比较结点的系数大小函数

int cmp(Term a, Term b) {

if (a.expn<b.expn) return -1;

else if (a.expn == b.expn) return 0;

else return 1;

}

将结点插入多项式链表的适当位置

void insertNode(LinkList &L, Term e) {

ploynomial\* q = L;

while (q->next != NULL) {

if (cmp(q->next->term, e)<0)

q = q->next;//q指向下一个结点

else break;//当前项的指数已经大于或等于要插入的项指数

}

if (q->next != NULL&&cmp(q->next->term, e) == 0) {//指数相同，系数相加

q->next->term.coef += e.coef;

}

else {

ploynomial\* node = (ploynomial\*)malloc(sizeof(ploynomial));

node->term.coef = e.coef;

node->term.expn = e.expn;

if (q->next == NULL)

node->next = NULL; //如果q结点为尾结点，则node的指针域设为NULL

else

node->next = q->next; //否则node的指针域指向q的下一个结点

q->next = node;

}

}

输入m项的系数和指数，建立表示一元多项式的有序链表L

void CreatPolyn(LinkList &L, int m) {

Term e;

InitList(L);

printf("\n输入%d项的系数和指数:", m);

for (int i = 1; i <= m; i++) {

printf("\n第%d项的系数和指数：", i);

scanf("%f%d", &e.coef, &e.expn);

insertNode(L, e);

}

}

void addPolyn(LinkList &L1, LinkList &L2) {

//完成多项式相加运算，即L1=L1+L2 ，并销毁一元多项式L2

ploynomial\* q;

for (q = L2->next; q != NULL; q = q->next) {

insertNode(L1, q->term);//将L2的每一项插入到L1中

}

free(L2);

}

完成多项式相减运算，即L1=L1-L2 ，并销毁一元多项式L2

减法的实现和加法的实现非常类似，就是把多次项的系数取相反数，然后在求和

void SubtracatPolyn(LinkList &L1, LinkList &L2) {

ploynomial\* q;

for (q = L2->next; q != NULL; q = q->next) {

q->term.coef = -(q->term.coef);

insertNode(L1, q->term);//将L2的每一项插入到L1中

}

free(L2);

}

void visitList(LinkList L) {

ploynomial\* q = L;

printf("\n");

while (q->next != NULL) {

q = q->next;

if (q->term.coef>0)

printf("+%fX^%d", q->term.coef, q->term.expn);

else printf("%fX^%d", q->term.coef, q->term.expn);

}

}

主函数

int main() {

LinkList L1, L2;

int n1, n2;

printf("请输入多项式L1的项数：");

scanf("%d", &n1);

CreatPolyn(L1, n1);

printf("请输入多项式L2的项数：");

scanf("%d", &n2);

CreatPolyn(L2, n2);

printf("\n多项式L1：");

visitList(L1);

printf("\n多项式L2：");

visitList(L2);

printf("\n相加:");

addPolyn(L1, L2);

//printf("\n相减:"); 相加和相减不能连续使用，因为执行相减或相加元素L2会被销毁。需要重新创建多项式L2

//SubtracatPolyn(L1,L2);

visitList(L1);

}

