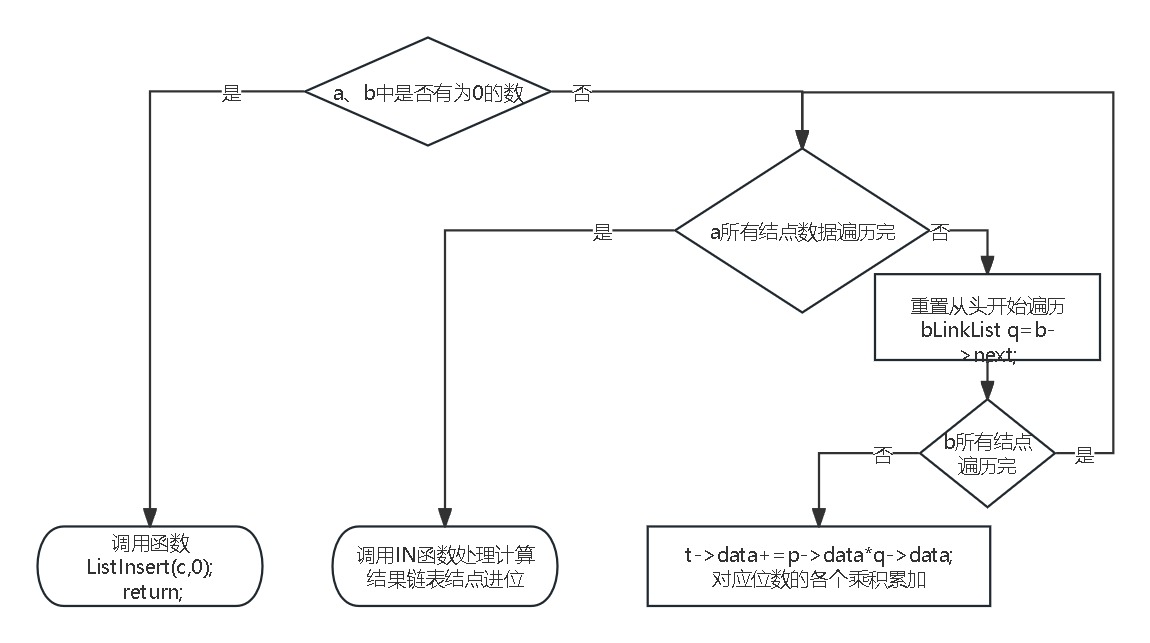


图9：乘法函数multiply流程图

**实现代码：**

void multiply(LinkList a,LinkList b,LinkList &c)

{//其中有一个数为0

if(abs(a->data)==1&&a->next->data==0||abs(b->data)==1&&b->next->data==0)

{ListInsert(c,0);

return;}

// 积的最大位数以0占位插入结点

for(int i=1;i<=(abs(a->data)+abs(b->data));i++)

ListInsert(c,0);

LinkList p=a->next;

for(int i=1;p!=a;p=p->next,i++)

{LinkList q=b->next;

for(int j=1;q!=b;q=q->next,j++)

{int k=i+j-1;

LinkList t=c;

while(k--)t=t->next;

t->data+=p->data\*q->data;//乘完后各个相应位数相加 }}

//ab异号，积为负数

if(a->data\*b->data<0)c->data=-1\*c->data;

IN(c);}

**10. 链表a、b数据相除a/b，求商存入c链表**

**实现思路：**

先调用compare函数判断被除数绝对值是否小于除数，若是，则直接得出计算结果为0.调用ListInsert函数插入0，返回结果。若不是，则要将除法转换为被除数的高位与除数低位做减法：从高位到低位，用被除数减去除数的多次倍数，以获得商的每一位。每次减法后，将得到的差值作为新的被除数。每次减法的次数即为当前位的商。将商记录下来。将新的被除数的低位与下一位对齐。继续进行减法直到被除数的位数小于除数的位数。最后得到的商即为结果，被除数的余数即为最后的被除数。

在此函数中先判断除数b是否为负数，通过设立一个标志flag=0,若b为负数，将b转为正数，并将flag=1。再调用InitList(tmp)初始化一个新的链表tmp,将存有被除数的链表a从高位到低位拷贝到tmp（相当于翻转链表a）,再调用subtract（tmp,b,c）函数让tmp减去b,得到结果存入c,同时通过flag判断是否要将b取相反数变回原始的正负号，最后调用IN（c）函数处理存有计算结果的链表c的进位结点

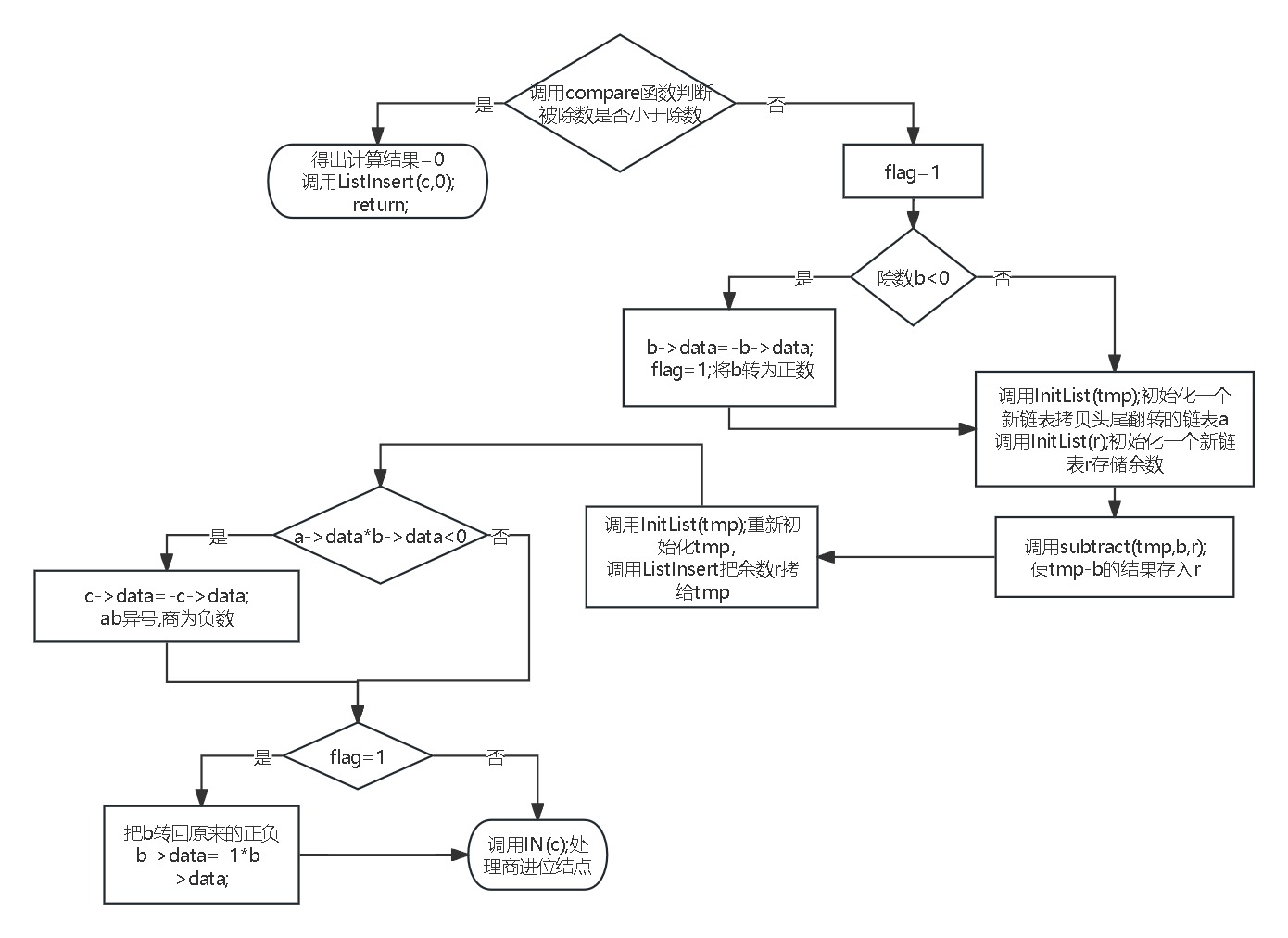


图10：除法函数divide流程图

**实现代码：**

void divide(LinkList a,LinkList b,LinkList &c)

{//a<b，商小于1，不显示小数，结果为0

if(Compare(a,b)==0)

{ListInsert(c,0);

return;}

for(int i=1;i<=abs(a->data);i++)

ListInsert(c,0);//商的最大位数

int flag=0; //flag=1,b由负转正数，=0没有改变

//b为负数时转正数，以便于做减法

if(b->data<0)

{b->data=-b->data;

flag=1;}

LinkList tmp;

InitList(tmp);

LinkList p=a->prior;

for(LinkList t=c->prior;t!=c;t=t->prior,p=p->prior)

{//将a高位部分拷给新建一个链表tmp，相当于翻转链表a存到tmp

LinkList q=(LinkList)malloc(sizeof(Node));

q->data=p->data;

q->next=tmp->next;

q->prior=tmp;

tmp->next->prior=q;

tmp->next=q;

tmp->data++;

//把除法转换为从高到低位的减法

while(Compare(tmp,b))

{t->data++;

LinkList r;//余数

InitList(r);

subtract(tmp,b,r);

// 删除链表尾部高位多余的0

while(tmp->data!=0){

if(tmp->data==0)return;

LinkList a=tmp->prior;

LinkList b=a->prior;

b->next=tmp;

tmp->prior=b;

free(a);

if(tmp->data>=0)tmp->data--;

else tmp->data++;}

InitList(tmp);

for(LinkList p=r->next;p!=r;p=p->next)

ListInsert(tmp,p->data);//把余数r拷给tmp

}}

// 把b转回原来的正负

if(flag)b->data=-1\*b->data;

if(a->data\*b->data<0)c->data=-c->data;//ab异号

IN(c);}

11. void Impl乘方运算a^x=c

**实现思路：**

首先，对输入的数a进行一些预处理，包括将负数转为正数，并将链表中的数值转换为一个长整型变量 sum。1.处理特殊情况：0 的任何次乘方都没有意义。

对指数 x 进行判断：如果 x 为负数，输出无法计算指数n为负数的结果。如果x为 0输出 1。如果 x 为 1，直接将 a 复制到 c。2.对指数 x 大于等于 2 的情况进行循环计算：首先将 a 复制到临时链表 d。使用 multiply 函数计算 a 和 d 的乘积，结果存储在链表 r 中。将结果链表 r 复制回 d，继续下一轮循环。最后将结果链表 r 复制回输出链表 c。

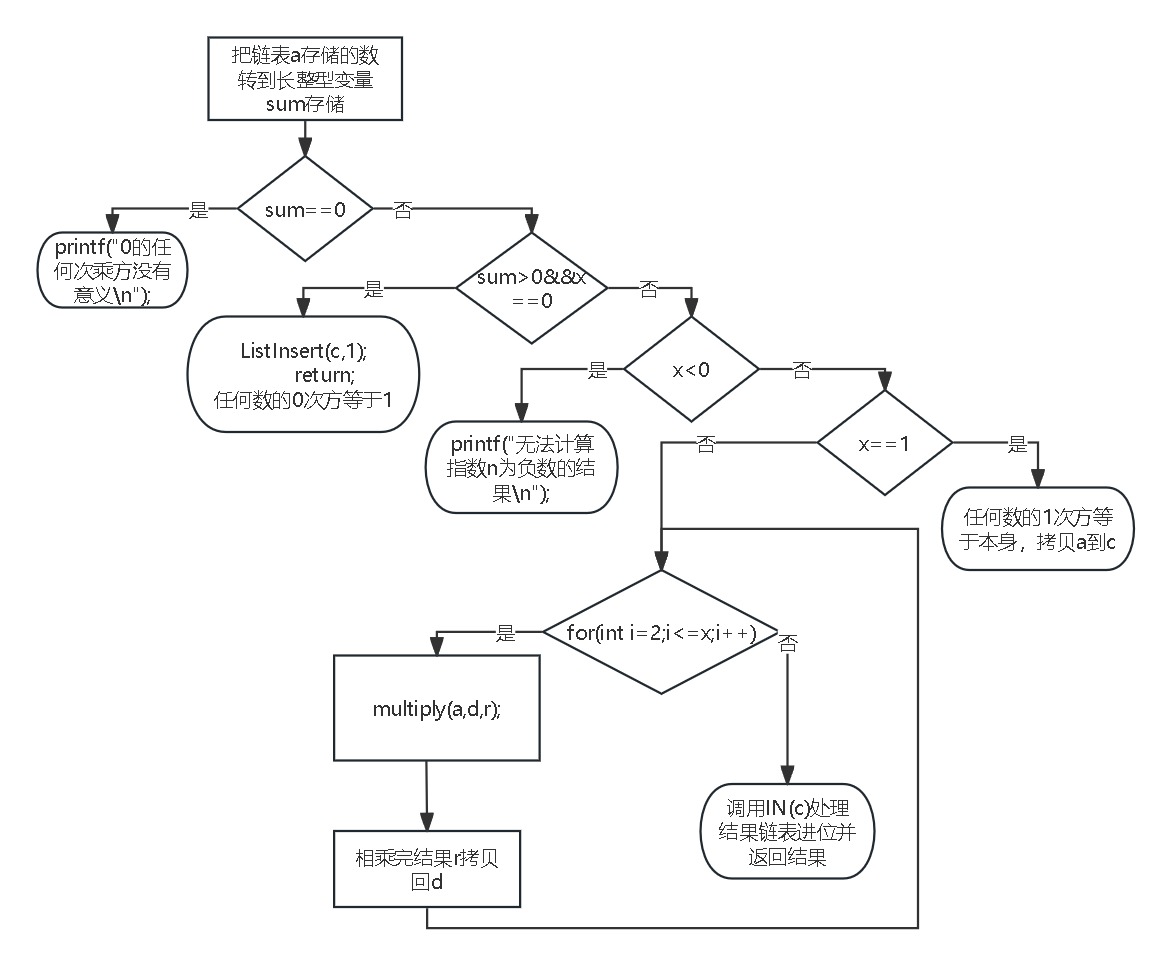


图11：乘方函数Impl流程图

**实现代码：**

void Impl(LinkList a,LinkList &c,int x){

LinkList d;

InitList(d);

LinkList r;

InitList(r);

int flag=0;//标志输入数据是否为负数，负要变为正数，同时flag=1

long int sum=0;

if(a->data<0){

a->data=-a->data;

flag=1;}

LinkList p=a->prior;

while(p!=a){//将链表存储的数转到长整型变量sum存储

sum=sum\*1000+p->data;

p=p->prior;}

if(sum==0){

printf("0的任何次乘方没有意义\n");

return ; }

if(sum>0&&x==0){//任何数的0次方等于1

ListInsert(c,1);

return; }

if(x<0){//链表存储的数据无法表示小数，小数只能表示为0，无法计算

printf("无法计算指数n为负数的结果\n");}

if(x==1){//任何数的1次方等于本身，拷贝a到c

for(LinkList h=a->next;h!=a;h=h->next){

ListInsert(c,h->data);}}

//x>=2

for(LinkList p=a->next;p!=a;p=p->next){

ListInsert(d,p->data);

}//a拷贝到d

for(int i=2;i<=x;i++){

multiply(a,d,r);

while(d->data!=0){//删除尾部最高位多余的结点0000

if(d->data==0){return;}

LinkList g=d->prior;

LinkList z=g->prior;

z->next=d;

d->prior=z;

free(g);

if(d->data>=0){

d->data--;}

else{

d->data++;} }

InitList(d);

for(LinkList p=r->next;p!=r;p=p->next){

ListInsert(d,p->data);

}//相乘完结果r拷贝回d 再次进入循环a\*d=r

if(i<x){

while(r->data!=0){//删除尾部最高位多余的结点0000

if(r->data==0) return;

LinkList g=r->prior;

LinkList z=g->prior;

z->next=r;

r->prior=z;

free(g);

if(r->data>=0){

r->data--; }

else{r->data++; }}

InitList(r);} }

// 最后乘方结果考回c返回c

for(LinkList h=r->next;h!=r;h=h->next){

ListInsert(c,h->data);}

if(flag){//转回原来正负号

a->data=-a->data;}

if(a->data<0&&x%2!=0){//判断结果正负号，负数的奇数次方为负

c->data=-c->data;}

IN(c);}

**12.阶乘运算a!=c**

实现思路：将链表中的数值转换为长整型变量，然后利用一个for循环计算阶乘。在计算过程中，使用了辅助链表d、f、g来存储中间结果，并同时删除尾部多余的结点。最后，将计算得到的阶乘结果存储在链表c中。

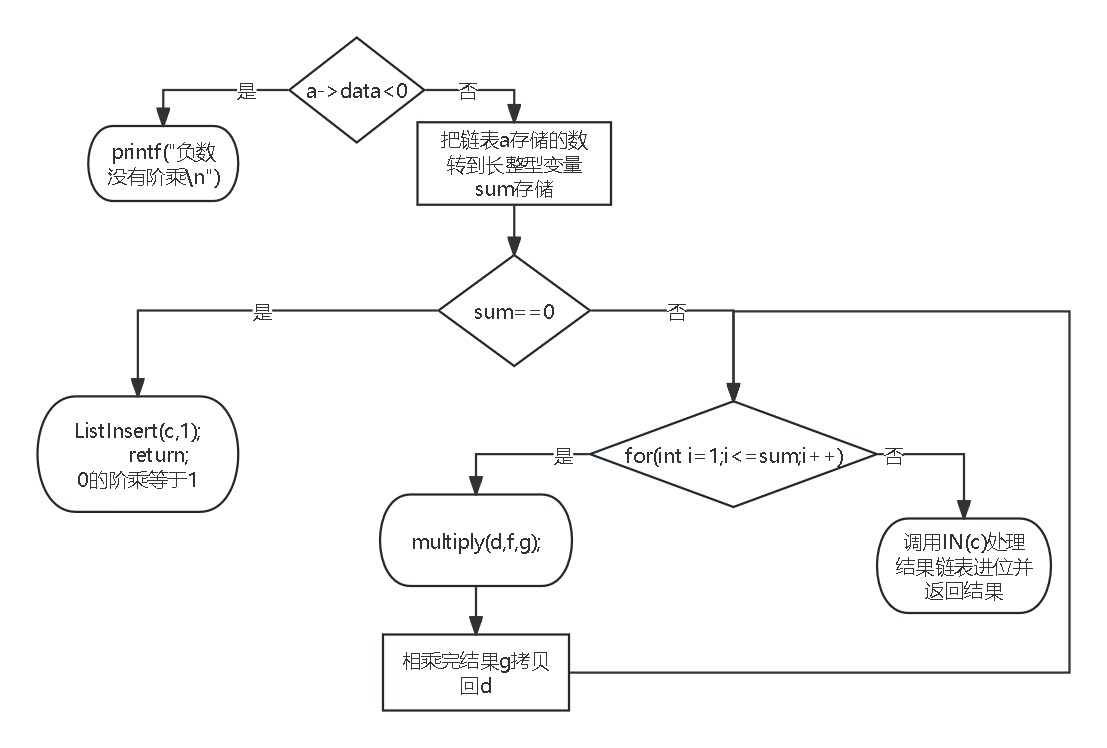


图12：阶乘运算Factorial函数流程图

**实现代码：**

void Factorial(LinkList a,LinkList &c){

LinkList p=a->prior;

long int sum=0;

LinkList d,f,g;

InitList(d);

InitList(f);

InitList(g);

ListInsert(d,1);

if(a->data<0){

printf("负数没有阶乘\n");

return;}

while(p!=a){//将链表存储的数转到长整型变量sum存储

sum=sum\*1000+p->data;

p=p->prior;}

if(sum==0){//0的阶乘等于1

ListInsert(c,1);

return; }

for(int i=1;i<=sum;i++){

ListInsert(f,i);

multiply(d,f,g);

if(i<sum){//删除尾部最高位多余的结点0000

while(d->data!=0){

if(d->data==0){

return;}

LinkList g=d->prior;

LinkList z=g->prior;

z->next=d;

d->prior=z;

free(g);

if(d->data>=0){

d->data--;}

else{d->data++; } }

InitList(d);

for(LinkList p=g->next;p!=g;p=p->next){

ListInsert(d,p->data);

}//g乘积拷贝回d，继续进入循环d\*f=g

while(f->data!=0){//删除尾部最高位多余的结点0000

if(f->data==0){return;}

LinkList g=f->prior;

LinkList z=g->prior;

z->next=f;

f->prior=z;

free(g);

if(f->data>=0){

f->data--;}

else{f->data++;}}

InitList(f);

while(g->data!=0){//删除尾部最高位多余的结点0000

if(g->data==0){return; }

LinkList t=g->prior;

LinkList z=t->prior;

z->next=g;

g->prior=z;

free(t);

if(g->data>=0){

g->data--;

} else{ g->data++;}}

InitList(g);}

for(LinkList p=g->next;p!=g;p=p->next){

ListInsert(c,p->data);} }//结果拷贝回c

IN(c);}

**13.主函数main**

实现思路：

先通过printf显示提示用户分别输入两个计算数据，然后调用getnum函数判断用户输入数据是否格式正确并成功构建链表，输入错误退出程序，输入正确则提示用户选择加减乘除运算方式，通过switch来调用用户选择的运算方式对应的add、subtract、multiply、divide函数，接着调用printdlist函数打印计算算结果。

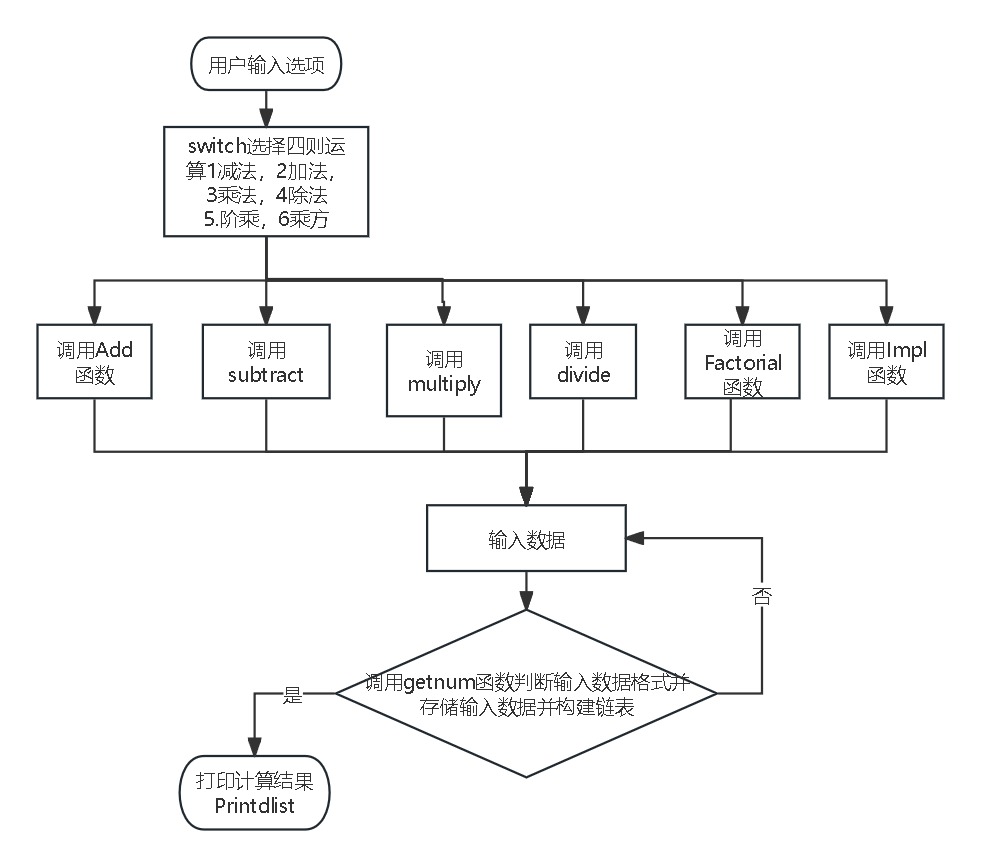


图13：main函数运行流程图

**实现代码：**

int main()

{ printf("22级软件工程卓越班\n----3222004465---- \n------陈婉瑜------\n\n");

printf("==========================\n");

printf("~~~~~~~~长整数运算~~~~~~~~\n");

printf("==========================\n");

while(1){

LinkList a,b,c;InitList(a);InitList(b);InitList(c);

int x=0;

printf("请选择要进行的计算操作\n");

printf("[1]======== 加法+ ========\n[2]======== 减法- ========\n[3]======== 乘法\* ========\n[4]======== 除法/ ========\n[5]======== 阶乘！========\n[6]======== 乘方^n========\n");

printf("\n");

char op=getch();

switch(op)

{case '1':

printf("======== 加法运算+ ========\n");

do{printf("请输入运算的第一个加数，输入完成按回车键\n");

int r= getnum(a);

if(r==0){

printf("ERROR!数字输入格式错误\n");

}}while(a->next==a);

do{printf("请输入运算的第二个加数，输入完成按回车键\n");

int r= getnum(b);

if(r==0){

printf("ERROR!数字输入格式错误\n");}

}while(b->next==b);

Printdlist(a);

printf("+");Printdlist(b);printf("="); Add(a,b,c);

Printdlist(c);destroyList(a);destroyList(b);destroyList(c);break;

case '2':

printf("======== 减法运算- ========\n");

do{

printf("请输入被减数，输入完成按回车键\n");

if(getnum(a)==0){

printf("ERROR!数字输入格式错误\n");}

}while(a->next==a);

do{printf("请输入减数，输入完成按回车键\n");

if(getnum(b)==0){

printf("ERROR!数字输入格式错误\n");}

}while(b->next==b);

Printdlist(a);printf("-");Printdlist(b);printf("=");subtract(a,b,c);

Printdlist(c);destroyList(a);

destroyList(b);destroyList(c);break;

case '3':printf("======== 乘法运算\* ========\n");

do{printf("请输入运算的第一个因数，输入完成按回车键\n");

if(getnum(a)==0){printf("ERROR!数字输入格式错误\n");

}}while(a->next==a);

do{printf("请输入运算的第二个因数，输入完成按回车键\n");

if(getnum(b)==0){printf("ERROR!数字输入格式错误\n");}}while(b->next==b);

Printdlist(a);

printf("\*");Printdlist(b);printf("=");multiply(a,b,c);

Printdlist(c);destroyList(a);destroyList(b);destroyList(c);break;

case '4':printf("======== 除法运算/ ========\n");

do{printf("请输入被除数，输入完成按回车键\n");

if(getnum(a)==0){

printf("ERROR!数字输入格式错误\n");}

}while(a->next==a);

do{

printf("请输入除数，输入完成按回车键\n");

if(getnum(b)==0){

printf("ERROR!数字输入格式错误\n");}

}while(b->next==b);

Printdlist(a);printf("/");Printdlist(b);printf("=");

if(abs(b->data)==1&&b->next->data==0)//0做除数

{ printf("ERROR!公式错误，0不可以做除数\n");

break;}

divide(a,b,c);Printdlist(c);

destroyList(a);destroyList(b);destroyList(c);break;

case '5':printf("======== 阶乘运算！ ========\n");

do{printf("请输入要进行阶乘运算的数，输入完成按回车键\n");

if(getnum(a)==0){ printf("ERROR!数字输入格式错误\n"); }

}while(a->next==a);

Printdlist(a);printf("!=");

Factorial(a,c);

Printdlist(c);

if(c->next==c) printf("[ERROR]!!!负数没有阶乘\n");

printf("\n");

destroyList(a);destroyList(c);break;

case '6':

printf("======== 乘方运算^ ========\n");

do{printf("请输入乘方运算的底数，输入完成按回车键\n"); if(getnum(a)==0){

printf("ERROR!数字输入格式错误\n");}

}while(a->next==a);

printf("请输入乘方运算的指数n，输入完成按回车键\n");

scanf("%d",&x);

Impl(a,c,x);Printdlist(a);printf("^%d=",x);

Printdlist(c);destroyList(a);destroyList(c);

printf("\n退出程序\n");system("pause");return 0;default:{

printf("您选择退出程序\n");

system("pause");

return 0;}}

system("pause");system("cls"); }}

# }四、调试分析

**（1）调试过程遇到的问题：**

1.在输入数据时，发现输入不是数字的数据比如s13;1234g23时没有报错，但是运算结果显示=0；因此在读取输入数据函数getnum中加入了判断语句

if(!(i==0&&(ss[0]=='-'||ss[0]=='+'))&&(!(ss[i]>='0'&&ss[i]<='9')))

return 0;

当输入数据错误，无法构成链表存储数据并返回0，接着在main函数调用时加入判断if(getnum(a)==0){

printf("数字输入格式错误\n");

return 0;

}

若输入数据有误就会立马弹出报错并退出程序

2.输出数据时发现数字前有无效的0，例如00012345；000283494；因此在进位函数IN中又添加上了删除尾结点无用的存储数据0的结点代码：删除链表尾部高位多余的0

while(L->prior->data==0&&abs(L->data)>1){

if(L->data==0)return;

LinkList a=L->prior;

LinkList b=a->prior;

b->next=L;

L->prior=b;

free(a);}

修改过后输出的数据就正确了，开头没有多余的0

**（2）算法的时空分析**：

双向循环链表尾部结点插入和删除的时间复杂度为O(1),遍历链表查询的时间复杂度为O(n),初始化的时间复杂度O(1)

# 五、用户使用说明

打开程序后会进入如下图所示的菜单界面，提示用户选择计算操作

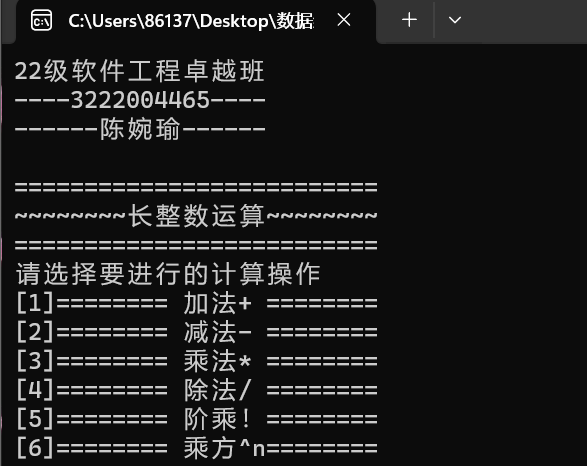


图14：程序使用界面（选择操作）

用户可根据想要使用的运算方式输入对应序号（1~6），若输入其他序号则会提示“您选择退出程序”并退出程序。例如想进行加法运算，可按下数字1，然后屏幕提示输入数据，输入完成后会显示计算结果，如下图所示

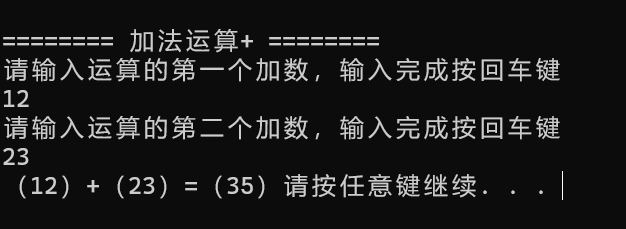


图15：程序使用界面（运算结果）

此时用户按任意键后将会清屏，提示输入数据，这时用户可继续重复上述操作输入另一组数据进行计算。

若用户输入数据混入非数字数据，则会弹出提示ERROR!数字输入格式错误，并再次提示输入数据，这时用户继续输入正确数据即可计算运行程序完成计算，如下图所示

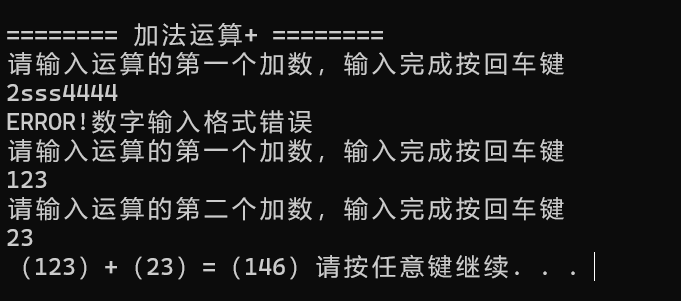
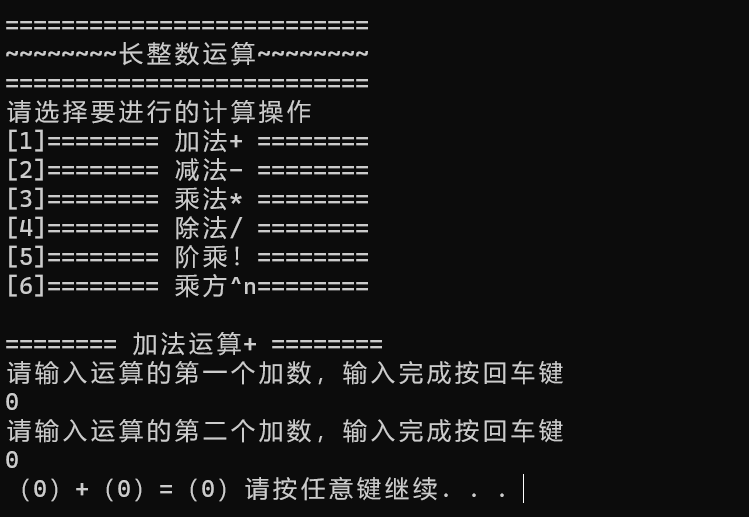
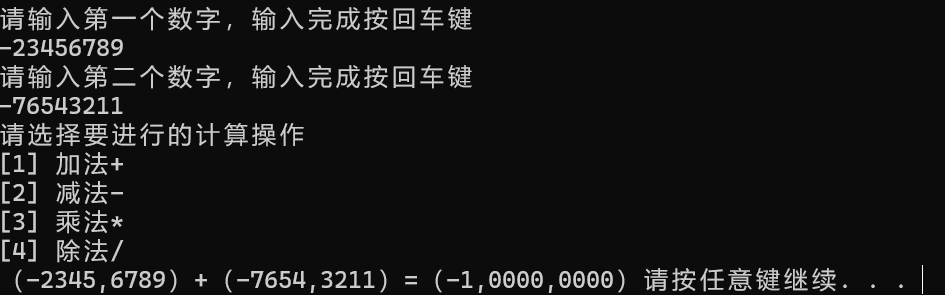


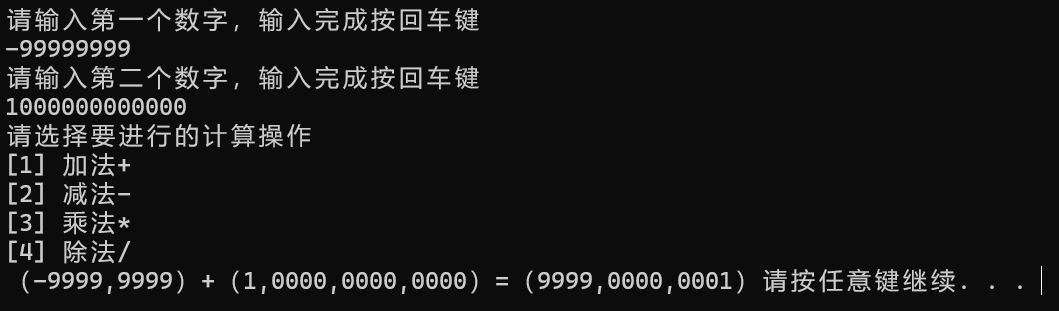
图16：数据输入错误提示1

# 六、测试结果

**1．加法功能测试：**







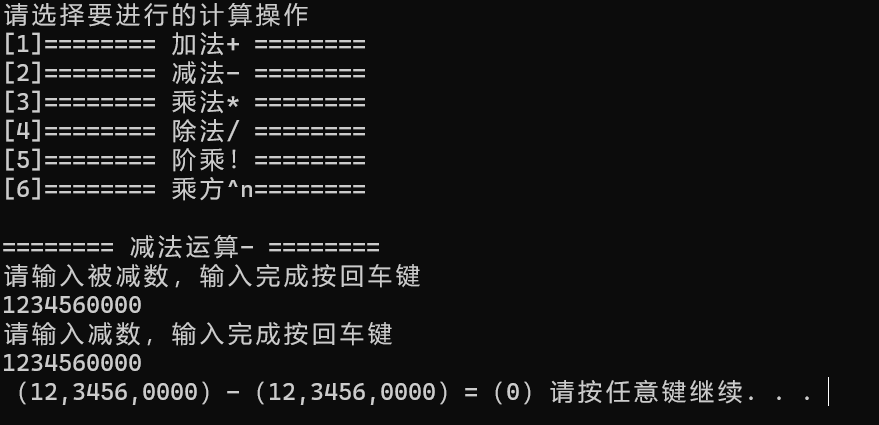








**2.减法功能测试：**

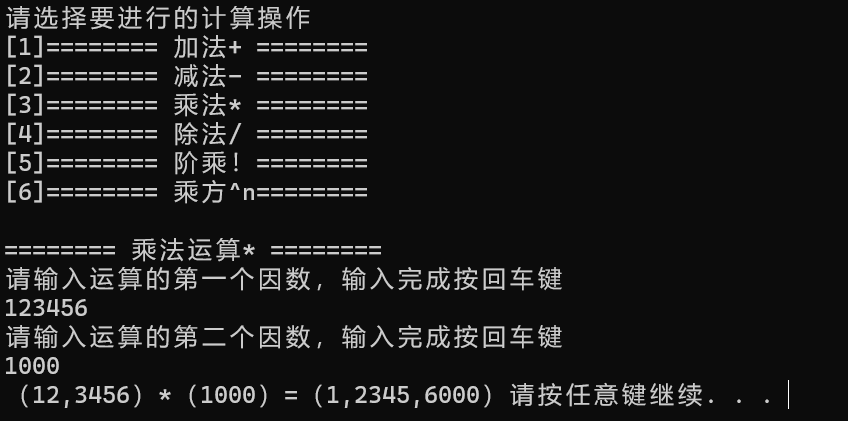








**3.乘法功能测试：**

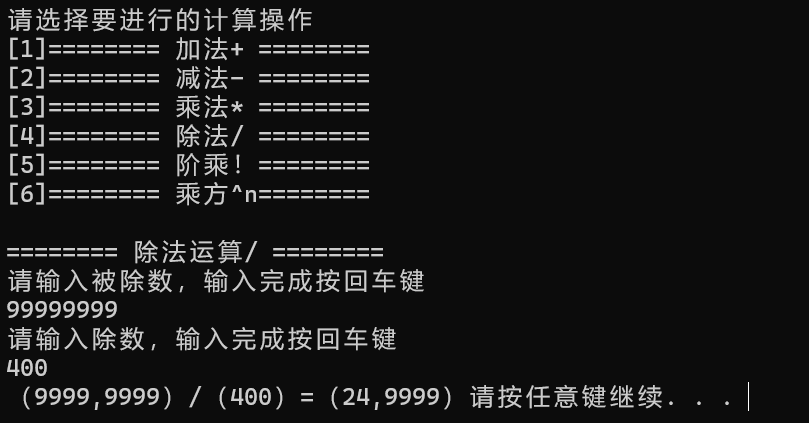








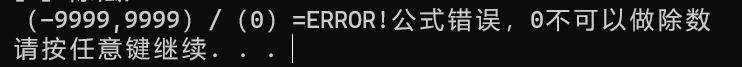
**4.除法功能测试：**





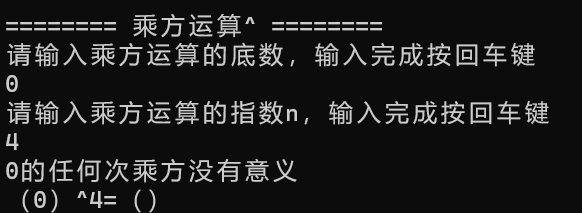








**5.乘方功能测试：**







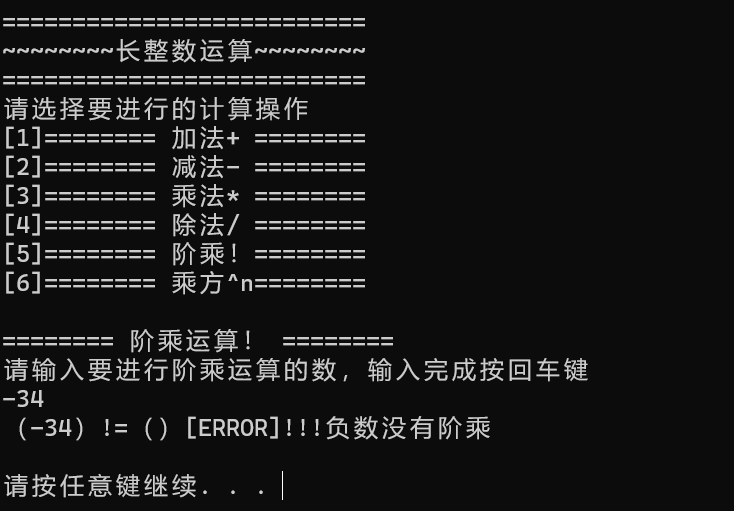


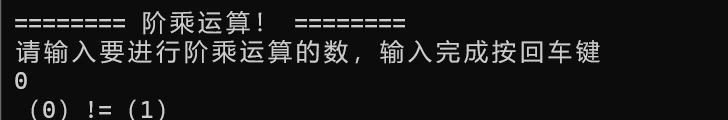






**6.阶乘功能测试：**









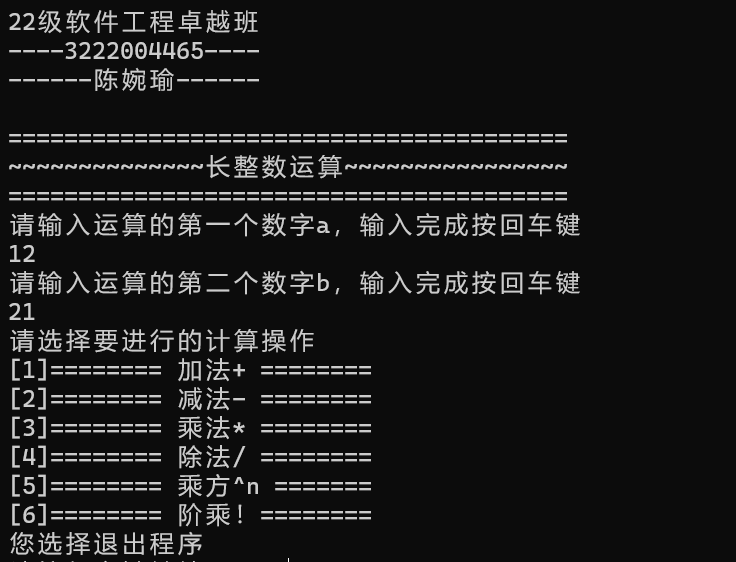
**7.输入数据错误测试**：

123ss21; 应输出“ERROR!数字输入格式错误”



**8.选择操作时除（1~6）外的选项测试：**

按下数字0 ；输出“您选择退出程序”并退出程序



上述测试中，加减乘除功能测试将正负数，负数，正数之间的都测试了

乘法和除法还测试了其中一个数为0的情况

除法中还测试了0作为除数时没有意义无法计算的反馈

乘方功能测试了0和1分别作底数和指数，底数为负数时奇数次和偶数次的情况

阶乘功能测试了0和1，17的阶乘，还测试了负数阶乘-34！无意义的反馈情况

还测试了输入数据有误，选择操作选项有误时的反馈

所有测试结果均正确符合预期

# 七、附录

长整数四则运算课设.cpp

长整数四则运算课设.exe