

数据结构课程设计

**2020/2021(1)**



实验题目 大整数的运算

学生姓名 单启涛

学生学号 201906061403

学生班级 软工1901

任课教师 毛国红

提交日期 2021年1月10日

**计算机科学与技术学院**

**大整数的运算 实验报告**

1. **实验的内容**

【问题描述】

密码学分为两类密码：对称密码和非对称密码。对称密码主要用于数据的加/解密，而非对称密码则主要用于认证、数字签名等场合。非对称密码在加密和解密时，是把加密的数据当作一个大的正整数来处理，这样就涉及到大整数的加、减、乘、除和指数运算等，同时，还需要对大整数进行输出。请采用相应的数据结构实现大整数的加、减、乘、除和指数运算，以及大整数的输入和输出。

【基本要求】

1. 要求采用链表来实现大整数的存储和运算，不允许使用标准模板类的链表类(list)和函数。 同时要求可以从键盘输入大整数，也可以文件输入大整数，大整数可以输出至显示器，也可以输出至文件。大整数的存储、运算和显示，可以同时支持二进制和十进制，但至少要支持十进制。大整数输出显示时，必须能清楚地表达出整数的位数。测试时，各种情况都需要测试，并附上测试截图；要求测试例子要比较详尽，各种极限情况也要考虑到，测试的输出信息要详细易懂，表明各个功能的执行正确。
2. 要求大整数的长度可以不受限制，即大整数的十进制位数不受限制，可以为十几位的整数，也可以为500多位的整数，甚至更长；大整数的运算和显示时，只需要考虑正的大整数。如果可能的话，请以秒为单位显示每次大整数运算的时间。
3. 要求采用类的设计思路，不允许出现类以外的函数定义，但允许友元函数。主函数中只能出现类的成员函数的调用，不允许出现对其它函数的调用。
4. 要求采用多文件方式：.h文件存储类的声明，.cpp文件存储类的实现，主函数main存储在另外一个单独的cpp文件中。如果采用类模板，则类的声明和实现都放在.h文件中。
5. 不强制要求采用类模板，也不要求采用可视化窗口；要求源程序中有相应注释。
6. 要求采用Visual C++ 6.0及以上版本进行调试。
7. **运行环境**

大整数的运算在Visual Studio2017平台下开发，操作系统：Windows 10。

硬件环境：

处理器：Intel(R) Core(TM) i5-6300HQ CPU @ 2.30GHz 2.30 GHz

内存：12.00GB

系统类型：64位操作系统，基于x64的处理器

1. **实验课题分析**

**3.1 系统总体设计**

实现大整数运算需要实现加、减、乘、除、指数运算5个基本功能，实现这些功能需要在链表上进行操作，题目中还要求能够以秒为单位显示每次大整数运算的时间。所以我在该项目中设计了三个类和一个主调函数，分别为链表操作类Calculate，大整数链表模板类List，节点类node主调函数main。

功能概要如下：

1. Calculate类：

主要对存储大整数的链表进行操作处理。如检查链表存储的数据是否合法，去链表存储数据的前置0，比较两链表的大小，设置运算的进制，基于进制数的两链表的加法运算，减法运算，乘法运算，除法运算，指数运算等。

1. List模板类：

将输入的数字字符串转化为双向链表的形式存储。含有等号运算符重载，获得字符串的长度，插入节点到尾或首，删除尾或首节点，显示链表存储的大整数，将链表数据保存到文件，构造和析构等函数。

1. node类：

含有前后的双指针。作为链表的构成部分。

1. main函数

主要用来实现菜单功能，供用户选择操作并调用相应的函数以实现操作。会对用户输入的每一项信息进行检验，保证得到的指令正确。并可输出运算时间。

**3.2 系统功能设计**

主要功能及说明如表1所示。

|  |  |  |
| --- | --- | --- |
| 编号 | 功能 | 简要说明 |
| 1 | 加法运算 | 实现两个大整数相加并得到结果 |
| 2 | 减法运算 | 实现两个大整数相减并得到结果 |
| 3 | 乘法运算 | 实现两个大整数相乘并得到结果 |
| 4 | 除法运算 | 实现两个大整数相除并得到结果 |
| 5 | 指数运算 | 根据指数和底数计算得到结果值 |
| 6 | 选择是否输出至文件 | 用户可以选择是否向文件输出  计算结果 |
| 7 | 从文件读取数据 | 用户可以选择是否从文件读入数据作为要运算的数据，而运算符从键盘输入 |
| 8 | 进制转换 | 用户可通过该功能切换从二到十的任意进制，初始默认为十进制 |

表1 系统功能说明表

**3.3 类的设计**

本系统主要实现了链表操作类Calculate，大整数链表模板类List，节点类node这三个个类，其中，Calculate类在实现大整数的运算过程中对链表的操作主要通过调用List类的成员函数完成

1. **node类：**

data用来存储数据，prev为指向前一个节点的指针，next为指向下一个节点的指针。

1. **List模板类：**

**List含root哨兵，tail尾节点。**

表2为List类中的成员函数及简要说明。

|  |  |
| --- | --- |
| 成员函数名 | 简要说明 |
| **List()** | 构造函数，初始化链表 |
| ~List() | 析构函数，销毁链表 |
| List(const string str) | 用输入的字符串类型的大整数创建链表 |
| void insert( T member) | 向链表末尾添加节点 |
| void preInsert(T member) | 向链表前面添加节点 |
| void erase( int boolLast) | 删除节点(尾或首） |
| void display() | 显示链表中大整数值 |
| void save(ofstream &fout) | 将结果保存到文件 |
| int getlength() | 返回链表的长度 |
| void operator=(const List<T>&s) | 等号运算符重载 |

表2 List类成员函数说明表

**关键成员函数实现说明：**

1. List(const string str)//用输入的字符串类型的大整数创建链表

该函数将输入的类为字符串的大整数存储在链表中，逐位取出字符转换成整形数据并在链表中新建节点，将其添加到链表中。主要代码如下：

root = new node<T>;

int n = str.length();

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

{

insert(int(str[i] - '0'));

}

1. void insert( T member)//向链表末尾添加节点

先新建一个data为member的节点，然后插入到链表后面。

流程图如图1所示

新建data为number的节点

长度是否为零

不是

是

Root和新节点建立双向链接，tail指向新节点

添加到尾部，更新tail

1. void preInsert(T member)//向链表前面添加节点

先新建一个data为member的节点，然后插入到链表前面。主要代码如下：

node<T>\* a = new node<T>;

a->data = member;

if (getlength() == 0)

{

root->next = a;

a->prev = root;

tail = a;

}

else

{

a->next = root->next;

root->next->prev = a;

root->next = a;

a->prev = root;

}

length++;

流程图如图2所示

新建data为number的节点

Root和新节点建立双向链接，tail指向新节点

添加到root->next的前面，更新root的next

长度是否为零

不是

是

1. void erase( int boolLast)//删除节点(尾或首）

如果booLast为1则删除尾节点，为0则删除root（哨兵）的下一个节点。

流程图如图3所示

root指向空删除tail

Root的next往前移，删除第一个有效节点

boolLast是否为0

判断链表长度是否大于1

tail往前移，删除尾节点

判断链表长度是否大于1

Root的next指向空删除tail

否

是

否 否

是 是

1. void operator=(const List<T>&s)//等号运算符重载

先将等号左边链表析构掉，然后建一个数据和等号右边的链表一样的链表。主要代码如下：

node<T>\* p = root->next;

while (root->next)

{

root->next = p->next;

delete p;

p = root->next;

}

tail = NULL;

length = 0;

node<T>\* b = s.root->next;

while(b)

{

insert(b->data);

b = b->next;

}

1. **Calculate类：**

表3为Calculate类中的成员函数及简要说明。

|  |  |
| --- | --- |
| 成员函数名 | 简要说明 |
| List<int>& add(List<int> &s1, List<int> &s2) | 加法运算 |
| List<int>& sub(List<int> &s1, List<int> &s2) | 减法运算 |
| List<int>& multiply(List<int> &s1, List<int> &s2) | 乘法运算 |
| List<int>& division(List<int> &s1, List<int> &s2) | 除法运算 |
| void exOperation(const char \*, const char \* , int, int) | 指数运算 |
| void preZeroDelete(List<int> &s1) | 去链表前面的零节点 |
| bool compare(List<int> &s1,List<int> &s2) | 比较函数（若s1>=s2则返回1,否则返回0） |
| bool check(const string str) | 检查输入是否正确 |
| void trBinary(int n) | 转换进制为n进制 |

表3 Calculate类成员函数说明表

**关键成员函数实现说明：**

1. List<int>& add(List<int> &s1, List<int> &s2)//加法

采用小学数学加法竖式计算的方法，从后往前逐位相加，大于进制数就往前进一位的方法。

向前进一位

从后往前逐位把s2加到s1上

从后往前逐位把s1加到s2上

某一位相加后是否大于等于进制数

向前进一位

循环进行完，Return s1

循环进行完，Return s2

S1和s2去前置零

S1是否比s2长

某一位相加后是否大于等于进制数

流程图如图4所示

否

是

否 否

是 是

1. List<int>& sub(List<int> &s1, List<int> &s2)//减法

先比较大小，然后采用小学数学加法竖式计算的方法用大的数建小的数，从后往前逐位相减不够减就借位，最后确定返回结果。

流程图如图5所示

从后往前逐位把s2减s1

循环进行完，对s2去前置零，加负号Return s2

某一位相减是不是不够减

向借进一位

S1和s2去前置零

S1是否比s2大

向前借一位

从后往前逐位用s1减s2

循环进行完，对s1去前置零，Return s1

某一位相减是不是不够减

否

是

否 否

是 是

1. List<int>& multiply(List<int> &s1, List<int> &s2)//乘法

采用小学数学乘法竖式计算的方法，从后往前逐位相乘，大于进制数就往前进一位的方法。新建一个链表用来存储每一层乘后的结果，新建另一个链表用来存储最终结果。

流程图如图6所示

S1和s2去前置零

向前进位

将链表d补零后加到链表c上，用s1乘s2的下一位

循环进行完，对c去前置零，将c赋值给s1，return s1.

某一位相乘是不是大于进制数

用s1的每一位从后往前和s2的某一位相乘，结果保留到链表d

否

是

1. List<int>& exOperation(List<int> &s1, List<int> &s2)//指数运算

建立在乘法的基础上，执行指数-1次底数的乘法，每次执行完都指数减一。最后累乘的结果即为最后的结果。

流程图如图7所示

S1和s2去前置零，新建一个大整数为1的链表b和两个复制s1和s2的链表a,k

循环进行完，return s1.

乘法函数和减法函数，执行指数-1次底数的乘法。S1和a相乘一次，k减一次b

1. List<int>& division(List<int> &s1, List<int> &s2)//除法

除法是建立在减法的基础上，先把除数乘以两数的位数之差的10的乘方倍。被除数能够减去除数的个数，即为商，剩下的即为余数，再把余数当做被除数，除数不变，如此循环，把每个循环的得到的商乘以除数扩大的倍数，相加。直到所得的余数小于除数结束运算。

流程图如图8所示

S1和s2去前置零，新建一个大整数为0的链表num用来存最终结果和一个空链表m用来存储s2放大后的数

条件为真的while循环

S1是否小于s2

将减的次数插入到num的后面

将S2赋值给m,然后在m后面添适当个数的0，然后s1一直减m直到小于m

对num后面补零，然后对num去前置0，将num赋值给s1，return s1.

是

否

1. void preZeroDelete(List<int> &s1)//去前置零

将大整数前面多余的零去掉。

主要代码如下：

while (s1.root->next&&s1.root->next->data == 0)

{

s1.erase(0);

}

if (s1.getlength() == 0)

s1.insert(0); //若全为零则补一个零

1. bool check(const string str)//检查输入是否合法

逐位遍历，若出现一位字符不在0~（进制数-1）之间则返回0；否则返回1。

主要代码如下：

int a = str.length();

for (int i = 0; i < a; i++)

{

if (str[i]-'0'<0 || str[i]-'0'>=radix)

return 0;

}

return 1;

1. bool compare(List<int> &s1,List<int> &s2)//比较s1和s2（若s1>=s2则返回1,否则返回0）

先去前置0，然后比较长度长度长的大，若长度相等则从左到右逐位比较。

主要代码如下：

preZeroDelete(s1);

preZeroDelete(s2);

if (s1.getlength() > s2.getlength())

return 1;

else if (s1.getlength() < s2.getlength())

return 0;

else

{

node<int>\*a = s1.root;

node<int>\*b = s2.root;

while (a->next)

{

if (a->next->data > b->next->data)

return 1;

else if (a->next->data < b->next->data)

return 0;

else

{

a = a->next;

b = b->next;

}

}

return 1;

}

**3.3 主程序的设计**

**主要运用三个类里面的函数和数据来完成文件读写工作和计算工作。对用户的每一个输入都有输入检错机制（输入不合要求时重输）。可输出计算所用时间。**

1. **调试分析**
2. **技术难点分析**

除法运算实现难点：

在除法运算中，可以采用逐个减的方法，但是考虑到这样效率太低，先把除数乘以两数的位数之差的10的乘方倍。被除数能够减去除数的个数，即为商，剩下的即为余数，再把余数当做被除数，除数不变，如此循环，把每个循环的得到的商乘以除数扩大的倍数，相加。直到所得的余数小于除数结束运算。

1. **调试错误分析**
   * **问题1：**

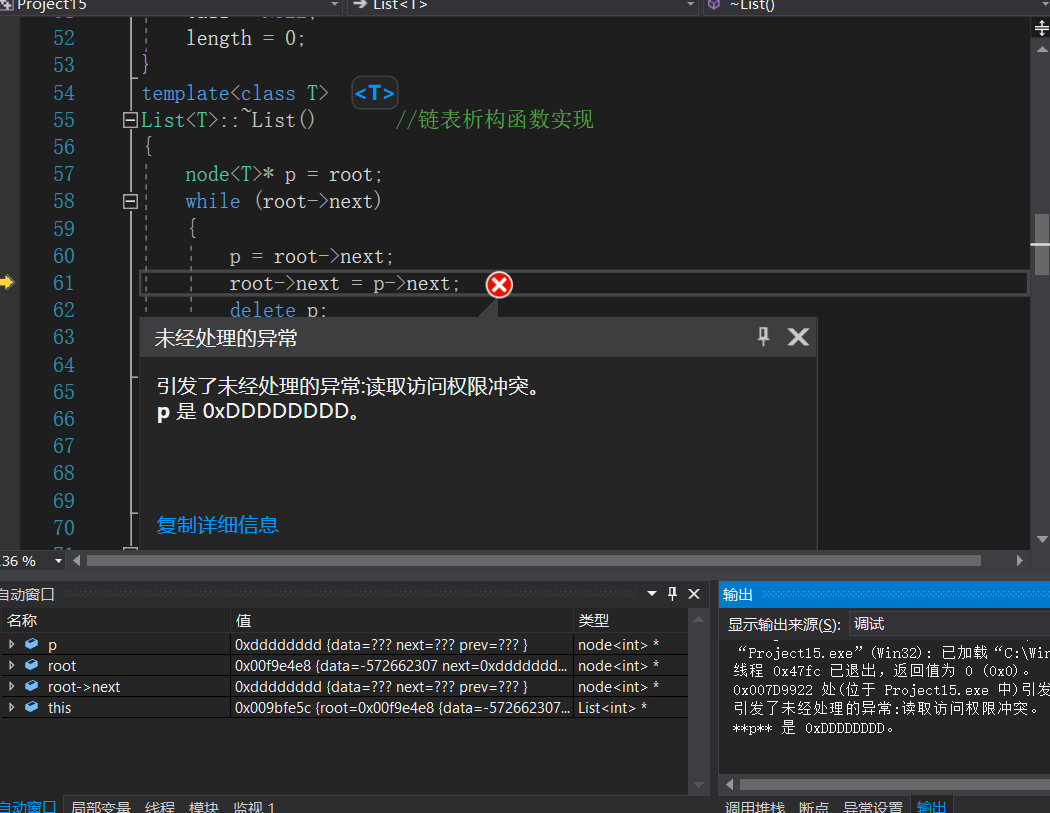
**问题描述：**

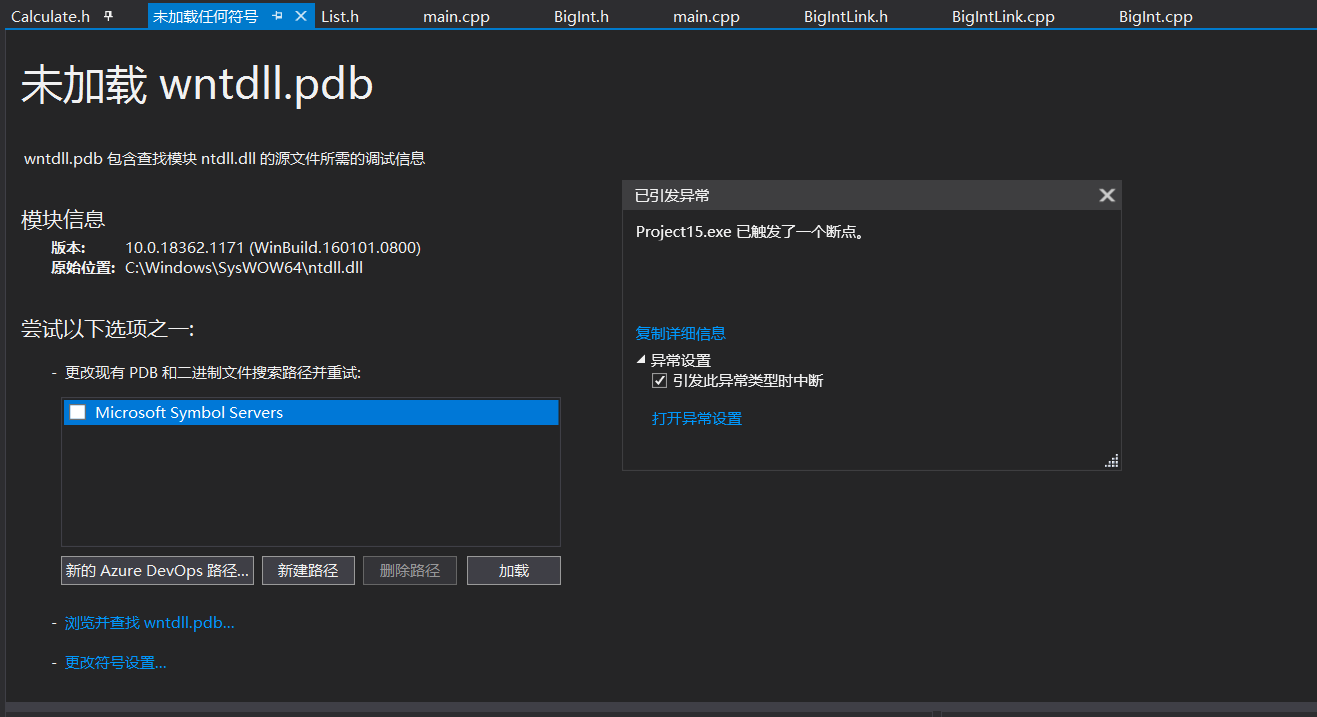
调试时老是出现读取权限访问冲突报错，不是野指针就是空指针。

**解决方法：**

通过单步调试和检查代码发现在节点删除函数中没有及时更新链表长度。然而运行的时候还是出错。我就请教同学，他建议我将函数参数类型改为引用，并仔细给我解释了原因。我按照他的建议改过之后确实把问题解决了。

修改前：





* + **问题2：**

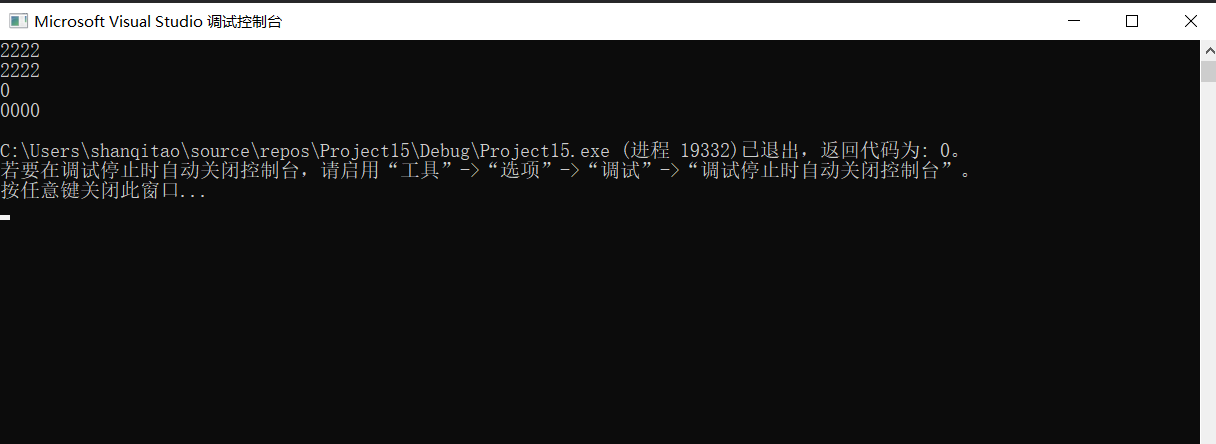
**问题描述：**

调试乘法时结果出现了多个零的情况。

**解决方法：**

写一个去前置零的函数，在输出结果前，去掉结果前面无用的零。

修改前：



* + **问题3：**

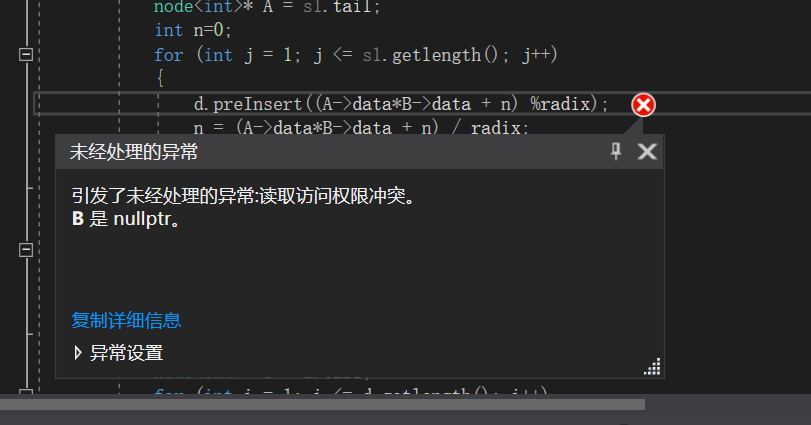
**问题描述：**

写完指数函数，调试时又出现了读取权限冲突。

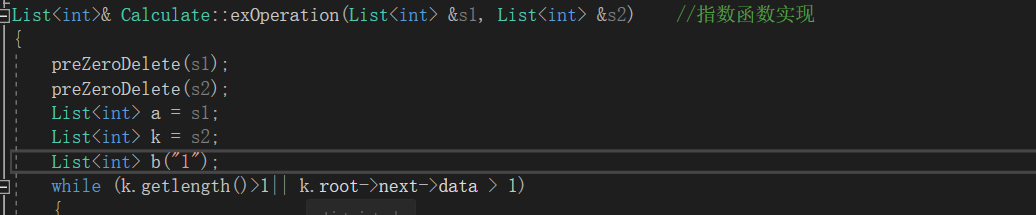
**解决方法：**

写指数函数时为了方便调用乘法函数我写了一个链表类的赋值重载函数。但在运行时老是出现读取权限冲突问题。我检查了几个小时，又请同学看了几个小时，最后自己又反复测试，单步调试，逐渐缩小范围，终于找到了根源所在。原来我调用赋值重载函数的方法不对。

修改前：



错误源码



应改为：

List<int> a;

A=s1;

List<int> k;

K=s2;

* + **问题4：**

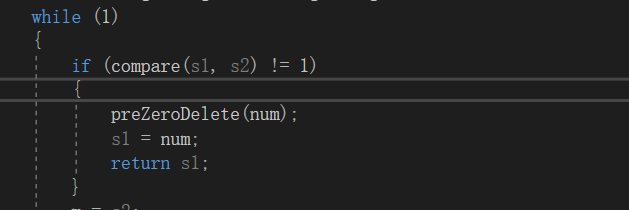
**问题描述：**

除法在调试时，如果是800/400则结果输出了2而不是200。

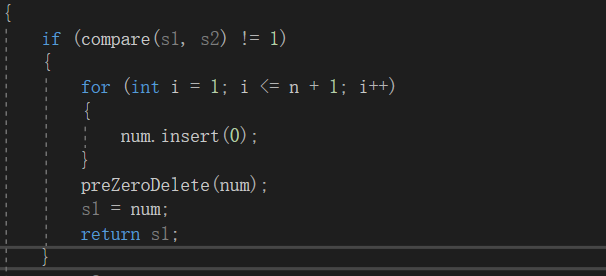
**解决方法：**

检查函数后，发现可以在返回结果前，忘链表后面添上相应个数的0。

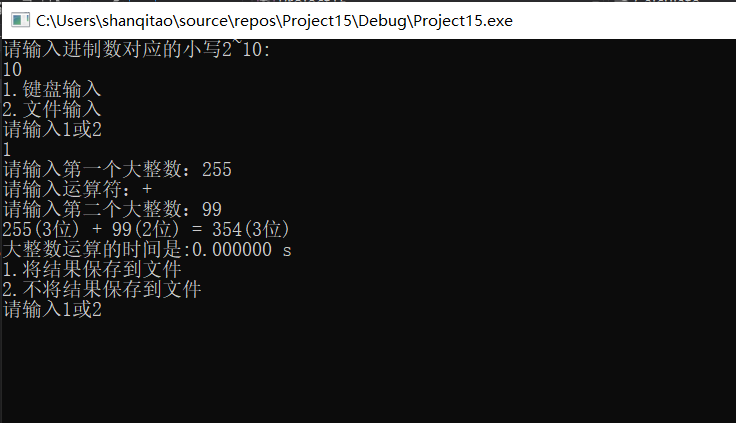
修改前：

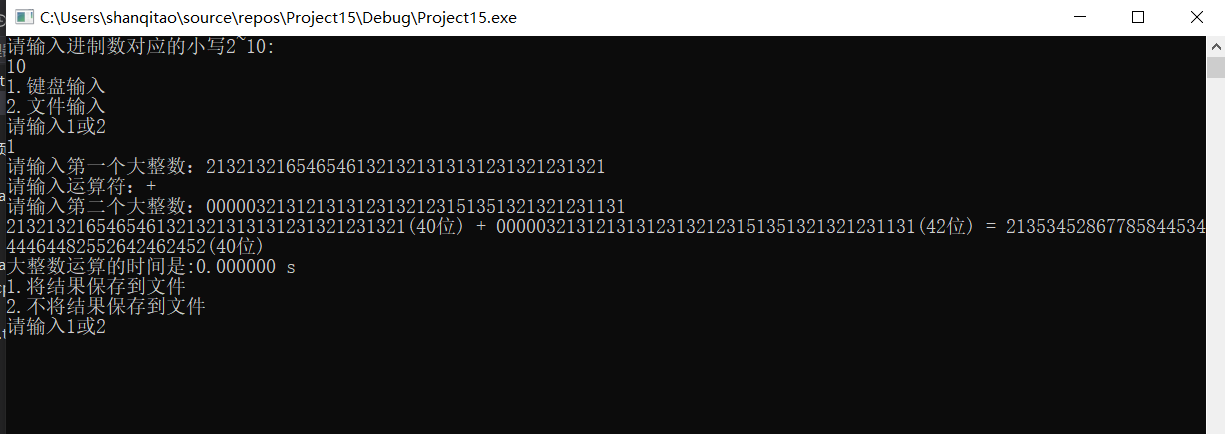


修改后：

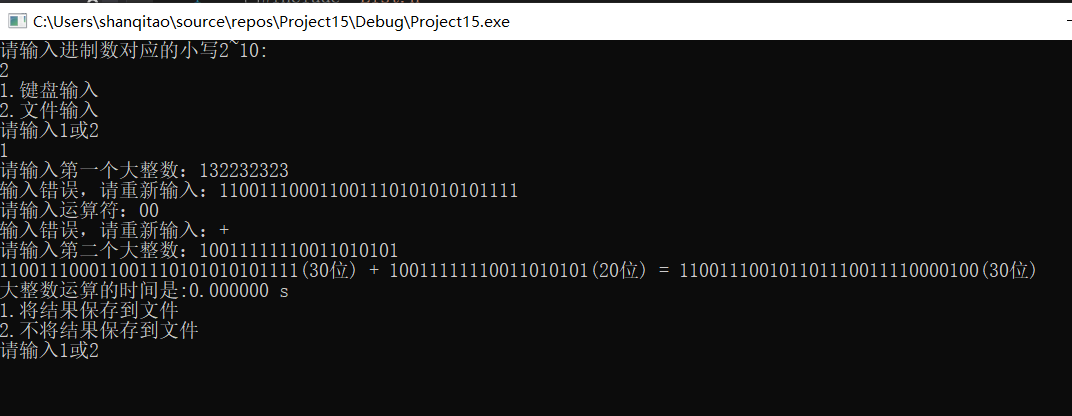


1. **测试结果分析**
2. **加法运算**

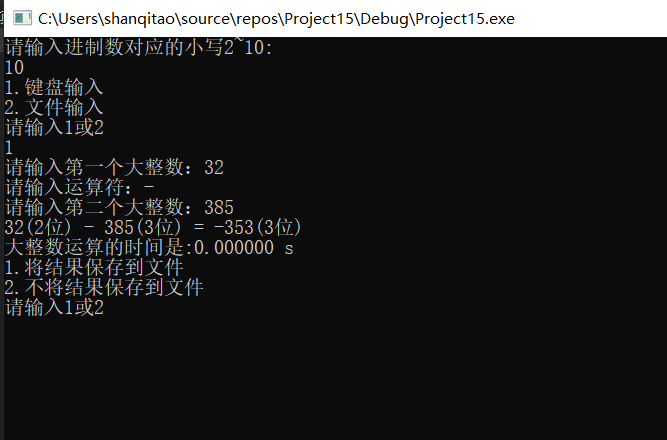


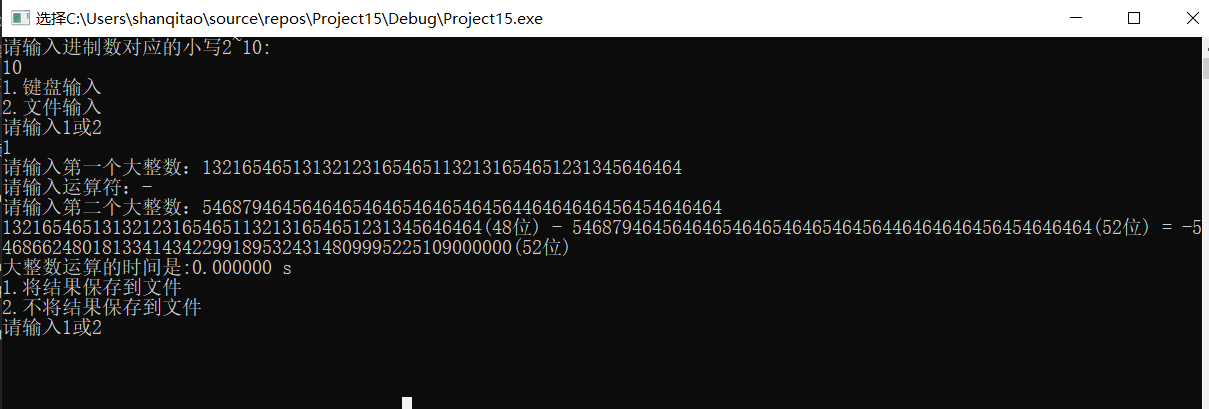


二进制加法：

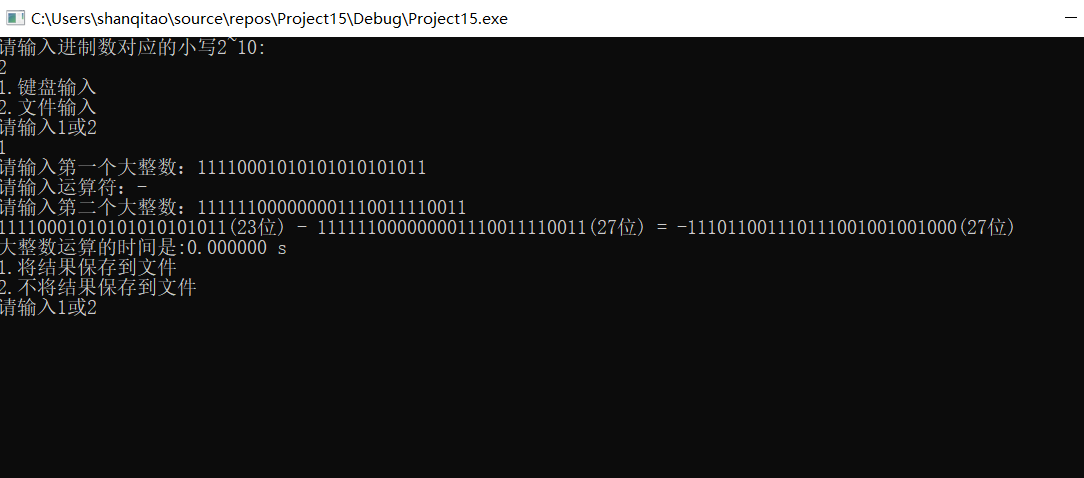


1. **减法运算**

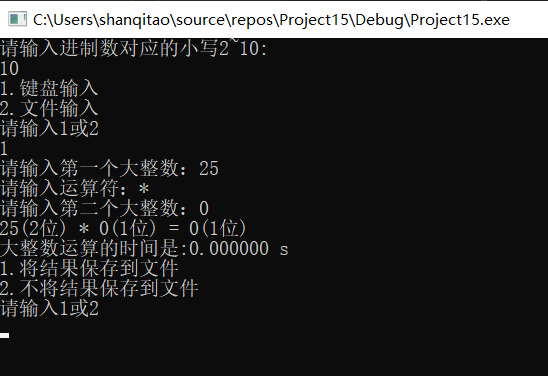




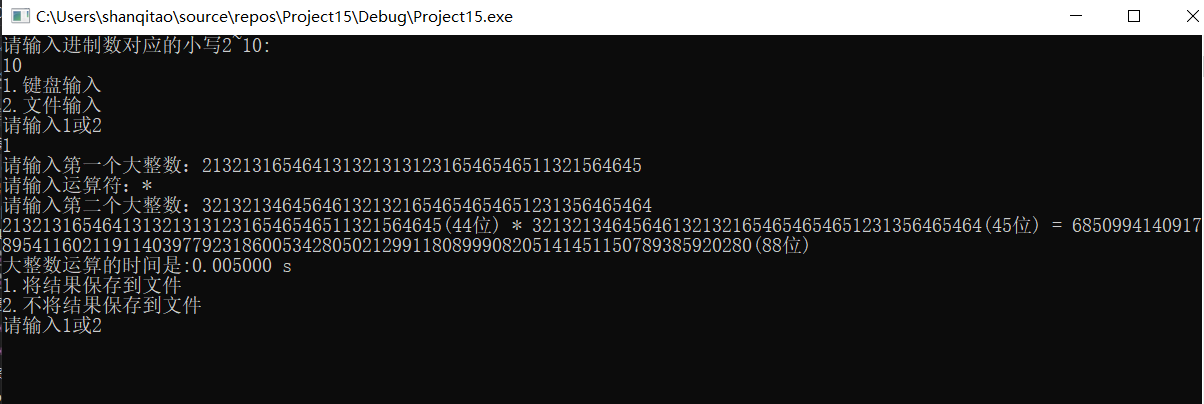
二进制减法：



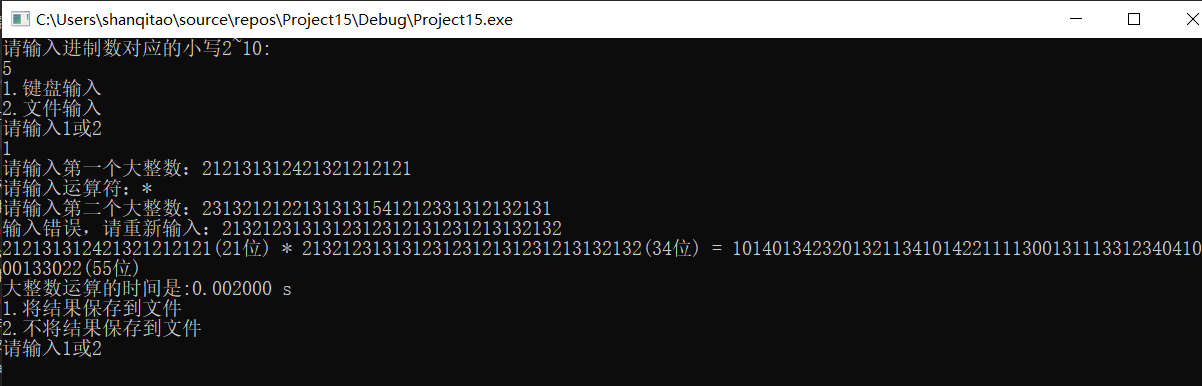
1. **乘法运算**



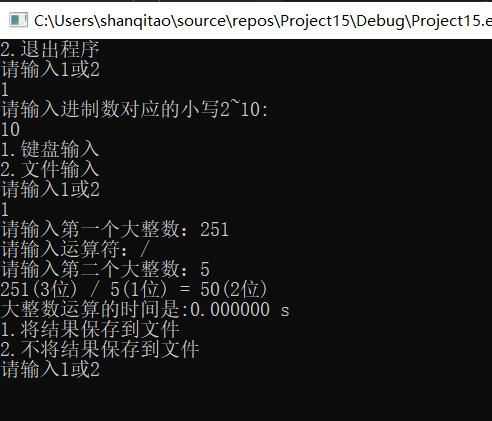
****

****

**五进制乘法：**

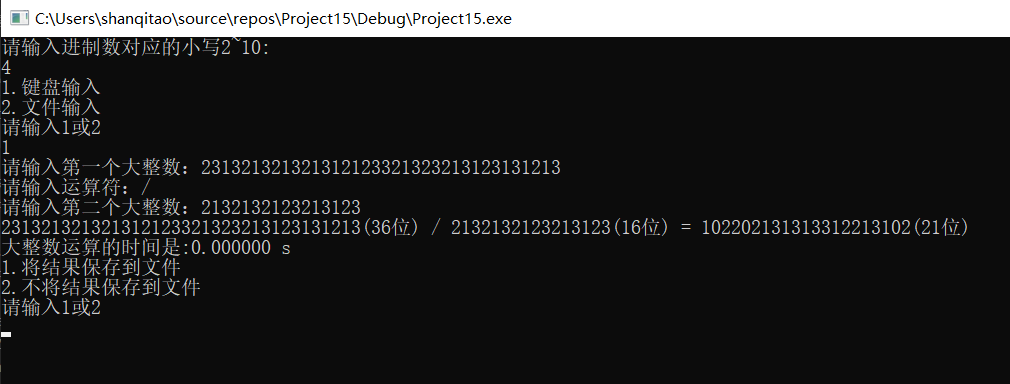
****

1. **除法运算**

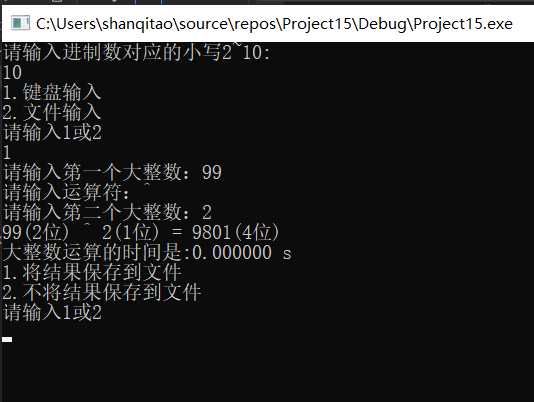
****

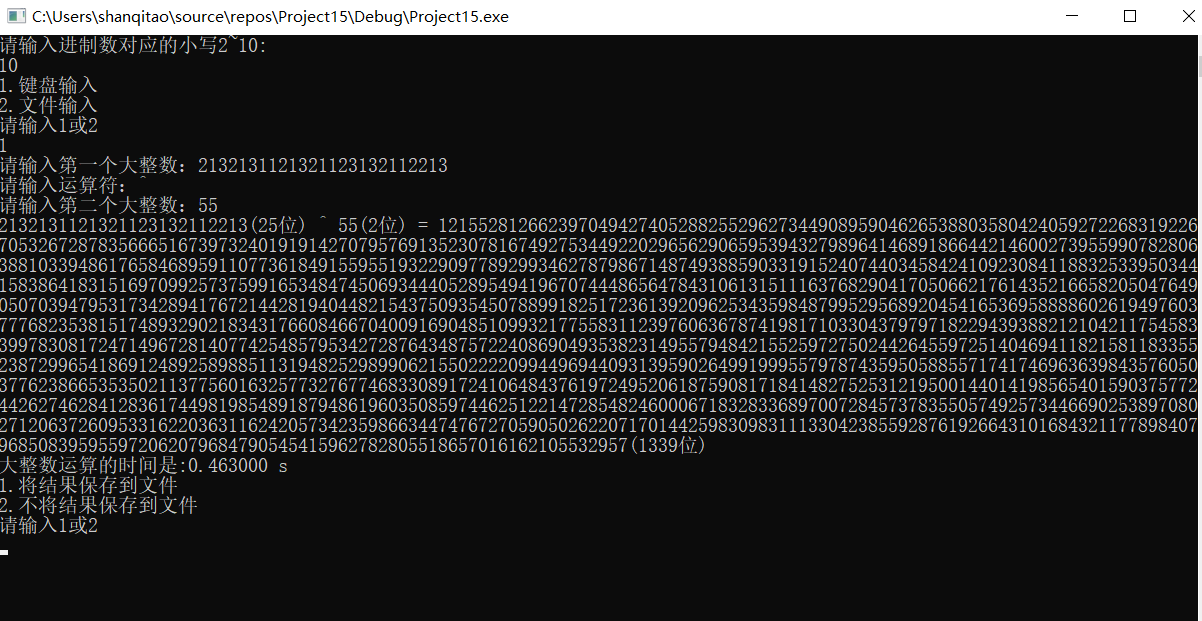
****

四进制除法：

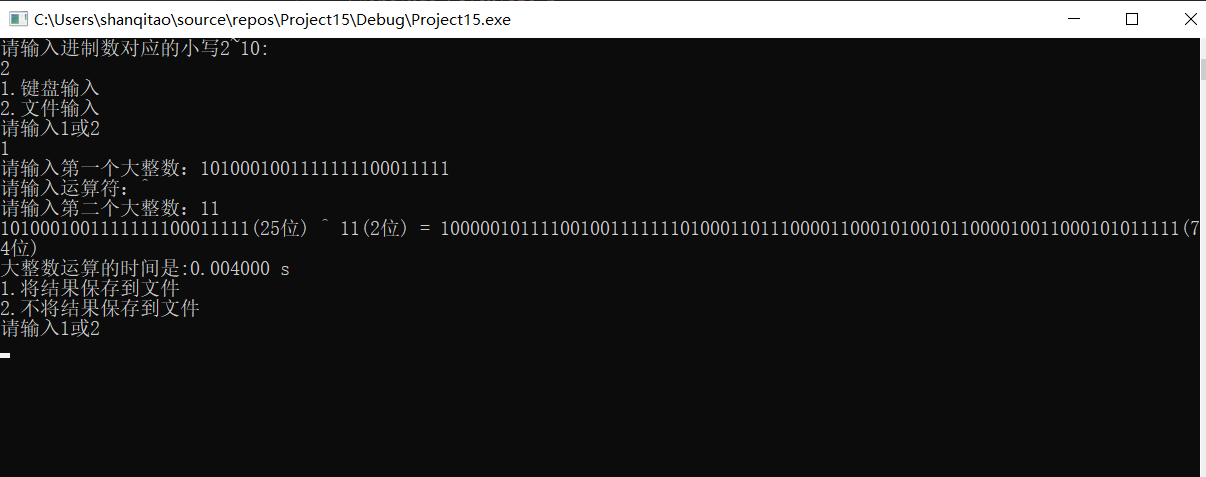


1. **指数运算**

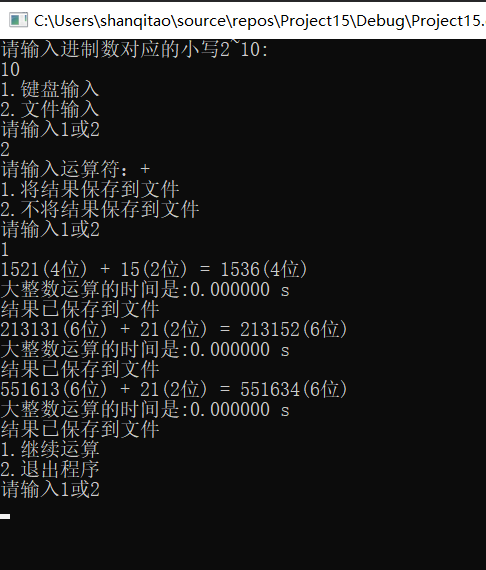




二进制指数运算



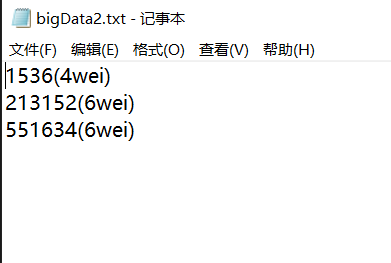
1. **文件读写**



读入的文件内容：



写出的文件内容：



1. **实验总结**

优点：

1.能兼容二进制到十进制之间的任意进制计算；2.链表设计为模板类；3.对用户输入的每一个信息都有检错机制；4.文件读时支持多组数据计算且用户可选择计算符。

缺点：

1.没有把链表类放在计算类里面；2.计算的函数在计算时会更改原链表的值；3.用户输入繁琐。

收获及体会：

1.自己没有提前对整个项目构思一遍就开始做，导致后边要对前面写过的函数进行大的改动，严重影响了效率。比如我刚开始设计的加法和减法和乘法函数都是void型，都是在函数内部将结果输出。然而到设计指数函数时，我发现要是直接写代码会特别长，而如果能够调用乘法和减法函数则非常简洁，导致我不得不对前面的函数进行大的更改。还有我当时没有设计尾指针，到后面用到的时候才想起来，又要对前面的代码进行大的更改。

2.还有就是没考虑到后面的内容，就针对眼前的方便设计函数，导致后面的大麻烦。如我设计加法函数和减法函数时，为了提高运行效率，直接更改了原链表，在原链表的基础上生成了结果。导致后来需要多个计算函数配合才能计算结果时，产生了不必要的错误和麻烦。

3.最后就是感觉自己的基础知识不扎实。比如那个等号运算符重载函数调用错误导致产生了空指针，我找了几个小时才找出来错误的根源。

4.这次课设大大的丰富了我的代码经验，也让我看到了自己的不足，今后我会更加努力，弥补不足。