数据结构实验报告

**实验一：一元多项式的加减**

姓 名： 彭 思 翔

学 号： 2017301500061

班 级： 计科二班

日 期：2018年10月21日

上机环境：Win10 VSCode

**一、实验题目**

实验一：一元多项式的加减运算

【问题描述】

请编写一个程序，完成一元多项式的存储，并实现两个多项式的相加减及相乘运算。

【基本要求】

（1）随机从键盘上输入多项式，构建单链表存储医院多项式，其中单链表的数据域包括系数和指数两项

（2）最后的结果输出形式为result=3+4X^2+5X^2

**二、实验项目的目的**

深入理解单链表并进行应用，顺便进行字符串处理，学会合理地设置函数，使函数复用率高。

**三、实验项目程序结构**

printPolynomial

addPolynomial

subPolynomial

initPolynomial

init

main

mulPolynomial

insertPolynomial

readPolynomial

destoryPolynomial

**四、实验项目中各文件函数功能描述**

void init(); //读入数据，控制进程

void initPolynomial(Polynomial\*&); //初始化多项式单链表

void readPolynomial(Polynomial\*&); //读取多项式，进行字符串处理

void printPolynomial(Polynomial\*); //打印多项式

void destoryPolynomial(Polynomial\*&); //销毁多项式单链表

void insertPolynomial(Polynomial\*&, Polynomial\*&); //插入排序单式至多项式的正确位置

void addPolynomial(Polynomial\*&, Polynomial\*&, Polynomial\*&); //多项式加法，由第三个参数返回结果

void subPolynomial(Polynomial\*&, Polynomial\*&, Polynomial\*&); //多项式减法，由第三个参数返回结果

void mulPolynomial(Polynomial\*&, Polynomial\*&, Polynomial\*&); //多项式乘法，由第三个参数返回结果

**五、算法描述**

【数据结构】

有序单链表：用于存储和表示指数由高到低的多项式，链表中存储指数、系数和下一项的地址，只存储系数不为0的项。

typedef struct Poly {

double coefficient, index;

struct Poly \* next;

}Polynomial;

【设计思路】

（1）读取实现：对于一个多项式的字符串，以加号分割，加号后第一个为数字则为系数，否则系数为1,字符‘x’后若为‘^’则指数为其后面的数字，否则指数为1。对于数，遇见连续数字在小数点前则前面的数×10加上当前的数字，在小数点后则前面的数+当前的数字/小数位数。

时间复杂度O(Lenth)

（2）插入实现：插入排序，从前至后找到比需要插入单式的指数更小的位置前一个为止，若指数相等则合并，若指数不相等则插入。

时间复杂度O(n)

void insertPolynomial(Polynomial \*&L, Polynomial \*&x) {

Polynomial \*p = L;

while (p下一个不为空 && p下一个指数大于x的指数) p = p->next;

if (p与x指数相等 && p != L) 合并系数; else 插入x至p后;

}

（3）加法实现：两个多项式归并，指数上若A[i]>B[j]则i++，若A[i]<B[j]则j++，若A[i]==B[j]则相加系数并i++j++。

时间复杂度O(m+n)

void addPolynomial(Polynomial \*&A, Polynomial \*&B, Polynomial \*&C) {

Polynomial \*p = A->next, \*q = B->next, \*r = C, \*s;

while (p, q不为null) {

if (p->index > q -> index) 插入p至C, p后移;

else if (p->index < q -> index) 插入q至C, q后移;

else if (p->index == q -> index) 系数相加不为0则插入至C, p和q后移;

}

while (p枚举剩余A) p加入C;

while (q枚举剩余B) q加入C;

}

（4）减法实现：两个多项式归并，指数上若A[i]>B[j]则i++，若A[i]<B[j]则j++，若A[i]==B[j]则相减系数并i++j++。代码和加法类似。

时间复杂度O(m+n)

（5）乘法实现：暴力枚举两个多项式的每一位，并相乘，将结果项插入至新的多项式。

时间复杂度O(m\*n^2)

void mulPolynomial(Polynomial \*&A, Polynomial \*&B, Polynomial \*&C) {

Polynomial \*p = A->next, \*q = B->next, \*r = C, \*s;

while (p枚举A)

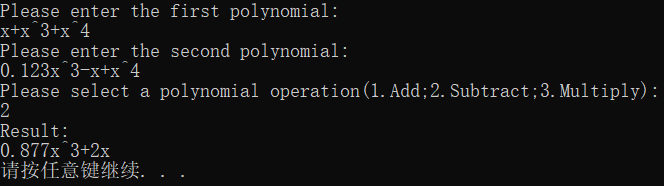
while (q枚举B)

系数相乘，指数相加，并插入C；

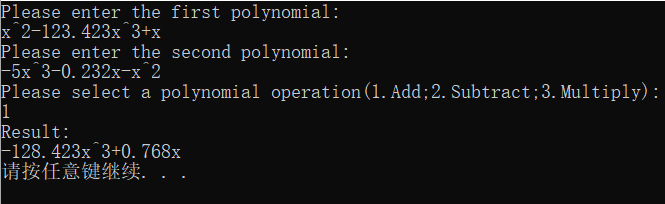
}

**六、实验数据和实验结果分析**

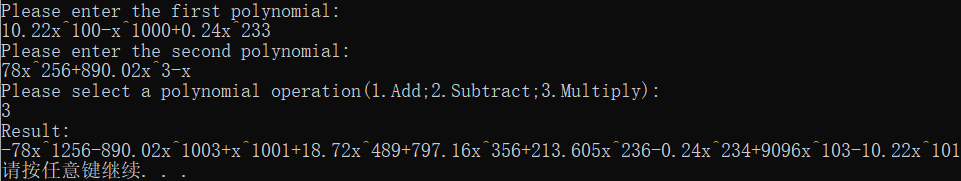
相加：



相减：



相乘：



运行结果良好。

**七、实验体会**

对单链表有了更加深入的认识，熟悉了二路归并等相关算法。运算的实现倒还算是简单，处理输入的字符串却废了不少功夫，比如系数指数省略，负数、小数的处理等，看来对字符串的操作要更熟悉一点才行。另外乘法运算过于暴力，暂时未想出以当前数据结构的更优解决方案。