《数据结构》上机报告

2018 年 10 月 10 日

姓名：刘思源 学号：1651390 班级：电子三班 得分：

<table>
<tr><td>实验题目</td><td colspan="2">链表的多项式应用</td></tr>
<tr><td>问题描述</td><td colspan="2">链表本身的特性使得其可以进行一些数学上的多项式的操作，如多项式的乘法，加法，和多项式的求值。</td></tr>
<tr><td rowspan="2">基本要求</td><td colspan="2">1. (p1)实现两个多项式的加法。
2. (p2) 实现两个多项式的乘法。
3. (p3)实现多项式的求值</td></tr>
<tr><td>已完成基本内容（序号）:</td><td>1，3</td></tr>
<tr><td rowspan="2">选做要求</td><td colspan="2"></td></tr>
<tr><td>已完成选做内容（序号）</td><td>2</td></tr>
<tr><td>数据结构设计</td><td colspan="2">

```
typedef struct LNode{
    ElemType coef;
    ElemType exp;
    struct LNode *next;
    LNode(ElemType x, ElemType y) :coef(x), exp(y),next(nullptr) {}
 } LNode,*LinkList;
```

</td></tr>
</table>

功能(函数)说明	1、 多项式链表的初始化 多项式链表一个节点包含两个数据域，底数和指数。初始化的时候同时进行初始化。 2、 多项式的加法 使用两个指针同时遍历两个多项式链表。如果此刻两个指针所指的指数域的值相同，则底数相加，指数不变，存入新的链表中，两个指针同时后移；如果不相等，则将指数较小者存入新链表，该指针向后移，直到某一个链表为空。将另一不为空的链表（如果存在），剩余的元素存储进新链表。

```
void initial(LinkList &L, ElemType n)
{
    LNode *p;
    L = new LNode(0, 0);
    p = L;
    while (n) {
        p ->next= new LNode(0, 0);
        p = p->next;
        cin >> p->coef >> p->exp;
        n--;
    }
    p->next = nullptr;
   }
```

```
void add(LinkList L1, LinkList L2, LinkList &L3)
{
    LNode *p,*p1,*p2;
    L3 = new LNode(0, 0);
    p1 = L1->next;
    p2 = L2->next;
    p = L3;

    while (p1 && p2) {
        if (p1->exp > p2->exp) {

            p->next = new LNode(p2->coef, p2->exp);
            p = p->next;
            p2 = p2->next;
        }
        else if (p1->exp < p2->exp) {

            p->next = new LNode(p1->coef, p1->exp);
            p = p->next;
            p1 = p1->next;
        }
        else {

            p->next = new LNode(p1->coef+p2->coef, p1->exp);
```

```
            p = p->next;
            p1 = p1->next;
            p2 = p2->next;
        }
    }
    while (p1) {

        p->next = new LNode(p1->coef, p1->exp);
        p = p->next;
        p1 = p1->next;
    }
    while (p2) {

        p->next = new LNode(p2->coef, p2->exp);
        p = p->next;
        p2 = p2->next;
    }
}
```

3、 多项式的乘法

多项式的乘法可以理解为若干个多项式的加法。不妨设有多项式 A 和多项式 B 相乘。A 的每一个单项式和 B 中的所有单项式底数相乘，指数相加，可以得到一个新的多项式。如此可以得到 n 个多项式（n 的个数为 A 的单项式个数），再将这 n 个多项式相加。

在实际的实现中，每产生一个新的多项式，就立刻将他和目标结果的多项式相加，来减少空间的复杂度。

```
void mutiply(LinkList L1, LinkList L2, LinkList &L3)
{
    LNode *p, *p1, *p2,*t;

    L3 = new LNode(0, 0);

    p = L3;
    p1 = L1->next;
    p2 = L2->next;

    while (p1) {
        LinkList Lt;
        Lt = new LNode(0, 0);
        t = Lt;
        while (p2) {
            t->next = new LNode(p1->coef*p2->coef, p1->exp+p2->exp);
            t = t->next;
            p2 = p2->next;
        }
```

```
            p2 = L2->next;
            add(Lt, L3, L3);
            p1 = p1->next;
        }

    }
```

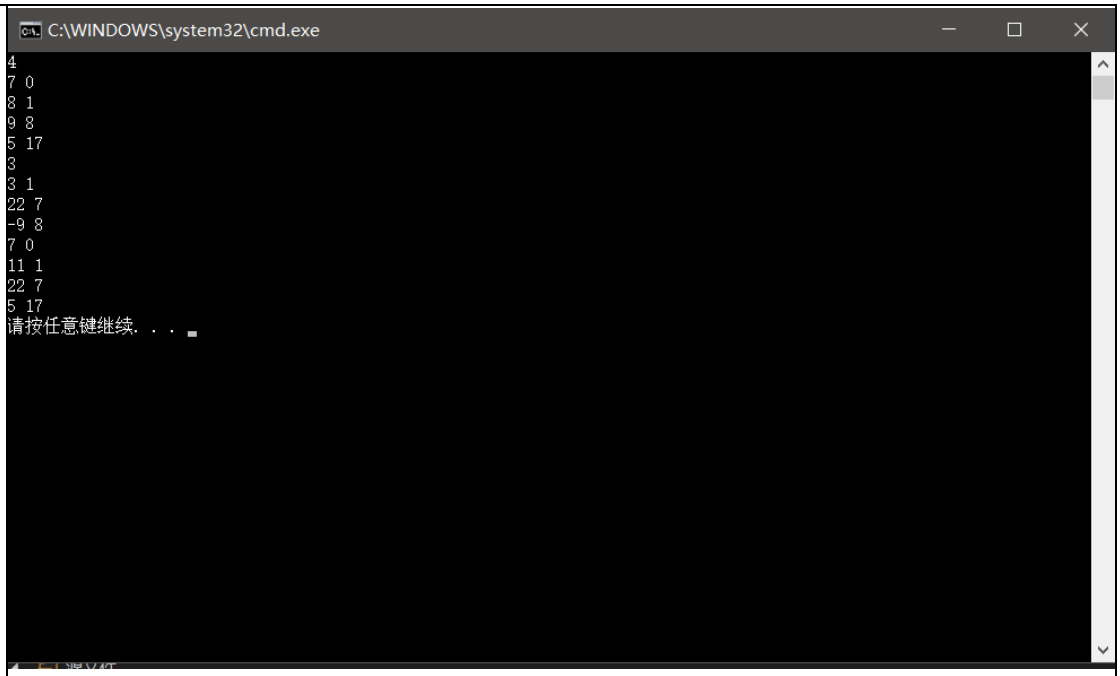
4、 多项式的求值

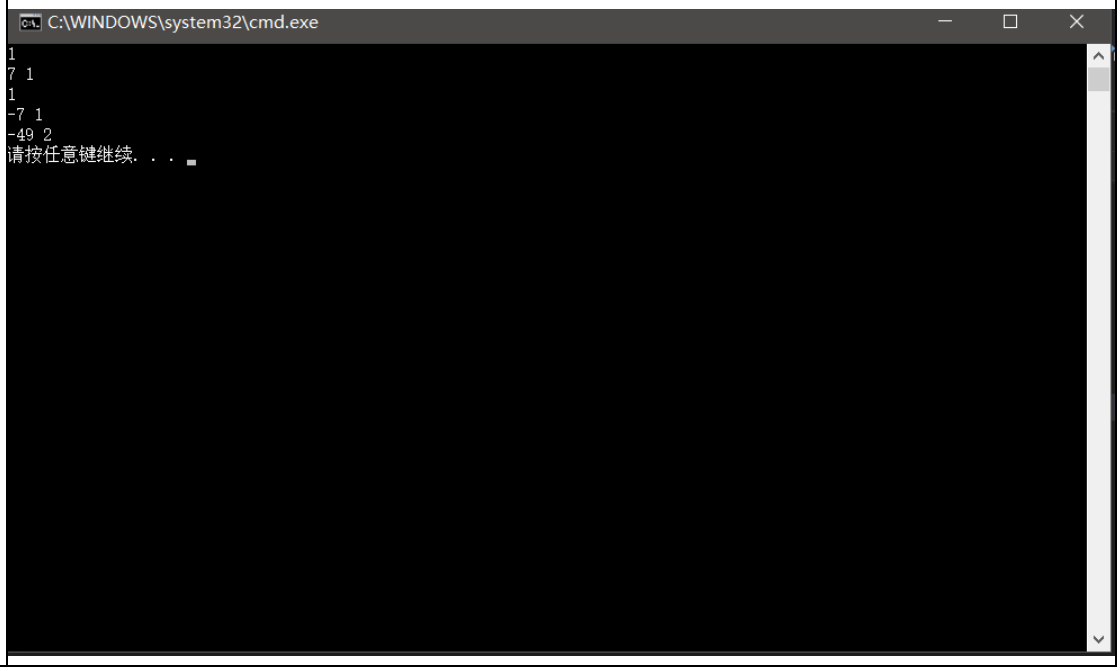
多项式的求值可以使用 pow 函数，但该函数时间较慢，不如直接使用循环进行运算。

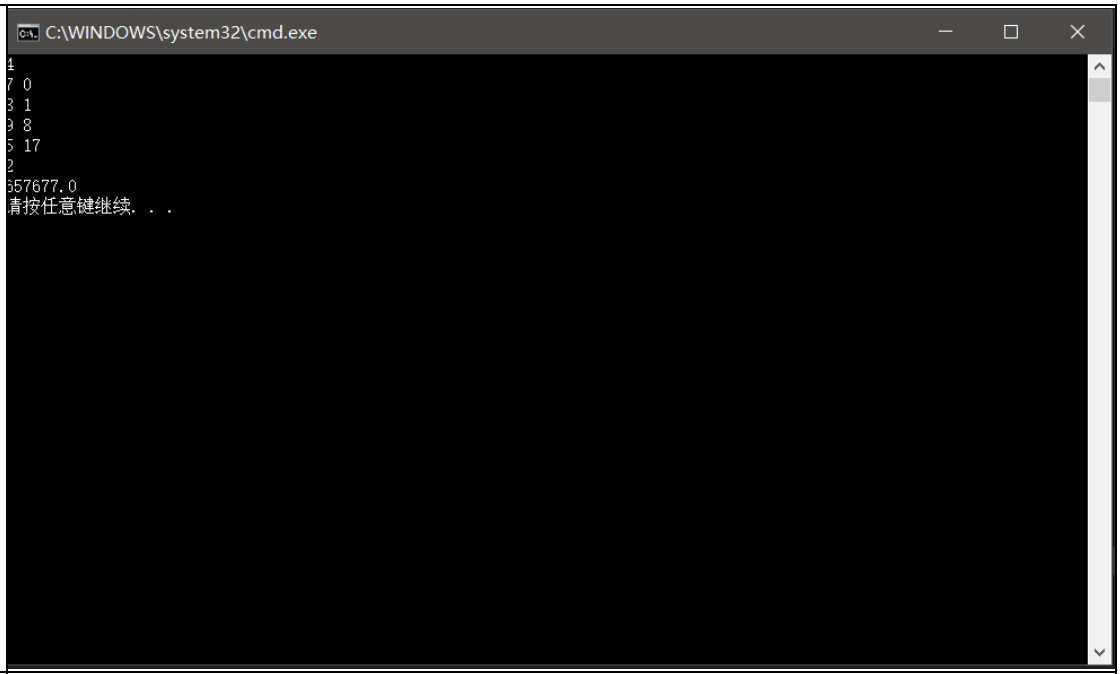
```
Status calculate(LinkList L, double x)
{
    double res = 0;
    int exp = 0;
    LNode *p;
    p = L->next;
    while (p) {
        double x1 = 1;
        if (p->exp == 0) {
            x1 = 1;
        }
        else {
            for (exp = p->exp ; exp > 0; exp--) {
                x1 *= x;
            }
        }
        x1 *= p->coef;
        res += x1;
        p = p->next;

    }
    return res;
   }
```

开发环境	Visual studio 2017

调试分析


C:\WINDOWS\system32\cmd.exe
4
7 0
8 1
9 8
5 17
3
3 1
22 7
-9 8
7 0
11 1
22 7
5 17
请按任意键继续. . .


C:\WINDOWS\system32\cmd.exe
1
7 1
1
-7 1
-49 2
请按任意键继续. . .

<table>
<tr><td></td><td> 
</td></tr>
<tr><td>心得体会</td><td>1、 在第三题的求值中，因为题目数据类型为 double 型，不太可能会有指数过大的情况出现。所以使用连乘的方式进行计算，可以稍微的快于 pow 函数。
2、 在进行多项式乘法时，由于直接进行乘法十分复杂，可以将其转化为第一问中已经解决的问题进行求解。同时在乘的同时进行叠加运算，可以减少空间的复杂度。</td></tr>
</table>