实验一 线性表的基础训练

题目：一元稀疏多项式计算
班级：少年班学院少年班 姓名：邱子悦 学号：PB18000028 完成日期：2019.10.18

一.实验要求

编写一个程序，该程序根据输入的命令行参数创建两个单链表表示多项式，然后输出计算结果。

命令行格式：〈可执行程序名〉〈第一个一元多项式〉〈第二个一元多项式〉〈运算符号〉

其中输入的两个一元多项式为不包含空格的字符串，指数不一定有序，形式为类数学表达式。

注意，系数值为 1 的非零次项的输出形式中略去系数 1，如项 1x^8 的输出形式为 x^8。

基本要求：

1. 能够计算两个多项式的加减；
2. 输出计算结果按指数降序排列的类数学表达式。

选作要求：

1. 增加乘积或求导运算；
2. 计算多项式在 x 处的值。

输入输出样例：

Input:

*.exe x+3x^2-1 -2x^2-x+3 +

Output:

x^2+2

二.设计思路

概要设计

ADT Polynomial {

数据对象：$D=\{a_i|a_i\in TermSet,\ i=1,2,\ldots,m,m\geq 0,TermSet$ 中的每个元素包含一个表示系数的实数和表示指数的整数$\}$

数据对象：$R1=\{<a_i,a_{i-1}>|a_i,a_{i-1}\in D,$ 且 a_{i-1} 中的指数值小于 ai 中的指数$,i=2,\ldots,m\}$

基本操作：

OrderInsert(L, p, e)

初始条件：有序表 L 已存在

操作结果：如无节点，则创建；如已有 e，作加法，判断 p 是否为 0，为 0 则释放节点。返回有序表

PrintPolyn(L)

初始条件：有序表 L 已存在

操作结果：输出类数学表达式

CreatePolyn(char po[])

初始条件：输入类数学表达式已存在

操作结果：转化为链表存储，返回有序表

AddPolyn(La, Lb, Lc)

初始条件：有序表 La, Lb 已存在

操作结果：返回有序表 Lc 存储加法结果

SubPolyn(La, Lb, Lc)

初始条件：有序表 La, Lb 已存在

操作结果：返回有序表 Lc 存储减法结果

Differential_once(L)

初始条件：有序表 L 已存在

操作结果：返回求导结果

Value(L, x)

初始条件：有序表 L 已存在

操作结果：返回 x 处求值结果

CopyPolyn(La, Lb)

初始条件：有序表 La 已存在

操作结果：La 复制到 Lb 处

} ADT Polynomial

1.头文件

后续调用到的函数包括：strcmp(), atof(), memset(), strncpy(), malloc(), pow()等，所以引入了这四个头文件：

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<math.h>
```

2.预编译

```
#define SIZE 100 //类数学表达式的最大字符数
#define MAX 10000 //链表创建时头指针赋指数值
```

3.数据类型定义

```
typedef struct Node{
    double p;
    int e;//e 意味着 expn（指数）
    struct Node *next;
}LNode,*LinkList;    //LinkList 作为头指针，增加可读性
```

4. 函数声明

```
void OrderInsert(LinkList L, double p, int e);//有序插入新结点
```

初始条件：

```
void PrintPolyn(LinkList L);//打印多项式
```

LinkList CreatePolyn(char po[]);//创建多项式
LinkList AddPolyn(LinkList La, LinkList Lb, LinkList Lc);//求和
LinkList SubPolyn(LinkList La, LinkList Lb, LinkList Lc);//求差
LinkList Differential_once(LinkList L);//求导数
double Value(LinkList L, double x);//求多项式值
LinkList CopyPolyn(LinkList La, LinkList Lb);//复制多项式（La 复制到 Lb 处）

三.关键代码讲解

1.主函数
四种功能的输入分别是：
加法：*.exe A B +
减法：*.exe A B -
求导：*.exe A d
求值：*.exe A X
其中，命令行 argc 为 4，即为前两种情况；argc 为 3，即为后两种，判断 argv[2] 为 d，则是求导，其他情况为求值，“X”可以是任何数字。

2. CreatePolyn()函数建立在 OrderInsert()有序插入的基础上，融合了多重特殊情况的判断，从类数学表达式中提取每个项的系数和指数，代码如下：

```
LinkList CreatePolyn(char po[]){
    char p[SIZE];
    char e[SIZE];
    double px;
    int ex;
    int i=0,j,k;

   LinkList head,L=(LinkList)malloc(sizeof(LNode));//申请头结点
    L->p=0;
    L->e=MAX;
    L->next=NULL;
    head=L;

    while(po[i]!='\0'){
        memset(p, '\0', SIZE*sizeof(char));//初始化
        //求出 px,ex;
         for(j=i+1;po[j]!='\0';j++){
            if(po[j]=='+' || po[j]=='-')
                if(po[j-1]!='^')
                    break;
         }
         strncpy(p,po+i,(j-i)*sizeof(char));
```

```
            i=j;

            if(p[0]=='x' || (p[0]=='+'&&p[1]=='x'))
                px=1;//一次函数 kx
            else if(p[0]=='-'&&p[1]=='x')
                px=-1;
            else{
                //double atof(const char *str)
                for(j=0;p[j]!='x'&&p[j]!='\0'; j++)
                    ;
                if(p[j]=='\0')
                    px=atof(p);//常数 C
                else{
                    memset(e, '\0', SIZE*sizeof(char));
                    strncpy(e, p, j*sizeof(char));
                    px=atof(e);
                }
            }

            for(j=0;p[j]!='x'&&p[j]!='\0'; j++)
                ;
            if(p[j]=='\0')
                ex=0;
            else if(p[j+1]!='^')
                ex=1;
            else{
                j+=2;
                memset(p, ' ', j);
                ex=atof(p);
            }
           OrderInsert(L,px,ex);

        }

        return head;
}
```

打印多项式的函数 PrintPolyn()，融合了输出类数学表达式的多重判断。加法和减法功能主要调用了 OrderInsert()，求导和求值功能都是基于已存在的线性表，只需对每一项作处理即可。函数 CopyPolyn()仅是辅助函数。

四.调试分析

分析程序的时空复杂度，以及在实验中遇到的问题。

1. 类数学表达式的转化

CreatePolyn()：

最初对于 1，-1，(+)x，-x 等特殊情况的考量太少，以至于输出结果总是有小错误。

最初没有考虑指数符号后可以为负数，例如 x^-3，导致样例测试结果出错。

头节点考虑假设指数为 MAX，但对于指数大于 MAX 的输入无法处理，可能会出现问题。

PrintPolyn()：

同理，对于输出时首项“+”的省略，以及后续“+”的强调，系数为 1 的项中的省略等等，是主要问题。

2. 求导函数

重点要考虑指数为 0 的情况，不能直接指数-1，而是要释放该节点。

3. 算法的时空分析

加法、减法、求导和求值、OrderInsert()复杂度都是 O(n)。CreatePolyn()复杂度是 O(n)。

五.代码测试

可采用文字说明加截图的方式来展示测试运行结果。

1. 加法

考虑了乱序输入、常数、一次项、系数不取整数、抵消、-1/-x/0 等特殊情况。

```
C:\Users\mi>C:\Users\mi\Desktop\main.exe 2x+5x^^8-3.1x^^11 7-5x^^8+11x^^9 +
-3.1x^11+11x^9+2x+7
```

```
C:\Users\mi>C:\Users\mi\Desktop\main.exe -1-x 2x+1 +
x
```

```
C:\Users\mi>C:\Users\mi\Desktop\main.exe -2x+1 0 +
-2x+1
```

2. 减法

考虑了乱序输入、常数、一次项、系数不取整数、抵消、-1/-x 等特殊情况。

```
C:\Users\mi>C:\Users\mi\Desktop\main.exe 2x+5x^^8-3.1x^^11 7-5x^^8+11x^^9 -
-3.1x^11-11x^9+10x^8+2x-7

C:\Users\mi>C:\Users\mi\Desktop\main.exe -1-x 2x+1 -
-3x-2
```

3. 求导

考虑了常数项、系数不取整数、一次项等易出错的情况。

```
C:\Users\mi>C:\Users\mi\Desktop\main.exe 7-x d
-1

C:\Users\mi>C:\Users\mi\Desktop\main.exe 2x+5x^^8-3.1x^^11 d
-34.1x^10+40x^7+2
```

4. 求值

六. 实验总结

第一次尝试较大的链表程序，使用 sublime text 书写，devc++执行，并学会了命令行输入参数。从最初的思路混乱到后来分块功能清晰、细节处理到位，收获很大，实验进行较为顺利，按时完成实验。

七.附录

main.c　　//主程序