哈尔滨工业大学计算机科学与技术学院

实验报告

课程名称：数据结构与算法

课程类型：必修

实验项目：线性表的链式存储结构与应用

实验题目：一元多项式的代数运算

实验日期：2017.11.5

设计成绩	报告成绩	指导老师
		张岩

一、实验目的

设计线性表的链式存储结构，并实现一元多项式的代数运算。

二、实验要求及实验环境

要求：

以链表存储一元多项式，在此基础上完成对多项式的代数操作。

1.能够输入多项式(可以按各项的任意输入顺序,建立按指数降幂排列的多项式)和输出多项式（按指数降幂排列）,以文件形式输入和输出，并显示。

2.能够计算多项式在某一点 x=x 0 的值，其中 x 0 是一个浮点型常量，返回结果为浮点数。

3.能够给出计算两个多项式加法、减法、乘法和除法运算的结果多项式，除法运算的结果包括商多项式和余数多项式。

4.要求尽量减少乘法和除法运算中间结果的空间占用和结点频繁的分配与回收操作。

环境：

64 位 Codeblocks16.01

三、设计思想（本程序中的用到的所有数据类型的定义，主程序的流程图及各程序模块之间的调用关系）

1. 逻辑设计

（1） 函数功能：

① ReadDoubleExpr()：从文件读取字符串

② polypointer trans(char expr[],polypointer m)：将字符串内信息解读出来

③ polypointer Insert(float coef1,int exp1,polypointer L)：按照降幂插入排序插入传入的结点

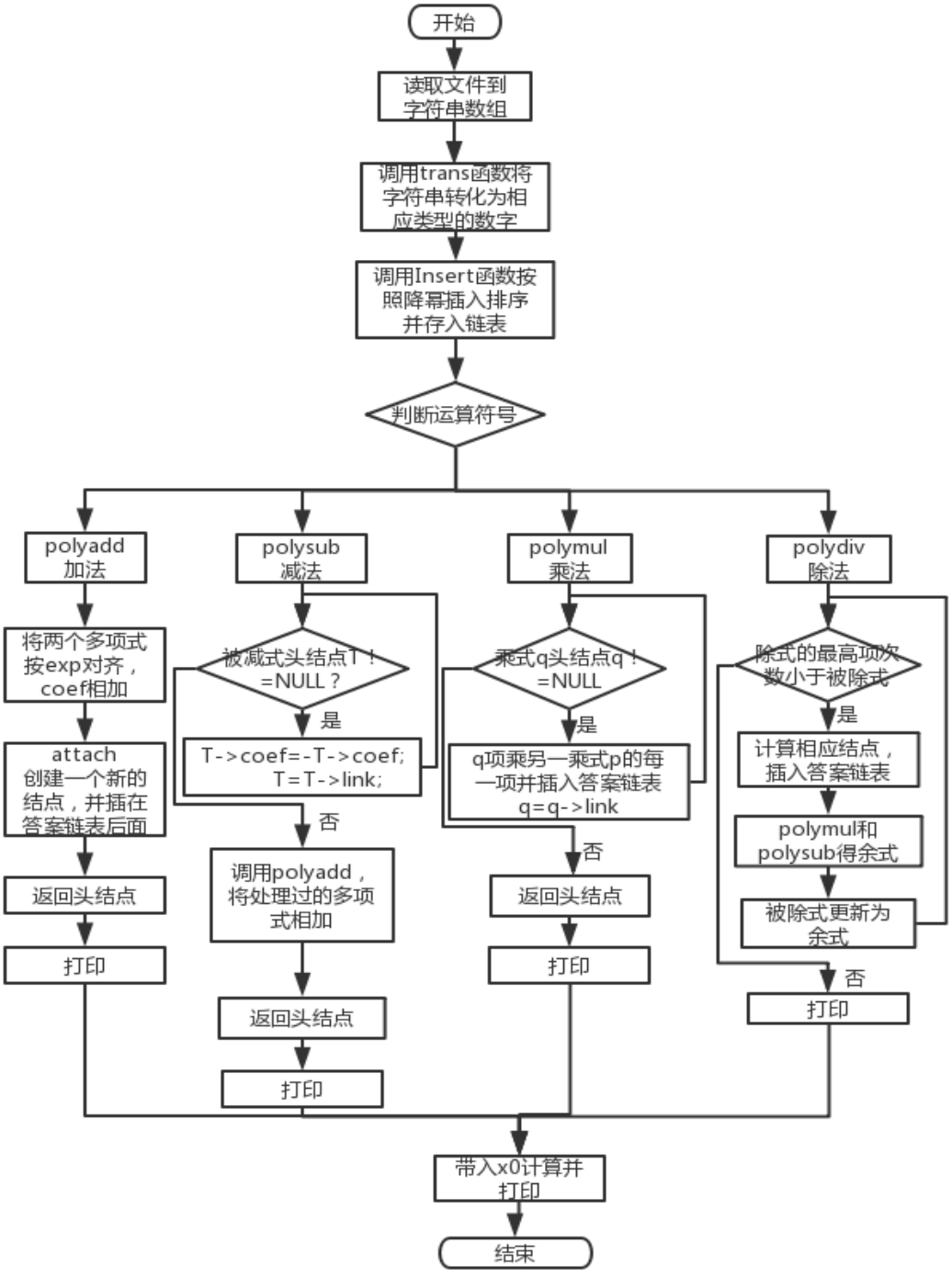
④ int Print(polypointer L)：打印链表表示的式子

⑤ void calcu(float x,polypointer ans)：带入 x0 求值

⑥ polypointer PolyAdd(polypointer a,polypointer b)：加

⑦ PolySub，PolyM，PolyDiv 分别是减法乘法和除法

（2）程序流程图


开始
读取文件到字符串数组
调用trans函数将字符串转化为相应类型的数字
调用Insert函数按照降幂插入排序并存入链表
判断运算符号
polyadd 加法
将两个多项式按exp对齐，coef相加
attach 创建一个新的结点，并插在答案链表后面
返回头结点
打印
polysub 减法
被减式头结点T！=NULL？
是
T->coef=-T->coef;
T=T->link;
否
调用polyadd，将处理过的多项式相加
返回头结点
打印
polymul 乘法
乘式q头结点q！=NULL
是
q项乘另一乘式p的每一项并插入答案链表
q=q->link
否
返回头结点
打印
polydiv 除法
除式的最高项次数小于被除式
是
计算相应结点，插入答案链表
polymul和polysub得余式
被除式更新为余式
否
打印
带入x0计算并打印
结束

2．物理设计

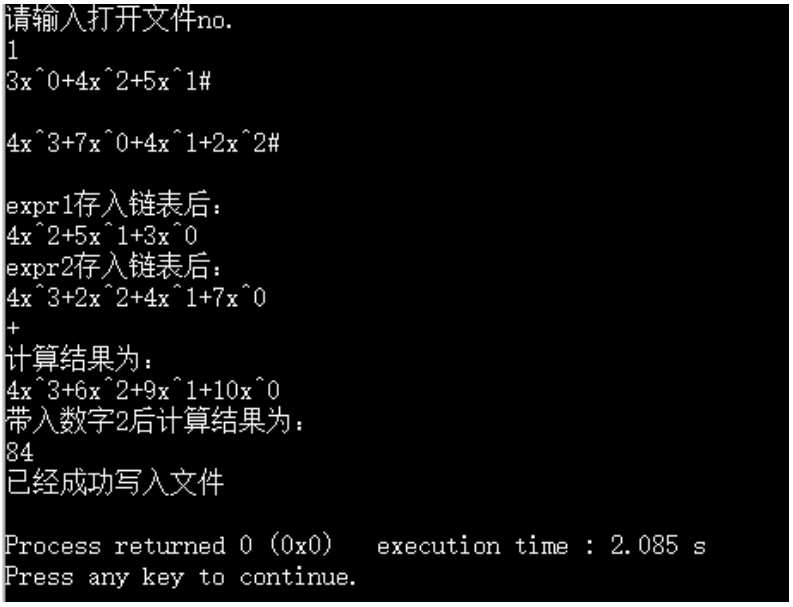
每个结点的结构：

```
struct polynode{
    float coef;
    int exp;
    polynode *link;
};
```

物理结构的优化：刚开始使用将各项与另外一个多项式相乘得到中间结果然后相加的方法，发现占用空间很多。于是想到在处理乘除法的时候，直接用一个链表存储答案，每次获得的每一项都按降幂插入答案链表，省去了中间过程的空间浪费。

四、测试结果

（1）加法



（2）减法

```
请输入打开文件no.
2
3x^0+4x^2+5x^1#

4x^3+7x^0+4x^1+2x^2#

expr1存入链表后：
4x^2+5x^1+3x^0
expr2存入链表后：
4x^3+2x^2+4x^1+7x^0
-
计算结果为：
-4x^3+2x^2+1x^1-4x^0
带入数字2后计算结果为：
-26
已经成功写入文件

Process returned 0 (0x0)   execution time : 1.175 s
Press any key to continue.
```

（3）乘法

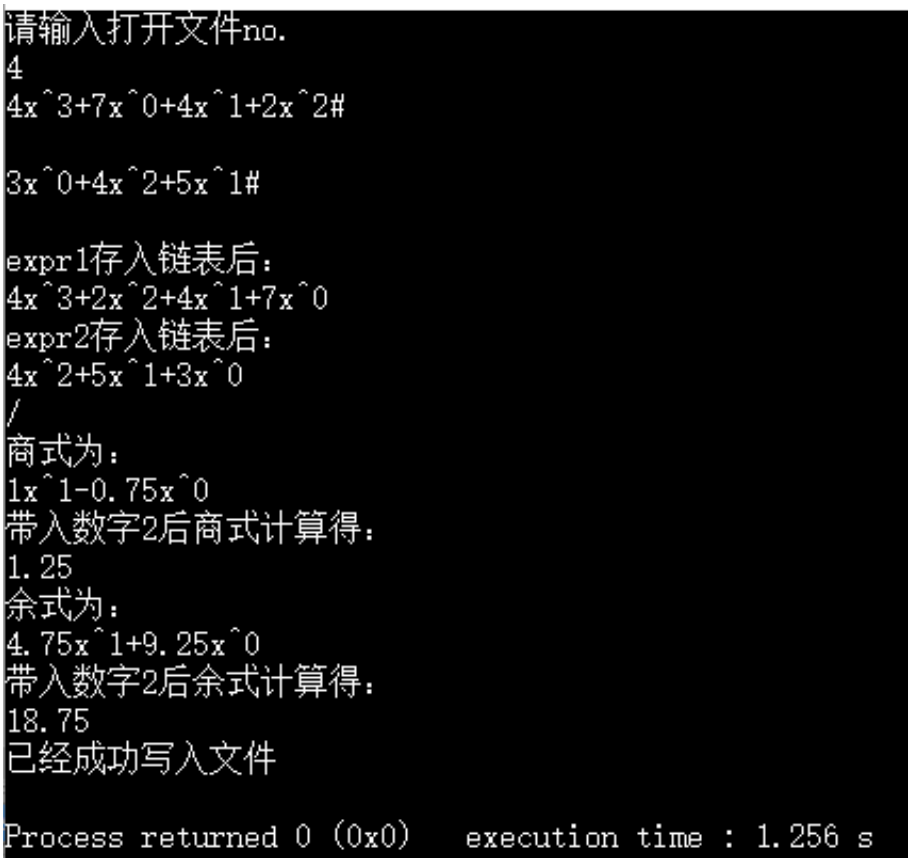
```
请输入打开文件no.
3
3x^0+4x^2+5x^1#

4x^3+7x^0+4x^1+2x^2#

expr1存入链表后：
4x^2+5x^1+3x^0
expr2存入链表后：
4x^3+2x^2+4x^1+7x^0
*
计算结果为：
16x^5+28x^4+38x^3+54x^2+47x^1+21x^0
带入数字2后计算结果为：
1595
已经成功写入文件

Process returned 0 (0x0)   execution time : 1.587 s
Press any key to continue.
```

（4）除法



（5）附加测试（一些特殊情况）

两个式子相同时的除法：

```
请输入打开文件no.
0
4.5x^3#

4.5x^3#

expr1存入链表后：
4.5x^3
expr2存入链表后：
4.5x^3
/
商式为：
1x^0
带入数字3.1后商式计算得：
1
余式为：
0
带入数字3.1后余式计算得：
0
已经成功写入文件

Process returned 0 (0x0)   execution time : 1.238 s
Press any key to continue.
```

两个式子相同时的减法

```
请输入打开文件no.
0
4.5x^3#

4.5x^3#

expr1存入链表后：
4.5x^3
expr2存入链表后：
4.5x^3
-
计算结果为：
0
带入数字3.1后计算结果为：
0
已经成功写入文件

Process returned 0 (0x0)   execution time : 1.572 s
Press any key to continue.
```

五、经验体会与不足

经验：

在处理乘法的时候为了节省空间,可以将中间结果直接经过处理之后存入答案链表，这样使得程序的思路更加清晰，也节省了空间。使用 sscanf()函数可以很好的处理字符串里的信息。

不足：

最后在显示结果的时候不够完美。

六、附录：源代码（带注释）

```
#include<cstdio>
#include<iostream>
#include<string.h>
#include<fstream>
#include<math.h>
using namespace std;

struct polynode{
    float coef;
    int exp;
    polynode *link;
};
typedef polynode *polypointer;
polypointer a=NULL;
polypointer b=NULL;

ofstream outfile,fout;
float x0;
void Destory(polypointer head)//销毁链表函数
{
    polypointer p = head,pr = NULL;
    while (p != NULL)
```

```
    {
        pr = p;
        p = p->link;
        delete pr;
    }
    cout<<"空间已释放"<<endl;
    return;
}
void calcu(float x,polypointer ans)
{
    float p1=0;
    if(ans==NULL){cout<<0<<endl;outfile<<0<<endl;return;}
    while(ans!=NULL)
    {
        p1=p1+ans->coef*pow(x,ans->exp);
        ans=ans->link;
    }
    cout<<p1<<endl;
    outfile<<p1<<endl;
}
int Print(polypointer L)
{

    if(L==NULL){cout<<"0"<<endl;outfile<<"0"<<endl;return 1;}
    do{
        cout<<L->coef<<"x^"<<L->exp;
        outfile<<L->coef<<"x^"<<L->exp;
        if(L->link!=NULL&&L->link->coef>0){cout<<"+";outfile<<"+";}
        L=L->link;
```

```
        }while(L!=NULL&&L->link!=NULL);


if(L!=NULL){cout<<L->coef<<"x^"<<L->exp<<endl;outfile<<L->coef<<"x^"<<L->exp<<endl;
}
        else {cout<<endl;outfile<<endl;}
        return 0;
}
polypointer Insert(float coef1,int exp1,polypointer L)
{
      // int cu=2;
        if(L==NULL)
        {
           //    cout<<1<<endl;
                L=new polynode;
                L->coef=coef1;
                L->exp=exp1;
                L->link=NULL;
                return L;
        }
        else{


                int doflag=0;
                int done=0;
                polypointer temp=L;
                polypointer temp1=L;
                polypointer tem;
                tem=new polynode;
                tem->coef=coef1;
                tem->exp=exp1;
```

```
tem->link=NULL;
while(temp1!=NULL)
{
    if(tem->exp==temp1->exp){temp1->coef+=tem->coef;done=1;}
    temp1=temp1->link;
    //return L;
}
if(done==1)return L;
if(temp->exp<exp1)
    {
        polypointer x1;
        x1=L;
        L=tem;
        tem->link=x1;
        return L;
    }

if(temp->link==NULL)
{
    if(temp->exp<exp1)
    {
        polypointer x1;
        x1=L;
        L=tem;
        tem->link=x1;
    }

    else{L->link=tem;}
    doflag=1;
```

```
                return L;
            }
            while(temp->link!=NULL)
           {
                if((exp1<(temp->exp))&&(exp1>(temp->link->exp)))
                {
                    polypointer x;
                    x=temp->link;
                    temp->link=tem;
                    tem->link=x;
                    doflag=1;
                }
                temp=temp->link;
            }
            if(doflag==0)
            {
                polypointer x2=L;
                while(x2->link!=NULL)
                {
                    x2=x2->link;
                }
                x2->link=tem;
            }
            return L;
        }
}
polypointer trans(char expr[],polypointer m)
{
        polypointer temp=m;
```

```
int i=0;
int pflag=0;
int xflag=0;
int tag=0;
int tag1=0;
float ans=0;
int counte=0;
int counte1=0;
int exp;
float coef;
while(expr[i]!='#')
{
    if(expr[i]=='+')
    {
        i++;
        tag1=0;
        xflag=0;
        coef=ans;
        temp=Insert(coef,exp,temp);
        pflag=0;
        counte=0;
        counte1=0;
        ans=0;
        continue;
    }
    else if(expr[i]=='^')
    {
        i++;
        continue;
```

```
}
else if(expr[i]=='x')
{
    xflag=1;
    i++;
    tag=0;
    continue;
}
else if((expr[i]>='0'&&expr[i]<='9')||expr[i]=='.')
{
    if(expr[i]=='.')
    {
        pflag=1;
        i++;
        continue;
    }
    if(xflag==0)
    {
        int tem1=expr[i]-'0';//把数字由字符转变为数字
        //cout<<"tem1:"<<tem1;
        if(pflag==0)//如果是小数点前的位数做一种处理
        {
        counte++;
        if(counte==1){ans=ans+tem1;}
        else ans=ans*10+tem1;
        }
        else{//如果是小数点后的位数，做另一种处理
        counte1++;
        ans=ans+pow(0.1,counte1)*tem1;
```

```
                        //cout<<ans<<endl;
                        }
                }
                else{
                        if(tag1>1)
                        {
                                exp=exp*10+(expr[i]-'0');
                        }
                        else{
                                exp=expr[i]-'0';
                                tag1=2;
                        }
                }
                i++;
            }
        }
      // cout<<coef<<" "<<exp<<endl;
        coef=ans;
        temp=Insert(coef,exp,temp);
        pflag=0;
        counte=0;
        counte1=0;
        ans=0;
        return temp;
}
char op3[3];
char exp3[10];
int ReadDoubleExpr()
{
```

```
  char expr1[100];
  char expr2[100];
  FILE *fp;
// int contr=0;
  cout<<"请输入打开文件 no."<<endl;
  int file;
  cin>>file;
  switch(file)
  {
        case 0:
              fp=fopen("exp.txt","r");
              break;
        case 1:
              fp=fopen("exp1.txt","r");
              break;
        case 2:
              fp=fopen("exp2.txt","r");
              break;
        case 3:
              fp=fopen("exp3.txt","r");
              break;
        case 4:
              fp=fopen("exp4.txt","r");
              break;
        default:
              cout<<"not found!"<<endl;
              return -1;
  }
  fgets(expr1,1024,fp);
```

```
        fgets(expr2,1024,fp);
        fgets(op3,1024,fp);
        fgets(exp3,1024,fp);
        sscanf(exp3,"%[^#]",exp3);
        sscanf(exp3,"%f",&x0);
        cout<<expr1<<endl;
        outfile<<expr1<<endl;
        cout<<expr2<<endl;
        outfile<<expr2<<endl;



        a=trans(expr1,b);
        cout<<"expr1 存入链表后："<<endl;
        outfile<<"expr1 存入链表后："<<endl;
        Print(a);

        b=trans(expr2,b);
        cout<<"expr2 存入链表后："<<endl;
        outfile<<"expr2 存入链表后："<<endl;
        Print(b);
}
polypointer Attach(float c,int e,polypointer d)
/*
建立一个新结点，系数 coef=c,指数 exp=e,把它连接到 d 所指结点之后，并且返回这个新结
点的指针
*/
{
        polypointer x;
        x=new polynode;
```

```
        x->coef=c;
        x->exp=e;
        d->link=x;
        return x;
}
char Compare(int x,int y)
{
        if(x==y)return '=';
        if(x>y) return '>';
        if(x<y) return '<';
}
polypointer PolyAdd(polypointer a,polypointer b)
{
        polypointer p,q,d,c;
        float x;
        p=a;q=b;
        c=new polynode;d=c;
        d->link=NULL;
        while((p!=NULL)&&(q!=NULL))
        {
                switch(Compare(p->exp,q->exp))
                {
                        case '=':
                                x=p->coef+q->coef;
                                if(x!=0)
                                        d=Attach(x,p->exp,d);
                                p=p->link;q=q->link;
                                break;
                        case '>':
```

```
                    d=Attach(p->coef,p->exp,d);
                    p=p->link;
                    break;
                case '<':
                    d=Attach(q->coef,q->exp,d);
                    q=q->link;
                    break;
            }
        }
        while(p!=NULL)
        {
            d=Attach(p->coef,p->exp,d);
            p=p->link;
        }
          while(q!=NULL)
        {
            d=Attach(q->coef,q->exp,d);
            q=q->link;
        }
        d->link=NULL;
        p=c;c=c->link;delete p;
       // Print(c);
        return c;
}
polypointer PolySub(polypointer a,polypointer b)
{
        polypointer sub;
        polypointer temans;
        sub=b;
```

```
        do{
                sub->coef=-sub->coef;
                sub=sub->link;
        }while(sub!=NULL);
    // if(sub!=NULL) sub->coef=-sub->coef;
        temans=PolyAdd(a,b);
       // Print(temans);
        return temans;
}
polypointer PolyMul(polypointer a,polypointer b)
{
        polypointer p,q,c;
        p=a;q=b;
        c=NULL;
        while(q!=NULL)
        {
                p=a;
                while(p!=NULL)
                {
                        c=Insert(p->coef*q->coef,p->exp+q->exp,c);
                        p=p->link;
                }
                q=q->link;
        }
        return c;
}
polypointer PolyDiv(polypointer a,polypointer b)
{
        polypointer p,q;
```

```
polypointer ans=NULL;
polypointer ans1=NULL;
polynode uniq;
p=a;
q=b;
while(p->exp>=q->exp)
{
        polypointer ans2=NULL;
        ans=Insert(p->coef/q->coef,p->exp-q->exp,ans);
        uniq.coef=p->coef/q->coef;
        uniq.exp=p->exp-q->exp;
        uniq.link=NULL;
        ans1=PolyMul(&uniq,q);//不是 ans,是除得的结果
        ans2=PolySub(p,ans1);
        if(ans2==NULL){p=ans2;break;}
        p=ans2;
      // if(p->coef==0)break;
}
cout<<"商式为："<<endl;
outfile<<"商式为："<<endl;
Print(ans);
cout<<"带入数字"<<x0<<"后商式计算得："<<endl;
outfile<<"带入数字"<<x0<<"后商式计算得："<<endl;
calcu(x0,ans);
cout<<"余式为："<<endl;
outfile<<"余式为："<<endl;
Print(p);
cout<<"带入数字"<<x0<<"后余式计算得："<<endl;
outfile<<"带入数字"<<x0<<"后余式计算得："<<endl;
```

```
    calcu(x0,p);
    return ans;
}

int main()
{
    outfile.open("ans.txt");
    int r;
    r=ReadDoubleExpr();
    if(r==-1){cout<<"未成功打开文件,请重试"<<endl;return -1;}
    polypointer ans;
    cout<<op3[0]<<endl;

    outfile<<op3[0]<<endl;
    switch(op3[0])
    {
        case '+':
            ans=PolyAdd(a,b);
            cout<<"计算结果为："<<endl;
            outfile<<"计算结果为："<<endl;
            Print(ans);
            cout<<"带入数字"<<x0<<"后计算结果为："<<endl;
            outfile<<"带入数字"<<x0<<"后计算结果为："<<endl;
            calcu(x0,ans);
            break;
        case '-':
            ans=PolySub(a,b);
            cout<<"计算结果为："<<endl;
            outfile<<"计算结果为："<<endl;
```

```
            Print(ans);
            cout<<"带入数字"<<x0<<"后计算结果为："<<endl;
            outfile<<"带入数字"<<x0<<"后计算结果为："<<endl;
            calcu(x0,ans);
            break;
        case '*':
            ans=PolyMul(a,b);
            cout<<"计算结果为："<<endl;
            outfile<<"计算结果为："<<endl;
            Print(ans);
            cout<<"带入数字"<<x0<<"后计算结果为："<<endl;
            outfile<<"带入数字"<<x0<<"后计算结果为："<<endl;
            calcu(x0,ans);
            break;
        case '/':
            ans=PolyDiv(a,b);
            break;
        default:
            cout<<"error"<<endl;
            outfile<<"error"<<endl;
            return -1;
    }
    Destory(ans);
    cout<<"已经成功写入文件"<<endl;
    outfile.close();
    return 0;
}
```