《数据结构与算法》实验报告

实验名称	设计实现一个一元稀疏多项式计算器				
姓名	陈思睿	学号	21030021007	日期	2024.3.13
实验内容	1. （3 分）用 C 或 C++语言编程实现“带头节点的线性链表”数据结构及基本操作，例如创建，定位，查找，遍历，插入，删除，归并，销毁等。 2. （2 分）设计输入输出格式，实现一元多项式的输入、存储、输出。用编程实现的 LinkList 进行多项式链表的创建、数值输入、输出等功能。默认多项式序列按指数降序或升序排列。 3. （2 分）实现两个多项式相加并输出结果。 4. （3 分）实现两个多项式相乘并输出结果。				
实验目的	利用线性表来存储多项式，并进行基本的操作				

实验步骤

1、基本结构

创建数据对象，采用单链表存储，数据元素为数据域（指数 expn 和系数 coef）和指针域 next。

其中指数为 int 类型，而系数为 float 类型。

```
typedef struct Poly{
    float coef; //系数
    int expn;   //指数
    struct Poly *next;
}ElemType;
```

2、基本操作

由于目的是实现一个两个一元多项式的相加与相乘，因此除了单链表的本身基本操作初始化、销毁、插入、删除（包含在在相加相乘操作中）、打印外，还有相加和相乘操作。

```
ElemType *LinkList();//建立空链表
void DestroyPolyn(ElemType *p);//销毁一元多项式P
void PrintPolyn(ElemType *p);//打印输出一元多项式P
ElemType *CreatPolyn(ElemType *head);//输入系数和指数，建立表示一元多项式的有序链表P
ElemType *ListInsert_L(ElemType *head, ElemType *e);//处理并插入元素e
ElemType *AddPolyn(ElemType *Pa, ElemType *Pb);//相加
ElemType *MultiplyPolyn(ElemType *Pa, ElemType *Pb);//相乘
```

3、初始化和销毁

此处建立的是一个不带头结点的单链表，因此头指针 head 直接指向第一个元素结点，初始时没有元素，因此 head—>NULL 即可。

销毁时考虑从头遍历整个链表，时间复杂度为 O（n）。同时，在删除是，设置了一个指针用来保存每次要释放的指针时指向的下一结点，防止销毁后整个链表丢失。

```
void DestroyPolyn(ElemType *p){//销毁一元多项式
    ElemType *q;
    if(!p)//空表就不用销毁了
        exit(0);
    while(p->next){
        q = p->next;
        p->next = q->next;
        free(q);
    }//while
    if(! p->next)//删除头结点
        free(p);
}//DestroyPolyn
```

4、输入

此处输入时、本身默认为按照指针序来排列，采取的输入方式为 coef，expn 的形式，以 0,0 为结束输入。

```
ElemType *CreatPolyn(ElemType *head){
    //输入系数和指数，建立表示一元多项式的有序链表P
    ElemType *p1;
    p1 = (ElemType*)malloc(LEN);
    scanf("%f,%d",&p1->coef,&p1->expn);//输入时，
    while( (p1->coef != 0) || (p1->expn != 0) ){
        if(p1->coef != 0){//系数为零，无效处理
            head = ListInsert_L(head, p1);
            p1 = (ElemType*)malloc(LEN);
        }//if
        scanf("%f,%d",&p1->coef,&p1->expn);
    }//while
    return head;
}//CreatPolyn
```

5、相加操作：

由于多项式默认为降序或升序（此处采取降序操作）

因此对于相加来说，定义一个 C 链表用来保存 A+B 的结果。

```
ElemType *AddPolyn(ElemType *Pa, ElemType *Pb){
    //多项式加法：Pc = Pa + Pb
    ElemType *ha,*hb,*qa,*qb,*Pc,*c;//qa指向Pa中
    ha = Pa;
    hb = Pb;
    if(!Pa)
        return Pb;
    else if(!Pb)
        return Pa;
    Pc = LinkList();
    qa = ha;
    qb = hb;
```

```
while(qa && qb){
    c = (ElemType*)malloc(LEN);
    if(qa->expn < qb->expn){
        c->coef = qb->coef;
        c->expn = qb->expn;
        qb = qb->next;
    }
    else if(qa->expn > qb->expn){
        c->coef = qa->coef;
        c->expn = qa->expn;
        qa = qa->next;
    }
    else{
        c->coef = qa->coef + qb->coef;
        c->expn = qa->expn;
        qa = qa->next;
        qb = qb->next;
    }
    if(c->coef != 0)
        Pc = ListInsert_L(Pc, c);
    else
        free(c);
}
```

分别定义一个 pa、pb 指针。平行依次遍历，谁的指数大则把谁的结点中的 expn、coef 值放在 c 中新建的结点里，然后指针后移，再次比较。注意当两指数相同时系数如果相加为 0 则不加入 c 中。

```
while(qa){
    c = (ElemType*)malloc(LEN);
    c->coef = qa->coef;
    c->expn = qa->expn;
    Pc = ListInsert_L(Pc, c);
    qa = qa->next;
}

while(qb){
    c = (ElemType*)malloc(LEN);
    c->coef = qb->coef;
    c->expn = qb->expn;
    Pc = ListInsert_L(Pc, c);
    qb = qb->next;
}
```

直到比较到某一个指针为空，则将 a、b 中不为空的链表中的值加入到 c 中。

5、多项式相乘

多项式相乘其实和多项式相加类似，只不过是 b 的每个结点都和 a 的结点运算。

```
ElemType *MultiplyPolyn(ElemType *Pa, ElemType *Pb){
    //多项式乘法：Pc = Pa * Pb
    ElemType *ha,*hb,*qa,*qb,*Pc,*c;//qa指向Pa中的当前
    float ccoef;
    int cexpn;
    ha = Pa;
    hb = Pb;

    if(!Pa || !Pb)
        return NULL;
    Pc = LinkList();
    for(qa = ha; qa; qa = qa->next){//当qa非空，逐个检
        for(qb = hb; qb; qb = qb->next){
            ccoef = qa->coef * qb->coef;
            cexpn = qa->expn + qb->expn;
            if(ccoef != 0){
                c = (ElemType*)malloc(LEN);
                c->coef = ccoef;
                c->expn = cexpn;
                Pc = ListInsert_L(Pc, c);
            }
        }//for(qb)
    }//for(qa)
```

注意在运算完后释放指针。

这块采用二重循环来操作。每一个 b 的元素都要在 a 中遍历一遍。时间复杂度为 0（n^2）

6、输出：

输出格式应该仍然为降序，A1x^n1+A2x^n2+A3x^n3+……

其中对于特殊输出有默认格式如下：

① 系数为 1，指数不为 1 时，为 x^n.

② 系数不为 1，指数为 1 时，为 Ax^n

③ 系数指数都为 1 时，为 x

④ 指数为 0 则为 A

⑤ 其他

```
while(q){
    if(q->expn == 0)
        printf("%g ",q->coef);//指数为0，直接输出系数
    else if(q->coef == 1 && q->expn != 1)
        printf("x^%d ",q->expn);//系数为1，可省略
    else if(q->coef == 1 && q->expn == 1)
        printf("x ");//指数为1，可省略
    else if(q->coef != 1 && q->expn == 1)
        printf("%gx ",q->coef);
    else
        printf("%gx^%d ",q->coef,q->expn);

    if(q->next != NULL && q->coef != 0 && q->next->coef > 0)
    //打印符号，若当前项为最后一项或下一项系数非正，则不打印
        printf("+ ");
    q = q->next;
}//while
```

7、实验结果

样例 1:

A：5x^5+4x^4+3x^3+2x^2+x

B：3x^3+2x^2+x

```
一元多项式相加
请输入多项式Pa:5,5 4,4 3,3 2,2 1,1 0,0
Pa = 5x^5 + 4x^4 + 3x^3 + 2x^2 + x
请输入多项式Pb:3,3 2,2 1,1 0,0
Pb = 3x^3 + 2x^2 + x

Pc = Pa + Pb
Pc = 5x^5 + 4x^4 + 6x^3 + 4x^2 + 2x

Pd = Pa*Pb
Pd = 15x^8 + 22x^7 + 22x^6 + 16x^5 + 10x^4 + 4x^3 + x^2
PS D:\code>
```

样例 2:

A：x+1

B：x+1

```
一元多项式相加
请输入多项式Pa:1,1 1,0 0,0
Pa = x + 1
请输入多项式Pb:1,1 1,0 0,0
Pb = x + 1

Pc = Pa + Pb
Pc = 2x + 2

Pd = Pa*Pb
Pd = x^2 + 2x + 1
PS D:\code>
```

8、源代码

```
#include<stdio.h>
#include<stdlib.h>
#define LEN sizeof(ElemType)

typedef struct Poly{
    float coef; //系数
    int expn;   //指数
    struct Poly *next;
}ElemType;

ElemType *LinkList();//建立空链表
void DestroyPolyn(ElemType *p);//销毁一元多项式 P
void PrintPolyn(ElemType *p);//打印输出一元多项式 P
ElemType *CreatPolyn(ElemType *head);//输入系数和指数，建立表示一元多项式的有序链表 P
ElemType *ListInsert_L(ElemType *head, ElemType *e);//处理并插入元素 e
ElemType *AddPolyn(ElemType *Pa, ElemType *Pb);//相加
ElemType *MultiplyPolyn(ElemType *Pa, ElemType *Pb);//相乘

ElemType *LinkList(){
    ElemType *head;
    head = NULL;
    return head;
}

void DestroyPolyn(ElemType *p){//销毁一元多项式 P
    ElemType *q;
    if(!p)//空表就不用销毁了
        exit(0);
    while(p->next){
        q = p->next;
        p->next = q->next;
        free(q);
    }//while
    if(! p->next)//删除头结点
        free(p);
}//DestroyPolyn

void PrintPolyn(ElemType *p){//打印输出一元多项式 P
    ElemType *q;
    q = p;
```

```
    if(!p)//空表即表达式为0
        printf("0");

    while(q){
        if(q->expn == 0)
            printf("%g ",q->coef);//指数为0，直接输出系数
        else if(q->coef == 1 && q->expn != 1)
            printf("x^%d ",q->expn);//系数为1，可省略
        else if(q->coef == 1 && q->expn == 1)
            printf("x ");//指数为1，可省略
        else if(q->coef != 1 && q->expn == 1)
            printf("%gx ",q->coef);
        else
            printf("%gx^%d ",q->coef,q->expn);

        if(q->next != NULL && q->coef != 0 && q->next->coef > 0)
        //打印符号，若当前项为最后一项或下一项系数非正，则不打印
            printf("+ ");
        q = q->next;
    }//while
    printf("\n");
}//PrintPolyn
ElemType *CreatPolyn(ElemType *head){
    //输入系数和指数，建立表示一元多项式的有序链表P
    ElemType *p1;
    p1 = (ElemType*)malloc(LEN);
    scanf("%f,%d",&p1->coef,&p1->expn);//输入时，同一项的系数和
指数用逗号隔开，项与项之间用空格隔开
    while( (p1->coef != 0) || (p1->expn != 0) ){
        if(p1->coef != 0){//系数为零，无效处理
            head = ListInsert_L(head, p1);
            p1 = (ElemType*)malloc(LEN);
        }//if
        scanf("%f,%d",&p1->coef,&p1->expn);
    }//while
    return head;
}//CreatPolyn
ElemType *AddPolyn(ElemType *Pa, ElemType *Pb){
    //多项式加法：Pc = Pa + Pb
    ElemType *ha,*hb,*qa,*qb,*Pc,*c;//qa指向Pa中的当前结点，qb
同理
    ha = Pa;
    hb = Pb;
    if(!Pa)
```

```
        return Pb;
    else if(!Pb)
        return Pa;
    Pc = LinkList();
    qa = ha;
    qb = hb;
    while(qa && qb){
        c = (ElemType*)malloc(LEN);
        if(qa->expn < qb->expn){
            c->coef = qb->coef;
            c->expn = qb->expn;
            qb = qb->next;
        }
        else if(qa->expn > qb->expn){
            c->coef = qa->coef;
            c->expn = qa->expn;
            qa = qa->next;
        }
        else{
            c->coef = qa->coef + qb->coef;
            c->expn = qa->expn;
            qa = qa->next;
            qb = qb->next;
        }
        if(c->coef != 0)
            Pc = ListInsert_L(Pc, c);
        else
            free(c);
    }

    while(qa){
        c = (ElemType*)malloc(LEN);
        c->coef = qa->coef;
        c->expn = qa->expn;
        Pc = ListInsert_L(Pc, c);
        qa = qa->next;
    }

    while(qb){
        c = (ElemType*)malloc(LEN);
        c->coef = qb->coef;
        c->expn = qb->expn;
        Pc = ListInsert_L(Pc, c);
        qb = qb->next;
```

```
    }

    return Pc;
}//AddPolyn

ElemType *MultiplyPolyn(ElemType *Pa, ElemType *Pb){
    //多项式乘法：Pc = Pa * Pb
    ElemType *ha,*hb,*qa,*qb,*Pc,*c;//qa 指向 Pa 中的当前结点，qb
同理
    float ccoef;
    int cexpn;
    ha = Pa;
    hb = Pb;

    if(!Pa || !Pb)
        return NULL;
    Pc = LinkList();
    for(qa = ha; qa; qa = qa->next){//当 qa 非空，逐个检索 qa
        for(qb = hb; qb; qb = qb->next){
            ccoef = qa->coef * qb->coef;
            cexpn = qa->expn + qb->expn;
            if(ccoef != 0){
                c = (ElemType*)malloc(LEN);
                c->coef = ccoef;
                c->expn = cexpn;
                Pc = ListInsert_L(Pc, c);
            }
        }//for(qb)
    }//for(qa)

    //计算完毕，释放 Pa 和 Pb
    DestroyPolyn(ha);
    DestroyPolyn(hb);

    return Pc;
}//MultiplyPolyn

int main(){
    ElemType *Pa,*Pb,*Pc,*Pd;
    printf("一元多项式相加\n");

    printf("请输入多项式 Pa:");
    Pa = LinkList();
    Pa = CreatPolyn(Pa);
```

```
    printf("Pa = ");
    PrintPolyn(Pa);

    printf("请输入多项式 Pb:");
    Pb = LinkList();
    Pb = CreatPolyn(Pb);
    printf("Pb = ");
    PrintPolyn(Pb);

    Pc = AddPolyn(Pa, Pb);
    printf("\nPc = Pa + Pb\n");
    printf("Pc = ");
    PrintPolyn(Pc);
    DestroyPolyn(Pc);

    Pd = MultiplyPolyn(Pa,Pb);
    printf("\nPd = Pa*Pb\n");
    printf("Pd = ");
    PrintPolyn(Pd);
    DestroyPolyn(Pd);

    return 0;
}
```

实验总结	本次实验实现了多项式链表的基本操作，熟悉了链表这一数据结构的基本操作。理解了链表和顺序表的优缺点，在什么样的情况下使用不同的线性表，加深了我对于课堂知识的理解和应用。