## 1、

#include <stdio.h>

#include <stdlib.h>

#define Maxsize 1024

typedef int elemtype;

typedef struct

{

elemtype v[Maxsize];

int len;

}sqlist;

void init(sqlist \*L)

{

int i;

int n;

printf("请给定顺序表的长度：\n");

scanf("%d",&n);

printf("请初始化该顺序表：\n");

for(i=0;i<n;i++)

{

scanf("%d",&L->v[i]);

L->len++;

}

}

int lenth(sqlist \*L)

{

int lenth;

lenth=L->len;

return lenth;

}

void output(sqlist \*L)

{

int i;

for(i=0;i<lenth(L);i++)

printf("%d:%d\n",i,L->v[i]);

}

int insert(sqlist \*L, int i, elemtype x)

{

int j;

if(L->len==Maxsize)

{

printf("溢出\n");

return 0;

}

else

if((i<1)||i>L->len+1)

{

printf("插入位置不正确\n");

return 0;

}

else

{

for(j=L->len-1;j>=i-1;j--)

L->v[j+1]=L->v[j];

L->v[i-1]=x;

L->len=L->len+1;

return 1;

}

}

int search(sqlist \*L, int x)

{

int i;

for(i=0;i<L->len;i++)

if(x==L->v[i])

break;

if(i<L->len)

return (i+1);

else

return 0;

}

int dele1(sqlist \*L, int i, elemtype \*y)

{

int j;

if((i<1)||(i>L->len))

{

printf("删除位置不正确\n");

return 0;

}

else

{

\*y=L->v[i-1];

for(j=i;j<L->len;j++)

L->v[j-1]=L->v[j];

L->len=L->len-1;

return 1;

}

}

int dele2(sqlist \*L, elemtype x, elemtype \*y)

{

int j;

if(!search(L,x))

{

printf("无要删除的目标元素\n");

return 0;

}

else

{

\*y=L->v[search(L,x)-1];

for(j=search(L,x);j<L->len;j++)

L->v[j-1]=L->v[j];

L->len=L->len-1;

return 1;

}

}

elemtype geti(sqlist \*L,int i)

{

return L->v[i];

}

int main()

{

sqlist \*L;

elemtype y=0;

L=(sqlist \*)malloc(sizeof(sqlist));

L->len=0;

init(L);

output(L);

//printf("%d\n",geti(L,0));

//dele1(L,1,&y);

//dele2(L,3,&y);

//printf("%d\n",y);

//insert(L,1,0);

//insert(L,lenth(L)+1,0);

//insert(L,2,0);

if(search(L,2))

printf("存在,位置为%d\n",search(L,2));

else

printf("不存在，位置为%d\n");

output(L);

return 0;

}

## 2、

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define m 20

typedef struct node

{

int k;

char v[m];

struct node \* next;

}Lnode, \*linklist;

Lnode \*creat()

{

Lnode \*h, \*p;

h=(Lnode\*)malloc(sizeof(Lnode));

h->next=NULL;

while(1)

{

p=(Lnode\*)malloc(sizeof(Lnode));

scanf("%d%s",&p->k,p->v);

if(p->k==0)

{

free(p);

break;

}

p->next=h->next;

h->next=p;

}

return h;

}

void untwist(Lnode \*h)

{

Lnode \*p;

p=h->next;

while(p)

{

if(p->k==5&&!strcmp(p->v,"cs."))

printf("cat\n");

if(p->k==101&&!strcmp(p->v,"thqqxw.lui.qswer"))

printf("this\_is\_a\_secret\n");

if(p->k==3&&!strcmp(p->v,"b\_ylxmhzjsys.virpbkr"))

printf("beware.\_dogs\_barking\n");

p=p->next;

}

}

int main()

{

Lnode \*h;

h=creat();

untwist(h);

}

## 5、

#include <stdio.h>

#include <stdlib.h>

typedef struct node

{

int exp;

float coef;

struct node \*next;

}polynode;

polynode \*creat()

{

polynode \*h, \*p, \*t;

h=(polynode\*)malloc(sizeof(polynode));

h->next=NULL;

t=h;

while(1)

{

p=(polynode\*)malloc(sizeof(polynode));

scanf("%f%d",&p->coef,&p->exp);

if(p->exp==0&&p->coef==0)

{

free(p);

break;

}

p->next=NULL;

t->next=p;

t=p;

}

return h;

}

void dxsxj(polynode \*ha, polynode \*hb)

{

polynode \*p, \*q, \*r, \*pre;

float x;

p=ha->next;

q=hb->next;

pre=ha;

while(p&&q)

{

if(p->exp==q->exp)

{

x=p->coef+q->coef;

if(x!=0)

{

p->coef=x;

pre=p;

}

else

{

pre->next=p->next;

free(p);

}

p=pre->next;

r=q;

q=q->next;

free(r);

}

else if(p->exp>q->exp)

{

r=q->next;

q->next=p;

pre->next=q;

pre=q;

q=r;

}

}

if(q)

pre->next=q;

free(hb);

}

void output(polynode \*h)

{

polynode \*p;

p=h->next;

while(p)

{

printf("%fx^%d+",p->coef,p->exp);

p=p->next;

}

printf("\n");

}

int main()

{

polynode \*h1, \*h2;

h1=creat();

output(h1);

h2=creat();

output(h2);

dxsxj(h1,h2);

output(h1);

return 0;

}