实验报告- 一元稀疏多项式简单计算器

一、需求分析

设计一个一元多项式简单计算器。
一元稀疏多项式简单计算器的基本功能是：

1. 输入并建立多项式；
2. 输出多项式，输出形式为整数序列： $n, c_1, e_1, c_2, e_2, \ldots, c_n, e_n$，其中 n 是多项式的项数，c_i 和 e_i 分别是第 i 项的系数和指数，序列按指数降序排列；
3. 多项式 a 和 b 相加，建立多项式 $a+b$；
4. 多项式 a 和 b 相减，建立多项式 $a-b$。

进阶功能是：
5. 计算多项式在 x 处的值。
6. 求多项式 a 的导函数 a'。
7. 多项式 a 和 b 相乘，建立乘积多项式 ab。
8. 多项式的输出形式为类数学表达式。例如，多项式 $-3x^8+6x^3-18$ 的输出形式为 $-3x\text{^}8+6x\text{^}3-18$，$x^{15}+(-8)x^7-14$ 的输出形式为 $x\text{^}15-8x\text{^}7-14$。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^6，项 $-1x^3$ 的输出形式为 $-x^3$。
9. 计算器的仿真界面。

其中选择了$1, 2, 3, 4, 5, 6, 7, 8$ 进行实现。

二、详细设计

数据类型

为了实现多项式的存储，需要构造一个链表，使用结构体，数据域中要存放单项的指数、系数，以及`flag`用来只是这一项的状态，指针域是指向下一个结构体的指针。

```
typedef struct LNode{
    double          coef;               //系数
    int             expn;               //指数
    struct LNode    *next;
    int             flag;               //特征，确定是否有定义或者是否已经销毁清除
}LNode,*LinkList;
```

模块

main函数

在`main`函数中，通过打印选项的方式向用户呈现计算器功能，再通过读入数字指令，利用`case`

来实现向不同功能的导向。
为了使操作更加方便，另加了复制和删除功能。

```
int main(){
	LinkList poly[10];
	int i,j,k,menu,n,d;
	double m = 0,x = 0,a,b;
	for(i=0;i<12;i++)
		InitList(poly[i]);
	printf("\n--------------欢迎使用多项式计算器！--------------\n");
	while(1){
	printf("\n请输入序号0~13：\n");
		printf("【1】  创建多项式		【2】  显示多项式\n");
		printf("【3】  所有多项式		【4】  多项式求和\n");
		printf("【5】  多项式求差		【6】  删除多项式\n");
		printf("【7】  多项式求积		【8】  清空多项式\n");
		printf("【9】  复制多项式		【10】 多项式求导\n");
		printf("【11】 多项式求值		【0】 退出计算器\n\n");
		scanf("%d",&menu);
		switch(menu){
			case 0:{
				printf("\n-------------非常感谢您使用本计算器！-------
------\n");
				return 0;
			}
			case 1:{
				printf("请输入要创建的多项式位置下标0~9\n");
				scanf("%d",&i);
				poly[i] = CreatPolyn();
				printf("--------操作完成--------\n");
				break;
			}
			case 2:{
				printf("请输入要显示的多项式位置下标0~9\n");
				scanf("%d",&i);
				PrintPolyn(poly[i],i);
				printf("\n--------操作完成--------\n");
				break;
			}
			case 3:{
				printf("已建立所有多项式为：\n");
				for(i=0;i<12;i++)
					if(!poly[i]->flag)
```

```
PrintPolyn(poly[i],i);
                                printf("\n--------操作完成--------\n");
                                break;
                        }
                        case 4:{
                                printf("请输入要求和的两个多项式位置下标0~9\n");
                                scanf("%d%d",&i,&j);
                                printf("请输入要存放到的位置下标0~9\n");
                                scanf("%d",&k);
                                AddPolyn(poly[i],poly[j],poly[k]);
                                printf("--------操作完成--------\n");
                                break;
                        }
                case 5:{
                                printf("请输入要求差的两个多项式位置下标0~9\n");
                                scanf("%d%d",&i,&j);
                                printf("请输入要存放到的位置下标0~9\n");
                                scanf("%d",&k);
                                SubtractPolyn(poly[i],poly[j],poly[k]);
                                printf("--------操作完成--------\n");
                                break;
                        }
                case 6:{
                                printf("请输入要删除的多项式位置下标0~9\n");
                                scanf("%d",&i);
                                DestroyList(poly[i]);
                                for(;i<11;i++)
                                        poly[i] = poly[i+1];
                                DestroyList(poly[i]);
                                printf("--------操作完成--------\n");
                                break;
                        }
                        case 7:{
                                printf("请输入要求积的两个多项式位置下标0~9\n");
                                scanf("%d%d",&i,&j);
                                printf("请输入要存放到的位置下标0~9\n");
                                scanf("%d",&k);
                                poly[k] = MutiPolyn(poly[i],poly[j]);
                                printf("--------操作完成--------\n");
                                break;
                        }
                        case 8:{
                                for(i = 0;i < 10;i++)
```

```
                                DestroyList(poly[i]);
                        printf("--------操作完成--------\n");
                        break;
                }
        case 9:{
                        printf("请输入要复制的多项式位置下标0~9\n");
                        scanf("%d",&i);
                        printf("请输入要复制到的位置下标0~9\n");
                        scanf("%d",&j);
                        CopyPolyn(poly[i],poly[j]);
                        printf("--------操作完成--------\n");
                        break;
                }
                case 10:{
                        printf("请输入要求导的多项式位置下标0~9\n");
                        scanf("%d",&i);
                        printf("请输入要存放到的位置下标0~9\n");
                        scanf("%d",&k);
                        printf("请输入求导的阶数\n");
                        scanf("%d",&d);
                        poly[k] = poly[i];
                        for(j = 0;j < d;j++)
                        poly[k] = DiffPolyn(poly[k]);
                        printf("--------操作完成--------\n");
                        break;
                }
                case 11:{
                        printf("请输入要求值的多项式的位置下标0~9\n");
                        scanf("%d",&i);
                        printf("请输入要赋给x的值\n");
                        scanf("%lf",&x);
                        ValuePolyn(poly[i],x);
                        printf("--------操作完成--------\n");
                        break;
                }
        default: printf("输入错误！\n");
        }
    }
    printf("\n-------------非常感谢您使用本计算器！-------------\n");
    return 0;
}
```

多项式的创建

用户输入1 时，会被导向多项式创建，用 `CreatePolyn` 函数创建多项式，保存在数组元素 `poly[用户输入序号]` 中。

```
LinkList CreatPolyn(){//指数降序创建多项式，同序号重复创建则覆盖
        LinkList L,q,t,r;
        int e = 0;        //exponent
        double c = 0;     //coefficient
        InitList(L);
        L->flag = NULL;                                    //标记有定义
    printf("请输入x的系数和指数,输入 0空格0 时结束\n");
    scanf("%lf%d",&c,&e);
        while(c){
                q = (LinkList)malloc(sizeof(LNode));//辅助结点
                if(!q)  exit(OVERFLOW);
                q->coef = c;    q->expn = e;
                q->next = NULL;
                for(t = L;t != NULL;r = t,t = t->next){
                        if(q->expn == t->expn){ //重复输入同一指数的系数，累加系数值
                            t->coef += q->coef;
                            break;
                        }
                        else if((abs(t->expn) < abs(q->expn))&&(t == L)){//指向第一个
结点
                                    q->next = L;
                                L = q;
                                    break;
                        }
                        else if((abs(t->expn) > abs(q->expn))&&(abs(t->next->expn) <
abs(q->expn))){
                                        q->next = t->next;
                                    t->next = q;
                                        break;
                        }
                }
                if(t == NULL){//遍历完没找到合适位置，直接插入
                        if(r->expn == q->expn) r->coef += q->coef;
                        else if(abs(r->expn) > abs(q->expn)) r->next = q;
                }
        printf("请输入x的系数和指数,输入 0空格0 时结束\n");
        scanf("%lf%d",&c,&e);
        }
        L->flag = NULL;
```

```
        return L;
}
```

显示多项式

选择显示多项式时，调用 `PrintPolyn` 函数，打印要显示的多项式 `poly[i]` 。其中要注意判断指数和系数是否为 1 ，如果是则不显示。

```
Status PrintPolyn(LinkList L,int i){//显示多项式,规范输出格式
        LinkList p;
        printf("%d号多项式:      ",i);
        if(!L->flag){
                p = L;
                while(p && p->coef == 0)//多项式为0
                        p = p->next;
                if(!p){
                        printf("0");
                        return 0;
                }
                while(L){//对多项式首项的处理
                        if(L->coef != 0){
                                if(L->expn == 0)        printf("%g",L->coef);//常数
                                else {
                                        if(L->expn == 1){       //指数为1不显示
                                                if(L->coef == 1) printf("x");
                                                else if(L->coef == -1) printf("-x");
                                                else    printf("%gx",L->coef);
                                        }
                                        else {
                                                if(L->coef == 1)
printf("x^%d",L->expn); //系数为1不显示
                                                else if(L->coef == -1) printf("-
x^%d",L->expn);
                                                else printf("%gx^%d",L->coef,L-
>expn);
                                        }
                                }
                                    break;
                        }
                        L = L->next;
                }
                L = L->next;
                while(L){
                        if((L->coef) > 0){//系数为正带+
```

```
                                if(L->expn == 0)        printf("+%g",L->coef);//常数
                                else {
                                        if(L->expn == 1){//指数为1不显示
                                                if(L->coef == 1)
printf("+x");//系数为1不显示
                                                else    printf("+%gx",L->coef);
                                        }
                                        else{
                                                if(L->coef == 1)
printf("+x^%d",L->expn);
                                                else printf("+%gx^%d",L->coef,L-
>expn);
                                        }
                                }
                        }
                        else if(L->coef < 0){//系数为负
                                if(L->expn == 0)        printf("%g",L->coef);
                                else {
                                        if(L->expn == 1){
                                                if(L->coef == -1)       printf("-
x");//系数为1不显示
                                                else    printf("%gx",L->coef);
                                        }
                                        else{
                                                if(L->coef == -1)       printf("-x^%d",L-
>expn);
                                                else printf("%gx^%d",L->coef,L->expn);
                                        }
                                        }
                        }
                L=L->next;
                }
                printf("\n");
        }
        else printf("多项式未定义\n");
}
```

删除与复制

根据链表的性质和所学线性表的销毁与复制不难得到这两个函数。

```
Status DestroyList(LinkList &L){//销毁多项式
        LinkList p,q;
        for(p = L;p;p = q){
```

```
                q = p->next;
                p->coef = 0;    p->expn = 0;
                free(p);
        }
        L->flag = 1;
        return OK;
}
```

```
Status CopyPolyn(LinkList L1,LinkList &L2){//多项式复制
        LinkList p,q,t;
    if(!(L1->flag)){
                L2->flag = NULL;
//标记有定义
                for(t = L2,p = L1;p != NULL;p = p->next){
                        q = (LinkList)malloc(sizeof(LNode));
                        if(!q)  exit(OVERFLOW);
                        q->coef = p->coef;
                        q->expn = p->expn;
                        q->next = p->next;
                        q->flag = NULL;
                        t->next = q;
                        t = t->next;
//复制
                 }
        }
        else printf("多项式未定义，请创建多项式。\n");
        return OK;
}
```

加法、减法

调用 `AddPolyn` 函数进行两个多项式的相加，比较相加多项式的指数，类比将两个线性表有序合并的操作，把两个多项式合并。

```
Status AddPolyn(LinkList L1,LinkList L2,LinkList &L3){//多项式求和
        LinkList p,q,t,r;
        if(!(L1->flag) && !(L2->flag)){//L1、L2都有定义
                L3->flag = NULL;                        //定义L3
                for(p = L1,q = L2,t = L3;(p != NULL)&&(q != NULL);){
                        r = (LinkList)malloc(sizeof(LNode));
                        if(!r)  exit(OVERFLOW);
                        if(p->expn > q->expn){//利用两链表的有序性
                                r->coef = p->coef;
                                r->expn = p->expn;
```

```
                r->next = NULL;
                r->flag = NULL;
                p = p->next;
                t->next = r;
                t = t->next;
        }
        else if(p->expn < q->expn){
                r->coef = q->coef;
                r->expn = q->expn;
                r->next = NULL;
                q = q->next;
                t->next = r;
                t = t->next;
        }
        else if(p->expn == q->expn){
                r->coef = p->coef + q->coef;
                r->expn = p->expn;
                r->next = NULL;
                p = p->next;
                q = q->next;
                t->next = r;
                t = t->next;
        }
}
if(!p){//p、q若有剩余
        for(;q != NULL;){
                r = (LinkList)malloc(sizeof(LNode));
                if(!r)  exit(OVERFLOW);
            r->coef = q->coef;
                r->expn = q->expn;
                r->next = NULL;
            q = q->next;
                t->next = r;
                t = t->next;
        }
}
else if(!q){
        for(;p != NULL;){
                r = (LinkList)malloc(sizeof(LNode));
                if(!r)  exit(OVERFLOW);
                r->coef = p->coef;
                r->expn = p->expn;
                r->next = NULL;
```

```
                                r->flag = NULL;
                                p = p->next;
                                t->next = r;
                                t = t->next;
                        }
                }
        }
        else printf("存在多项式未定义，请检查并创建多项式。\n");
        return OK;
}
```

减法的操作思路类似。

```
Status SubtractPolyn(LinkList L1,LinkList L2,LinkList &L3){//多项式求差
        LinkList p,q,t,r;
        if(!(L1->flag) && !(L2->flag)){//L1、L2都有定义
                L3->flag = NULL;                        //定义L3
                for(p = L1,q = L2,t = L3;(p != NULL) && (q != NULL);){
                        r = (LinkList)malloc(sizeof(LNode));
                        if(!r)  exit(OVERFLOW);
                        if(p->expn > q->expn){//利用有序性
                                r->coef = p->coef;
                                r->expn = p->expn;
                                r->next = NULL;
                                r->flag = NULL;
                                p = p->next;
                                t->next = r;
                                t = t->next;
                        }
                        else if(p->expn < q->expn){
                                r->coef = -q->coef;
                                r->expn = q->expn;
                                r->next = NULL;
                                q = q->next;
                                t->next = r;
                                t = t->next;
                        }
                        else if(p->expn == q->expn){
                                r->coef = p->coef - q->coef;
                                r->expn = p->expn;
                                r->next = NULL;
                                p = p->next;
                                q = q->next;
```

```
                                t->next = r;
                                t = t->next;
                        }
                }
                if(!p){//有剩余
                        for(;q != NULL;){
                                r = (LinkList)malloc(sizeof(LNode));
                                if(!r)  exit(OVERFLOW);
                            r->coef = -q->coef;
                                r->expn = q->expn;
                                r->next = NULL;
                                r->flag = NULL;
                            q = q->next;
                                t->next = r;
                                t = t->next;
                        }
                }
                else if(!q){
                        for(;p != NULL;){
                                r = (LinkList)malloc(sizeof(LNode));
                                if(!r)  exit(OVERFLOW);
                                r->coef = p->coef;
                                r->expn = p->expn;
                                r->next = NULL;
                                r->flag = NULL;
                                p = p->next;
                                t->next = r;
                                t = t->next;
                        }
                }
        }
        else printf("存在多项式未定义，请检查并创建多项式。\n");
        return OK;
}
```

乘法

```
LinkList MutiPolyn(LinkList L1, LinkList L2){//多项式相乘
    LinkList L3,t,k,r,q,w,p,o,m;
    if(!(L1->flag) && !(L2->flag)){
                t = m = (LinkList)malloc(sizeof(LNode));
                if(!m)  exit(OVERFLOW);
                m->coef = 0;
```

```
            m->expn = 0;
            m->next = NULL;
            m->flag = NULL;
            k = L1;
            while(k){
            r = L2;
            while(r){//相乘
            o = (LinkList)malloc(sizeof(LNode));
            if(!o)  exit(OVERFLOW);
            o->coef = k->coef * r->coef;
            o->expn = k->expn + r->expn;
            o->flag = NULL;
            m->next = o;
            m = o;
            r = r->next;
            }
            k = k->next;
    }
    m->next = NULL;
    L3 = (LinkList)malloc(sizeof(LNode));
    if(!L3) exit(OVERFLOW);
            L3->coef = 0;
            L3->expn = 0;
            L3->next = NULL;
            L3->flag = NULL;
            for(p = t;p != NULL;p = p->next){//排序
                    q = (LinkList)malloc(sizeof(LNode));
                    if(!q)  exit(OVERFLOW);
                    q->coef = p->coef;
                    q->expn = p->expn;
                    q->next = NULL;
                    if(L3->next == NULL){
                            if(q->expn == L3->expn)         L3->coef += q->coef;
                    else if(L3->expn < q->expn){
                                    q->next = L3;
                                    L3 = q;
                            }
                    else L3->next = q;
                    }
                    else{
                            for(w = L3;w->next != NULL;w = w->next){
                            if(q->expn == w->expn){
                                    w->coef += q->coef;
```

```
                                        break;
                                        }
                                else if((w->expn < q->expn) && (w == L3)){
                                                q->next = L3;
                                        L3 = q;
                                        break;
                                        }
                                else if((w->expn > q->expn) && (w->next->expn < q-
>expn)){
                                                q->next = w->next;
                                        w->next = q;
                                                break;
                                        }
                                }
                                if(w->next == NULL){
                                        if(w->expn == q->expn) w->coef += q->coef;
                                        else if(w->expn > q->expn) w->next = q;
                                }
                        }
                }
                return L3;
        }
        else printf("存在多项式未定义，请检查并创建多项式。\n");
}
```

求导

根据求导规则，修改数据域的值。

```
LinkList DiffPolyn(LinkList L1){//多项式求导
        LinkList L2 = NULL,p,q,t;
        if(!(L1->flag)){
                p = L2;
                q = L1;
                while(q){
                        t = (LinkList)malloc(sizeof(LNode));
                        if(!t)  exit(OVERFLOW);
                        t->flag = NULL;
                        t->next = NULL;
                        t->coef = q->coef * q->expn;
                        t->expn  =q->expn - 1;
                        if(L2 == NULL)  L2 = t;//头插
                        else    p->next = t;
                        p = t;
```

```
                        q = q->next;
                }
                p->next = NULL;
                return L2;
        }
        else printf("多项式未定义，请创建多项式。\n");
}
```

求值

调用 `ValuePolyn` 函数，直接把 x 的值代入计算即可。

```
Status ValuePolyn(LinkList L,double x){//多项式求值
        double n = 0,m = 0;
        if(!(L->flag)){
                while(L){
                        m = pow(x,L->expn);
                        n = n + (L->coef) * m;
                        L = L->next;
                }
                printf("所求多项式的值为%g\n",n);
        }
        else printf("多项式未定义，请创建多项式\n");
        return OK;
}
```

三、困难与改进

1. 在实验之初，考虑过做计算器仿真界面，最后迫于C语言的局限性与本人对编程语言掌握的程度不得不放弃这种想法。
2. 一开始为了配合设想中的仿真界面考虑采取字符串读入和输出。即使用者直接输入一个数学多项式，而不是只输入系数和指数。但是在读入字符串时需要对^号，运算符进行识别，同时还要根据数字的位置判断它是系数还是指数。在写完程序后发现bug实在太多难以修复，所以只能放弃这条思路，然后采用直接输入系数和指数的方法。

四、测试结果

经过测试，实验题目中要求的以下几组数据计算结果都符合预期。

1. $(2x+5x^8-3.1x^{11})+(7-5x^8+11x^9)=(-3.1x^{11}+11x^9+2x+7)$
2. $(6x^{-3}-x+4.4x^2-1.2x^9)-(-6x^{-3}+5.4x^2-x^2+7.8x^{15})=(-7.8x^{15}-1.2x^9+12x^{-3}-x)$
3. $(1+x+x^2+x^3+x^4+x^5)+(-x^3-x^4)=(1+x+x^2+x^5)$
4. $(x+x^3)+(-x-x^3)=0$
5. $(x+x^{100})+(x^{100}+x^{200})=(x+2x^{100}+x^{200})$

6. $\left(x+x^2+x^3\right)+0=x+x^2+x^3$
7. 互换上述测试数据中的前后两个多项式

五、实验收获

作为转系学生，我在大一下学期完全没有接触编程，这次实验很好地起到了热身的作用，让我温习了许多C语言的知识和实操。同时，我在各个函数的构造中也尽可能地结合了课程中有关线性表的一些算法，诸如有序合并，对理论课也起到了很好的复习作用。总体来说收获良多。