# 第1章 线性表

## 1.1 线性表的顺序存储结构

### 1.1.1 顺序表（静态数组）

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>

#define MAXSIZE 100                /*  顺序表空间的存储量  */

typedef int DataType;                   /*  顺序表元素类型  */

typedef    struct{
     DataType data[MAXSIZE];       /*  顺序表元素数组  */
     int    last;                  /*  顺序表最后一个元素位置  */
}SeqList;                          /*  顺序表类型  */

void wait()
{
     printf("\n 请按任意键...\n");
     getch();
}

int go_on()
{
     int flag=1;
     char choice;

     while(1){
          printf("\n 继续吗？[Y/N]");
          choice=getche();
```

```
            if(choice=='Y' || choice=='y')
                break;
            else if(choice=='N' || choice=='n'){
                flag=0;
                break;
            }
        }
        return(flag);
}

/* 初始化，构造一个空顺序表 */
SeqList *Init_SeqList( )
{
        SeqList *L;

        L=(SeqList *)malloc(sizeof(SeqList));
        if(!L){
            printf("\n 内存分配失败.\n");
            exit(-1);
        }
        L->last=-1;
        return(L);
}

/* 插入元素，在顺序表第 i 个位置之前插入元素 x，即插入在第 i 个位置 */
int Insert_SeqList(SeqList *L,int i,DataType x)
{
        int j;

        if(L->last==MAXSIZE-1){          /* 表满，不能插入 */
            printf("\n 表满，不能插入.\n");
            return(-1);                  /* 不能插入，返回-1 */
        }

        if(i<1 || i>L->last+2){          /* 插入位置错，不能插入 */
```

```
        printf("\n 插入位置错，不能插入.\n");
        return(0);                    /* 插入失败，返回 0 */
    }

    for(j=L->last;j>=i-1;j--)         /* 元素向后移动 */
        L->data[j+1]=L->data[j];
    L->data[i-1]=x;                   /* 插入元素 x */
    L->last++;                        /* 表最后位置值增加 1 */
    return(1);                        /* 插入成功，返回 1 */
}

void Insert(SeqList *L)
{
    DataType x;
    int i,flag=1,insert_flag;

    while(flag){

        printf("\n 请输入要插入元素的位置：");
        scanf("%d",&i);
        printf("请输入要插入元素:");
        scanf("%d",&x);

        insert_flag=Insert_SeqList(L,i,x);

        if(insert_flag==1)
            printf("\n 插入成功.\n");
        else
            printf("\n 插入失败.\n");

        flag=go_on();
    }
}

/* 删除元素，将顺序表第 i 个位置元素删除 */
```

```
int Delete_SeqList(SeqList *L,int i)
{
    int j;

    if(L->last==-1){                    /* 表空，不能删除 */
        printf("\n 表空，不能删除.\n");
        return(-1);                     /* 删除失败，返回-1 */
    }

    if(i<1 || i>L->last+1){             /* 删除位置错，不能删除. */
        printf ("\n 删除位置错，不能删除.\n");
        return(0);                      /* 删除失败，返回 0 */
    }

    for(j=i; j<=L->last; j++)
        L->data[j-1]=L->data[j];    /* 结点向前移动 */
    L->last--;                          /* last 仍指向最后元素 */
    return(1);                          /* 删除成功，返回 1 */
}

void Delete(SeqList *L)
{
    int i,flag=1,delete_flag;

    while(flag){
        printf("\n 请输入要删除元素的位置：");
        scanf("%d",&i);

        delete_flag=Delete_SeqList(L,i);

        if(delete_flag==1)
            printf("\n 删除成功.\n");
        else
            printf("\n 删除失败.\n");
```

```
            flag=go_on();
        }
}

/* 按值查找元素，在顺序表中查找元素 x，*/
/* 查找成功，返回 x 的位置序号，查找失败，返回-1 */
int Locate_SeqList(SeqList *L, DataType x)
{
    int i=0;

    while(i<=L->last && L->data[i]!= x)
        i++;
    if(i>L->last)
        return(-1);                    /* 查找失败，返回-1 */
    else
        return(i+1);                   /* 查找成功，返回 x 的位置序号 */
}

void Locate(SeqList *L)
{
    DataType x;
    int flag=1,locate_flag;

    while(flag){
        printf("\n 请输入要查找元素：");
        scanf("%d",&x);

        locate_flag=Locate_SeqList(L,x)    ;

        if(locate_flag>0)
            printf("\n 查找成功，%d 是第 %d 个元素.\n",x,locate_flag);
        else
            printf("\n 查找失败，没有元素 %d.\n",x);

        flag=go_on();
```

```
    }
}

/* 输出顺序表 */
void Display_SeqList(SeqList *L)
{
    int i;

    printf("\n 顺序表全部元素\n");
    for(i=0;i<=L->last;i++)
        printf("%4d",L->data[i]);
    printf("\n");
}

main()
{
    SeqList *L;

    char choice;
    int flag=1;

    L=Init_SeqList( );

    do{
        printf("\n");
        printf("----顺序表（静态数组实现）----\n");
        printf("      1.......插入元素\n");
        printf("      2.......删除元素\n");
        printf("      3.......查找元素\n");
        printf("      4.......输出元素\n");
        printf("      0.......退出\n");
        printf("------------------------------\n");
        printf("请选择[1/2/3/4/0]:");

        choice=getche();
```

```
        switch(choice){
            case '1':Insert(L);break;
            case '2':Delete(L);break;
            case '3':Locate(L);break;
            case '4':Display_SeqList(L);break;
            case '0':flag=0;break;
        }
        wait();
    }while(flag==1);
}
```

### 1.1.2 顺序表（动态数组）

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#define MAXSIZE 100                 /* 顺序表空间的存储量 */
typedef int DataType;               /* 顺序表元素类型 */
typedef   struct{
    DataType   *data;               /* 顺序表元素存储空间基址 */
    int   length;                   /* 顺序表当前长度 */
}SeqList;                           /* 顺序表类型 */
void wait( )
{
    printf("\n 请按任意键...\n");
    getch( );
}
int go_on( )
{
    int flag=1;
    char choice;
    while(1){
        printf("\n 继续吗？[Y/N]");
        choice=getche( );
        if(choice=='Y' || choice=='y')
```

```
            break;
        else if(choice=='N' || choice=='n'){
            flag=0;
            break;
        }
    }
    return(flag);
}
/* 初始化，构造一个空顺序表 */
void Init_SeqList(SeqList *L,int n)
{
    L->data=(DataType *)malloc(n*sizeof(DataType));
    if(!L->data){
        printf("\n 内存分配失败.\n");
        exit(-1);
    }
    L->length=0;
}
/* 插入元素，在顺序表第 i 个位置之前插入元素 x，即插入在第 i 个位置 */
int   Insert_SeqList(SeqList *L,int i,DataType x)
{
    DataType *p,*q;
    if(L->length == MAXSIZE){          /* 表满，不能插入 */
        printf("\n 表满，不能插入.\n");
        return(-1);                    /* 不能插入，返回-1 */
    }
    if(i<1 || i>L->length+1){          /* 插入位置错，不能插入 */
        printf("\n 插入位置错，不能插入.\n");
        return(0);                     /* 插入失败，返回 0 */
    }
    q=&(L->data[i-1]);                 /* q 指向插入位置 */
    for(p=&(L->data[L->length-1]);p>=q;p--)    /* 元素向后移动    */
        *(p+1)=*p;
    *q=x;                              /* 插入元素 x */
    L->length++;                       /* 表当前长度增加 1 */
```

```
    return(1);                          /* 插入成功，返回 1 */
}
void Insert(SeqList *L)
{
    DataType x;
    int i,flag=1,insert_flag;
    while(flag){
        printf("\n 请输入要插入元素的位置：");
        scanf("%d",&i);
        printf("请输入要插入元素:");
        scanf("%d",&x);
        insert_flag=Insert_SeqList(L,i,x);
        if(insert_flag==1)
            printf("\n 插入成功.\n");
        else
            printf("\n 插入失败.\n");
        flag=go_on( );
    }
}
/* 删除元素，将顺序表第 i 个位置元素删除 */
int Delete_SeqList(SeqList *L,int i)
{
    DataType *p,*q;
    if(L->length==0){                   /* 表空，不能删除 */
        printf("\n 表空，不能删除.\n");
        return(-1);                     /* 删除失败，返回-1 */
    }
    if(i<1 || i>L->length){             /* 删除位置错，不能删除 */
        printf ("\n 删除位置错，不能删除.\n");
        return(0);                      /* 删除失败，返回 0*/
    }
    q=&(L->data[L->length-1]);
    for(p=&(L->data[i]);p<=q;p++)       /* 结点向前移动 */
        *(p-1)=*p;
    L->length--;                        /* 表当前长度减少 1 */
```

```
    return(1);                          /* 删除成功，返回 1 */
}
void Delete(SeqList *L)
{
    int i,flag=1,delete_flag;
    while(flag){
        printf("\n 请输入要删除元素的位置：");
        scanf("%d",&i);
        delete_flag=Delete_SeqList(L,i);
        if(delete_flag==1)
            printf("\n 删除成功.\n");
        else
            printf("\n 删除失败.\n");
        flag=go_on( );
    }
}
/* 按值查找元素，在顺序表中查找元素 x，查找成功，返回 x 的位置序号，查找失败，返回-1 */
int Locate_SeqList(SeqList *L, DataType x)
{
    DataType *p;
    int i;
    i=0;
    p=L->data;
    while(i<=L->length-1 && *p!= x){
        i++;
        p++;
    }
    if(i>L->length-1)
        return(-1);                     /* 查找失败，返回-1 */
    else
        return(i+1);                    /* 查找成功，返回 x 的位置序号 */
}
void Locate(SeqList *L)
{
```

```
    DataType x;
    int flag=1,locate_flag;
    while(flag){
        printf("\n 请输入要查找元素： ");
        scanf("%d",&x);
        locate_flag=Locate_SeqList(L,x)    ;
        if(locate_flag>0)
            printf("\n 查找成功，%d 是第%d 个元素.\n",x,locate_flag);
        else
            printf("\n 查找失败，没有元素 %d.\n",x);
        flag=go_on( );
    }
}
/* 输出顺序表 */
void Display_SeqList(SeqList *L)
{
    int i;
    printf("\n 顺序表全部元素\n");
    for(i=0;i<=L->length-1;i++)
        printf("%4d",L->data[i]);
    printf("\n");
}
main( )
{
    SeqList L;
    char choice;
    int flag=1;
    Init_SeqList(&L,MAXSIZE);
    do{
        printf("\n");
        printf("----顺序表（动态数组实现）----\n");
        printf("      1.......插入元素\n");
        printf("      2.......删除元素\n");
        printf("      3.......查找元素\n");
        printf("      4.......输出元素\n");
```

```
        printf("       0.......退出\n");
        printf("-----------------------------\n");
        printf("请选择[1/2/3/4/0]:");
        choice=getche( );
        switch(choice){
            case '1':Insert(&L);break;
            case '2':Delete(&L);break;
            case '3':Locate(&L);break;
            case '4':Display_SeqList(&L);break;
            case '0':flag=0;break;
        }
        wait( );
    }while(flag==1);
}
```

## 1.2 线性表的链式存储结构

### 1.2.1 单链表（带头结点）

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
typedef int DataType;                    /* 单链表元素类型 */
typedef struct node{
    DataType data;                       /* 单链表元素 */
    struct node *next;                   /* 单链表元素后继指针 */
}LNode,*LinkList;                        /* 单链表结点类型、单链表类型 */
void wait( )
{
    printf("\n 请按任意键...\n");
    getch( );
}
int go_on( )
{
    int flag=1;
```

```
    char choice;
    while(1){
        printf("\n 继续吗？[Y/N]");
        choice=getche( );
        if(choice=='Y' || choice=='y')
            break;
        else if(choice=='N' || choice=='n'){
            flag=0;
            break;
        }
    }
    return(flag);
}
/* 初始化，构造一个空的带头结点单链表 */
void Init_LinkList(LinkList *L)
{
    *L=(LNode *)malloc(sizeof(LNode));
    if(*L==NULL){
        printf("\n 内存分配失败.\n");
        exit(-1);
    }
    (*L)->next=NULL;
}
/* 判断带头结点单链表是否为空 */
int Empty_LinkList(LinkList L)
{
    if(L->next==NULL)
        return(1);
    else
        return(0);
}
/* 求带头结点单链表长度 */
int Length_LinkList(LinkList L)
{
    LNode *p=L->next;
```

```
    int k;
    k=0;
    while(p!=NULL){
        k++;
        p=p->next;
    }
    return(k);
}
void Length(LinkList L)
{
    int k;
    k=Length_LinkList(L);
    printf("\n 表长：%d\n",k);
}
```

/* 按序号查找元素，在带头结点单链表中查找第 i 个元素，查找成功，返回指向第 i 个元素结点的指针，否则，返回空 */

```
LNode *Locatei_LinkList(LinkList L,int i)
{
    LNode *p=L;
    int j=0;
    while(p->next!=NULL && j<i){
        j++;
        p=p->next;
    }
    if(j==i)
        return(p);
    else
        return(NULL);
}
```

/* 按值查找元素，在带头结点单链表中查找元素值为 x 的第 1 个元素，查找成功，返回指向 x 所在结点的指针，*k 返回其位置序号，否则，返回空，*k 无意义 */

```
LNode *Locatex_LinkList(LinkList L,int x,int *k)
{
    LNode *p=L->next;
    int j=1;
```

```
    while(p!=NULL && p->data!=x){
        j++;
        p=p->next;
    }
    *k=j;
    return(p);
}
void Locatei(LinkList L)
{
    LNode *p;
    int flag=1,i;
    while(flag){
        printf("\n 请输入要查找元素的位置： ");
        scanf("%d",&i);
        p=Locatei_LinkList(L,i);
        if(p!=NULL && i!=0)
            printf("\n 查找成功，第 %d 个元素是 %d.\n",i,p->data);
        else
            printf("\n 查找失败，没有第 %d 个元素.\n",i);
        flag=go_on( );
    }
}
void Locatex(LinkList L)
{
    LNode *p;
    DataType x;
    int flag=1,k;
    while(flag){
        printf("\n 请输入要查找元素： ");
        scanf("%d",&x);
        p=Locatex_LinkList(L,x,&k)    ;
        if(p!=NULL)
            printf("\n 查找成功，%d 是第 %d 个元素.\n",p->data,k);
        else
            printf("\n 查找失败，没有元素 %d.\n",x);
```

```
            flag=go_on( );
        }
    }
    /* 插入元素，在带头结点单链表中第 i 个位置之前插入元素 x，即插入在第 i 个位置 */
    int Insert_LinkList(LinkList L,int i,DataType x)
    {
        LNode *p,*s;
        int len;
        len=Length_LinkList(L);
        if(i<1 || i>len+1){                    /* 插入位置错，不能插入 */
            printf("\n 插入位置错，不能插入.\n");
            return(0);                         /* 插入失败，返回 0 */
        }
        p=Locatei_LinkList(L,i-1);             /* 查找第 i-1 个结点 */
        s=(LNode *)malloc(sizeof(LNode));
        if(s==NULL){
            printf("\n 内存分配失败.\n");
            exit(-1);
        }
        s->data=x;
        s->next=p->next;                       /* 新结点插入在第 i-1 个结点的后面 */
        p->next=s;
        return(1);                             /* 插入成功，返回 1 */
    }
    void Insert(LinkList L)
    {
        DataType x;
        int i,flag=1,insert_flag;
        while(flag){
            printf("\n 请输入要插入元素的位置：");
            scanf("%d",&i);
            printf("请输入要插入元素:");
            scanf("%d",&x);
            insert_flag=Insert_LinkList(L,i,x);
```

```
            if(insert_flag==1)
                printf("\n 插入成功.\n");
            else
                printf("\n 插入失败.\n");
            flag=go_on( );
        }
}
/* 删除元素，在带头结点单链表中删除第 i 个位置元素 */
int Delete_LinkList(LinkList L,int i)
{
    LNode *p,*s;
    int len;
    if(Empty_LinkList(L)==1){
        printf("\n 表空，不能删除.\n");
        return(0);
    }
    len=Length_LinkList(L);
    if(i<1 || i>len){                    /* 删除位置错，不能删除. */
        printf ("\n 删除位置错，不能删除.\n");
        return(0);                       /* 删除失败，返回 0 */
    }
    p=Locatei_LinkList(L,i-1);           /* 查找第 i-1 个结点 */
    s=p->next;                           /* s 指向第 i 个结点 */
    p->next=s->next;                     /* 从链表中删除*/
    free(s);                             /* 释放*s */
    return(1);                           /* 删除成功，返回 1 */
}
void Delete(LinkList L)
{
    int i,flag=1,delete_flag;

    while(flag){
        printf("\n 请输入要删除元素的位置：");
        scanf("%d",&i);
        delete_flag=Delete_LinkList(L,i);
```

```
            if(delete_flag==1)
                printf("\n 删除成功.\n");
            else
                printf("\n 删除失败.\n");
            flag=go_on( );
        }
    }
    /* 输出带头结点单链表中所有数据元素 */
    void Display_LinkList(LinkList L)
    {
        LNode *p=L;
        if(Empty_LinkList(L)==1)
            printf("\n 表空，没有元素.\n");
        else{
            printf("\n 链表所有元素\n");
            while(p->next!=NULL){
                p=p->next;
                printf("%4d",p->data);
            }
        }
    }
    main( )
    {
        LinkList L;
        char choice;
        int flag=1;
        Init_LinkList(&L);
        do{
            printf("\n");
            printf("------单链表（带头结点）------\n");
            printf("          1.......插入元素\n");
            printf("          2.......删除元素\n");
            printf("          3.......按位置查找元素\n");
            printf("          4.......按值查找元素\n");
            printf("          5.......输出表长\n");
```

```
        printf("          6.......输出元素\n");
        printf("          0.......退出\n");
        printf("------------------------------\n");
        printf("请选择[1/2/3/4/5/6/0]:");
        choice=getche( );
        switch(choice){
            case '1':Insert(L);break;
            case '2':Delete(L);break;
            case '3':Locatei(L);break;
            case '4':Locatex(L);break;
            case '5':Length(L);break;
            case '6':Display_LinkList(L);break;
            case '0':flag=0;break;
        }
        wait( );
    }while(flag==1);
}
```

### 1.2.2 单链表（不带头结点）

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>

typedef int DataType;                   /*  单链表元素类型  */
typedef struct node{
    DataType data;                      /*  单链表元素  */
    struct node *next;                  /*  单链表元素后继指针  */
}LNode,*LinkList;                       /*  单链表结点类型、单链表类型  */

void wait()
{
    printf("\n 请按任意键...\n");
    getch();
}
```

```
int go_on()
{
    int flag=1;
    char choice;

    while(1){
        printf("\n 继续吗？[Y/N]");
        choice=getche();
        if(choice=='Y' || choice=='y')
            break;
        else if(choice=='N' || choice=='n'){
            flag=0;
            break;
        }
    }
    return(flag);
}

/* 初始化，构造一个空的不带头结点单链表 */
void Init_LinkList(LinkList *PL)
{
    *PL=NULL;
}

/* 判断不带头结点单链表是否为空 */
int Empty_LinkList(LinkList L)
{
    if(L==NULL)
        return(1);
    else
        return(0);
}

/* 求不带头结点单链表长度 */
```

```
int Length_LinkList(LinkList L)
{
    LNode *p=L;
    int k;

    k=0;
    while(p!=NULL){
        k++;
        p=p->next;
    }
    return(k);
}

void Length(LinkList L)
{
    int k;

    k=Length_LinkList(L);
    printf("\n 表长：%d\n",k);
}

/* 按序号查找元素，在不带头结点单链表中查找第 i 个元素， */
/* 查找成功，返回指向第 i 个元素结点的指针，查找失败，返回空 */
LNode *Locatei_LinkList(LinkList L,int i)
{
    LNode *p=L;
    int j=1;

    while(p!=NULL && j<i ){
        j++;
        p=p->next;
    }

    if (p!=NULL && j==i)
        return(p);
```

```
    else
        return(NULL);
}

/* 按值查找元素，在不带头结点单链表中查找元素值为 x 的第 1 个元素， */
/* 查找成功，返回指向 x 所在结点的指针，*k 返回其位置序号，否则，返回空,*k 无意义 */
LNode *Locatex_LinkList(LinkList L,int x,int *k)
{
    LNode *p=L;
    int j=1;

    while(p!=NULL && p->data!= x){
        j++;
        p=p->next;
    }
    *k=j;
    return(p);
}

void Locatei(LinkList L)
{
    LNode *p;
    int flag=1,i;

    while(flag){
        printf("\n 请输入要查找元素的位置：");
        scanf("%d",&i);

        p=Locatei_LinkList(L,i)    ;

        if(p!=NULL && i!=0)
            printf("\n 查找成功，第 %d 个元素是 %d.\n",i,p->data);
        else
            printf("\n 查找失败，没有第 %d 个元素.\n",i);
```

```
            flag=go_on();
        }
}

void Locatex(LinkList L)
{
        LNode *p;
        DataType x;
        int flag=1,k;

        while(flag){
            printf("\n 请输入要查找元素： ");
            scanf("%d",&x);

            p=Locatex_LinkList(L,x,&k)      ;

            if(p!=NULL)
                printf("\n 查找成功，%d 是第 %d 个元素。\n",p->data,k);
            else
                printf("\n 查找失败，没有元素 %d.\n",x);

            flag=go_on();
        }
}

void Locate(LinkList L)
{
        char choice;
        int flag=1;

        do{
            printf("\n");
            printf("-----------查找元素-----------\n");
            printf("           1.......按位置查找\n");
```

```
            printf("          2.......按值查找\n");
            printf("          0.......返回\n");
            printf("------------------------------\n");
            printf("请选择[1/2/0]:");

            choice=getche();
            switch(choice){
                case '1':Locatei(L);break;
                case '2':Locatex(L);break;
                case '0':flag=0;break;
            }
        }while(flag==1);
    }
```

/* 插入元素，在不带头结点单链表中第 i 个位置之前插入数据元素 x，即插入在第 i 个位置 */

```
    int Insert_LinkList(LinkList *L,int i,DataType x)
    {
        LNode *p,*s;
        int len;

        len=Length_LinkList(*L);
        if(i<1 || i>len+1){                    /* 插入位置错，不能插入 */
            printf("\n 插入位置错，不能插入.\n");
            return(0);                         /* 插入失败，返回 0 */
        }

        if(i==1){
            s=(LNode *)malloc(sizeof(LNode));
            if(s==NULL){
                printf("\n 内存分配失败.\n");
                exit(-1);
            }
            s->data=x;
            s->next=*L;                        /* 新结点插入在第 i-1 个结点的后面 */
```

```
            *L=s;
        }
        else if(i>=2 && i<=len+1){
            p=Locatei_LinkList(*L,i-1);/* 查找第 i-1 个结点 */
            s=(LNode *)malloc(sizeof(LNode));
            if(s==NULL){
                printf("\n 内存分配失败.\n");
                exit(-1);
            }
            s->data=x;
            s->next=p->next;              /* 新结点插入在第 i-1 个结点的后面 */
            p->next=s;
        }
        return(1);                        /* 插入成功，返回 1 */
}

void Insert(LinkList *L)
{
        DataType x;
        int i,flag=1,insert_flag;

        while(flag){
            printf("\n 请输入要插入元素的位置：");
            scanf("%d",&i);
            printf("请输入要插入元素:");
            scanf("%d",&x);

            insert_flag=Insert_LinkList(L,i,x);

            if(insert_flag==1)
                printf("\n 插入成功.\n");
            else
                printf("\n 插入失败.\n");

            flag=go_on();
```

```
    }
}

/* 删除元素，在不带头结点单链表中删除第 i 个位置元素 */
int Delete_LinkList(LinkList *L,int i)
{
    LNode *p,*s;
    int len;

    if(Empty_LinkList(*L)==1){
        printf("\n 表空，不能删除.\n");
        return(0);
    }

    len=Length_LinkList(*L);
    if(i<1 || i>len){                    /* 删除位置错，不能删除. */
        printf ("\n 删除位置错，不能删除.\n");
        return(0);                       /* 删除失败，返回 0 */
    }

    if(i==1){
        s=*L;                            /* s 指向第 i 个结点 */
        *L=s->next;                      /* 从链表中删除 */
        free(s);                         /* 释放*s */
    }
    else if(i>=2 && i<=len){
        p=Locatei_LinkList(*L,i-1);/* 查找第 i-1 个结点 */
        s=p->next;                       /* s 指向第 i 个结点 */
        p->next=s->next;                 /* 从链表中删除 */
        free(s);                         /* 释放*s */
    }
    return(1);                           /* 删除成功，返回 1 */
}

void Delete(LinkList *L)
```

```
{
    int i,flag=1,delete_flag;

    while(flag){
        printf("\n");
        printf("请输入要删除元素的位置：");
        scanf("%d",&i);

        delete_flag=Delete_LinkList(L,i);

        if(delete_flag==1)
            printf("\n 删除成功.\n");
        else
            printf("\n 删除失败.\n");

        flag=go_on();
    }
}

/* 输出带头结点单链表中所有数据元素 */
void Display_LinkList(LinkList L)
{
    LNode *p=L;

    if(Empty_LinkList(L)==1)
        printf("\n 表空，没有元素.\n");
    else{
        printf("\n 链表所有元素\n");
        while(p!=NULL){
            printf("%6d",p->data);
            p=p->next;
        }
    }
}
```

```
main()
{
    LinkList L;
    char choice;
    int flag=1;

    Init_LinkList(&L);

    do{
        printf("\n");
        printf("-----单链表（不带头结点）-----\n");
        printf("          1.......插入元素\n");
        printf("          2.......删除元素\n");
        printf("          3.......查找元素\n");
        printf("          4.......输出长度\n");
        printf("          5.......输出元素\n");
        printf("          0.......退出\n");
        printf("----------------------------\n");
        printf("请选择[1/2/3/4/5/0]:");

        choice=getche();
        switch(choice){
            case '1':Insert(&L);break;
            case '2':Delete(&L);break;
            case '3':Locate(L);break;
            case '4':Length(L);break;
            case '5':Display_LinkList(L);break;
            case '0':flag=0;break;
        }
        wait();
    }while(flag==1);
}
```

### 1.2.3 循环单链表（带头结点）

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>

typedef int DataType;                    /* 循环单链表元素类型 */
typedef struct node{
    DataType data;                       /* 循环单链表元素 */
    struct node *next;                   /* 循环单链表元素后继指针 */
}LNode,*CLinkList;                       /* 循环单链表结点类型、循环单链表类型 */

void wait()
{
    printf("\n 请按任意键...\n");
    getch();
}

int go_on()
{
    int flag=1;
    char choice;

    while(1){
        printf("\n 继续吗？[Y/N]");
        choice=getche();
        if(choice=='Y' || choice=='y')
            break;
        else if(choice=='N' || choice=='n'){
            flag=0;
            break;
        }
    }
    return(flag);
}
```

```
/* 初始化，构造一个空的带头结点的循环单链表 */
void Init_CLinkList(CLinkList *L)
{
    *L=(LNode *)malloc(sizeof(LNode));
    if(*L==NULL){
        printf("\n 内存分配失败.\n");
        exit(-1);
    }
    (*L)->next=*L;
}

/* 判断带头结点循环单链表是否为空 */
int Empty_CLinkList(CLinkList L)
{
    if(L->next==L)
        return(1);
    else
        return(0);
}

/* 求带头结点循环单链表的长度 */
int Length_CLinkList(CLinkList L)
{
    LNode *p=L;
    int k;

    k=0;
    while(p->next!=L){
        k++;
        p=p->next;
    }
    return(k);
}
```

```
void Length(CLinkList L)
{
    int k;

    k=Length_CLinkList(L);
    printf("\n 表长：%d\n",k);
}

/* 按序号查找元素，在带头结点循环单链表中查找第 i 个数据元素，*/
/* 查找成功，返回指向第 i 个元素结点的指针，查找失败，返回空 */
LNode *Locatei_CLinkList(CLinkList L,int i)
{
    LNode *p=L;
    int j=0;

    while(p->next!=L && j<i ){
        j++;
        p=p->next;
    }

    if (j==i)
        return(p);
    else
        return(NULL);
}

/* 按值查找元素，在带头结点单链表中查找元素值为 x 的第 1 个元素，*/
/* 查找成功，返回指向 x 所在结点的指针，*k 返回其位置序号，否则，返回空，
*k 无意义 */
LNode *Locatex_CLinkList(CLinkList L,int x,int *k)
{
    LNode *p=L->next;
    int j=1;

    while(p!=L && p->data!= x){
```

```
            j++;
            p=p->next;
        }

        if(p!=L){
            *k=j;
            return(p);
        }
        else
            return(NULL);
}

void Locatei(CLinkList L)
{
        LNode *p;
        int flag=1,i;

        while(flag){
            printf("\n 请输入要查找元素的位置：");
            scanf("%d",&i);

            p=Locatei_CLinkList(L,i);

            if(p!=NULL && i!=0)
                printf("\n 查找成功，第 %d 个元素是 %d.\n",i,p->data);
            else
                printf("\n 查找失败，没有第 %d 个元素.\n",i);

            flag=go_on();
        }
}

void Locatex(CLinkList L)
{
        LNode *p;
```

```
    DataType x;
    int flag=1,k;

    while(flag){
        printf("\n 请输入要查找元素的值：");
        scanf("%d",&x);

        p=Locatex_CLinkList(L,x,&k)  ;

        if(p!=NULL)
            printf("\n 查找成功，%d 是第 %d 个元素。\n",p->data,k);
        else
            printf("\n 查找失败，没有元素 %d.\n",x);

        flag=go_on();
    }
}

void Locate(CLinkList L)
{
    char choice;
    int flag=1;

    do{
        printf("\n");
        printf("----------查找元素----------\n");
        printf("          1.......按位置查找\n");
        printf("          2.......按值查找\n");
        printf("          0.......返回\n");
        printf("-----------------------------\n");
        printf("请选择[1/2/0]:");

        choice=getche();
        switch(choice){
            case '1':Locatei(L);break;
```

```
            case '2':Locatex(L);break;
            case '0':flag=0;break;
        }
    }while(flag==1);
}
```

/* 插入元素，在带头结点循环单链表中第 i 个位置之前插入数据元素 x，即插入在第 i 个位置 */

```
int Insert_CLinkList(CLinkList L,int i,DataType x)
{
    LNode *p,*s;
    int len;

    len=Length_CLinkList(L);
    if(i<1 || i>len+1){                 /* 插入位置错，不能插入 */
        printf("\n 插入位置错，不能插入.\n");
        return(0);                      /* 插入失败，返回 0 */
    }

    p=Locatei_CLinkList(L,i-1);         /* 查找第 i-1 个结点*/
    s=(LNode *)malloc(sizeof(LNode));
    if(s==NULL){
        printf("\n 内存分配失败.\n");
        exit(1);
    }
    s->data=x;
    s->next=p->next;                    /* 新结点插入在第 i-1 个结点的后面 */
    p->next=s;
    return(1);                          /* 插入成功，返回 1 */
}

void Insert(CLinkList L)
{
    DataType x;
    int i,flag=1,insert_flag;
```

```
    while(flag){
        printf("\n 请输入要插入元素的位置： ");
        scanf("%d",&i);
        printf("请输入要插入元素的值:");
        scanf("%d",&x);

        insert_flag=Insert_CLinkList(L,i,x);

        if(insert_flag==1)
            printf("\n 插入成功.\n");
        else
            printf("\n 插入失败.\n");

        flag=go_on();
    }
}

/* 删除元素，在带头结点循环单链表中删除第 i 个位置元素 */
int Delete_CLinkList(CLinkList L,int i)
{
    LNode *p,*s;
    int len;

    if(Empty_CLinkList(L)==1){
        printf("\n 表空，不能删除.\n");
        return(0);
    }

    len=Length_CLinkList(L);
    if(i<1 || i>len){                    /* 删除位置错，不能删除. */
        printf ("\n 删除位置错，不能删除.\n");
        return(0);                       /* 删除失败，返回 0 */
    }
```

```
    p=Locatei_CLinkList(L,i-1);          /* 查找第 i-1 个结点 */
    s=p->next;                           /* s 指向第 i 个结点 */
    p->next=s->next;                     /* 从链表中删除 */
    free(s);                             /* 释放*s */
    return(1);                           /* 删除成功，返回 1 */
}

void Delete(CLinkList L)
{
    int i,flag=1,delete_flag;

    while(flag){
        printf("\n");
        printf("请输入要删除元素的位置：");
        scanf("%d",&i);

        delete_flag=Delete_CLinkList(L,i);

        if(delete_flag==1)
            printf("\n 删除成功.\n");
        else
            printf("\n 删除失败.\n");

        flag=go_on();
    }
}

/* 输出带头结点循环单链表中所有元素 */
void Display_CLinkList(CLinkList L)
{
    LNode *p=L;

    if(Empty_CLinkList(L)==1)
        printf("\n 表空，没有元素.\n");
    else{
```

```
        printf("\n 循环单链表所有元素\n");
        while(p->next!=L){
            p=p->next;
            printf("%4d",p->data);
        }
    }
}

main()
{
    CLinkList L;
    char choice;
    int flag=1;

    Init_CLinkList(&L);

    do{
        printf("\n");
        printf("----循环单链表（带头结点）----\n");
        printf("          1.......插入元素\n");
        printf("          2.......删除元素\n");
        printf("          3.......查找元素\n");
        printf("          4.......输出长度\n");
        printf("          5.......输出元素\n");
        printf("          0.......退出\n");
        printf("-----------------------------\n");
        printf("请选择[1/2/3/4/5/0]:");

        choice=getche();
        switch(choice){
            case '1':Insert(L);break;
            case '2':Delete(L);break;
            case '3':Locate(L);break;
            case '4':Length(L);break;
            case '5':Display_CLinkList(L);break;
```

```
                case '0':flag=0;break;
            }
            wait();
        }while(flag==1);
    }
```