



《算法与数据结构》实验

实验一（1） 顺序表的应用

实验一（2） 单链表的应用

实验一（3） 栈的应用

实验一（4） 队列的应用

实验二 二叉树的应用

实验三 图的应用

实验四 查找

实验五 排序

实验报告内容



实验名称 实验一 线性表的应用-顺序表

一、实验目的

1. 熟悉顺序表的类型定义和基本运算；
2. 要求利用顺序表来解决实际问题。

二、实验内容

1. 已知有两个按元素值递增有序的顺序表A和B，设计一个算法将表A和表B的全部元素归并为一个按元素值递增有序的顺序表C。

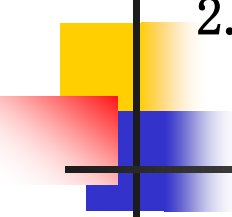
要求：

从键盘输入顺序表A和B的各元素，编程实现上述算法，输出顺序表A、顺序表B和顺序表C的所有元素值。

◆算法思路：

◆实现程序：参考教材例2.3

◆运行结果：



2. 已知线性表A按顺序存储，且每个元素都是互不相等的整数。
编程实现把所有偶数移到所有的奇数前边的算法。

要求：

- (1) 时间最少，辅助空间最少；
- (2) 线性表A的各元素初始值从键盘输入；
- (3) 输出结果。

◆ **算法思路：** 从左向右找到奇数`sq.elem[i]`，从右向左找到偶数`sq.elem[j]`，将两者交换；重复此过程直到*i*大于*j*为止。

◆ **实现程序：** 参考教材上例**2.4**

◆ **运行结果：**

三、实验设备

- 1. 硬件：一台PC机。
- 2. 软件：Visual C++6.0。

四、实验小结

遇到的主要问题及如何解决的，经验、体会及建议。

1.程序框架

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

```
#define MAXSIZE 100
```

```
typedef int ElemType;
```

```
typedef struct
```

```
{
```

```
    ElemType data[MAXSIZE];
```

```
    int len;
```

```
}SqList;
```

```
void merge(SqList A, SqList B, SqList &C)
```

```
{.....}
```

```
void main()
```

```
{SqList sqA,sqB,sqC; int i;
```

```
    for (i=0;i<8;i++) scanf("%d",&sqA.data[i]);
```

```
    sqA.length=8;
```

```
    .....
```

```
    merge(sqA,sqB,sqC);
```

```
    .....
```

```
    printf("data of sqC:\n");
```

```
    for (i=0;i<sqC.len;i++) printf("%5d",sqC.data[i]);
```

```
}
```



实验一（2） 线性表的应用-单链表

■ 实验目的

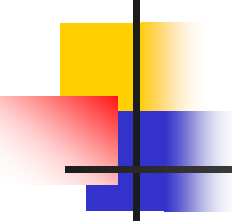
1. 熟悉单链表的类型定义和基本运算；
2. 学会利用单链表来解决实际问题。

■ 实验内容

1. 设单链表的数据为互不相等的整数，建立一个单链表，并设计一个算法, 找出单链表中元素值最大的结点。

要求：

- (1) 单链表的数据从键盘输入；
- (2) 输出单链表所有结点的数据和最大值结点序号。

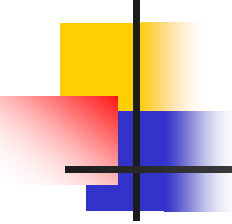
- 
-
2. 设计算法，根据输入的学生人数和成绩建立一个单链表，并累计成绩不及格的人数。

要求：

- (1) 学生人数和成绩均从键盘输入；
- (2) 输出所有学生的成绩和不及格的人数。

■ 实验设备

1. 硬件：一台PC机。
2. 软件：Visual C++6.0。



```
■ #include "iostream.h"
■ #include "malloc.h"
■ #define N 10
■ typedef int ElemType;
■ typedef struct LNode
■ {
■     ElemType data;
■     struct LNode *next;
■ }LNode,*LinkList;
■ void main()
■ {   LinkList head;
■     ElemType a[N];
■     int i,k;
■     for(i=0;i<N;i++) scanf("%d",&a[i]);
■     CreateLink(head,a,N);
■
■         .....
■     k=MaxNode(head);
■     cout<<k<<" ";   cout<<endl;
■ }
```



- **void CreateLink(LinkList &h, ElemType a[],int n)**
- {
-
- }

- **int MaxNode(LinkList h)**
- {
-
- }



```
int MaxNode(LinkList sl)
```

```
    {int j,k;
```

```
        LNode *p,*q;
```

```
        if (sl->next ==NULL) return 0;
```

```
        q=sl->next;  p=q->next;
```

```
        k=1; j=2;
```

```
        while (p!=NULL)
```

```
        {
```

```
            if (p->data>q->data) {q=p;k=j;}
```

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

```
        }
```

```
        return k;
```

```
    }//MaxNode
```



实验一（3） 线性表的应用-栈

■ 实验目的

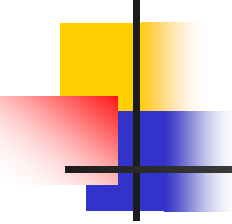
1. 熟悉顺序栈和链栈的类型定义和基本运算；
2. 学会利用栈的特点来解决实际问题。

■ 实验内容

1. 编写一个算法，将非负的十进制整数转换为其他进制的数输出，10及其以上的数字从‘A’开始的字母表示。

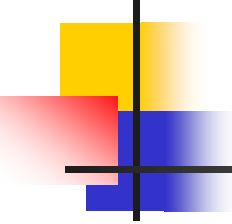
要求：

- 1) 采用顺序栈实现算法；
- 2) 从键盘输入一个十进制的数，输出相应的八进制数和十六进制数。



- #include "stdio.h"
- #include "iostream.h"
- #include "malloc.h"
- # define STACK_INIT_SIZE 100
- # define STACKINCREMENT 10
- typedef char SElemType;
- typedef struct {
- SElemType *base;
- SElemType *top;
- int stacksize;
- }SqStack;
- int trans(int d,int b)
- {
- SqStack st;
- int digit;
- char ch;
-
- }

```
void main()
{
    int d,b,t;
    cout<<"input d:";
    cin>>d;
    cout<<"input b:";
    cin>>b;
    t=trans(d,b);
}
```



2.回文指的是一个字符串从前面读和从后面读都一样，编写一个算法判断一个字符串是否为回文。

要求：

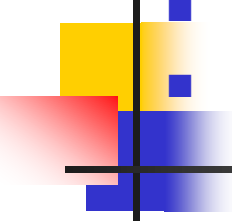
1) 采用链栈实现算法；

2) 从键盘输入一个字符串，输出判断结果。

■ 实验设备

1. 硬件：一台PC机。

2. 软件：Visual C++6.0。



- **#include <stdio.h>**
- **#include <malloc.h>**
- **typedef struct stnode**
- **{char data;**
- **struct stnode *next;**
- **}StackNode;**
- **int ishw(char str[])**
- **{int i=0;**
- **char ch;**
- **StackNode *st=NULL,*p;**
- **.....**
- **}**

```
void main()  
{char string[20];  
int t;  
printf("input a string:");  
scanf("%s",string);  
t=ishw(string);  
if(t) printf("The string is  
                HUIWEN.");  
else printf("The string is  
                not HUIWEN.");  
}
```



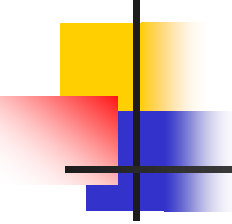
实验一（4） 线性表的应用-队列

■ 实验目的

1. 熟悉循环队列和链队列的类型定义和基本运算;
2. 学会利用队列的特点来解决实际问题。

■ 实验内容

1. 设从键盘输入一整数序列 a_1, a_2, \dots, a_n ，试编程实现：
当 $a_i < 0$ 时， a_i 进队；当 $a_i > 0$ 时，将队首元素出队；当 $a_i = 0$ 时，表示输入结束。最后输出队列中的所有元素。
要求：
 - 1) 采用循环队列存储结构；
 - 2) 有异常处理功能。

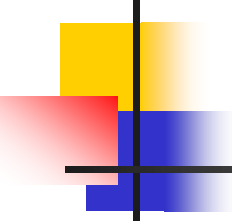


2. 设计一个程序，反映病人到医院看病、排队看医生的情况。

要求：采用链队列存储结构

■ 实验设备

1. 硬件：一台PC机。
2. 软件：Visual C++6.0。

- 
- #include "malloc.h"
 - #include "iostream.h"
 - typedef char QElemType;
 - typedef struct QNode{
 - QElemType data[10];
 - struct QNode *next;
 - }Qnode,*QueuePtr;
 -
 - typedef struct {
 - QueuePtr front;
 - QueuePtr rear;
 - }LinkQueue;



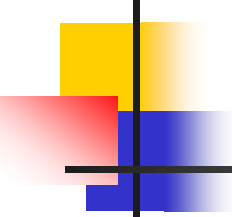
实验二 二叉树的应用

■ 实验目的

1. 熟悉二叉树的顺序存储结构和二叉链存储结构；
2. 熟悉二叉树的基本运算和遍历算法；
3. 学会利用二叉树的遍历算法来解决实际问题。

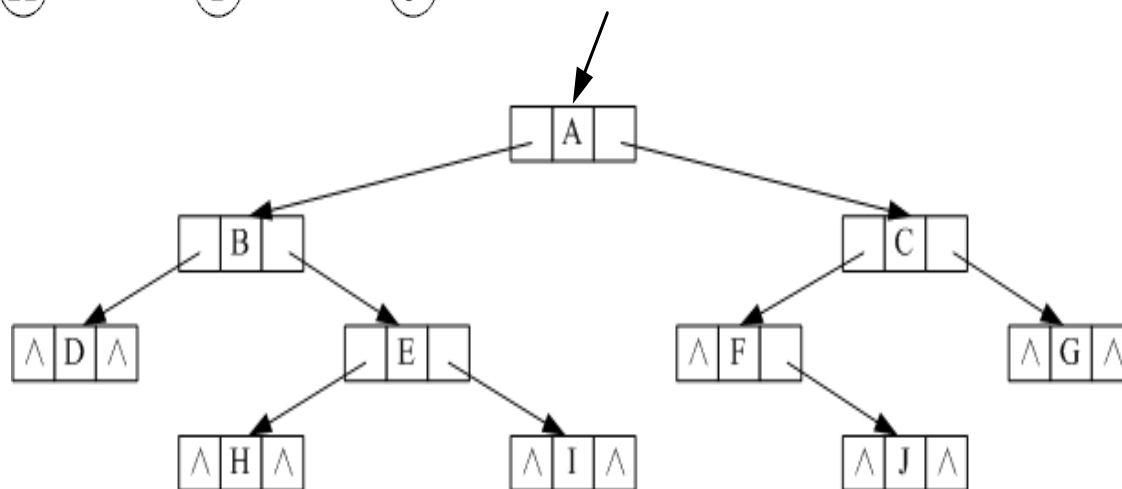
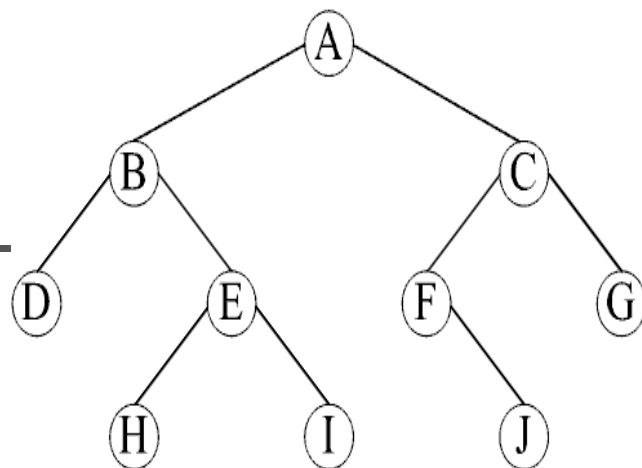
■ 实验内容

1. 设计一个程序，由给定的二叉树先序序列，建立其二叉链表存储结构，并求出二叉树的中序序列和后序序列。

- 
-
2. 假设二叉树采用二叉链存储结构，编写一个算法，求出二叉树中的叶子结点数。并设计主函数调用上述算法。

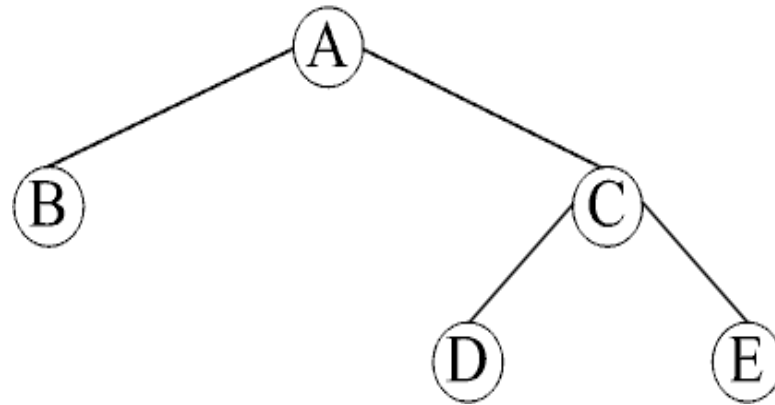
■ 实验设备

1. 硬件：一台PC机。
2. 软件：Visual C++6.0。



先序序列: **ABDEHICFJG**

输入: **ABD##EH##I##CF#J##G##**



输入: AB##CD##E##



```
#include "iostream.h"
```

```
#include "stdio.h"
```

```
#include "malloc.h"
```

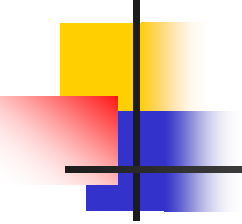
```
typedef struct Bnode
```

```
{
```

```
    ElemType data;
```

```
    struct Bnode *lchild,*rchild;
```

```
} BNode ,*BTree;
```



```
void preCreateBTree(BTree &BT)
{
    char data;
    data=getchar();
    if(data == '#')    BT = NULL;
    else
    {
        BT = (BTree)malloc(sizeof(Bnode));
        BT->data = data;
        preCreateBTree (BT->lchild);
        preCreateBTree (BT->rchild);
    }
}
```



```
void PreOrder(BTree bt)
```

```
{
```

```
    if(bt!=NULL)
```

```
    {
```

```
        cout<<bt->data;    //visit(bt)
```

```
        PreOrder( bt->lchild);
```

```
        PreOrder( bt->rchild);
```

```
    }
```

```
}
```



```
void InOrder(BTree *bt)
```

```
{
```

```
    if(bt!=NULL)
```

```
    {
```

```
        InOrder( bt->lchild);
```

```
        cout<<bt->data;
```

```
        InOrder( bt->rchild);
```

```
    }
```

```
}
```




```
void PostOrder(BTree *bt)
```

```
{
```

```
    if(bt!=NULL)
```

```
    {
```

```
        PostOrder( bt->lchild);
```

```
        PostOrder( bt->rchild);
```

```
        cout<<bt->data;
```

```
    }
```

```
}
```



```
int LeafCount(BTree bt)
```

```
{ int num1,num2;
```

```
if (bt==NULL) return 0;
```

```
else if (bt->lchild==NULL && bt->rchild==NULL)
```

```
return 1;
```

```
else
```

```
{ num1=LeafCount(bt->lchild);
```

```
num2=LeafCount(bt->rchild);
```

```
return(num1+num2);
```

```
}
```

```
}
```



```
void main()
```

```
{
```

```
    BiTree t;
```

```
    int leafs;
```

```
    printf("\n请输入结点 数据(包括虚结点): ");
```

```
    preCreateBTree(t);
```

```
    printf("\n先序序列: ");
```

```
    PreOrder(t); cout<<endl;
```

```
    printf("\n中序序列: ");
```

```
    InOrder(t); cout<<endl;
```

```
    printf("\n后序序列: ");
```

```
    PostOrder(t); cout<<endl;
```

```
    leafs= LeafCount(t);
```

```
    printf("\n叶子结点数: %d",leafs);
```

```
}
```



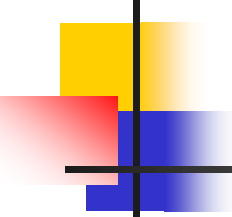
实验三 图的应用

■ 实验目的

1. 熟悉图的邻接矩阵存储结构和邻接表存储结构;
2. 掌握建立图的邻接矩阵和邻接表的算法;
3. 掌握图的遍历算法和顶点的度的求解方法。

■ 实验内容

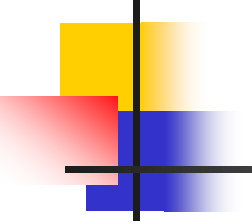
1. 设计一个程序, 采用交互方式建立一个网的邻接矩阵表示, 并且:
 - (1) 分行输出该邻接矩阵;
 - (2) 求出各顶点的度并输出。



2.设计一个程序，采用交互方式建立一个无向图的邻接表表示，并输出该图的深度优先搜索遍历得到的顶点序列。

■ 实验设备

1. 硬件：一台PC机。
2. 软件：Visual C++6.0。



```
#include "iostream.h"
#include "stdio.h"
#include "malloc.h"
#define MAXVEX 100

typedef char VertexType;
typedef struct edgenode
{
    ...
}ArcNode;
typedef struct vexnode
{
    ...
}VHeadNode;
typedef struct
{
    ...
}AdjList;
```



```
#include "iostream.h"
```

```
#include "stdio.h"
```

```
#define MAX_VERTEX_NUM 20
```

```
#define INFINITY 32555
```

```
typedef char VertexType;
```

```
typedef int AdjMatrix[MAX_VERTEX_NUM][MAX_VERTEX_NUM];
```

```
typedef struct {
```

```
    VertexType vexs[MAX_VERTEX_NUM];
```

```
    AdjMatrix arcs;
```

```
    int vexnum, arcnum;
```

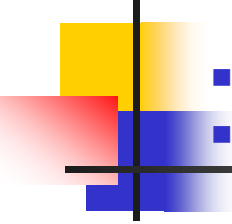
```
}MGraph;
```

```
void createUDN(MGraph &g)
```

```
{.....}
```

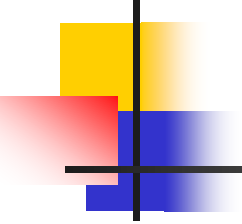
```
void dispUDN(MGraph g)
```

```
{.....}
```



```
void createUDN(MGraph &g){
    int i,j,k,w;
    cout<<"vexnum arcnum:";
    cin>>g.vexnum>>g.arcnum;
    cout<<"vexs:";
    for(i=0;i<g.vexnum;i++) cin>>g.vexs[i];

    for(i=0;i<g.vexnum;i++)
        for(j=0;j<g.vexnum;j++)
            g.arcs[i][j]=INFINITY;
    cout<<"i j w:\n";
    for(k=0;k<g.arcnum;k++){
        cin>>i>>j>>w;
        g.arcs[i][j]=w;
        g.arcs[j][i]=w;
    }
}
```

```
■ void dispUDN(MGraph g){
■   int i,j;
■   for(i=0;i<g. vexnum;i++)
■   { for(j=0;j<g. vexnum;j++)
■       if (g.arcs[i][j]==INFINITY) cout<<"& ";
■       else cout<<g.arcs[i][j]<<" ";
■       cout<<"\n";
■   }
■ }
■ void main()
■ { MGraph g;
■
■   createUDN(g);
■   cout<<"MGraph:\n";
■   dispUDN(g);
■ }
```



```
void CountDu(MGraph g)
```

```
{.....}
```

```
void main()
```

```
{
```

```
    MGraph g;
```

```
    createUDN(g);
```

```
    dispUDN(g);
```

```
    CountDu(g);
```

```
}
```



```
void CreateAdjList(ALGraph &g)
```

```
{.....}
```

```
void DispAdjList(ALGraph g)
```

```
{.....}
```

```
void DFS(ALGraph g,int vi)
```

```
{.....}
```

```
void main()
```

```
{
```

```
    ALGraph g;
```

```
    CreateAdjList(g);
```

```
    DispAdjList(g);
```

```
    DFS(g);
```

```
}
```



实验四 查找

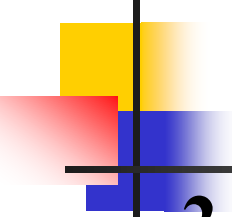
■ 实验目的

1. 熟悉二分查找算法;
2. 理解二叉排序树的定义和特性;
3. 掌握二叉排序树的相关算法;

■ 实验内容

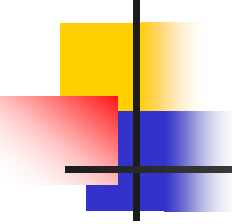
1. 在关键字有序序列中采用二分查找法查找关键字为给定值 k 的元素, 输出查找结果。

要求:有序序列和给定值 k 都从键盘输入。

- 
2. 给定关键字序列为{16, 5, 17, 29, 11, 3, 15, 20}，按表中元素的顺序依次插入，建立相应的二叉排序树，给出其中序序列。

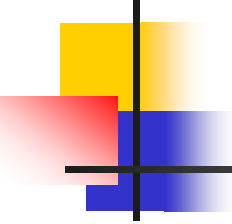
■ 实验设备

1. 硬件：一台PC机。
2. 软件：Visual C++6.0。

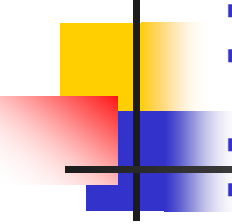
- 
- **#include "stdio.h"**
 - **typedef int KeyType;**
 - **typedef struct{**
 - **KeyType key;**
 - **} ElemType;**

 - **typedef struct**
 - **{ ElemType *elem;**
 - **int length;**
 - **} SStable;**

 - **int Search_Bin (SStable ST,KeyType k)**
 - **{ int low=1,high=ST.length,mid;**
 - **...**
 - **}**



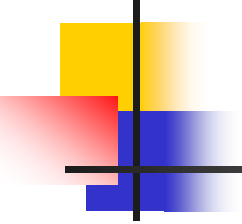
```
■ void main()
■ {
■     SStable table;
■     ElemType telem[11];
■     int i,key,location;
■     printf("input 10 elements:");
■     for(i=1;i<=10;i++)
■         scanf("%d",&telem[i].key);
■     table.elem=telem;
■     table.length=10;
■     printf("input the key to search:");
■     scanf("%d",&key);
■     location=Search_Bin (table,key);
■     if(location!=0)
■         printf("The location is %d\n",location);
■     else
■         printf("Not found!\n");
■ }
```



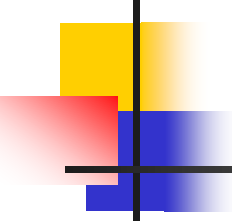
- typedef struct BitNode{
- ElemType data;
- struct BitNode *lchild,*rchild;
- }BitNode,*BiTree;

- BiTree insert(BiTree b, BiTree s)
- {
- if (b==NULL) b=s;
- else if (s->data.key>b->data.key) b->rchild=insert(b->rchild,s);
- else if (s->data.key<b->data.key) b->lchild=insert(b->lchild,s);
- return b;
- }

- BiTree creat(){
- int k;
- BiTree t,s;
- t=NULL;
- scanf("%d",&k);
- while (k!=-1){
- s=(BiTree)malloc(sizeof(BitNode));
- s->data.key=k;
- s->lchild=NULL;
- s->rchild=NULL;
- t=insert(t,s);
- scanf("%d",&k);
- }
- return t;
- }



```
■ void inorder(BiTree t){
■   if (t){
■       inorder(t->lchild);
■       printf("%3d",t->data);
■       inorder(t->rchild);
■   }
■ }
■ void main(){
■   BiTree t;
■   printf("input data,end in -1:");
■   t=creat();
■   printf("The inorder sequence is:");
■   inorder(t);
■ }
```



实验五 排序

■ 实验目的

1. 熟悉希尔排序的过程和算法;
2. 熟悉直接选择排序的过程和算法;
3. 熟悉快速排序的过程和算法;

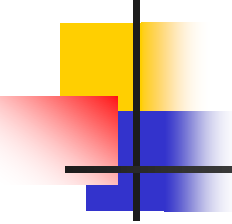
■ 实验内容

给出一组关键字序列{29,18,25,47,58,12,51,10}, 分别给出用希尔排序、直接选择排序和快速排序算法从小到大排序的结果。



■ 实验设备

1. 硬件：一台PC机。
2. 软件：Visual C++6.0。

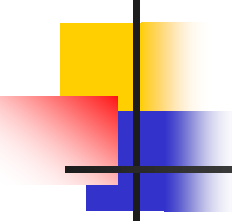


- #include "stdio.h"
- #define MAXSIZE 20

- typedef int KeyType;
- typedef struct{
- KeyType key;
- } RedType;

- typedef struct
- { RedType r[MAXSIZE+1];
- int length;
- } SqList;

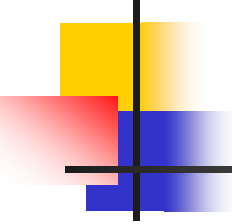
- void BInsertSort(SqList &L)
- { int i,j,low,high,mid;
- ...
- }



```
void main()
{
    SqList L;
    int i;
    printf("input 10 elements:");
    for(i=1;i<=10;i++)
        scanf("%d",&L.r[i].key);
    L.length=10;
    BInsertSort(L);
    printf("input the ordList:\n");
    for(i=1;i<=10;i++)
        printf("%5d",L.r[i].key);
}
```

- 
- 设计算法求两个元素值递增有序的顺序表L1和L2中的公共元素,并将其置入顺序表L3中。
-

- **typedef struct{**
- **ElemType *elem;**
- **int length;**
- **int listsize;**
- **}SqList;**



```
■ SqList_Jiao(SqList L1, SqList L2, SqList &L3)
■ {
■     if (L1.length<=L2.length)
■         L3.listsize=L1.length;
■     else L3.listsize=L2.length;
■     L3.elem=(ElemType*)malloc(L3.listsize*sizeof(ElemType));
■     if(!L3.elem) exit(OVERFLOW);
■     p1=L1.elem;p2=L2.elem;p3=L3.elem;
■     L3.length=0;
■     p1_last=L1.elem+L1.length-1;
■     p2_last=L2.elem+L2.length-1;
■     while(p1<=p1_last&& p2<=p2_last)
■     {
■         if (*p1<*p2) p1++;
■         else if (*p1>*p2 ) p2++;
■         else {*p3=*p1; p1++;p2++;p3++;L3.length++;}
■     }
■ }//SqList_Jiao
```