湖南大學

数据结构

课程实验报告

题　　目：线性表的应用

学生姓名　李晓彤

学生学号　201726010128

专业班级　软件 1701 班

完成日期　2018.11.6

一、 需求分析

0 问题描述

在数学上，一个一元 n 次多项式 $P_n(x)$可按降序写成：

$$P_n(x) = p_n x^n + p_{n-1}x^{n-1} + \cdots + p_1 x + p_0$$

它是由 n＋1 个系数唯一确定。因此，在计算机里它可以用一个线性表 P 来表示：

$$P=(P_n, P_{n-1}, \cdots, P_1, P_o)$$

一元多项式的运算包括加法、减法和乘法，而多项式的减法和乘法都可以用加法来实现。

线性表ADT的应用，基于线性表设计和实现两个一元多项式的加法运算。

要求：用有序线性表表示一元多项式。表内元素按照多项式的次数递减的次序排列。

并且输入项的个数范围为 1~100。

1.问题分析

问题中的需求：

1.按有序线性表表示一元多项式。

2.基于线性表设计和实现两个一元多项式的加法运算。

3.输入不用按递减次序输入，但是到线性表中要按递减次序存储

功能：

1.乱序输入后要先存入，再按递减次序存储。

2.存储完之后输出存好的一元多项式。

3.进行两个一元多项式加法。

4.输出运算结果。

2.输入数据

问题需处理多项式的系数和次数，数据输入格式为：

输入第一个大于 0 的整数表示系数，第二个大于 0 的整数表示次数；若为常数，则输入次数为 0，当输入系数为 0 时结束操作。（系数和次数中间用空格隔开）

例如输入：3 5 4 1 -2 4 5 0 0

则表示你输入了多项式：3x^5+4x-2x^4+5

3.输出数据

1.先输出排好序的多项式，按 3x5+4x-2x4+5 输出，前面的数字表示系数，后面的数字表示次数。

2. 进行多项式加法后输出结果，还是按 1 的格式来输入。

例如输出：

请输入要添加的多项式 s1：输入第一个大于 0 的整数表示系数，

第二个大于 0 的整数表示次数；若为常数，则输入次数为 0，当输入系数为 0 时结束操作，例如 3 5 4 1 -2 4 5 0 0

现在的一元 n 次多项式为：

3x5-2x4+4x+5

请输入要添加的多项式 s2：输入第一个大于 0 的整数表示系数，

第二个大于 0 的整数表示次数；若为常数，则输入次数为 0，当输入系数为 0 时结束操作，例如 3 5 4 1 -2 4 5 0 0

现在的一元 n 次多项式为：

3x5-2x4+4x+5

计算后的多项式为：

现在的一元 n 次多项式为：

6x5-4x4+8x+10

4.测试样例设计

测试样例一：

样例输入：

3 5 4 1 -2 4 5 0 0

3 5 4 1 -2 4 5 0 0

输出：

======欢迎来到基于链表实现的计算一元多项式项目======

请输入要添加的多项式 s1：输入第一个大于 0 的整数表示系数，

第二个大于 0 的整数表示次数；若为常数，则输入次数为 0，当输入系数为 0 时结束操作，例如 3 5 4 1 -2 4 5 0 0

3 5 4 1 -2 4 5 0 0

现在的一元 n 次多项式为：

3x5-2x4+4x+5

请输入要添加的多项式 s2：输入第一个大于 0 的整数表示系数，

第二个大于 0 的整数表示次数；若为常数，则输入次数为 0，当输入系数为 0 时结束操作，例如 3 5 4 1 -2 4 5 0 0

3 5 4 1 -2 4 5 0 0

现在的一元 n 次多项式为：

3x5-2x4+4x+5

计算后的多项式为：

现在的一元 n 次多项式为：

6x5-4x4+8x+10

说明：该组样例测试目的是测试两个相同的多项式的相加。

测试样例二：

样例输入：

3 5 4 1 -2 4 5 0 0

3 6 4 1 -2 4 5 0 0

样例输出：

======欢迎来到基于链表实现的计算一元多项式项目======

请输入要添加的多项式 s1：输入第一个大于 0 的整数表示系数，

第二个大于 0 的整数表示次数；若为常数，则输入次数为 0，当输入系数为 0 时结束操作，例如 3 5 4 1 -2 4 5 0 0

3 5 4 1 -2 4 5 0 0

现在的一元 n 次多项式为：

3x5-2x4+4x+5

请输入要添加的多项式 s2：输入第一个大于 0 的整数表示系数，

第二个大于 0 的整数表示次数；若为常数，则输入次数为 0，当输入系数为 0 时结束操作，例如 3 5 4 1 -2 4 5 0 0

3 6 4 1 -2 4 5 0 0

现在的一元 n 次多项式为：

3x6-2x4+4x+5

计算后的多项式为：

现在的一元 n 次多项式为：

3x6+3x5-4x4+8x+10

说明：本组测试样例的目的是测试输入两个有不同次数的多项式相加能否得到正确结果。

测试样例三：（特殊情况）

输入：

3 5 4 1 -2 4 5 0 0

4 0 0

输出：

======欢迎来到基于链表实现的计算一元多项式项目======

请输入要添加的多项式 s1：输入第一个大于 0 的整数表示系数，

第二个大于 0 的整数表示次数；若为常数，则输入次数为 0，当输入系数为 0 时结束操作，例如 3 5 4 1 -2 4 5 0 0

3 5 4 1 -2 4 5 0 0

现在的一元 n 次多项式为：

3x5-2x4+4x+5

请输入要添加的多项式 s2：输入第一个大于 0 的整数表示系数，

第二个大于 0 的整数表示次数；若为常数，则输入次数为 0，当输入系数为 0 时结束操作，例如 3 5 4 1 -2 4 5 0 0

4 0 0

现在的一元 n 次多项式为：

+4

计算后的多项式为：

现在的一元 n 次多项式为：

3x5-2x4+4x+9

说明：该数据是一个多项式和一个常数相加，测试系统能否可以计算多项式和常数相加。

测试样例四：（边界样例及极易错误样例）

输入：

3 5 4 1 -2 4 5 0 0

3 3 4 6 5 7 0

错误输出：

Program received signal SIGSEGV，Segmentation fault.

正确输出：

======欢迎来到基于链表实现的计算一元多项式项目======

请输入要添加的多项式 s1：输入第一个大于 0 的整数表示系数，

第二个大于 0 的整数表示次数；若为常数，则输入次数为 0，当输入系数为 0 时结束操作，例如 3 5 4 1 -2 4 5 0 0

3 5 4 1 -2 4 5 0 0

现在的一元 n 次多项式为：

3x5-2x4+4x+5

请输入要添加的多项式 s2：输入第一个大于 0 的整数表示系数，

第二个大于 0 的整数表示次数；若为常数，则输入次数为 0，当输入系数为 0 时结束操

作，例如 3 5 4 1 -2 4 5 0 0

3 3 4 6 5 7 0

现在的一元 n 次多项式为：

5x7+4x6+3x3

计算后的多项式为：

现在的一元 n 次多项式为：

5x7+4x6+3x5-2x4+3x3+4x+5

说明：本例子是两个次数都不同的多项式相加，输出相加结果，这个例子会遇到指针越界问题，如果处理不当会让程序崩溃，电脑黑屏。

测试样例五：（特殊样例）

输入：

3 5 4 1 -2 4 5 0 0

3 5 -4 1 2 4 5 0 0

输出：

======欢迎来到基于链表实现的计算一元多项式项目======

请输入要添加的多项式 s1：输入第一个大于 0 的整数表示系数，

第二个大于 0 的整数表示次数；若为常数，则输入次数为 0，当输入系数为 0 时结束操作，例如 3 5 4 1 -2 4 5 0 0

3 5 4 1 -2 4 5 0 0

现在的一元 n 次多项式为：

3x5-2x4+4x+5

请输入要添加的多项式 s2：输入第一个大于 0 的整数表示系数，

第二个大于 0 的整数表示次数；若为常数，则输入次数为 0，当输入系数为 0 时结束操作，例如 3 5 4 1 -2 4 5 0 0

3 5 -4 1 2 4 5 0 0

现在的一元 n 次多项式为：

3x5+2x4-4x+5

计算后的多项式为：

现在的一元 n 次多项式为：

6x5+10

说明：本样例输入了两组有次数相同但是系数不同的多项式，目的为测试输出会不会输出系数为 0 的计算结果。

二、概要设计

1.抽象数据类型

问题虽然可以乱序输入，但是存储的时候是按多项式的次数降序存储，因为后来存储数据满足：通过 for 循环依次存入，按照存储的先后次序，满足线性特征，即 <ai,ai+1>(0≤a＜n)。

又因为要比较两组数据和插入并计算数据，所以用链表会更方便一些，并且题目明确要求用链表存储。

2.算法的基本思想

本题先用结构体存储输入 x 的系数和次数，这样方便后续按 x 的次数降序排列。排列好后再存入链表中。本题使用三个链表，分别存输入的两个多项式和多项式的计算结果。

在计算时：如果两个链表的次数相等，则将系数相加并存入链表 3；如果两个链表的次

数不相等，则将次数大的那个存入链表 3，然后次数大的那个链表的当前指针向后移动，移到下一个存储次数和系数的地方；如果其中一个链表移到头了，则接下来直接存储另外一个链表后面的值即可。

3.程序的流程

程序由四个模块组成：

（1）结构体存储模块：

先定义一个结构体，结构体中存储输入的系数和次数，并用快速排序将存储的数据按次数降序排列。这一个模块的功能是：完成了乱序输入的要求，并且已经暂时完成了降序排列的要求。

（2）链表存储模块：

将排好序的结构体中的系数和次数依次存入链表中，这一步完成了按降序存储的要求。

（3）多项式计算功能模块：

将已经降序排好的多项式进行加法运算，如果两个链表中只要有一个有下一个结点，则进行运算。先判断两个链表 1 和 2 结点的次数值是否相等，如果相等则将系数相加，并且只有相加值不为 0 时才存入链表 3 中，然后将当前指针移到下一个结点。不相等则将次数大的结点的系数和次数存入链表 3 中，如果一个链表已经遍历完，则接下来直接存储未遍历完的那个链表。

（4）多项式输出模块：

当多项式降序存储进入后会调用这个功能输出多项式，当多项式计算完成后将调用该功能输出多项式。

该模块的基本思想是：

如果它是存储值的第一个结点，则输出系数 x 次数；如果它不是第一个结点，若他的系数的值大于 0，则输出+系数 x 次数，若系数小于 0 则输出系数 x 次数；如果他的次数等于 1 则输出系数 x；当次数为 0（即为常数时），若系数大于 0，则输出+系数，若系数小于 0，则输出系数。

三、详细设计

1.物理数据类型

数据由两部分组成：系数、次数，结构特性满足线性特性。则根据插入迅速并且时间复杂度低的性能需求选择链表来存储数据。

抽象数据类型的具体设计：

ADT IntegerSet {

数据对象：D = { ai | ai Î 整数，i = 1,2，…, n, n>=0}

数据关系：R = {< ai-1，ai> | ai-1, ai ÎD}

成员：

```
Link* head;//头指针
 Link* tail;//尾指针
 Link* curr;//当前指针
 int cnt;//链表长度
```

基本操作：

```
void init();//操作功能：初始化，制造出一个空表
void removeall();//操作功能：删除表中所有内容
int havecnt();//操作功能：得到表长
bool havenext();//操作功能：判断是否有下一个元素,若有则返回真，没有
```

```
        则返回假。
        void next();//操作功能：将目前为止移到下一个位置
        void print() const;//操作功能：输出多项式
        void clear();//操作功能：清除多项式
        void append(const int& it,const int& cs);//操作功能：尾端插入
        const int& geteleValue() const;//操作功能：得到一个 x 的系数
        const int& getcsValue() const;//操作功能：得到一个 x 的次数
};
```

2.输入和输出的格式

输入流程及格式：输入的第一个数字为系数，第二个正数为次数，中间用空格隔开，如果输入常数，则输入的次数为 0，如果要结束输入则输入系数为 0 即可。

输出流程：

先输出系统名称：======欢迎来到基于链表实现的计算一元多项式项目======

然后输出提示信息和输入例子：

请输入要添加的多项式 s1：输入第一个大于 0 的整数表示系数，

第二个大于 0 的整数表示次数；若为常数，则输入次数为 0，当输入系数为 0 时结束操作，例如 3 5 4 1 -2 4 5 0 0

当用户按要求输入后输出排好序的多项式，然后提醒用户输入多项式 s2，再输出排好序的多项式，最后输出计算结果。

3.算法的具体步骤

1.先用结构体存储输入的系数和次数，然后用 sort 函数对结构体的次数进行降序排列。

2.再将排好序的结构体存入链表中。

3.然后将两个链表进行加法操作，如果两个链表中只要有一个有下一个结点，则进行运算。先判断两个链表 1 和 2 结点的次数值是否相等，如果相等则将系数相加，并且只有相加值不为 0 时才存入链表 3 中，然后将当前指针移到下一个结点。不相等则将次数大的结点的系数和次数存入链表 3 中，如果一个链表已经遍历完，则接下来直接存储未遍历完的那个链表。

4.最后输出链表：如果它是存储值的第一个结点，则输出系数 x 次数；如果它不是第一个结点，若他的系数的值大于 0，则输出+系数 x 次数，若系数小于 0 则输出系数 x 次数；如果他的次数等于 1 则输出系数 x；当次数为 0（即为常数时），若系数大于 0，则输出+系数，若系数小于 0，则输出系数。

伪代码：

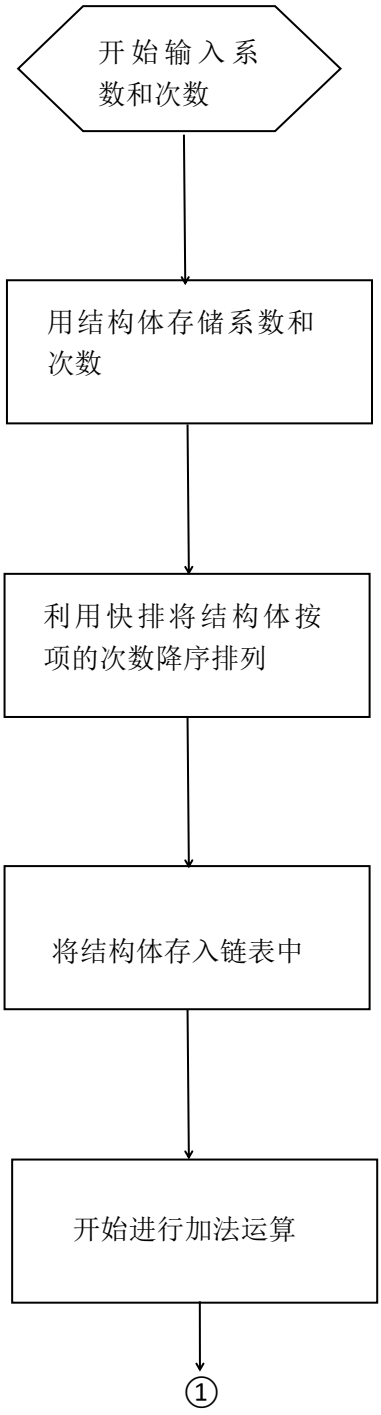
```
int main() {
    void test();
        test();
}
bool bj (DXS d1,DXS d2)//结构体为 DXS
{
    return d1.cshu>d2.cshu;
}
void test()
{
    int a=0;
```

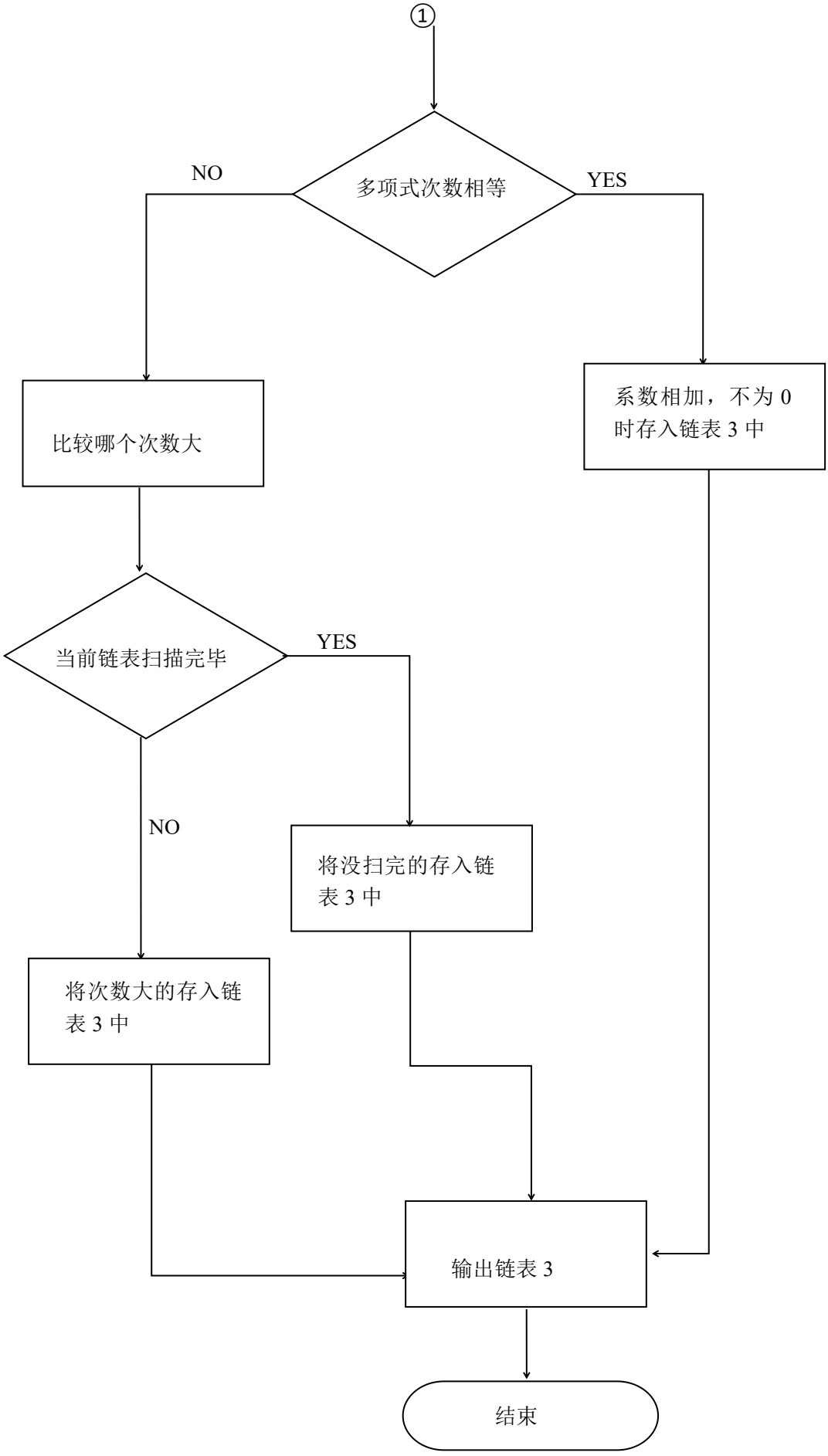
```
LList list1;
LList list2;
LList list3;
DXS d[100];//输入项的个数不大于100，所以这里将100个结构体实例化
int i;
for(i=0;;i++)
{
    cin>>d[i].xs;//输入系数
    if(d[i].xs==0)
    {
        break;
    }
    else
    {
        cin>>d[i].cshu;//输入次数
    }
}
sort(d,d+i,bj);//排序
for(int j=0;j<i;j++)
{
    list1.append(d[j].xs,d[j].cshu);//将排好序的结构体存入链表中
}
list1.print();//链表输出

//第二个多项式的输入：
DXS c[100];
int k;
for(k=0;;k++)
{
    cin>>c[k].xs;
    if(c[k].xs==0)
    {
        break;
    }
    else
    {
        cin>>c[k].cshu;
    }
}
sort(c,c+k,bj);
for(int j=0;j<k;j++)
{
    list2.append(c[j].xs,c[j].cshu);
}
```

```
    list2.print();
    //接下来开始多项式加法
    int xishu=0,cishu=0,l1=1,l2=1;//l1、l2 为当前指针移动到第几个结点，从 1 开始计数
    //当两个链表中其中一个有下一个结点时执行循环体
    while(list1.havenext()||list2.havenext())
    {
        if(list1.getcsValue()==list2.getcsValue())//当次数相等时
        {
            xishu = list1.geteleValue()+list2.geteleValue();
            if(xishu!=0)//计算后的系数不等于 0 时才存入新链表中
            {
                list3.append(xishu,list1.getcsValue()); //将值从尾部插入
            }
            list1.next();//到下一个结点
            l1++;
            list2.next();
            l2++;
            xishu=0;
        }
        else if(list2.getcsValue()>list1.getcsValue())//如果次数不相等则存储大的
        {
            if(list2.havecnt()<l2)//如果一个链表遍历完了
            {
            //则直接存储未遍历完的链表
                list3.append(list1.geteleValue(),list1.getcsValue());
                list1.next();
            }
            else {//否则存储次数大的链表
                list3.append(list2.geteleValue(),list2.getcsValue());
                list2.next();
                l2++;
            }
        }
        else
        {
            if(list1.havecnt()<l1)
            {
                list3.append(list2.geteleValue(),list2.getcsValue());
                list2.next();
            }
            else
            {
```

```
                list3.append(list1.geteleValue(),list1.getcsValue());
                list1.next();
                l1++;
            }

        }
    }
    list3.print();//输出计算结果
}
```

流程图：




①
多项式次数相等
NO
YES
比较哪个次数大
系数相加，不为 0 时存入链表 3 中
当前链表扫描完毕
YES
NO
将没扫完的存入链表 3 中
将次数大的存入链表 3 中
输出链表 3
结束

4.算法的时空分析

（1）结构体存储模块：

由于是 for 循环存储，具体由输入数据规模决定，所以时间复杂度为Θ(n)

（2）链表存储模块：

将排好序的结构体中的系数和次数依次存入链表中，

这一步运用 for 循环存储，具体由结构体中存储的数据规模决定，故时间复杂度为Θ(n)

（3）多项式计算功能模块：

使用 while 和 if，while 复杂度取决于两个链表的长度，故时间复杂度为Θ(n)

（4）多项式输出模块：

类中自定义的函数成员，用 while 来判断是否到尾指针，所以时间复杂度为Θ(n)

四、调试分析

1.调试方案设计

调试目的：为了发现代码是否有语法错误、连接错误、逻辑错误和运算错误，有不严谨之处。

样例：

3 5 4 1 -2 4 5 0 0

3 3 4 6 5 7 0

设置断点处：计算模块的 while 设为断点

调试计划：留心观察当一个链表遍历完之后指针会指向哪里。

设置观察变量；list2.geteleValue()、list1.geteleValue()、list1.getcsValue()、list2.getcsValue()、list1.havenext()、list2.havenext()；

2.调试过程和结果，及分析

调试过程：当两个次数相等时进行运算，然后两个指针分别向后移一位，如果两个次数不等则存储大的那个，并且大的那个指针向后移一位。

问题：在一个链表遍历完之后，再进行后续次数比较和存储时，已经遍历完的链表的指针会越界，系统会崩溃电脑会黑屏。

解决方法：获取当前链表的长度，如果遍历的结点数大于链表长度，则直接存储未遍历完的，此时不再将已遍历完的链表指针向后移动。

五、测试结果

测试样例一：（两个相同的多项式相加）

```
======欢迎来到基于链表实现的计算一元多项式项目======
请输入要添加的多项式s1：输入第一个大于0的整数表示系数，
第二个大于0的整数表示次数；若为常数，则输入次数为0，当输入系数为0时结束操作，例如3 5 4 1 -2 4 5 0 0
3 5 4 1 -2 4 5 0 0
现在的一元n次多项式为：
3x5-2x4+4x+5
请输入要添加的多项式s2：输入第一个大于0的整数表示系数，
第二个大于0的整数表示次数；若为常数，则输入次数为0，当输入系数为0时结束操作，例如3 5 4 1 -2 4 5 0 0
3 5 4 1 -2 4 5 0 0
现在的一元n次多项式为：
3x5-2x4+4x+5
计算后的多项式为：
现在的一元n次多项式为：
6x5-4x4+8x+10
请按任意键继续. . .
```

分析结论：输入两个相同的多项式，则系数相加次数不变后输出。

测试样例二：（一般情况：两个有次数不同的多项式相加）

```
======欢迎来到基于链表实现的计算一元多项式项目======
请输入要添加的多项式s1：输入第一个大于0的整数表示系数，
第二个大于0的整数表示次数；若为常数，则输入次数为0，当输入系数为0时结束操作，例如3 5 4 1 -2 4 5 0 0
3 5 4 1 -2 4 5 0 0
现在的一元n次多项式为：
3x5-2x4+4x+5
请输入要添加的多项式s2：输入第一个大于0的整数表示系数，
第二个大于0的整数表示次数；若为常数，则输入次数为0，当输入系数为0时结束操作，例如3 5 4 1 -2 4 5 0 0
3 6 4 1 -2 4 5 0 0
现在的一元n次多项式为：
3x6-2x4+4x+5
计算后的多项式为：
现在的一元n次多项式为：
3x6+3x5-4x4+8x+10
请按任意键继续. . .
```

分析结论：两个有次数不同的多项式相加，将次数相同的项的系数相加，不同的项按降序排列输出。

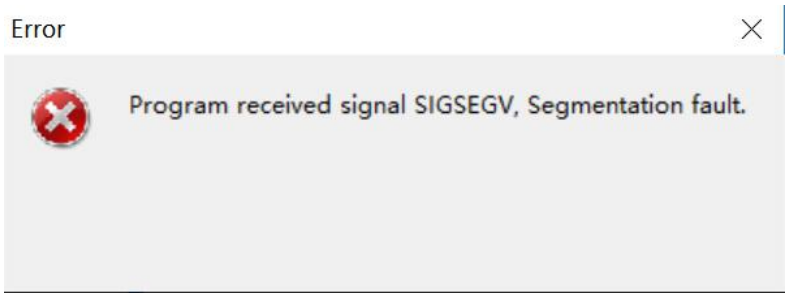
测试样例三：（特殊情况：一个多项式和一个常数相加）

```
======欢迎来到基于链表实现的计算一元多项式项目======
请输入要添加的多项式s1：输入第一个大于0的整数表示系数，
第二个大于0的整数表示次数；若为常数，则输入次数为0，当输入系数为0时结束操作，例如3 5 4 1 -2 4 5 0 0
3 5 4 1 -2 4 5 0 0
现在的一元n次多项式为：
3x5-2x4+4x+5
请输入要添加的多项式s2：输入第一个大于0的整数表示系数，
第二个大于0的整数表示次数；若为常数，则输入次数为0，当输入系数为0时结束操作，例如3 5 4 1 -2 4 5 0 0
4 0 0
现在的一元n次多项式为：
+4
计算后的多项式为：
现在的一元n次多项式为：
3x5-2x4+4x+9
请按任意键继续. . .
```

分析结论：一个多项式和常数相加，则多项式先输出，然后再将常数相加的结果输出。

测试样例四：（特殊情况；两个次数都不同的多项式相加）

错误输出：



正确输出：

```
======欢迎来到基于链表实现的计算一元多项式项目======
请输入要添加的多项式s1：输入第一个大于0的整数表示系数，
第二个大于0的整数表示次数；若为常数，则输入次数为0，当输入系数为0时结束操作，例如3 5 4 1 -2 4 5 0 0
3 5 4 1 -2 4 5 0 0
现在的一元n次多项式为：
3x5-2x4+4x+5
请输入要添加的多项式s2：输入第一个大于0的整数表示系数，
第二个大于0的整数表示次数；若为常数，则输入次数为0，当输入系数为0时结束操作，例如3 5 4 1 -2 4 5 0 0
3 3 4 6 5 7 0
现在的一元n次多项式为：
5x7+4x6+3x3
计算后的多项式为：
现在的一元n次多项式为：
5x7+4x6+3x5-2x4+3x3+4x+5
请按任意键继续. . .
```

分析结论：两个次数不同的多项式相加，则按次数降序输出，这里极易出现指针越界问题。

测试样例五：（特殊情况：两个次数相同的但是系数为相反数的多项式相加）

```
======欢迎来到基于链表实现的计算一元多项式项目======
请输入要添加的多项式s1：输入第一个大于0的整数表示系数，
第二个大于0的整数表示次数；若为常数，则输入次数为0，当输入系数为0时结束操作，例如3 5 4 1 -2 4 5 0 0
3 5 4 1 -2 4 5 0 0
现在的一元n次多项式为：
3x5-2x4+4x+5
请输入要添加的多项式s2：输入第一个大于0的整数表示系数，
第二个大于0的整数表示次数；若为常数，则输入次数为0，当输入系数为0时结束操作，例如3 5 4 1 -2 4 5 0 0
3 5 -4 1 2 4 5 0 0
现在的一元n次多项式为：
3x5+2x4-4x+5
计算后的多项式为：
现在的一元n次多项式为：
6x5+10
请按任意键继续. . .
```

分析结论：两个次数相同的但是系数为相反数的多项式相加后，不为 0 的结果存入链表中，为 0 的不存。

六、实验日志

2018/10/24

内容：实验二上机

确定了多项式的实现思想。设计了链表 ADT，添加了链表的成员变量 cishu 来存储多项式的次数。

2018/11/4

内容：确定 main 函数

输入：

1.初想法：让用户直接输入一个表达式：3x5-2x4+4x+5

遇到的问题：处理表达式过程及其复杂，并且不能按实际要求存入链表。

2. 修改后的想法：直接让用户输入项的系数和次数，中间用逗号隔开，然后用结构体接收，再排序后存入链表中。

链表计算后插入：

1. 初想法：将链表 1 和 2 计算的结果插入链表 2

遇到的问题：因为链表中的结点是声明的结构体，链表里的成员变量只有头指针、尾指针和当前指针，无法将计算好的结果赋给链表 2 结点的系数。

2. 修改后的想法：

将链表 1 和 2 的计算结果存入链表 3

遍历变一个指针后遇到指针越界问题：

在一个链表遍历完之后，再进行后续次数比较和存储时，已经遍历完的链表的指针会越界，系统会崩溃电脑会黑屏。

解决方法：获取当前链表的长度，如果遍历的结点数大于链表长度，则直接存储未遍历完的，此时不再将已遍历完的链表指针向后移动。

经验和体会：在使用链表时要注意指针越界问题；sort 函数快排使用方法是 sort（数组开始，数组结束，基于排序的 bool 函数名）；在使用类继承时要注意子类都要重写父类的虚基类。通过本次实验，更加了解了链表的使用方法和应用。