实验二 循环单链表、循环双链表及其应用

实验周次：第8周 学时：2学时 地点：学院机房

**【实验目的】**

1.领会循环单链表存储结构和掌握循环单链表中各种基本运算算法设计。

2.领会循环双链表存储结构和掌握循环双链表中的各种基本运算算法设计。

**【实验内容】**

实验题1.编写一个程序clinklist.cpp,实现循环单链表的各种基本运算和整体建表算法（假设循环单链表的元素类型Elem Type为char），并在此基础上设计一个程序exp2-4.cpp完成以下功能。

1. 初始化循环单链表h。
2. 依次采用尾插法插入a,b,c,d,e元素。
3. 输出循环单链表h。
4. 输出循环单链表L的长度。
5. 判断循环单链表L是否为空。
6. 输出循环单链表h的第3个元素。
7. 输出元素a的位置。
8. 在第4个元素位置上插入f元素。
9. 输出循环单链表h。
10. 删除循环单链表h的第3个元素。
11. 输出循环单链表h。
12. 释放循环单链表h。

//循环单链表运算算法

#include <stdio.h>

#include <malloc.h>

typedef *int* ElemType;

typedef *struct* LNode        //定义单链表结点类型

{

    ElemType data;

*struct* LNode \*next;

} LinkNode;

*void* CreateListF(LinkNode \*&*L*,ElemType *a*[],*int* *n*) //头插法建立循环单链表

{

    LinkNode \*s;*int* i;

*L*=(LinkNode \*)malloc(sizeof(LinkNode));     //创建头结点

*L*->next=NULL;

    for (i=0;i<*n*;i++)

    {

        s=(LinkNode \*)malloc(sizeof(LinkNode));//创建新结点

        s->data=*a*[i];

        s->next=*L*->next;            //将结点s插在原开始结点之前,头结点之后

*L*->next=s;

    }

    s=*L*->next;

    while (s->next!=NULL)           //查找尾结点,由s指向它

        s=s->next;

    s->next=*L*;                      //尾结点next域指向头结点

}

*void* CreateListR(LinkNode \*&*L*,ElemType *a*[],*int* *n*) //尾插法建立循环单链表

{

    LinkNode \*s,\*r;*int* i;

*L*=(LinkNode \*)malloc(sizeof(LinkNode));     //创建头结点

*L*->next=NULL;

    r=*L*;                    //r始终指向终端结点,开始时指向头结点

    for (i=0;i<*n*;i++)

    {

        s=(LinkNode \*)malloc(sizeof(LinkNode));//创建新结点

        s->data=*a*[i];

        r->next=s;          //将结点s插入结点r之后

        r=s;

    }

    r->next=*L*;              //尾结点next域指向头结点

}

*void* InitList(LinkNode \*&*L*) //初始化线性表

{

*L*=(LinkNode \*)malloc(sizeof(LinkNode)); //创建头结点

*L*->next=*L*;

}

*void* DestroyList(LinkNode \*&*L*)  //销毁线性表

{

    LinkNode \*pre=*L*,\*p=pre->next;

    while (p!=*L*)

    {

        free(pre);

        pre=p;                  //pre、p同步后移一个结点

        p=pre->next;

    }

    free(pre);                  //此时p=L,pre指向尾结点,释放它

}

*bool* ListEmpty(LinkNode \**L*)     //判线性表是否为空表

{

    return(*L*->next==*L*);

}

*int* ListLength(LinkNode \**L*)     //求线性表的长度

{

    LinkNode \*p=*L*;*int* i=0;      //p指向头结点,n置为0(即头结点的序号为0)

    while (p->next!=*L*)

    {

        i++;

        p=p->next;

    }

    return(i);                  //循环结束,p指向尾结点,其序号n为结点个数

}

*void* DispList(LinkNode \**L*)      //输出线性表

{

    LinkNode \*p=*L*->next;

    while (p!=*L*)                //p不为L,输出p结点的data域

    {

        printf("%c ",p->data);

        p=p->next;

    }

    printf("\n");

}

*bool* GetElem(LinkNode \**L*,*int* *i*,ElemType &*e*) //求线性表中第i个元素值

{   *int* j=1;

    LinkNode \*p=*L*->next;

    if (*i*<=0 || *L*->next==*L*)     //i错误或者空表返回假

        return false;

    if (*i*==1)                   //求第1个结点值，作为特殊情况处理

    {

*e*=*L*->next->data;

        return true;

    }

    else                        //i不为1时

    {

        while (j<=*i*-1 && p!=*L*)  //找第i个结点p

        {

            j++;

            p=p->next;

        }

        if (p==*L*)               //没有找到返回假

            return false;

        else                    //找到了提取它的值并返回整

        {

*e*=p->data;

            return true;

        }

    }

}

*int* LocateElem(LinkNode \**L*,ElemType *e*)  //查找第一个值域为e的元素序号

{

    LinkNode \*p=*L*->next;

*int* i=1;

    while (p!=*L* && p->data!=*e*)  //查找第一个值域为e的结点p

    {

        p=p->next;

        i++;                    //i对应结点p的序号

    }

    if (p==*L*)

        return(0);              //没有找到返回0

    else

        return(i);              //找到了返回其序号

}

*bool* ListInsert(LinkNode \*&*L*,*int* *i*,ElemType *e*)  //插入第i个元素

{

*int* j=1;

    LinkNode \*p=*L*,\*s;

    if (*i*<=0) return false;     //i错误返回假

    if (p->next==*L* || *i*==1)     //原单链表为空表或i=1作为特殊情况处理

    {

        s=(LinkNode \*)malloc(sizeof(LinkNode)); //创建新结点s

        s->data=*e*;

        s->next=p->next;        //将结点s插入到结点p之后

        p->next=s;

        return true;

    }

    else

    {

        p=*L*->next;

        while (j<=*i*-2 && p!=*L*)  //找第i-1个结点p

        {

            j++;

            p=p->next;

        }

        if (p==*L*)               //未找到第i-1个结点

            return false;

        else                    //找到第i-1个结点p

        {

            s=(LinkNode \*)malloc(sizeof(LinkNode)); //创建新结点s

            s->data=*e*;

            s->next=p->next;                        //将结点s插入到结点p之后

            p->next=s;

            return true;

        }

    }

}

*bool* ListDelete(LinkNode \*&*L*,*int* *i*,ElemType &*e*) //删除第i个元素

{

*int* j=1;

    LinkNode \*p=*L*,\*q;

    if (*i*<=0 || *L*->next==*L*)

        return false;           //i错误或者空表返回假

    if (*i*==1)                   //i=1作为特殊情况处理

    {

        q=*L*->next;              //删除第1个结点

*e*=q->data;

*L*->next=q->next;

        free(q);

        return true;

    }

    else                        //i不为1时

    {

        p=*L*->next;

        while (j<=*i*-2 && p!=*L*)  //找第i-1个结点p

        {

            j++;

            p=p->next;

        }

        if (p==*L*)               //未找到第i-1个结点

            return false;

        else                    //找到第i-1个结点p

        {

            q=p->next;          //q指向要删除的结点

*e*=q->data;

            p->next=q->next;    //从单链表中删除q结点

            free(q);            //释放q结点

            return true;

        }

    }

}

//文件名:exp2-4.cpp

#include "clinklist.cpp"

*int* main()

{

    LinkNode \*h;

    ElemType e;

    printf("循环单链表的基本运算如下:\n");

    printf("  (1)初始化循环单链表h\n");

    InitList(h);

    printf("  (2)依次采用尾插法插入a,b,c,d,e元素\n");

    ListInsert(h,1,'a');

    ListInsert(h,2,'b');

    ListInsert(h,3,'c');

    ListInsert(h,4,'d');

    ListInsert(h,5,'e');

    printf("  (3)输出循环单链表h:");

    DispList(h);

    printf("  (4)循环单链表h长度:%d\n",ListLength(h));

    printf("  (5)循环单链表h为%s\n",(ListEmpty(h)?"空":"非空"));

    GetElem(h,3,e);

    printf("  (6)循环单链表h的第3个元素:%c\n",e);

    printf("  (7)元素a的位置:%d\n",LocateElem(h,'a'));

    printf("  (8)在第4个元素位置上插入f元素\n");

    ListInsert(h,4,'f');

    printf("  (9)输出循环单链表h:");

    DispList(h);

    printf("  (10)删除h的第3个元素\n");

    ListDelete(h,3,e);

    printf("  (11)输出循环单链表h:");

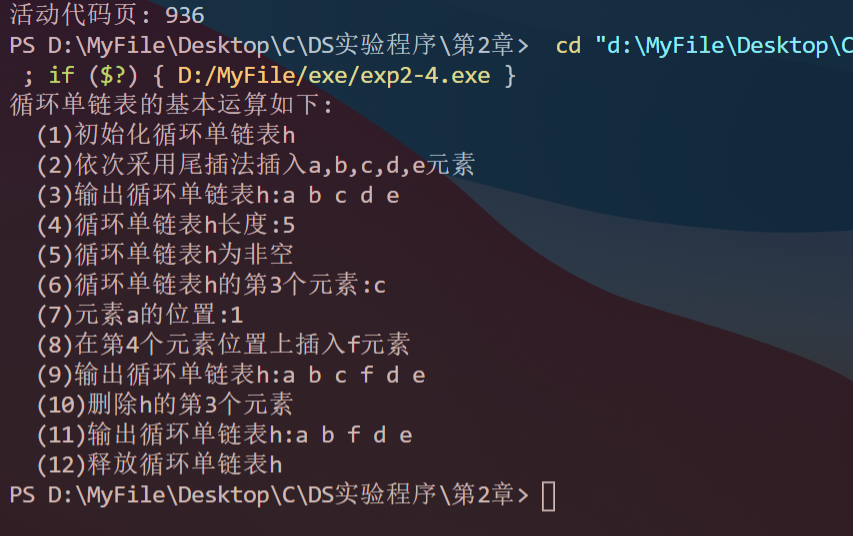
    DispList(h);

    printf("  (12)释放循环单链表h\n");

    DestroyList(h);

    return 1;

}



实验题2.编写一个程序cdlinklist.cpp,实现循环双链表的各种基本运算和整体建表算法（假设循环双链表的元素类型Elem Type为char），并在此基础上设计一个程序exp2-5.cpp完成以下功能。

1. 初始化循环双链表h。
2. 依次采用尾插法插入a,b,c,d,e元素。
3. 输出循环双链表h。
4. 输出循环双链表L的长度。
5. 判断循环双链表L是否为空。
6. 输出循环双链表h的第3个元素。
7. 输出元素a的位置。
8. 在第4个元素位置上插入f元素。
9. 输出循环双链表h。
10. 删除循环双链表h的第3个元素。
11. 输出循环双链表h。
12. 释放循环双链表h。

//循环双链表运算算法

#include <stdio.h>

#include <malloc.h>

typedef *int* ElemType;

typedef *struct* DNode        //定义双链表结点类型

{

    ElemType data;

*struct* DNode \*prior;    //指向前驱结点

*struct* DNode \*next;     //指向后继结点

} DLinkNode;

*void* CreateListF(DLinkNode \*&*L*,ElemType *a*[],*int* *n*) //头插法建立循环双链表

{

    DLinkNode \*s;

*L*=(DLinkNode \*)malloc(sizeof(DLinkNode));   //创建头结点

*L*->next=NULL;

    for (*int* i=0;i<*n*;i++)

    {

        s=(DLinkNode \*)malloc(sizeof(DLinkNode));//创建新结点

        s->data=*a*[i];

        s->next=*L*->next;            //将结点s插在原开始结点之前,头结点之后

        if (*L*->next!=NULL) *L*->next->prior=s;

*L*->next=s;s->prior=*L*;

    }

    s=*L*->next;

    while (s->next!=NULL)           //查找尾结点,由s指向它

        s=s->next;

    s->next=*L*;                      //尾结点next域指向头结点

*L*->prior=s;                     //头结点的prior域指向尾结点

}

*void* CreateListR(DLinkNode \*&*L*,ElemType *a*[],*int* *n*) //尾插法建立循环双链表

{

    DLinkNode \*s,\*r;

*L*=(DLinkNode \*)malloc(sizeof(DLinkNode));  //创建头结点

*L*->next=NULL;

    r=*L*;                    //r始终指向尾结点,开始时指向头结点

    for (*int* i=0;i<*n*;i++)

    {

        s=(DLinkNode \*)malloc(sizeof(DLinkNode));//创建新结点

        s->data=*a*[i];

        r->next=s;s->prior=r;   //将结点s插入结点r之后

        r=s;

    }

    r->next=*L*;              //尾结点next域指向头结点

*L*->prior=r;             //头结点的prior域指向尾结点

}

*void* InitList(DLinkNode \*&*L*)    //初始化线性表

{

*L*=(DLinkNode \*)malloc(sizeof(DLinkNode));   //创建头结点

*L*->prior=*L*->next=*L*;

}

*void* DestroyList(DLinkNode \*&*L*) //销毁线性表

{

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

    while (p!=*L*)

    {

        free(pre);

        pre=p;          //pre、p同步后移一个结点

        p=pre->next;

    }

    free(pre);          //此时p=L,pre指向尾结点,释放它

}

*bool* ListEmpty(DLinkNode \**L*)    //判线性表是否为空表

{

    return(*L*->next==*L*);

}

*int* ListLength(DLinkNode \**L*)    //求线性表的长度

{

    DLinkNode \*p=*L*;

*int* i=0;

    while (p->next!=*L*)

    {

        i++;

        p=p->next;

    }

    return(i);              //循环结束,p指向尾结点,其序号i为结点个数

}

*void* DispList(DLinkNode \**L*) //输出线性表

{

    DLinkNode \*p=*L*->next;

    while (p!=*L*)

    {

        printf("%c ",p->data);

        p=p->next;

    }

    printf("\n");

}

*bool* GetElem(DLinkNode \**L*,*int* *i*,ElemType &*e*)    //求线性表中第i个元素值

{

*int* j=1;

    DLinkNode \*p=*L*->next;

    if (*i*<=0 || *L*->next==*L*)

        return false;           //i错误或者L为空表返回假

    if (*i*==1)                   //i=1作为特殊情况处理

    {

*e*=*L*->next->data;

        return true;

    }

    else                        //i不为1时

    {

        while (j<=*i*-1 && p!=*L*)  //查找第i个结点p

        {

            j++;

            p=p->next;

        }

        if (p==*L*)               //没有找到第i个节，返回假

            return false;

        else                    //找到了第i个节，返回真

        {

*e*=p->data;

            return true;

        }

    }

}

*int* LocateElem(DLinkNode \**L*,ElemType *e*) //查找第一个值域为e的元素序号

{

*int* i=1;

    DLinkNode \*p=*L*->next;

    while (p!=NULL && p->data!=*e*)

    {

        i++;

        p=p->next;

    }

    if (p==NULL)            //不存在值为e的结点,返回0

        return(0);

    else                    //存在值为e的结点,返回其逻辑序号i

        return(i);

}

*bool* ListInsert(DLinkNode \*&*L*,*int* *i*,ElemType *e*) //插入第i个元素

{

*int* j=1;

    DLinkNode \*p=*L*,\*s;

    if (*i*<=0) return false;             //i错误返回假

    if (p->next==*L*)                     //原双链表为空表时

    {

        s=(DLinkNode \*)malloc(sizeof(DLinkNode));   //创建新结点s

        s->data=*e*;

        p->next=s;s->next=p;

        p->prior=s;s->prior=p;

        return true;

    }

    else if (*i*==1)                      //L不为空，i=1作为特殊情况处理

    {

        s=(DLinkNode \*)malloc(sizeof(DLinkNode));   //创建新结点s

        s->data=*e*;

        s->next=p->next;p->next=s;      //将结点s插入到结点p之后

        s->next->prior=s;s->prior=p;

        return true;

    }

    else                                //i不为1时

    {

        p=*L*->next;

        while (j<=*i*-2 && p!=*L*)          //查找第i-1个结点p

        {   j++;

            p=p->next;

        }

        if (p==*L*)                       //未找到第i-1个结点

            return false;

        else                            //找到第i-1个结点\*p

        {

            s=(DLinkNode \*)malloc(sizeof(DLinkNode));   //创建新结点s

            s->data=*e*;

            s->next=p->next;            //将结点s插入到结点p之后

            if (p->next!=NULL) p->next->prior=s;

            s->prior=p;

            p->next=s;

            return true;

        }

    }

}

*bool* ListDelete(DLinkNode \*&*L*,*int* *i*,ElemType &*e*) //删除第i个元素

{

*int* j=1;

    DLinkNode \*p=*L*,\*q;

    if (*i*<=0 || *L*->next==*L*)

        return false;               //i错误或者为空表返回假

    if (*i*==1)                       //i==1作为特殊情况处理

    {

        q=*L*->next;                  //删除第1个结点

*e*=q->data;

*L*->next=q->next;

        q->next->prior=*L*;

        free(q);

        return true;

    }

    else                            //i不为1时

    {

        p=*L*->next;

        while (j<=*i*-2 && p!=NULL)   //查找到第i-1个结点p

        {

            j++;

            p=p->next;

        }

        if (p==NULL)                //未找到第i-1个结点

            return false;

        else                        //找到第i-1个结点p

        {

            q=p->next;              //q指向要删除的结点

            if (q==NULL) return 0;  //不存在第i个结点

*e*=q->data;

            p->next=q->next;        //从单链表中删除q结点

            if (p->next!=NULL) p->next->prior=p;

            free(q);                //释放q结点

            return true;

        }

    }

}

//文件名:exp2-5.cpp

#include "cdlinklist.cpp"

*int* main()

{

    DLinkNode \*h;

    ElemType e;

    printf("循环双链表的基本运算如下:\n");

    printf("  (1)初始化循环双链表h\n");

    InitList(h);

    printf("  (2)依次采用尾插法插入a,b,c,d,e元素\n");

    ListInsert(h,1,'a');

    ListInsert(h,2,'b');

    ListInsert(h,3,'c');

    ListInsert(h,4,'d');

    ListInsert(h,5,'e');

    printf("  (3)输出循环双链表h:");

    DispList(h);

    printf("  (4)循环双链表h长度:%d\n",ListLength(h));

    printf("  (5)循环双链表h为%s\n",(ListEmpty(h)?"空":"非空"));

    GetElem(h,3,e);

    printf("  (6)循环双链表h的第3个元素:%c\n",e);

    printf("  (7)元素a的位置:%d\n",LocateElem(h,'a'));

    printf("  (8)在第4个元素位置上插入f元素\n");

    ListInsert(h,4,'f');

    printf("  (9)输出循环双链表h:");

    DispList(h);

    printf("  (10)删除h的第3个元素\n");

        ListDelete(h,3,e);

    printf("  (11)输出循环双链表h:");

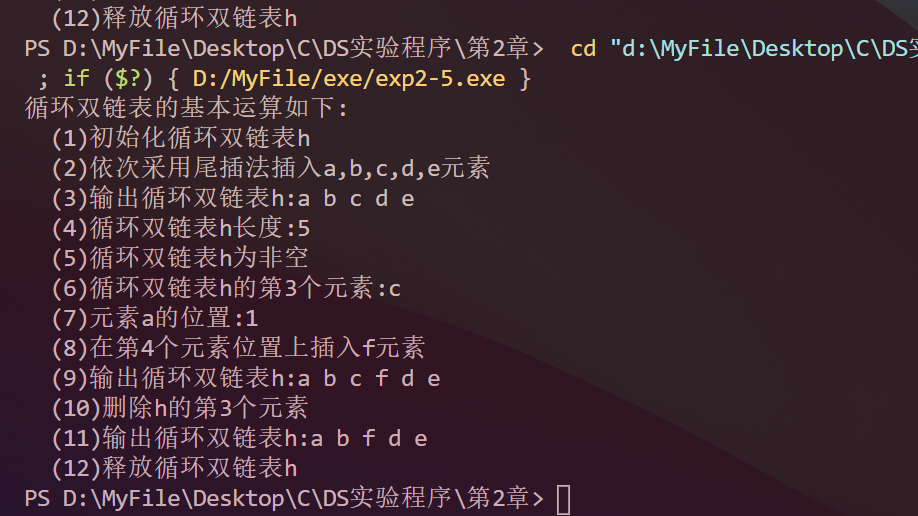
    DispList(h);

    printf("  (12)释放循环双链表h\n");

    DestroyList(h);

    return 1;

}



综合实验题：职工信息的综合运算

目的：深入掌握单链表应用的算法设计。

实验题3.假设有一个职工文件emp.dat，每个职工记录包含职工编号(no)、姓名(name)、部门号(depno)、工资数(salary)信息。设计一个程序exp2-6.cpp完成以下功能。

1. 从emp.dat文件中读出职工记录，并建立一个带头结点的单链表L。
2. 输入一个职工记录。
3. 显示所有职工记录。
4. 按编号no对所有职工记录进行递增排序。
5. 按部门号depno对所有职工记录进行递增排序。
6. 按工资数salary对所有职工记录进行递增排序。
7. 删除指定职工号的职工记录。
8. 删除职工文件中的全部记录。
9. 将单链表L中的所有职工记录存储到职工文件emp.dat中。

//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;

}