南京理工大学
NANJING UNIVERSITY OF SCIENCE & TECHNOLOGY

数据结构实验一报告

班　　级 9191062301

学生姓名 孙傲歆

学　　号 919106840333

题　　目 一元多项式的相加减算法

指导教师 练智超

一、实验要求

输入:基于链表存储的一元多项式
目标:基于链表存储两一元多项式相加减的结果，并输出结果

二、算法思路

该算法通过输入获取多项式，所以需要一个建立多项式的函数，由于输入的指数是无序的，并且可能存在需要合并相同指数项，我们需要在这个过程中进行合并，有两种方法，第一种，会比较繁琐浪费时间，就是先一次性建立一个多项式，然后给这个多项式按照指数大小进行排序，排完序后进行合并同类项，如果发现这个同类项的系数为 0，还要进行删除该项，所需要的步骤较多，不宜采用，但较相似我们人的手写计算的过程，第二种，边建边排序边合并同类项，一次最多插入一个节点，直到链表建立结束，相比第一种算法时间复杂度较小。建立 A,B 结束后在进行加法，减法运算。

三、代码运行结果

输入:以系数-次数的形式输入，以 0-0 表示输入的结束
输出:输出两个多项式相加、相减的结果，同样以系数-次数的形式给出
代码输入输出结果:

```
请输入第一个多项式:
4 3
2 2
1 1
5 0
0 0
请输入第二个多项式:
3 3
6 2
2 1
0 0
加法结果
5 0
3 1
8 2
7 3

减法结果
5 0
-1 1
-4 2
1 3
```

输入输出结果对应的多项式:
F1=4x^3+2x^2+x+5
F2=3x^3+6x^2+2x
F1+F2=7x^3+8x^2+3x+5
F1-F2=x^3-4x^2-x+5

四、代码

```
using namespace std;
#include<bits/stdc++.h>
struct node
{
    int num;
    int exp;
    node *next;
};
node *insert(node *head,node *p)//向链表中插入
{
    node *q,*qa,*qb;
    q=new node;
    q->num=p->num;
    q->exp=p->exp;
    q->next=NULL;
    if(head==NULL)
    {
        head=q;
        head->next=NULL;
        return head;
    }
    qa=qb=head;
    while(qa!=NULL)
    {
        while(qa->exp<p->exp)
        {
            qb=qa;
            if(qa->next==NULL)
                break;
            qa=qa->next;
        }
        if(qa->exp==p->exp)
        {
            qa->num=qa->num+p->num;
            if(qa->num==0)
            {
                if(qa==head)
                {
                    head=head->next;
                    break;
                }
                else
```

```
                    {
                        qb->next=qa->next;
                    }
                }
                else
                {
                    break;
                }
            }
            else if(qa->exp<p->exp)
            {
                qa->next=q;
                break;
            }
            else
            {
                if(qb==head&&qb->exp>q->exp)
                {
                    q->next=head;
                    head=q;
                    break;
                }
                else
                {
                    qb->next=q;
                    q->next=qa;
                    break;
                }
            }
        }
        return head;
}
node *create()//创建多项式
{
      node *p,*head;
       p=new node;
       head=NULL;
       while(cin>>p->num>>p->exp)
       {
           if(p->num==0&&p->exp==0)
               break;
            head=insert(head,p);
       }
       return head;
```

```
}
node *addsub(node *pa,node *pb,char s)//多项式相加减
{
      node *p,*q,*head;
       p=(node*)malloc(sizeof(node));
       head=q=p;
       while(pa!=NULL&&pb!=NULL)
       {
            if(pa==NULL)
            {
                 while(pb!=NULL)
                 {
                      q=p;
                         p->exp=pb->exp;
                         p->num=pb->num;
                         p=new node;
                         q->next=p;
                         pb=pb->next;
                 }
            }
            else if(pb==NULL)
            {
                 while(pa!=NULL)
                 {
                      q=p;
                      p->exp=pa->exp;
                      p->num=pa->num;
                      p=new node;
                      q->next=p;
                      pa=pa->next;
                 }
            }
            else
            {
                 if(pa->exp>pb->exp)
                 {
                      q=p;
                      p->exp=pb->exp;
                      p->num=pb->num;
                      p=new node;
                      q->next=p;
                      pb=pb->next;
                 }
                 else if(pa->exp<pb->exp)
```

```
            {
                q=p;
                p->exp=pa->exp;
                p->num=pa->num;
                p=new node;
                q->next=p;
                pa=pa->next;
            }
            else
            {
                q=p;
                p->exp=pa->exp;
                if(s=='+')
                    p->num=pa->num+pb->num;
                if(s=='-')
                    p->num=pa->num-pb->num;
                if(p->num!=0)
                {
                    p=new node;
                    q->next=p;
                    pa=pa->next;
                    pb=pb->next;
                }
                else
                {
                    pa=pa->next;
                    pb=pb->next;
                }
            }
        }
    }
    q->next=NULL;
    return head;
}
int main()
{
    node *pa,*pb,*p;
    cout<<"请输入第一个多项式:"<<endl;
    pa=create();
    cout<<"请输入第二个多项式:"<<endl;
    pb=create();
    p=addsub(pa,pb,'+');
    cout<<"加法结果"<<endl;
    while(p!=NULL)
```

```
    {
        cout<<p->num<<" "<<p->exp<<endl;
        p=p->next;
    }
    cout<<endl;
    p=addsub(pa,pb,'-');
    cout<<"减法结果"<<endl;
    while(p!=NULL)
    {
        cout<<p->num<<" "<<p->exp<<endl;
        p=p->next;
    }
    cout<<endl;
}
```