实验一

/*顺序表的插入和删除算法 */

```
#include "stdio.h"
#define LIST_INIT_SIZE 100
#define LISTINCREMENT 10

typedef struct {                    //顺序表存储结构
  int *elem;
  int length;
  int listsize;
}sqlist;

sqlist InitList_sq()                //建立一个顺序表
{
  sqlist l;    int a,i;    i=0;
  l.elem=(int *)malloc(LIST_INIT_SIZE*sizeof(int));
  if(!l.elem) exit(1);
  l.length=0;
  l.listsize=LIST_INIT_SIZE;
  scanf("%d",&a);
  while (a!=0)
```

```
    { l.elem[i++]=a;
        scanf("%d",&a);
    }
  l.length=i;
  return(l);
}
printsqlist(sqlist l)                    //显示一个顺序表
{      int i;
     for (i=0;i<=l.length-1;i++)
          printf("%d\t",l.elem[i]);
     printf("%s"," the length is:");
     printf("%d\n",l.length);
}
sqlist insertsqlist(sqlist L,int e,int i) //在第 i 个元素前插入 e
{    int *newbase,*p,*q;
      if (L.length>=L.listsize)
          {newbase=(int*)malloc((LIST_INIT_SIZE
                        +LISTINCREMENT)*sizeof(int));
          if(!newbase)
            {printf("\n%s","the overflow"); exit(1);} }
          if (i<1 || i>L.length+1)
          {printf("\n%s","insert:the error i"); exit(1);}
```

```
        q=&(L.elem[i-1]);
        for (p=&(L.elem[L.length-1]);p>=q;--p)
              *(p+1)=*p;
        *q=e;        ++L.length;
        return(L);
}
sqlist deletesqlist(sqlist L,int i)        //删除在第 i 个元素
{    int *p,*q;
     if (i<1 || i>L.length)
        {printf("%s","delete:the error i"); exit(1);}
     p=&(L.elem[i-1]);
     q=L.elem+L.length-1;
     for (++p;p<=q;++p)        *(p-1)=*p;
     --L.length;
     return(L);
}
main()                    //主函数
{    sqlist q;
    q=InitList_sq();                          //初始化
    printsqlist(q);                           //输出
    q=insertsqlist(q,5,2);                    //在第 2 个元素前插入 5
    printf("%s\n","insert(i=2,e=5):");
```

```
    printsqlist(q);                          //输出
    q=deletesqlist(q,3);                     //删除第 3 个元素
    printf("%s\n","delete location 3:");
    printsqlist(q);                          //输出
    }

/*单链表的插入和删除算法  */

#include "stdio.h"
#define NULL 0

typedef struct Lnode       //链表存储结构
{ int    data;
   struct Lnode *next;
  } Lnode;
typedef struct Lnode *LinkList;

LinkList creatlklist(int n)     //建立一个 n 个元素的链表
{ int i;     LinkList L,p;
  L=(LinkList)malloc(sizeof(Lnode));
  L->next=NULL;
  for (i=n;i>0;--i)
```

```
  {p=(LinkList)malloc(sizeof(Lnode));
    scanf("%d",&p->data);
    p->next=L->next;L->next=p;
  }
 return(L);
}
printlklist(LinkList l)      //显示一个链表
{ LinkList p;
    p=l->next;
    while (p!=NULL)
        { printf("%d\t",p->data);      p=p->next; }
}
LinkList insertlklist(LinkList L,int x,int i)
              //在第 i 个元素前插入 e
{ int j;      LinkList p,s;      p=L; j=0;
    while(p && j<i-1)         { p=p->next;    j++;} ;
    if(!p || j>i-1)
         {printf("\n%s","insert:not exist i");exit(1);}
          s=(struct Lnode *)malloc(sizeof(struct Lnode));
          s->data=x;    s->next=p->next; p->next=s;
      return(L);
}
```

```
LinkList deletelklist(LinkList L,int i) //删除在第 i 个元素
{ int j;    LinkList p,q;      p=L; j=0;
   while(p->next && j<i-1)
        { p=p->next;    j++;};
   if(!(p->next) || j>i-1)
     {printf("\n%s","delete:not exist i");exit(1);}
   q=p->next;        p->next=q->next;        free(q);
   return(L);
}
main()       //主函数
{      LinkList q;
     q=creatlklist(5);
     printlklist(q);
     q=insertlklist(q,5,2);
     printf("\n%s\n","after location 2 insert data 5:");
     printlklist(q);
     q=deletelklist(q,3);
     printf("\n%s\n","delete location 3: ");
     printlklist(q);
  }
```