**exp2-6**

#include "c:/cpp/linklist.cpp"

void depart( LinkNode \*&L,ElemType x)

{

LinkNode\*p,\*q,\*r;

p=L->next;

L->next=NULL;

r=L;

while(p!=NULL)

{

if(p->data<x)

{

q=p->next;

p->next=L->next;

L->next=p;

if(p->next==NULL)

r=p;

p=q;

}

else

{

r->next=p;

r=p;

p=p->next;

}

}

r->next=NULL;

}

int main()

{

LinkNode\*L;

ElemType a[]="abcdefgh";

int n=8;

CreateListR(L,a,n);

printf("L:");

DispList(L);

ElemType x='d';

printf("按%c进行划分:",x);

depart(L,x);

printf("L:");

DispList(L);

DestroyList(L);

return 1;

}

**exp2-7**

#include"c:/cpp/linklist.cpp"

void add(LinkNode \*L1,LinkNode \*L2,LinkNode \*&L3)

{

LinkNode\*p1=L1->next,\*p2=L2->next,\*r;

L3=L1;

r=L3;

free(L2);

while(p1!=NULL && p2!=NULL)

{

r->next=p1;

r=r->next;

p1=p1->next;

r->next=p2;

r=r->next;

p2=p2->next;

}

r->next=NULL;

if(p2!=NULL)

p1=p2;

r->next=p1;

}

int main()

{

LinkNode \*L1, \*L2, \*L3;

ElemType a[]="aaaaa";

ElemType b[]="bbb";

int n1=5,n2=3;

CreateListR(L1,a,n1);

CreateListR(L2,b,n2);

printf("L1:");

DispList(L1);

printf("L2:");

DispList(L2);

add(L1,L2,L3);

printf("L3:");

DispList(L3);

return 1;

}

**exp2-8**

#include"c:/cpp/linklist.cpp"

void sort(LinkNode \*&L)

{

LinkNode\*p,\*q,\*pre;

p=L->next->next;

L->next->next=NULL;

while(p!=NULL)

{

q=p->next;

pre=L;

while(pre->next!=NULL && pre->next->data < p->data)

pre=pre->next;

p->next=pre->next;

pre->next=p;

p=q;

}

}

void add(LinkNode \*L1,LinkNode \*L2,LinkNode \*&L3)

{

LinkNode \*p1,\*p2,\*r,\*s;

p1=L1->next;

p2=L2->next;

L3=(LinkNode\*)malloc(sizeof(LinkNode));

r=L3;

while(p1!=NULL && p2!=NULL)

{

if(p1->data < p2->data)

{

s=(LinkNode\*)malloc(sizeof(LinkNode));

s->data=p1->data;

r->next=s;

r=s;

p1=p1->next;

}

else if(p1->data > p2->data)

{

s=(LinkNode\*)malloc(sizeof(LinkNode));

s->data=p2->data;

r->next=s;

r=s;

p2=p2->next;

}

else

{

s=(LinkNode\*)malloc(sizeof(LinkNode));

s->data=p1->data;

r->next=s;

r=s;

p1=p1->next;

p2=p2->next;

}

}

if(p2!=NULL)

p1=p2;

while(p1!=NULL)

{

s=(LinkNode\*)malloc(sizeof(LinkNode));

s->data=p1->data;

r->next=s;

r=s;

p1=p1->next;

}

s->next=NULL;

}

void Insect(LinkNode\*L1,LinkNode\*L2,LinkNode\*&L3)

{

LinkNode\*p1,\*p2,\*r,\*s;

p1=L1->next;

L3=(LinkNode\*)malloc(sizeof(LinkNode));

r=L3;

while(p1!=NULL)

{

p2=L2->next;

while(p2!=NULL && p2->data < p1->data)

{

p2=p2->next;

}

if(p2!=NULL && p2->data==p1->data)

{

s=(LinkNode\*)malloc(sizeof(LinkNode));

s->data=p1->data;

r->next=s;

r=s;

}

p1=p1->next;

}

r->next=NULL;

}

void Sub(LinkNode\*L1,LinkNode\*L2,LinkNode\*&L3)

{

LinkNode\*p1,\*p2,\*s,\*r;

p1=L1->next;

p2=L2->next;

L3=(LinkNode\*)malloc(sizeof(LinkNode));

r=L3;

while(p1!=NULL)

{

p2=L2->next;

while(p2!=NULL && p2->data < p1->data)

{

p2=p2->next;

}

if(p2!=NULL && p2->data > p1->data)

{

s=(LinkNode\*)malloc(sizeof(LinkNode));

s->data=p1->data;

r->next=s;

r=s;

}

p1=p1->next;

}

r->next=NULL;

}

int main()

{

LinkNode\*L1,\*L2,\*L3;

ElemType a[]={'a','q','v','l'};

int n1=4;

CreateListR(L1,a,n1);

printf("原始L1:");

DispList(L1);

ElemType b[]={'i','b','a','t','d','m'};

int n2=6;

CreateListR(L2,b,n2);

printf("原始L2:");

DispList(L2);

sort(L1);

printf("排序后L1:");

DispList(L1);

sort(L2);

printf("排序后L2:");

DispList(L2);

add(L1,L2,L3);

printf("L1与L2的并集为:");

DispList(L3);

Insect(L1,L2,L3);

printf("L1,L2的交集为:");

DispList(L3);

Sub(L1,L2,L3);

printf("L1,L2的差集为:");

DispList(L3);

}

**exp2-9**

#include<stdio.h>

#include<stdlib.h>

#include<malloc.h>

typedef struct

{

int a;//系数

int r;//指数

}xArray;

typedef struct pnode

{

int a;

int r;

struct pnode \*next;

}xNode;

void DispList(xNode\*y)

{

int first=1;

xNode \*p=y->next;

while(p!=NULL)

{

if(first==1)

first=0;

else if(p->a > 0)

printf("+");

else if(p->a < 0)

printf("-");

if(p->r==0)

printf("%d",p->a);

else if(p->r==1)

printf("%dx",p->a);

else

printf("%dx^%d",p->a,p->r);

p=p->next;

}

printf("\n");

}

void sort(xNode\*y1)

{

xNode\*p=y1->next,\*pre,\*q;

if(p!=NULL)

{

q=p->next;

p->next=NULL;

p=q;

}

while(p!=NULL)

{

q=p->next;

pre=y1;

while(pre->next!=NULL && pre->next->r > p->r)

pre=pre->next;

p->next=pre->next;

pre->next=p;

p=q;

}

}

void CreateListR(xNode\*&y,xArray a[],int n)

{

xNode \*s,\*r;

int i;

y=(xNode\* )malloc(sizeof(xNode));

y->next=NULL;

r=y;

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

{

s=(xNode\* )malloc(sizeof(xNode));

s->a=a[i].a;

s->r=a[i].r;

r->next=s;

r=s;

}

r->next=NULL;

}

void add(xNode\*y1,xNode \*y2,xNode\*&y3)

{

xNode \*p1,\*p2,\*s,\*r,\*q;

p1=y1->next;

p2=y2->next;

y3=(xNode\*)malloc(sizeof(xNode));

r=y3;

while(p1!=NULL && p2!=NULL)

{

if(p1->r > p2->r)

{

s=(xNode\*)malloc(sizeof(xNode));

s->a=p1->a;

s->r=p1->r;

r->next=s;

r=s;

p1=p1->next;

}

else if(p1->r < p2->r)

{

s=(xNode\*)malloc(sizeof(xNode));

s->a=p2->a;

s->r=p2->r;

r->next=s;

r=s;

p2=p2->next;

}

else

{

int add\_a=p1->a + p2->a;

s=(xNode\*)malloc(sizeof(xNode));

s->a=add\_a;

s->r=p2->r;

r->next=s;

r=s;

p2=p2->next;

p1=p1->next;

}

}

if(p2!=NULL)

p1=p2;

while(p1!=NULL)

{

s=(xNode\*)malloc(sizeof(xNode));

s->a=p1->a;

s->r=p1->r;

r->next=s;

r=s;

p1=p1->next;

}

r->next=NULL;

}

int main()

{

xNode \*y1,\*y2,\*y3;

xArray a[]={{1,0},{2,5},{3,4},{4,1}};

CreateListR(y1,a,4);

xArray b[]={{2,0},{3,7},{4,5},{7,9},{4,4}};

CreateListR(y2,b,5);

printf("原多项式y1：");

DispList(y1);

printf("原多项式y2：");

DispList(y2);

sort(y1);

sort(y2);

printf("排序后多项式y1：");

DispList(y1);

printf("排序后多项式y2：");

DispList(y2);

add(y1,y2,y3);

printf("相加的结果为:");

DispList(y3);

return 1;

}

**exp2-10**

#include<stdio.h>

#include<stdlib.h>

#include<malloc.h>

typedef struct

{

int a;//系数

int r;//指数

}xArray;

typedef struct pnode

{

int a;

int r;

struct pnode \*next;

}xNode;

void DispList(xNode\*y)

{

int first=1;

xNode \*p=y->next;

while(p!=NULL)

{

if(first==1)

first=0;

else if(p->a > 0)

printf("+");

else if(p->a < 0)

printf("-");

if(p->r==0)

printf("%d",p->a);

else if(p->r==1)

printf("%dx",p->a);

else

printf("%dx^%d",p->a,p->r);

p=p->next;

}

printf("\n");

}

void sort(xNode\*y1)

{

xNode\*p=y1->next,\*pre,\*q;

if(p!=NULL)

{

q=p->next;

p->next=NULL;

p=q;

}

while(p!=NULL)

{

q=p->next;

pre=y1;

while(pre->next!=NULL && pre->next->r > p->r)

pre=pre->next;

p->next=pre->next;

pre->next=p;

p=q;

}

}

void CreateListR(xNode\*&y,xArray a[],int n)

{

xNode \*s,\*r;

int i;

y=(xNode\* )malloc(sizeof(xNode));

y->next=NULL;

r=y;

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

{

s=(xNode\* )malloc(sizeof(xNode));

s->a=a[i].a;

s->r=a[i].r;

r->next=s;

r=s;

}

r->next=NULL;

}

void cheng(xNode\*y1,xNode\*y2,xNode \*&y3)

{

xNode\*p1,\*p2,\*s,\*r;

p1=y1->next;

y3=(xNode\*)malloc(sizeof(xNode));

r=y3;

while(p1!=NULL)

{

p2=y2->next;

while(p2!=NULL)

{

s=(xNode\*)malloc(sizeof(xNode));

s->a=p1->a \* p2->a;

s->r=p1->r + p2->r;

r->next=s;

r=s;

p2=p2->next;

}

p1=p1->next;

}

r->next=NULL;

}

int main()

{

xNode \*y1,\*y2,\*y3;

xArray a[]={{1,0},{2,5},{3,4},{4,1}};

CreateListR(y1,a,4);

xArray b[]={{2,0},{3,7},{4,5},{7,9},{4,4}};

CreateListR(y2,b,5);

printf("原多项式y1：");

DispList(y1);

printf("原多项式y2：");

DispList(y2);

sort(y1);

sort(y2);

printf("排序后多项式y1：");

DispList(y1);

printf("排序后多项式y2：");

DispList(y2);

printf("相乘后的多项式为:");

cheng(y1,y2,y3);

sort(y3);

DispList(y3);

return 0;

}

**exp2-11**

//文件名:exp2-11.cpp

#include <stdio.h>

#include <malloc.h>

typedef struct

{

int no; //职工号

char name[10]; //姓名

int depno; //部门号

float salary; //工资数

} EmpType; //职工类型

typedef struct node

{

EmpType data; //存放职工信息

struct node \*next; //指向下一个结点的指针

} EmpList; //职工单链表结点类型

void DestroyEmp(EmpList \*&L) //释放职工单链表L

{

EmpList \*pre=L,\*p=pre->next;

while (p!=NULL)

{

free(pre);

pre=p;

p=p->next;

}

free(pre);

}

void DelAll(EmpList \*&L) //删除职工文件中全部记录

{

FILE \*fp;

if ((fp=fopen("emp.dat","wb"))==NULL) //重写清空emp.dat文件

{

printf(" 提示:不能打开职工文件\n");

return;

}

fclose(fp);

DestroyEmp(L); //释放职工单链表L

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

L->next=NULL; //建立一个空的职工单链表L

printf(" 提示:职工数据清除完毕\n");

}

void ReadFile(EmpList \*&L) //读emp.dat文件建立职工单键表L

{

FILE \*fp;

EmpType emp;

EmpList \*p,\*r;

int n=0;

L=(EmpList \*)malloc(sizeof(EmpList)); //建立头结点

r=L;

if ((fp=fopen("emp.dat","rb"))==NULL) //不存在emp.dat文件

{

if ((fp=fopen("emp.dat","wb"))==NULL)

printf(" 提示:不能创建emp.dat文件\n");

}

else //若存在emp.dat文件

{

while (fread(&emp,sizeof(EmpType),1,fp)==1)

{ //采用尾插法建立单链表L

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

p->data=emp;

r->next=p;

r=p;

n++;

}

}

r->next=NULL;

printf(" 提示:职工单键表L建立完毕,有%d个记录\n",n);

fclose(fp);

}

void SaveFile(EmpList \*L) //将职工单链表数据存入数据文件

{

EmpList \*p=L->next;

int n=0;

FILE \*fp;

if ((fp=fopen("emp.dat","wb"))==NULL)

{

printf(" 提示:不能创建文件emp.dat\n");

return;

}

while (p!=NULL)

{

fwrite(&p->data,sizeof(EmpType),1,fp);

p=p->next;

n++;

}

fclose(fp);

DestroyEmp(L); //释放职工单链表L

if (n>0)

printf(" 提示:%d个职工记录写入emp.dat文件\n",n);

else

printf(" 提示:没有任何职工记录写入emp.dat文件\n");

}

void InputEmp(EmpList \*&L) //添加一个职工记录

{

EmpType p;

EmpList \*s;

printf(" >>输入职工号(-1返回):");

scanf("%d",&p.no);

if (p.no==-1) return;

printf(" >>输入姓名 部门号 工资:");

scanf("%s%d%f",&p.name,&p.depno,&p.salary);

s=(EmpList \*)malloc(sizeof(EmpList));

s->data=p;

s->next=L->next; //采用头插法插入结点s

L->next=s;

printf(" 提示:添加成功\n");

}

void DelEmp(EmpList \*&L) //删除一个职工记录

{

EmpList \*pre=L,\*p=L->next;

int no;

printf(" >>输入职工号(-1返回):");

scanf("%d",&no);

if (no==-1) return;

while (p!=NULL && p->data.no!=no)

{

pre=p;

p=p->next;

}

if (p==NULL)

printf(" 提示:指定的职工记录不存在\n");

else

{

pre->next=p->next;

free(p);

printf(" 提示:删除成功\n");

}

}

void Sortno(EmpList \*&L) //采用直接插入法单链表L按no递增有序排序

{

EmpList \*p,\*pre,\*q;

p=L->next->next;

if (p!=NULL)

{

L->next->next=NULL;

while (p!=NULL)

{

q=p->next;

pre=L;

while (pre->next!=NULL && pre->next->data.no<p->data.no)

pre=pre->next;

p->next=pre->next;

pre->next=p;

p=q;

}

}

printf(" 提示:按no递增排序完毕\n");

}

void Sortdepno(EmpList \*&L) //采用直接插入法单链表L按depno递增有序排序

{

EmpList \*p,\*pre,\*q;

p=L->next->next;

if (p!=NULL)

{

L->next->next=NULL;

while (p!=NULL)

{

q=p->next;

pre=L;

while (pre->next!=NULL && pre->next->data.depno<p->data.depno)

pre=pre->next;

p->next=pre->next;

pre->next=p;

p=q;

}

}

printf(" 提示:按depno递增排序完毕\n");

}

void Sortsalary(EmpList \*&L) //采用直接插入法单链表L按salary递增有序排序

{

EmpList \*p,\*pre,\*q;

p=L->next->next;

if (p!=NULL)

{

L->next->next=NULL;

while (p!=NULL)

{

q=p->next;

pre=L;

while (pre->next!=NULL && pre->next->data.salary<p->data.salary)

pre=pre->next;

p->next=pre->next;

pre->next=p;

p=q;

}

}

printf(" 提示:按salary递增排序完毕\n");

}

void DispEmp(EmpList \*L) //输出所有职工记录

{

EmpList \*p=L->next;

if (p==NULL)

printf(" 提示:没有任何职工记录\n");

else

{

printf(" 职工号 姓名 部门号 工资\n");

printf(" ----------------------------------\n");

while (p!=NULL)

{

printf(" %3d%10s %-8d%7.2f\n",p->data.no,p->data.name,p->data.depno,p->data.salary);

p=p->next;

}

printf(" ----------------------------------\n");

}

}

int main()

{

EmpList \*L;

int sel;

printf("由emp.dat文件建立职工单键表L\n");

ReadFile(L);

do

{

printf(">1:添加 2:显示 3:按职工号排序 4:按部门号排序 5:按工资数排序\n");

printf(">6:删除 9:全删 0:退出 请选择:");

scanf("%d",&sel);

switch(sel)

{

case 9:

DelAll(L);

break;

case 1:

InputEmp(L);

break;

case 2:

DispEmp(L);

break;

case 3:

Sortno(L);

break;

case 4:

Sortdepno(L);

break;

case 5:

Sortsalary(L);

break;

case 6:

DelEmp(L);

break;

}

} while (sel!=0);

SaveFile(L);

return 1;

}

**exp2-12**

#include<stdio.h>

#include<malloc.h>

#include<string.h>

#define MAX 50

typedef struct node

{

int data;

struct node \* next;

}nodetype;

void creatlink(nodetype \*&L,char a[],int n)

{

nodetype \*p,\*r;

int i=0;

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

r=L;

while(i<n)

{

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

p->data=a[n-i-1]-'0';

r->next=p;

r=p;

i++;

}

r->next=NULL;

}

displink(nodetype\*L)

{

nodetype \*p=L->next;

while(p!=NULL)

{

printf("%d",p->data);

p=p->next;

}

printf("\n");

}

void add(nodetype \*L1,nodetype\*L2,nodetype\*&L3)

{

nodetype \*p1=L1->next,\*p2=L2->next,\*p,\*r;

int carry=0;

L3=(nodetype\*)malloc(sizeof(nodetype));

r=L3;

while(p1!=NULL&&p2!=NULL)

{

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

p->data=(p1->data+p2->data+carry)%10;

r->next=p; r=p;

carry=(p1->data+p2->data+carry)/10;

p1=p1->next;

p2=p2->next;

}

if(p1==NULL) p1=p2;

while(p1!=NULL)

{

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

p->data=(p1->data+carry)%10;

r->next=p; r=p;

carry=(p1->data+carry)/10;

p1=p1->next;

}

if(carry>0)

{

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

p->data=carry;

r->next=p;

r=p;

}

r->next=NULL;

}

void reverse(nodetype \*&L)

{

nodetype\*p=L->next,\*q;

L->next=NULL;

while(p!=NULL)

{

q=p->next;

p->next=L->next;

L->next=p;

p=q;

}

}

int mid(nodetype\*L)

{

nodetype \*slow=L,\*quick=L;

while(quick!=NULL&&quick->next!=NULL)

{

slow=slow->next;

quick=quick->next->next;

}

return slow->data;

}

int main()

{

nodetype \*L1,\*L2,\*L3;

char s[MAX],t[MAX];

printf("输入第一组整数\n");

scanf("%s",s);

printf("输入第二组整数\n");

scanf("%s",t);

creatlink(L1,s,strlen(s));

creatlink(L2,t,strlen(t));

printf("整数单链表L1:");

displink(L1);

printf("整数单链表L2:");

displink(L2);

add(L1,L2,L3);

printf("结果单链表是:");

displink(L3);

reverse(L3);

printf("对应的整数:");

displink(L3);

printf("中间位:%d\n",mid(L3));

return 1;

}