带有图形界面的一元稀疏多项式

大数据学院1班 和泳毅 PB19010450

一、实验要求

1. 一个命令行菜单如：

```
**********OPERATIONS**********
*****1.   menu           *****
*****2.   create         *****
*****3.   display        *****
*****4.   display all    *****
*****5.   add            *****
*****6.   sub            *****
*****7.   delete         *****
*****8.   mul            *****
*****9.   calculate      *****
*****10.  derivative     *****
*****11.  optimized dis  *****
**********OPERATIONS**********
```

选作功能顺序可以自定义，但基本功能的顺序请尽量和以上保持一致。

2. 输入并建立多项式。输入格式:$n \quad c_1 \quad e_1 \quad c_2 \quad e_2 \ldots c_n \quad e_n$其中$n$是输入的项数，$c_i$、$e_i$分别是第$i$项的系数和指数，$e_i$可能等于$e_j$，注意合并同类项，输入并不会按指数排序。$c_i$为实数，$e_i$为整数。你可以用一个数组$Polynomials$保存所有创建的多项式的头节点。
3. 输出多项式。输入格式：i。i表示第i个多项式，输出$Polynomials[i-1]$指向的多项式。输出形式为：$n \quad c_1 \quad e_1 \quad c_2 \quad e_2 \quad \ldots$其中$n$为多项式的项数，$c_i, e_i$分别为第$i$项的系数和指数，输出序列按指数降序。输出的多项式应为最简形式，应合并同类型，删除系数为0的项。若多项式为0则输出$1 \quad 0 \quad 0$.
4. 输出当前存储的所有多项式。
5. 多项式加法。输入格式：$i \quad j$。其中i表示第i个多项式，即$Polynomials[i-1]$指向的多项式。创建一个新的多项式作为结果，并输出。输出格式同3。
6. 多项式减法，同5。
7. 删除多项式。输入格式：i。i表示删除第i个多项式，即$Polynomials[i-1]$指向的多项式。需要释放掉该多项式的空间，并类似于顺序表的删除，$Polynomials[i]$存入$Polynomials[i-1]$，$Polynomials[i+1]$存入$Polynomials[i+2]$...。
8. 乘法。输入格式:$i \quad j$。表示$Polynomials[i-1] * Polynomials[j-1]$。创建一个新的多项式作为结果，并输出。
9. 求值或者求导。
10. 优化输出格式。实现优化后，有两种输出格式：基本输出格式和优化输出格式。可以两种输出都保留，也可以只使用优化格式。
11. 图形界面。如果实现图形界面，直接在图形界面测试。

二、设计思路

单链表的特点在于插入、删除时非常方便，不用估计空间规模，开辟空间方便，但不易做到随机读取，是一个动态性良好的结构。而本实验难点在于多项式的创建与打印，设计重点将放于上述内容。

1、结构部分

实验要求使用单链表结构，每个结点data域应至少包括系数、指数。而为了在链表存在而多项式不存在或多项式销毁的情况下，多项式运算、打印正常，增加flag标记值，true表示多项式空（未建立），false表示多项式非空（已建立）。

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

2、创建部分

为保证算法的正确性与健壮性，需要考虑到用户输入的各种可能性，这里只讨论用户在"正确输入"的情况下的各种可能：无序、同指数项、系数累加为零项...为实现此功能，运用了直接插入排序，使得用户输入后，链表所存多项式为指数绝对值降序的、最简的。

> **直接插入排序** (Straight Insertion Sort)是一种最简单的排序方法，它的基本操作是讲一个「记录」插入到已排好序的有序表中，从而得到一个新的、「记录」数增1的有序表。

下面用伪代码讲述大致思路：

```
Begin
初始化链表，将头结点flag标记有定义，头结点存数据
输入coef,expn,当输入0 0时结束，先接受第一个值
将用户在Inputbox窗口输入的宽字符转换为相应类型
while 输入的系数不为0 do
    将输入赋值于辅助结点q
    for 头结点 to 遍历完 by 指针后移 do//for从无到有始终有序
        找到合适插入q位置以保持有序性
    end for
    继续下一项的输入
end while
返回头结点
End
```

> 该链表的直插法中，辅助结点q代表「监视哨」，但只用于比较与插入，不必用于判断越界。expn为用于比较的关键字。当待排序列中「记录」关键字非递减有序，比较次数达最小值$n-1$（即$\sum_{i=2}^{n}1$）;关键字非递增有序，比较次数达最大值$(n+2)(n-1)/2$ 即($\sum_{i=2}^{n}i$)，平均比较次数$\approx n^2/4$，时间复杂度为$O(n^2)$。

3、运算部分

多项式的加减乘法、求导求值等都是基于对链表data域的操作，以及生成新链表的存储。

下面用伪代码讲述大致思路：

(1) 加法

```
Begin
最先判断L1 L2上的多项式有定义，再将L3上的多项式定义
for L1 L2的头结点 to 其中之一遍历完 do
    利用链表有序性，指数相同系数累加，指数不同直接保留
    当其中之一遍历完，将另一个直接循环并入
end for
返回L3头结点
End
```

(2) 减法与加法同理

(3) 乘法

```
Begin
最先判断L1 L2上的多项式有定义，再将L3上的多项式定义
while L1没遍历完 do
    while L2没遍历完 do
        两项相乘，存入辅助链表L'
    end while
end while//直接依次相乘
for L'头结点 to 遍历完 by 指针后移 do
    直插法，一项一项插入L3，与创建多项式同理
end for
返回L3头结点
End
```

(4) 求导

```
Begin
最先判断L1上的多项式有定义，再将L2上的多项式定义
while L1非空 do
    系数*指数，指数减一
    头插法插入L2，指数降序不变
end while
返回L2头结点
End
```

e、求值

```
Begin
最先判断L上的多项式有定义
while L非空 do
    代入x值，对coef*x^expn累加
end while
将结果转化为宽字符
通过outtextxy显示在图像框中
End
```

4、打印部分

打印要考虑多种情况，且要优化输出格式:

```
Begin
最先判断L上的多项式有定义
if 所有结点系数、指数都为0 then
    打印 0
    return结束打印
end if
while L非空 do//处理首项
    if 系数不为0 then
        if 指数为0 then
            作为常数打印
        else then
            if 指数为1 then
                指数位不打印
            else then
                if 系数为1 then
                    系数位不显示
                else if 系数为-1 then
                    系数位显示 -
                else then
                     系数正常打印
        end if
        break
    end if
end while
while 继上述位置继续，L非空 do//处理剩余项
    if 系数>0 then
        if 指数为0 then
           打印 + coef
        else then
            if 指数为1 then
                if 系数为1 then
                    打印 +x
                eles then
                    打印 + coef x
            else then
                if 系数为1 then
                    打印 + x ^ expn
                eles then
                    打印 + coef x ^ expn

    else if 系数<0 then
        与系数为正同理
    end if
end while
将结果转化为宽字符
通过outtextxy显示在图像框中
End
```

5、总体结构

本实验完成前未了解结构与应用的隔离，所有具体函数都直接对链表操作。

故结构为：头文件、宏定义——定义结构体——各函数——主函数调用

三、关键代码讲解

分析本实验较为关键的为CreatPolyn、PrintPolyn、MutiPolyn、主函数四部分代码，采用代码+注释的方式讲解：

1、CreatPolyn

```
LinkList CreatPolyn() {//指数降序创建多项式，同序号重复创建会覆盖
    LinkList L, q, t, r = NULL;                          //辅助结点
    int e = 0, flag = 0, count = 0, i = 0, j = 0, k = 0;//变量初始化
    double c = 0;
    InitList(L);                          //初始化链表
    L->flag = NULL;                       //标记有定义
    wchar_t s1[100],s11[100],s12[100];  //宽字符串
    InputBox(s1, 100, _T("请输入x的系数和指数,都为0时结束"));//接受为宽字符串S1
    for (; s1[i] != 32 && s1[i] != '\0'&&i<100; i++)//分离系数
        s11[j++] = s1[i];
    s11[j] ='\0';
    i++;
    for (; s1[i] != '\0'&&i<100; i++)               //分离指数
        s12[k++] = s1[i];
    s12[k] = '\0';
    c = wcstod(s11,NULL);   //转换宽字符为浮点
    e = wcstol(s12,NULL,10);//转换宽字符为整型
    while (c) {             //系数不为零一直输入
        q = (LinkList)malloc(sizeof(LNode));
        if (!q) exit(OVERFLOW);
        q->coef = c;
        q->expn = e;
        q->next = NULL;
        for (t = L; t != NULL; r = t, t = t->next) {//直插遍历
            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))) {
                //在中间找到合适位置
                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;
        }
        i = 0, j = 0, k = 0;
        wchar_t s1[100],s11[100],s12[100];//再一次接受用户输入
        InputBox(s1, 100, _T("请输入x的系数和指数,都为0时结束"));
        for (; s1[i] != 32 && s1[i] != '\0' && i < 100; i++)//重复分离
            s11[j++] = s1[i];
        s11[j] = '\0';
        i++;
        for (; s1[i] != '\0' && i < 100; i++)
            s12[k++] = s1[i];
        s12[k] = '\0';
        c = wcstod(s11, NULL);//重复转换
        e = wcstol(s12, NULL, 10);
    }
    L->flag = NULL;
    return L;
}
```

2、PrintPolyn

```
Status PrintPolyn(LinkList L, int i,int x,int y) {  //显示多项式,规范输出格式
    LinkList p;
    TCHAR s[1000],s1[10000];              //宽字符串
    char poly[100] = "\0",poly1[100] = "\0";
    int n = 0;
    swprintf_s(s, _T("%d"), i);           //6-9行打印第几号多项式
    settextstyle(25, 0, _T("微软雅黑"));
    outtextxy(x-10, y,s);
    outtextxy(x, y, _T("号多项式："));
    if (!L->flag) {//多项式有定义
        p = L;
        while (p && p->coef == 0)
            //遍历多项式，如果系数全为0，则多项式为0
            p = p->next;
        if (!p) {
            strcat(poly,"0");
            return 0;
        }
        while (L) {                       //对多项式首项的处理
        //不断地将多项式归并在一个字符串上
            if (L->coef != 0) {           //常数
                if (L->expn == 0) {
                    gcvt(L->coef, 3, poly1); //浮点数转换成字符串
                    strcat(poly, poly1); //字符串连接
                }
                else {
                    if (L->expn == 1) { //指数为1不显示
                        if (L->coef == 1)   strcat(poly, "x");
                        //系数为1不显示1
                        else if (L->coef == -1)  strcat(poly, "-x");
                        //系数为-1显示负号
                        else {//其他系数
```

```
                            gcvt(L->coef, 3, poly1);
                            strcat(poly, poly1);
                            strcat(poly, "x");
                        }
                    }
                    else {//指数不为1
                        if (L->coef == 1) { //系数为1不显示
                            strcat(poly, "x^");
                            itoa(L->expn, poly1,10); //整型转换成字符串
                            strcat(poly, poly1);
                        }
                        else if (L->coef == -1) { //系数为-1显示负号
                            strcat(poly, "-x^");
                            itoa(L->expn, poly1,10);
                            strcat(poly, poly1);
                        }
                        else { //其他通用情况
                            gcvt(L->coef, 3, poly1);
                            strcat(poly, poly1);
                            strcat(poly, "x^");
                            itoa(L->expn, poly1,10);
                            strcat(poly, poly1);
                        }
                    }
                }
                break;//处理完第一个非空结点就退出
            }
            L = L->next;
        }
        L = L->next;//下一个结点
        while (L) {//打印剩下所有结点
            if ((L->coef) > 0) {            //系数为正带+
                if (L->expn == 0) {         //常数
                    strcat(poly, "+");
                    gcvt(L->coef, 3, poly1);
                    strcat(poly, poly1);
                }
                else {                      //指数不为0
                    if (L->expn == 1) {     //指数为1不显示
                        if (L->coef == 1)    strcat(poly, "+x");//系数为1不显示
                        else {
                            strcat(poly, "+");
                            gcvt(L->coef, 3, poly1);
                            strcat(poly, poly1);
                            strcat(poly, "x");
                        }
                    }
                    else {                  //指数不为1
                        if (L->coef == 1){  //系数为1不显示
                            strcat(poly, "+x^");
                            itoa(L->expn, poly1, 10);
                            strcat(poly, poly1);
                        }
                        else {
                            strcat(poly, "+");
                            gcvt(L->coef, 3, poly1);
                            strcat(poly, poly1);
```

```
                            strcat(poly, "x^");
                            itoa(L->expn, poly1, 10);
                            strcat(poly, poly1);
                        }
                    }
                }
            }
            else if (L->coef < 0) {     //系数为负，自带-
                if (L->expn == 0) {     //常数
                    gcvt(L->coef, 3, poly1);
                    strcat(poly, poly1);
                }
                else {                  //指数不为0
                    if (L->expn == 1) {
                        if (L->coef == -1)  strcat(poly, "-x"); //系数为-1
                        else {
                            gcvt(L->coef, 3, poly1);
                            strcat(poly, poly1);
                            strcat(poly, "x");
                        }
                    }
                    else {              //指数不为1
                        if (L->coef == -1) { //系数为-1
                            strcat(poly, "-x^");
                            itoa(L->expn, poly1, 10);
                            strcat(poly, poly1);
                        }
                        else {
                            gcvt(L->coef, 3, poly1);
                            strcat(poly, poly1);
                            strcat(poly, "x^");
                            itoa(L->expn, poly1, 10);
                            strcat(poly, poly1);
                        }
                    }
                }
            }
            L = L->next;
        }
        outtextxy(820, 620, _T("返回"));//返回按钮
        Char2TCHAR(poly, s1);//字符串转换为宽字符
        settextstyle(25, 0, _T("微软雅黑"));//字体设置
        outtextxy(x+200, y, s1);//在(x+200,y)显示宽字符
    }
    return OK;
}
```

3、MutiPolyn

```
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)) {//参与乘法的多项式有定义
        t = m = (LinkList)malloc(sizeof(LNode));
        if (!m) exit(OVERFLOW);
        m->coef = 0;
```

```
        m->expn = 0;
        m->next = NULL;
        m->flag = NULL;//初始化辅助结点
        k = L1;
        while (k) {//直接相乘存入辅助链表
            r = L2;
            while (r) {
                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;
        }//得到乘法结果，此时辅助链表data域混乱无序
        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) {//与建表操作【一模一样】的直插排序
            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;
                }
            }
        }
        L3->flag = NULL;
        return L3;
    }
    return NULL;//存在参与乘法的多项式未定义，返回空值
}
```

4、主函数

```
int main() {
    initgraph(1000, 700);    //设置窗口的大小；
    setbkcolor(WHITE);       //设置背景色为白色；
    cleardevice();
    calcul_memu();           //菜单界面函数
    LinkList poly[10];
    int i, j, k, n, d, o, u, y;
    double m = 0, x = 0, a, b;
    MOUSEMSG M;              //鼠标
    for (i = 0; i < 10; i++)
        InitList(poly[i]);
    while (1){
        setcolor(DARKGRAY); //选择颜色
        settextstyle(60, 20, _T("幼圆"));
        M = GetMouseMsg();//获取鼠标点击参数
        switch (M.uMsg){
        case WM_LBUTTONDOWN:
            if (M.x > 190 && M.x < 370 && M.y>110 && M.y < 170){
                //190<x<370 && 110<y<170范围内触发点击，执行
                wchar_t s1[100],s11[100];
                InputBox(s1, 100, _T("请输入要创建的多项式位置下标0~9"));
                //接收为宽字符串
                o = 0;
                for (u = 0; s1[u] != '\0' && u < 100; u++)//分离
                    s11[o++] = s1[u];
                s11[o] = '\0';
                i = wcstol(s11, NULL, 10);//转换
                poly[i] = CreatPolyn();
            }
            if (M.x > 590 && M.x < 770 && M.y>110 && M.y < 170){
                initgraph(1000, 700);      //设置新窗口的大小；
                setbkcolor(DARKGRAY);      //背景颜色
                cleardevice();
                settextcolor(WHITE);
                setbkmode(TRANSPARENT);
                wchar_t s2[100],s21[100];
                InputBox(s2, 100, _T("请输入要显示的多项式位置下标0~9"));
                o = 0;
                for (u=0; s2[u] != '\0' && u < 100; u++)//分离
                    s21[o++] = s2[u];
                s21[o] = '\0';
                i = wcstol(s21, NULL, 10);
```

```
                PrintPolyn(poly[i], i,110,100);
            }
        ......//其他调用同理
    }
    return 0;
}
```

四、调试分析

1、需求分析

(1) 程序所能达到的基本可能：本程序可以由用户创建多个稀疏多项式，并实现多项式的加法、减法、乘法，也可以实现多项式的n阶求导、求值、定积分运算、幂运算，最后也能完成多项式的复制与打印；

(2) 输入的形式及输入值范围： 在菜单输入0-9的整型数，建立多项式以'系数"空格"指数'的形式输入，其中系数为双精度浮点型，指数为正的整型数，当系数与指数都为0时结束建立；

(3) 输出的形式：输出以双精度浮点型的系数、字符'x'、字符'^'、正的整型的指数构成多项式形式，特别地，在求值运算与定积分运算中，输出双精度浮点型数；

(4) 测试数据要求：系数为双精度浮点型，指数为正的整型，求导的阶数与幂都为正的整型。

2、时空分析

多项式算法的时间复杂度和空间复杂度与多项式的长度有关，即与链表的结点个数有关。

CreatPolyn、CopyPolyn原理相似，每一个结点遍历一次，又生成同等结点数，所以时间复杂度和空间复杂度都为*O*(n)；

PrintPolyn、ValuePolyn不需要申请额外空间，只用遍历每一个结点，所以时间复杂度为*O*(n)，空间复杂度为*O*(1)；

AddPolyn、SubtractPolyn、DiffPolyn、IntegralPolyn需要额外的空间供新的多项式来运算与存储，同时又访问每一个结点，时间复杂度为*O*(n)，空间复杂度为*O*(n)，其中对于AddPolyn、SubtractPolyn来说，它们时间复杂度中的n = max[min（n_1, n_2），abs($n_1 - n_2$)]；

MutiPolyn需要使一个多项式的各单项与另一个多项式相乘，最后将结果相加，并且也需要额外空间来存储，所以时间复杂度为*O*(m*n)，空间复杂度为*O*(m*n)，m、n分别为两多项式长度;

最后，PowPolyn通过调用MutiPolyn来实现，不仅与多项式长度有关，还与幂的次数有关，时间复杂度为*O*(a*m*n)，空间复杂度为*O*(a*m*n), a表示幂的次数。

3、问题与体会

(1) 多项式可以用带头结点的单链表表示。

(2) 由于程序在执行时，需要操作多个多项式链表，采用数组存储这些链表的头指针。

(3) 用户界面可以采用菜单驱动方式，以方便操作。

(4) 注意输出格式，指数为0时只显示系数，系数或指数为1时不显示，首项为正时不输出'+'。

(5) 系数采用%g输出符合常规位数保留习惯。

(6) 结构体增加flag项，用来标记某序号多项式是否有定义，创建时记为0即有定义，销毁或清除时为1即无定义。

(7) 适当应用辅助结点，更加方便。

(8) 计算多项式的幂时可以反复应用多项式的乘积。

(9) 销毁多项式可以循环应用多项式的清除,多项式的n阶导可以循环应用多项式的一阶导。

(10)图形界面制作中，输入Inputbox只能接收为宽字符，且显示outtextxy也需要宽字符，需要做相应转换。

(11)确定鼠标点击范围前，可以画框确定坐标。

(12)VS中可以应用EasyX库完成图形界面设计。

五、代码测试

菜单界面

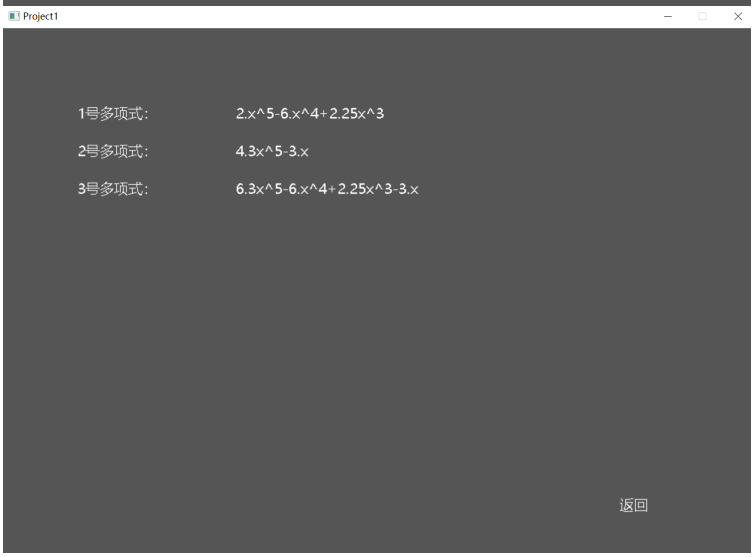
创建多项式

显示多项式


Project1
1号多项式:
2.x^5-6.x^4+2.25x^3
返回

多项式求和


Project1
多项式计算器
创建多项式
显示多项式
所有多项式
多项式求和
复制多项式
多项式求导
多项式求值
多项式的定积分
多项式的n次幂
退出计算器
Project1
请输入要求和的两个多项式位置下标0~9
确定
1 2


Project1
1号多项式:
2.x^5-6.x^4+2.25x^3
2号多项式:
4.3x^5-3.x
3号多项式:
6.3x^5-6.x^4+2.25x^3-3.x
返回

多项式求差


Project1
多项式计算器
创建多项式
显示多项式
所有多项式
多项式求和
多项式求
项式
多项式求积
清空多项式
复制多项式
多项式求导
多项式求值
多项式的定积分
多项式的n次幂
退出计算器
Project1
请输入要求差的两个多项式位置下标0~9
确定
1 2

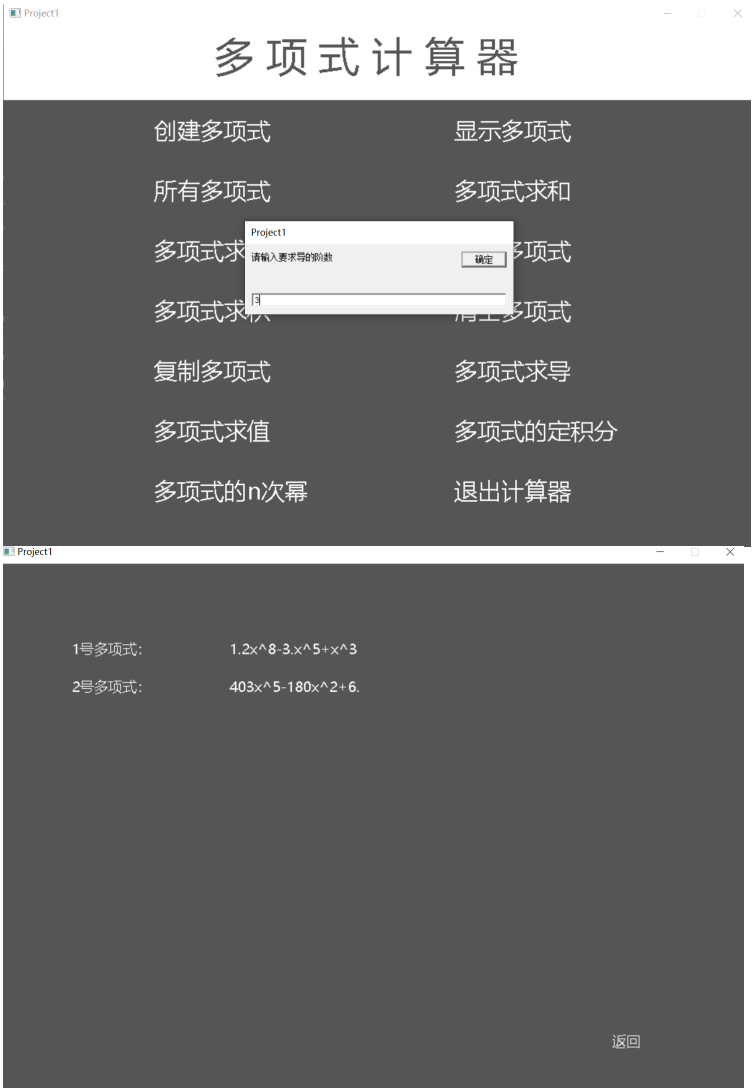
Project1
1号多项式:
2.x^5-6.x^4+2.25x^3
2号多项式:
4.3x^5-3.x
3号多项式:
6.3x^5-6.x^4+2.25x^3-3.x
4号多项式:
-2.3x^5-6.x^4+2.25x^3+3.x
返回

多项式求积


Project1
多项式计算器
创建多项式
显示多项式
所有多项式
多项式求和
多项式求
项式
多项式求
清空多项式
复制多项式
多项式求导
多项式求值
多项式的定积分
多项式的n次幂
退出计算器
Project1
请输入要求积的两个多项式位置下标0-9
确定
1 2

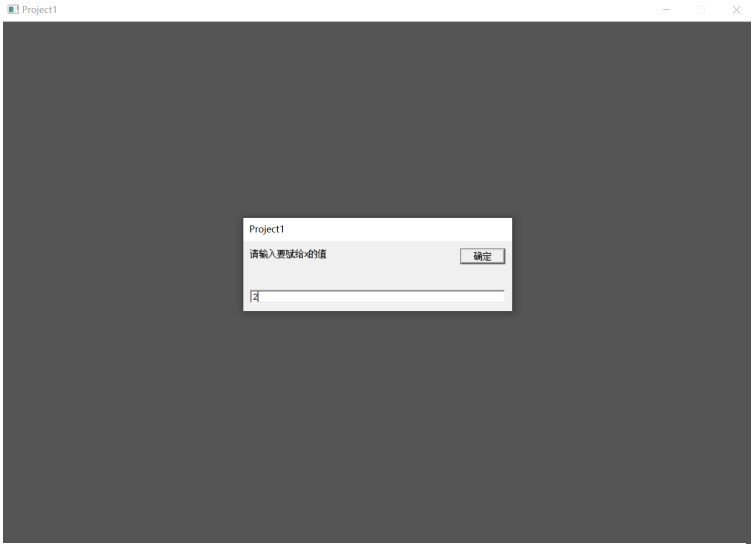
Project1
1号多项式:
2.x^5-6.x^4+2.25x^3
2号多项式:
4.3x^5-3.x
3号多项式:
6.3x^5-6.x^4+2.25x^3-3.x
4号多项式:
-2.3x^5-6.x^4+2.25x^3+3.x
5号多项式:
8.6x^10-25.8x^9+9.67x^8-6.x^6+18.x^5-6.75x^4
返回

多项式求导


Project1
多项式计算器
创建多项式
显示多项式
所有多项式
多项式求和
多项式求
项式
多项式求
清空多项式
复制多项式
多项式求导
多项式求值
多项式的定积分
多项式的n次幂
退出计算器
Project1
请输入要求导的阶数
确定
3

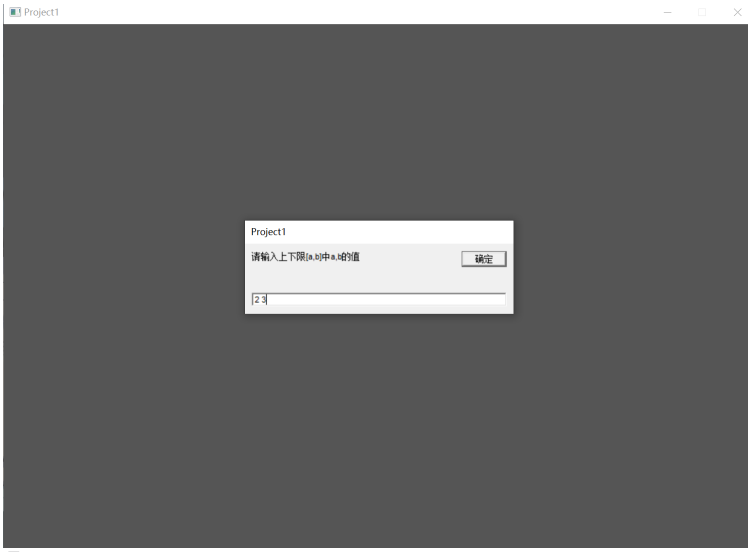
Project1
1号多项式:
1.2x^8-3.x^5+x^3
2号多项式:
403x^5-180x^2+6.
返回

多项式求值


Project1
Project1
请输入要赋给x的值
确定
2


Project1
当x=2.时，多项式的值为：219.2
返回

多项式定积分



六、实验总结

通过本实验，加深了对链表操作的熟练和理解，学会了使用VS，并学会用EasyX库做图形界面。