**测试用例**

按照降幂输入数据

第一个多项式

1 4

1 2

1 0

a

第二个多项式

1 5

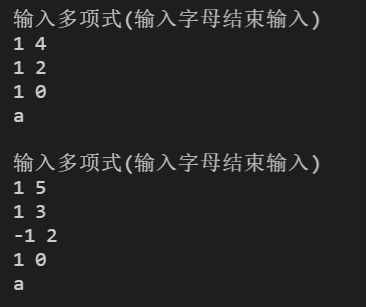
1 3

-1 2

1 0

a

进行加法和减法运算



**算法和数据结构设计**

1. 对多项式 a和 b分别建立链表，a作为第一个多项式，b作为第二个，建立新链表head\_sub和head\_add来存放a和b计算后的结果。
2. 进行加法和减法运算
3. 加法运算（函数 addition（） ），从a和b的头指针开始，比较每个结点的指数，在c中记录更大的结点的数据，同时，指向这个结点的指针向后挪动一位。若指数相同，在c中记录两者系数相加后的结果和指数，指向两个结点的指针都向后挪动一位。
4. 减法运算（函数 subtraction（） ），从a和b的头指针开始，比较每个结点的指数，若a中结点的指数更大，在c中记录这个结点的数据，同时，指向这个结点的指针向后挪动一位。若b中结点的指数更大，在c中记录这个结点的数据，但系数为相反数，指向这个结点的指针向后挪动一位。若两者指数相同，c中记录a结点的系数减去b结点的系数和指数，指向两者的指针都向后挪动一位。
5. （函数fre（） ）释放a、b链表的空间

#include <stdio.h>

#include <stdlib.h>

#define LEN sizeof(struct item)

typedef struct item{

    int x;

    float coeffi; //系数

    int index;   //指数

    struct item \*next;

}Item;

Item \*input();

Item \*addition(Item \*heada, Item \*headb);

Item \*subtraction(Item \*heada, Item \*headb); //链表a 减去链表b

void fre(Item \*head);

int main(){

    Item \*heada, \*headb, \*head\_add, \*head\_sub; //headc 是链表a 和b 相加后的结果

    heada = input();

    headb = input();

    head\_add = addition(heada, headb);  //进行加法运算

    head\_sub = subtraction(heada, headb); //进行减法运算

    fre(heada), fre(headb);

    return 0;

}

Item \*input(){

    Item \*p1, \*p2, \*head;

    head = p1 = (Item\*)malloc(LEN);

    head->next = NULL;

    p2 = (Item\*)malloc(LEN);

    if(p1 == NULL || p2 == NULL){

        printf("内存分配失败，程序终止\n");

        exit(-1);

    }

    printf("\n输入多项式(输入字母结束输入)\n");

    while(scanf("%f %d", &p2->coeffi, &p2->index) != 0){

        p1->next = p2;

        p1 = p2;

        p2 = (Item\*)malloc(LEN);

    }

    p2 = NULL;

    p1->next = p2;

    while (getchar() == '\n')

        continue;

    return head;

}

Item \*addition(Item \*heada, Item \*headb){

    Item \*p1c, \*p2c, \*pa, \*pb, \*head;

    head = p1c = (Item\*)malloc(LEN);

    head->next = NULL;

    if(p1c == NULL){

        printf("内存分配失败，程序终止\n");

        exit(-1);

    }

    pa = heada->next;

    pb = headb->next;

    do{

        p2c = (Item\*)malloc(LEN);

        if(pa->index > pb->index){

            p2c->index = pa->index;

            p2c->coeffi = pa->coeffi;

            pa = pa->next;

        }

        else if(pa->index < pb->index){

            p2c->index = pb->index;

            p2c->coeffi = pb->coeffi;

            pb = pb->next;

        }

        else{

            p2c->index = pb->index;

            p2c->coeffi = pb->coeffi + pa->coeffi;

            pa = pa->next;

            pb = pb->next;

        }

        p1c->next = p2c;

        p1c = p2c;

    }while(pa != NULL && pb != NULL);

    if(pa == NULL && pb != NULL){

        do{

            p2c = (Item\*)malloc(LEN);

            p2c->index = pb->index;

            p2c->coeffi = pb->coeffi;

            p1c->next = p2c;

            p1c = p2c;

            pb = pb->next;

        }while (pb != NULL);

    }

    else if(pb == NULL && pa != NULL){

        do{

            p2c = (Item\*)malloc(LEN);

            p2c->index = pa->index;

            p2c->coeffi = pa->coeffi;

            p1c->next = p2c;

            p1c = p2c;

            pa = pa->next;

        }while (pa != NULL);

    }

    p1c->next = NULL;

    return head;

}

Item \*subtraction(Item \*heada, Item \*headb){

    Item \*head, \*p1c, \*p2c, \*pa, \*pb;

    head = p1c = (Item\*)malloc(LEN);

    head->next = NULL;

    if(p1c == NULL){

        printf("内存分配失败，程序终止\n");

        exit(-1);

    }

    pa = heada->next;

    pb = headb->next;

    do{

        p2c = (Item\*)malloc(LEN);

        if(pa->index > pb->index){

            p2c->index = pa->index;

            p2c->coeffi = pa->coeffi;

            pa = pa->next;

        }

        else if(pa->index < pb->index){

            p2c->index = pb->index;

            p2c->coeffi = pb->coeffi\*(-1);

            pb = pb->next;

        }

        else{

            p2c->index = pb->index;

            p2c->coeffi = pa->coeffi - pb->coeffi;

            pa = pa->next;

            pb = pb->next;

        }

        p1c->next = p2c;

        p1c = p2c;

    }while(pa != NULL && pb != NULL);

    if(pa == NULL && pb != NULL){

        do{

            p2c = (Item\*)malloc(LEN);

            p2c->index = pb->index;

            p2c->coeffi = pb->coeffi\*(-1);

            p1c->next = p2c;

            p1c = p2c;

            pb = pb->next;

        }while (pb != NULL);

    }

    else if(pb == NULL && pa != NULL){

        do{

            p2c = (Item\*)malloc(LEN);

            p2c->index = pa->index;

            p2c->coeffi = pa->coeffi;

            p1c->next = p2c;

            p1c = p2c;

            pa = pa->next;

        }while (pa != NULL);

    }

    p1c->next = NULL;

    return head;

}

void fre(Item \*head){

    Item \*p1, \*p2;

    p1 = head->next;

    while(p1 != NULL){

        p2 = p1->next;

        free(p1);

        p1 = p2;

    }

    free(head);

    head = NULL;

}