一）数据结构

1.单链表的建立、插入、删除

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

typedef struct node
{
    int data;
    struct node *next;
}node;

//固定长度单链表的建立
node *Create(int n)
{
    node *p,*q,*h=0;
    int i;
    printf("请输入%d个数:\n",n);
    for(i=0;i<n;i++)
    {
        q=(node *)malloc(sizeof(node));
        scanf("%d",&q->data);
        q->next=NULL;

        if(h==0)
        {
            h=q;
            p=q;
        }
        else
        {
            p->next=q;
            p=q;
        }
    }
    return h;
}
//单链表的打印
void Print(node *h)
{
    node *p=h;
    while(p)
    {
        printf("%d",p->data);
        if(p->next)
        {
            printf("->");
        }
        p=p->next;
    }
    printf("\n");
}
//非固定长度单链表的建立
node *Createany()
{
    node *p,*q,*h=0;
    int x;
    printf("请输入任意多个正数:\n");
    while(1)
    {
        scanf("%d",&x);
        if(x!=-1)//输入-1退出录入
        {
```

```
            q=(node *)malloc(sizeof(node));
            q->data=x;
            q->next=NULL;
            if(h==0)
            {
                h=q;
                p=q;
            }
            else
            {
                p->next=q;
                p=q;
            }
        }
        else
        {
            break;
        }
    }
    return h;
}
//未知长度链表测长
int Findlen(node *h)
{
    node *q=h;
    int i=0;
    while(q)
    {
        i++;
        q=q->next;
    }
    return i;
}

//删除单链表节点
node *Delchain(node *h,int mydata)
{
    node *q=h,*t;
    while(q)
    {
        if(mydata==q->data)
        {
            if(q==h)//删除的是表头(包括只有表头一个节点和多于一个节点两种情况）
            {
                if(q->next)
                {
                    h=q->next;
                }
                else
                {
                    h=NULL;
                }
            }
            else if(q->next==NULL)//删除的是表尾
            {
                t->next=NULL;
            }
            else
            {
                t->next=q->next;
            }
            free(q);
            break;
        }
        else
        {
            t=q;
```

```
            q=q->next;

        }
    }
    if(q==NULL)//q一直遍历到了最后为NULL还没有找到相等的数
    {
        printf("没有这个结点!\n");
    }
    return h;

}

//增加单链表的节点到合适位置（按从小到大排列）
node *Insert(node *head, int mydata)
{
    node *q=head,*p=(node *)malloc(sizeof(node)),*t;
    p->data=mydata;
    p->next=NULL;
    if(mydata<head->data)//如果mydata比表头的data还小
    {
        p->next=head;
        head=p;
    }
    else
    {
        while(q!=NULL&&mydata>q->data)
        {
            t=q;
            q=q->next;
        }
        if(q)
        {
            t->next=p;
            p->next=q;
        }
        else
        {
            t->next=p;
        }
    }
    return head;


}
main()
{
    node *myhead,*myhead2,*myhead3,*myhead4;
    int i,len,mydata,mydata2;
    myhead=Create(5);
    Print(myhead);

    /*
    myhead2=Createany();
    Print(myhead2);
    len=Findlen(myhead2);
    printf("链表长度为%d\n",len);

    printf("请输入要删除的节点数据域:\n");
    scanf("%d",&mydata);
    myhead3=Delchain(myhead,mydata);
    Print(myhead3);
    */

    printf("请输入要添加的节点数据域:\n");
    scanf("%d",&mydata2);
    myhead4=Insert(myhead,mydata2);
    Print(myhead4);
```

```
    system("pause");
}
```

2.双链表

创建双链表，并实现其删除、插入结点。（见《程序员面试宝典》P139)

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

typedef struct student
{
    int data;
    struct student *next;
    struct student *pre;
}dnode;

dnode *creat()//创建双链表，以输入0为链表结束标志
{
    dnode *head,*p,*s;
    int x,cycle=1;
    head=(dnode *)malloc(sizeof(dnode));
    p=head;
    while(cycle)
    {
        printf("请输入链表数据域data：\n");
        scanf("%d",&x);
        if(x!=0)
        {
            s=(dnode *)malloc(sizeof(dnode));
            s->data=x;
            printf("%d\n",s->data);
            p->next=s;
            s->pre=p;
            p=s;

        }
        else
        {
            cycle=0;
        }
    }
    head=head->next;
    head->pre=NULL;
    p->next=NULL;

    printf("\n yyy %d",head->data);
    return(head);
}

void print(dnode *h)
{
    while(h)
    {
        printf("data=%d\n",h->data);
        h=h->next;
    }

}

dnode *del(dnode *h,int num)//双链表删除结点
{
    dnode *p,*q,*t;
    p=h;
    while(p->next!=NULL&&p->data!=num)
    {
```

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

    if(p->data==num)
    {
        if(p==h)
        {
            h=p->next;
            h->pre=NULL;
            free(p);
        }
        else
        {
            t=p->next;
            q->next=t;
            t->pre=q;
        }
    }
    else
    {
        printf("无此结点!\n");
    }
    return h;
}

dnode *insert(dnode *h,int n)//双链表插入结点
{
    dnode *p,*q,*t;
    q=(dnode *)malloc(sizeof(dnode));
    q->data=n;
    p=h;
    while(p->next!=NULL&&(p->data<q->data))
    {
        t=p;
        p=p->next;
    }

    if(p==h)//如果插入的这个值最小，插在链表最前端
    {
        h=q;
        h->next=p;
        h->pre=NULL;
    }
    else if(p->data>=q->data)
    {
        t->next=q;
        q->pre=t;
        q->next=p;
        p->pre=q;
    }
    else//p->next==NULL
    {
        p->next=q;
        q->pre=p;
        q->next=NULL;
    }

    return h;

}


main()
{
    dnode *head,*head1,*head2,*head3,*head4;
    head=creat();//创建双链表，数据域为2，3，4，5
```

```
    print(head);
    printf("\n");

    head1=del(head,3);//删除数据域data=3的结点
    print(head1);
    printf("\n");

    head2=insert(head1,3);//插入数据域data=3的结点
    print(head2);
    printf("\n");

    head3=insert(head2,6);//插入数据域data=6的结点
    print(head3);
    printf("\n");

    head4=insert(head2,1);//插入数据域data=1的结点
    print(head4);
    printf("\n");

}
```

3.循环单链表

已知n个人（编号为1，2，3，……n)围坐在一张圆桌周围。从编号为k的人开始报数，从1开始报数，数到m的那个人出列；他的下一个又从1报数，数到m出列。依次重复下去，使所有人全部出列。

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

typedef struct LNode
{
    int data;
    struct LNode *link;
}LNode,*LinkList;

void JOSEPHUS(int n,int k,int m)
{
    LinkList p,r,list,curr,t;//都是指针
    int i,s;
    p=(LinkList)malloc(sizeof(LNode));
    p->data=0;//第一个结点p装载数据域为0
    p->link=p;
    curr=p;
    for(i=1;i<n;i++)//链表的数据域装载的是从0到n-1的数值，循环从1开始装载。
    {
        t=(LinkList)malloc(sizeof(LNode));
        t->data=i;
        t->link=curr->link;
        curr->link=t;
        curr=t;
    }

    while(k--)
    {
        r=p;//p本来为序号0的结点，这样循环（k-1）次，最后p指向k-1,k-1就是第k个人。
        p=p->link;
    }
    while(n--)
    {
        for(s=m-1;s>=0;s--)//执行循环m次，最后p指向data=(k-1+m-1)，即报数为m的人
        {
            r=p;//r保存的是指向第k个人（即data=k-1）的人的指针
            p=p->link;
        }

        r->link=p->link;//指向第k个人（即data=k-1）的r的link指向报完数的下一个人（即data=k-1+m）
```

```
        printf("%d->",p->data);//输出报数为m(即data=k-1+m)的人的data
        free(p);//释放这个报数的人的存储空间
        p=r->link;//p指向报完数的下一个人（即data=k+m)
    }

}

main()
{
    JOSEPHUS(13,4,1);

    system("pause");
}
```

4.队列的建立，入队出队

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

typedef struct node
{
    int data;
    struct node *next;
}node;

typedef struct queue
{
    node *head,*rear;
}queue;

//创建队列
queue * Createqueue()
{
    node *q,*p,*h=NULL;
    queue *myqueue=(queue *)malloc(sizeof(queue));
    int x;

    printf("请输入队列节点数据域值:\n");
    while(1)
    {
        scanf("%d",&x);
        if(x!=-1)
        {
            q=(node *)malloc(sizeof(node));
            q->data=x;
            q->next=NULL;

            if(h==NULL)
            {
                h=q;
                p=q;
            }
            else
            {
                p->next=q;
                p=q;
            }
        }
        else
        {
            break;
        }
    }

    if(h==NULL)
    {
```

```
        myqueue->head=NULL;
        myqueue->rear=NULL;
    }
    else
    {
        myqueue->head=h;
        myqueue->rear=q;//若h==q，这样写也可以包含
    }
    return myqueue;

}

//打印队列
void Print(queue *myqueue)
{
    node *q;
    q=myqueue->head;
    while(q)
    {
        printf("%d",q->data);
        if(q->next)
        {
            printf("->");
        }
        q=q->next;
    }
}

//队列入列
queue *Insert(queue *myqueue,int x)
{
    node *q=(node *)malloc(sizeof(node));
    q->data=x;
    q->next=NULL;

    if(myqueue->head==NULL)
    {
        myqueue->head=q;
        myqueue->rear=q;
    }
    else
    {
        myqueue->rear->next=q;
        myqueue->rear=q;
    }
    return myqueue;
}
//队列出队
queue *Del(queue *myqueue)
{
    node *q;
    if(myqueue->head==NULL)
    {
        printf("溢出\n");
    }
    else if(myqueue->head==myqueue->rear)
    {
        myqueue->head=NULL;
        myqueue->rear=NULL;
    }
    else
    {
        q=myqueue->head;
        myqueue->head=q->next;
        free(q);
    }
    return myqueue;
```

```
}

main()
{
    queue *myqueue=Createqueue();
    int x;
    Print(myqueue);

    printf("请输入要入队的值:\n");
    scanf("%d",&x);
    myqueue=Insert(myqueue,x);
    Print(myqueue);
    myqueue=Del(myqueue);
    Print(myqueue);

    system("pause");

}
```

6.栈

编程实现入栈出栈操作

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

typedef struct student
{
    int data;
    struct student *next;
}node;

typedef struct stackqueue
{
    node *zhandi,*top;
}queue;

queue *push(queue *HQ,int x)//入栈
{
    node *s;
    s=(node *)malloc(sizeof(node));
    s->data=x;
    s->next=NULL;

    if(HQ->zhandi==NULL)
    {
        HQ->top=s;
        HQ->zhandi=s;

    }
    else
    {
        s->next=HQ->top;
        HQ->top=s;
    }


    return HQ;
}

queue *pop(queue *HQ)//出栈
{
    node *s;
    if (HQ->top==NULL)
    {
        printf("溢出!\n");
```

```
    }
    else
    {
        s=HQ->top;
        printf("弹出%d\n",s->data);
        if(HQ->top==HQ->zhandi)
        {
            HQ->top=NULL;
            HQ->zhandi=NULL;
        }
        else
        {
            HQ->top=s->next;
            HQ->top->next=s->next->next;
        }



    }
    return HQ;
}

main()
{
    queue *myHQ;
    int i;
    myHQ=(queue *)malloc(sizeof(queue));
    myHQ->top=NULL;
    myHQ->zhandi=NULL;//一定要先初始化

    for(i=1;i<=5;i++)
    {
        myHQ=push(myHQ,i);
        printf("现在栈顶为%d\n",myHQ->top->data);
        printf("现在栈底为%d\n",myHQ->zhandi->data);

    }

    for(i=1;i<=4;i++)
    {
      myHQ=pop(myHQ);
        printf("现在的栈顶是%d\n",myHQ->top->data);
        printf("现在栈底为%d\n",myHQ->zhandi->data);
    }

    system("pause");

}
```

5.二叉树的操作

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

typedef struct node
{
    char data;
    struct node *lchild,*rchild;
}node,*Tree;

Tree CreateTree()//创建二叉树
{
    node *p=NULL;
    char c;
```

```
    scanf("%c",&c);
    if(c!='#')
    {
        p=(node *)malloc(sizeof(node));
        p->data=c;
        p->lchild=CreateTree();
        p->rchild=CreateTree();
    }
    else
    {
        p=NULL;
    }
    return p;
}

void PrePrintTree(Tree mytree)
{
    node *p=mytree;
    char c;
    if(p)
    {
        c=p->data;
        printf("%c",c);
        PrePrintTree(p->lchild);
        PrePrintTree(p->rchild);
    }
}

void MidPrintTree(Tree mytree)
{
    node *p=mytree;
    char c;
    if(p)
    {
        c=p->data;
        MidPrintTree(p->lchild);
        printf("%c",c);
        MidPrintTree(p->rchild);
    }
}

void LastPrintTree(Tree mytree)
{
    node *p=mytree;
    char c;
    if(p)
    {
        c=p->data;
        LastPrintTree(p->lchild);
        LastPrintTree(p->rchild);
        printf("%c",c);
    }
}

void PrePrintLeaf(Tree mytree)//先序输出叶子节点
{
    node *p=mytree;
    char c;
    if(p)
    {
        if(p->lchild==NULL&&p->rchild==NULL)
        {
            c=p->data;
            printf("%c",c);
        }
        else
        {
```

```
            PrePrintLeaf(p->lchild);
            PrePrintLeaf(p->rchild);
        }

    }
}

int CountLeaf(Tree mytree)//统计叶子节点数目
{
    node *p=mytree;
    static int n=0;
    if(p)
    {
        if(p->lchild==NULL&&p->rchild==NULL)
        {
            n++;
        }
        else
        {
            CountLeaf(p->lchild);
            CountLeaf(p->rchild);
        }
    }
    return n;
}

int CountLeaf2(Tree mytree)//统计叶子节点数目方法二
{
    node *p=mytree;
    int n=0;
    if(p)
    {
        if(p->lchild==NULL&&p->rchild==NULL)
        {
            n=1;
        }
        else
        {
            n=CountLeaf2(p->lchild)+CountLeaf2(p->rchild);
        }
    }
    return n;
}



int CountTreeHeight(Tree mytree)//计算树的高度
{
    node *p=mytree;
    int n=0,l,r;
    if(p)
    {
        if(p->lchild==NULL&&p->rchild==NULL)
        {
            n=1;
        }
        else
        {
            l=CountTreeHeight(p->lchild);
            r=CountTreeHeight(p->rchild);
            n=l>r?(l+1):(r+1);

        }
    }
    return n;
}
```

```
void main()
{
    Tree mytree=NULL;
    int n1,n2,n3;
    printf("请输入二叉树各节点:\n");
    mytree=CreateTree();

    PrePrintTree(mytree);
    printf("\n");

    MidPrintTree(mytree);
    printf("\n");

    LastPrintTree(mytree);
    printf("\n");

    PrePrintLeaf(mytree);
    printf("\n");

    n1=CountLeaf(mytree);
    printf("%d\n",n1);

    n2=CountLeaf2(mytree);
    printf("%d\n",n2);

    n3=CountTreeHeight(mytree);
    printf("%d\n",n3);
    system("pause");

}
```

二）排序

1.交换排序

（1）普通冒泡

- 时间复杂度：最差、平均都是O(n^2)，最好是O(n)
- 空间复杂度：O(1)

```
# include<stdio.h>

void bubble(int *list,int len)
{
    int i,j,t,flag=0;
    for(i=0;i<len-1;i++)
    {
        flag=0;//设置标记，当某一轮交换没有交换任何数，那下一轮交换也不必进行了
        for(j=0;j<len-1-i;j++)
        {
            if(list[j]>list[j+1])
            {
                t=list[j];
                list[j]=list[j+1];
                list[j+1]=t;
                flag=1;
            }
        }
        if(flag==0)
        {
            break;
        }
    }
```

```
}

void main()
{
    int n,list[10];
    printf("请输入10个整数：");
    for(n=0;n<10;n++)
    {
        scanf("%d",&list[n]);
    }
    printf("\n");
    bubble(list,10);
    for(n=0;n<10;n++)
    {
        printf("%d\t",list[n]);
    }
    printf("\n");

}
```

（2）鸡尾酒排序：双向冒泡

- 时间复杂度：最差、平均都是O(n^2)，最好是O(n)

- 空间复杂度：O(1)

```
#include<stdio.h>

void CocktailBubble(int *list,int n)
{
    int low=0,high=n-1,j,t,flag;
    while(low<high)
    {
        flag=0;//一次进行两趟for循环，第一个for循环排最大值（次大值），第二个for循环排最小值（次小值），只要其中一趟没有交换任何数字就可以结束排序
        for(j=low;j<high;j++)
        {
            if(list[j]>list[j+1])
            {
                t=list[j];
                list[j]=list[j+1];
                list[j+1]=t;
                flag=1;
            }
        }
        if(flag==0)
        {
            break;
        }
        high--;//上述for循环第一次结束，排完最大值；第二次，排完次大值
        for(j=high;j>low;j--)
        {
            if(list[j]<list[j-1])
            {
                t=list[j];
                list[j]=list[j-1];
                list[j-1]=t;
            }
        }
        if(flag==0)
        {
            break;
        }
        low++;//上述for循环第一次结束，排完最小值；第二次，排完次小值
```

```
    }
}

void main(){
    int i,list[10];
    printf("请输入10个整数:");
    for(i=0;i<10;i++){
        scanf("%d",&list[i]);
    }
    for(i=0;i<10;i++){
        printf("%d ",list[i]);
    }
    printf("\n");
    CocktailBubble(list,10);
    for(i=0;i<10;i++){
        printf("%d ",list[i]);
    }
    printf("\n");

    while(1){
        ;
    }
}
```

（3）快速排序法

- 时间复杂度：平均O(nlogn),最坏O(n^2)
- 空间复杂度:O(logn)

include<stdio.h>

include<stdlib.h>

int Partition(int *list,int low,int high)//该划分算法的工作为：排好list[low]，然后返回list[low]排好后的位置 { int i=low,j=high; int t=list[low];

```
while(i<j)
{
    while(list[j]>=t&&i<j)
    {
        j--;
    }
    if(i<j)
    {
        list[i]=list[j];
        i++;
    }


    while(list[i]<=t&&i<j)
    {
        i++;
    }
    if(i<j)
    {
        list[j]=list[i];
        j--;
    }

}
list[i]=t;
return i;
```

}

void QuickSort(int *list,int low,int high) { int p; if(low<high) { p=Partition(list,low,high);//每运行一次，排好list[low]，然后返回list[low]值排好后的位置，然后对其左右区间进行递归 QuickSort(list,low,p-1); QuickSort(list,p+1,high); } }

main() { int list[10],i; printf("请输入10个整数：\n"); for(i=0;i<10;i++) { scanf("%d",&list[i]); }

```
QuickSort(list,0,9);

for(i=0;i<10;i++)
{
    printf("%d\t",list[i]);
}


system("pause");
```

}

2.插入排序

(1)直接插入法

时间复杂度：最差、平均都是O(n^2)，最好是O(n)

空间复杂度：1

```
#include<stdio.h>
#include<stdlib.h>
void insertion(int *list,int n)
{
    int i,j,t;
    for(i=1;i<n;i++)//待插入的是list[1]到list[n-1]
    {
        if(list[i]<list[i-1])
        {
            t=list[i];
            list[i]=list[i-1];
            j=i;
            while(t<list[j-1]&&j>=1)
            {
                list[j]=list[j-1];
                j--;
            }
            list[j]=t;
        }

    }
}
main()
{
    int list[10],n=10,i;

    printf("请输入10个整数：\n");
    for(i=0;i<10;i++)
    {
        scanf("%d",&list[i]);
    }

    insertion(list,10);

    for(i=0;i<10;i++)
    {
        printf("%d\t",list[i]);
    }
```

```
    system("pause");
}
```

2）shell排序：分组的插入排序

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

void ShellPass(int *list,int n,int incre)
{
    int i,j,t,k;//以第一个步长list[0]……list[incre-1]为基准，待插入的是list[incre]……list[n-1]，每个插的时候和自己的增量组比较。

    for(i=incre;i<n;i++)
    {
        if(list[i]<list[i-incre])
        {
            t=list[i];
            list[i]=list[i-incre];
            j=i;
            while(t<list[j-incre]&&j>=incre)
            {
                list[j]=list[j-incre];
                j-=incre;
            }
            list[j]=t;

        }
    }

}
void ShellSort(int *list,int n)
{
    int incre=n;
    while(1)
    {
        incre=incre/3+1;
        ShellPass(list,n,incre);
        if(incre==1)
        {
            break;
        }
    }
}

main()
{
    int list[10],n=10,i;

    printf("请输入10个整数：\n");
    for(i=0;i<10;i++)
    {
        scanf("%d",&list[i]);
    }

    ShellSort(list,10);

    for(i=0;i<10;i++)
    {
        printf("%d\t",list[i]);
    }


    system("pause");
}
```

3.选择排序

（1）普通选择排序

时间复杂度：最差、平均都是O(n^2)，最好是O(n)

空间复杂度：1

稳定性：**不稳定**

```
#include<stdio.h>
void SimpleSort(int *list,int n)
{
    int i,j,k,t;
    for(i=0;i<n;i++)
    {
        k=i;
        for(j=i+1;j<n;j++)
        {
            if(list[j]<list[k])
            {
                k=j;//将比list[k]小的list[j]的j存入k
            }
        }
        if(k!=i)//每进行一次循环找出list[i]到list[n-1]中最小的那个，将这个最小值与list[i]交换位置
        {
            t=list[i];
            list[i]=list[k];
            list[k]=t;
        }
    }
}

main()
{
    int list[10],n=10,i;
    printf("请输入10个整数：\n");
    for(i=0;i<10;i++)
    {
        scanf("%d",&list[i]);
    }
    SimpleSort(list,10);

    for(i=0;i<10;i++)
    {
        printf("%d",list[i]);
    }
    system("pause");
}
```

4.归并排序

时间复杂度：最差、平均、最好都是O(nlogn)

空间复杂度：O(n)

```
#include<stdio.h>
#include<stdlib.h>
void Merge(int *list,int *newlist,int s,int m,int t)
{
    int i,j,k;
    i=s;
    j=m+1;
    k=s;
    while(i<=m&&j<=t)
```

```
    {
        if(list[i]<=list[j])
        {
            newlist[k]=list[i];
            k++;
            i++;
        }
        else
        {
            newlist[k]=list[j];
            k++;
            j++;
        }
    }

    while(i<=m)//剩余的若是前一个序列，则其直接按并入newlist
    {
        newlist[k]=list[i];
        i++;
        k++;
    }
    while(j<=t)
    {
        newlist[k]=list[j];
        j++;
        k++;
    }
}

void MSort(int *list,int *newlist,int s,int t)
{
    int *newlist2;
    int m;
    newlist2=(int *)malloc((t-s+1)*sizeof(int));//分配一个新数组，和list等长


    if(s==t)
    {
        newlist[s]=list[s];
    }
    else
    {
        m=(s+t)/2;
        MSort(list,newlist2,s,m);//将list[s]……list[m]递归归并为有序的newlist2[s]……newlist2[m]
        MSort(list,newlist2,m+1,t);//将list[m+1]……list[t]递归归并为有序的newlist2[m+1]……newlist2[t]
        Merge(newlist2,newlist,s,m,t);//将newlist2[s]……newlist2[m]和newlist2[m+1]……newlist2[t]归并到newlist[s]……newlist[t]
    }
}
void MergeSort(int *list,int *newlist,int n)
{
    MSort(list,newlist,0,n-1);
}
main()
{
    int list[10],n=10,i,newlist[10];
    printf("请输入10个整数：\n");
    for(i=0;i<10;i++)
    {
        scanf("%d",&list[i]);
    }
    MergeSort(list,newlist,10);

    for(i=0;i<10;i++)
    {
        printf("%d",newlist[i]);
    }
```

```
    system("pause");
}
```

三）查找

1.二分法查找

时间复杂度： O(logn)

```
#include<stdio.h>
#include<stdlib.h>
void ErfenSearch(int *list,int n,int x,int *index)
{
    int low,high,k;
    low=0;
    high=n-1;
    if(x<=list[high]&&x>=list[low])
    {
        while(low<=high)
        {
            k=(low+high)/2;
            if(x<list[k])
            {
                high=k-1;
            }
            else if(x>list[k])
            {
                low=k+1;
            }
            else
            {
                break;
            }
        }

        *index=k;
    }
    else
    {
        *index=-1;
    }

}
main()
{
    int list[10],n=10,i,x;
    int rel;
    printf("请输入10个整数：\n");
    for(i=0;i<10;i++)
    {
        scanf("%d",&list[i]);
    }
    printf("请输入待查找的数:\n");
    scanf("%d",&x);

    ErfenSearch(list,10,x,&rel);

    printf("%d",rel);


    system("pause");
}
```

四）经典编程题

1.格雷码生成

格雷码生成方法：某数右移1位后，和原数按位异或，即得到该原数格雷码

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

//生成某整数的格雷码
void Graycode(int *t)
{
    int a,b;
    a=*t;
    b=a>>1;
    *t=a^b;

}

//把某数化为二进制
char *Turntobit(int data)//设data是小于256的正整数，那么二进制数用8位表示即可
{
    char a,t;
    char *k=(char *)malloc(8*sizeof(char));//一定要这样分配存储空间，才能返回k为指针
    int i=0;
    while(data)
    {
        a=(char)(data%2+'0');
        data=data/2;
        k[i]=a;
        i++;
    }
    while(i<=7)
    {
        k[i]='0';
        i++;
    }
    for(i=0;i<4;i++)
    {
        t=k[i];
        k[i]=k[7-i];
        k[7-i]=t;
    }
    return k;
}

void main()
{
    int d,i=0;
    char *c,*c1;
    c=(char *)malloc(8*sizeof(char));
    c1=(char *)malloc(8*sizeof(char));
    printf("请输入一个小于256的正整数:\n");
    scanf("%d",&d);
    c=Turntobit(d);
    printf("其化为二进制是:\n");
    for(i=0;i<8;i++)
    {
        printf("%c",c[i]);
    }
    printf("\n");

    Graycode(&d);
    printf("其格雷码是%d",d);
```

```
    c1=Turntobit(d);
    printf("其化为二进制是:\n");
    for(i=0;i<8;i++)
    {
        printf("%c",c1[i]);
    }
    printf("\n");
    system("pause");
}
```

//递归生成格雷码

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

void PrintBit(int Num,int n){
    char* a=(char*)malloc((1<<n)+1);
    int i;
    a[n]='\0';
    for(i=n-1;i>=0;i--){
        a[i]=Num%2+'0';
        Num/=2;
    }
    printf("%s ",a);
}

int* GrayCode(int n){
    int* Gray=(int*)malloc(1<<n);
    int* temp;
    int i;
    if(n==1){
        Gray[0]=0;
        Gray[1]=1;
        return Gray;
    }
    temp=GrayCode(n-1);
    for(i=0;i<((1<<n)>>1);i++){
        Gray[i]=temp[i];
        Gray[(1<<n)-i-1]=temp[i]+(1<<(n-1));
    }
    return Gray;
}


void main(){
    int* a;
    int n=4;
    int i;
    a=GrayCode(n);
    for(i=0;i<1<<n;i++){
        PrintBit(a[i],n);
    }
    system("pause");
}
```

2.大数相加

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#define N 1000
void Bigadd(char *a,char *b,char *c)
{
    int m=strlen(a),n=strlen(b);
```

```
    char *k=a;
    int i,j,h,t,jinwei;
    if(m<n)
    {
        a=b;
        b=k;
        m=strlen(a);
        n=strlen(b);
    }
    memset(c,'0',(m+1)*sizeof(char));
    c[m+1]='\0';

    i=m-1;
    j=n-1;
    h=m;
    jinwei=0;
    while(j>=0)
    {
        t=(int)(a[i]-'0')+(int)(b[j]-'0')+jinwei;
        c[h]=(char)(t%10+'0');
        jinwei=t/10;
        i--;
        j--;
        h--;
    }
    while(i>=0)
    {
        t=(int)(a[i]-'0')+jinwei;
        c[h]=(char)(t%10+'0');
        jinwei=t/10;
        i--;
        h--;
    }
    c[h]=(char)(jinwei+'0');//这时候h等于0

    if(c[0]=='0')
    {
        c[0]=' ';
    }

}

void main()
{
    char *a=(char *)malloc(N);
    char *b=(char *)malloc(N);
    char *c=(char *)malloc(N+1);

    printf("请输入第一个加数：\n");
    gets(a);
    printf("请输入第二个加数：\n");
    gets(b);

    Bigadd(a,b,c);
    printf("和为:\n");
    puts(c);

    system("pause");
}
```

3.大数相乘

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#define max 100
```

```
void Bigmulti(char *a,char *b,char *c)
{
    int m=strlen(a);
    int n=strlen(b);
    int t,jinwei;
    int i,j,h;

    memset(c,'0',m+n);
    c[m+n]='\0';

    for(i=n-1;i>=0;i--)
    {
        jinwei=0;
        j=m-1;
        h=m+i;
        while(j>=0)
        {
            t=(int)(b[i]-'0')*(int)(a[j]-'0')+(int)(c[h]-'0')+jinwei;
            c[h]=(char)(t%10+'0');
            jinwei=t/10;
            j--;
            h--;
        }
        while(jinwei)
        {
            t=(int)(c[h]-'0')+jinwei;
            c[h]=(char)(t%10+'0');
            jinwei=t/10;
            h--;
        }
    }

    if(c[0]=='0')
    {
        c[0]=' ';
    }
}

void main()
{
    char a[max],b[max],c[2*max];
    printf("请输入被乘数:\n");
    gets(a);
    printf("请输入乘数:\n");
    gets(b);
    Bigmulti(a,b,c);
    puts(c);
    system("pause");
}
```

4.发红包问题：某个数值的红包次数超过红包总数一半，求该红包

```
    #include<stdio.h>
    void CocktailSort(float *gift,int n)
    {
        int low=0,high=n-1,i,flag;
        float t;

        while(low<high)
        {
            flag=0;

            if(low<high)
            {
                for(i=low;i<high;i++)
```

```
            {
                if(gift[i]>gift[i+1])
                {
                    t=gift[i];
                    gift[i]=gift[i+1];
                    gift[i+1]=t;
                    flag=1;
                }
            }
        }

        if(flag==0)
        {
            break;
        }
        high--;

        flag=0;
        if(low<high)
        {
            for(i=high;i>low;i--)
            {
                if(gift[i]<gift[i-1])
                {
                    t=gift[i];
                    gift[i]=gift[i-1];
                    gift[i-1]=t;
                    flag=1;
                }
            }
       }
       if(flag==0)
       {
            break;
       }
       low++;

    }
}

int searchGift(float *sortedgifts,int n)
{
    float  t=sortedgifts[0];
    int i,count=1;

    for(i=1;i<n;i++)
    {
        if(sortedgifts[i]==t)
        {
           count++;
        }
        else
        {
            t=sortedgifts[i];
            count=1;
        }
        if(count>n/2)
        {
            return t;
        }
    }


}

void main()
{
```

```
        float gift[5]={1,2,3,2,2},rel;
        int n=5,i;

        CocktailSort(gift,5);
        for(i=0;i<5;i++)
        {
            printf("%f\t",gift[i]);
        }
        printf("\n");
        rel=searchGift(gift,5);
        printf("%f\n",rel);

        system("pause");
    }
```

5.A，B两个整数集合，设计一个算法求他们的交集。

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

#define max 100
int *qiujiao(int *a,int *b,int al,int bl, int *cll)
{
    int *p=a,t=al,i,j,h=0,*c=NULL,*rel=NULL;
    if(al<bl)
    {
        a=b;
        b=p;
        al=bl;
        bl=t;
    }

    c=(int *)malloc(bl*sizeof(int));
    memset(c,0,bl*sizeof(int));

    for(i=0;i<bl;i++)
    {
        for(j=0;j<al;j++)
        {
            if(a[j]==b[i])
            {
                c[h]=a[j];
                h++;
            }
        }
    }

    rel=(int *)malloc(h*sizeof(int));
    *cll=h;
    for(i=0;i<h;i++)
    {
        rel[i]=c[i];
    }
    return rel;
}

main()
{
    int a[max],b[max],al,bl,i=0,*c=NULL,cl;

    printf("请输入数组a,以回车结束:\n");

    while(1)
    {
        scanf("%d",&a[i]);
```

```
        i++;
        if(getchar()=='\n')
        {
            break;
        }
    }
    al=i;

    i=0;
    printf("请输入数组b,以回车结束:\n");
    while(1)
    {
        scanf("%d",&b[i]);
        i++;
        if(getchar()=='\n')
        {
            break;
        }


    }
    bl=i;

    c=qiujiao(a,b,al,bl,&cl);
    for(i=0;i<cl;i++)
    {
        printf("%d\t",c[i]);
    }

    system("pause");
}
```

6. 已知rand7()可以产生1~7的7个数(均匀概率),利用rand7() 产 生 rand10() 1~10(均匀概率)。

```
int rand10()
{
    int i=rand7()-1;
    int j=rand7()-1;
    int num=i*7+j;
    if(num>=40)
    {
        return rand10();
    }
    else
    {
        return num%10+1;
    }
}
```

and7()-1等概率地产生0~6的数。
两个（rand7()-1）可以等概率地产生0~48个数，每个数产生的概率都是 1/49;
然后当产生 40 - 48 这9个数时我们重新获取，这样相当于把 40 - 48 的概率均摊到 0 - 39 上面， 所以不会影响 0 - 39 的等概率产生。
然后余10加即可等概率得到0~9的数，加1即可得到1~10的数。

五）字符串相关编程题

1.整数化为二进制数

写法1：结果作为局部函数返回值

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

```
char *Intostr(int data)
{
    char *str,t;
    int i=0;
    str=(char *)malloc(8*sizeof(char));//必须这样写，才和返回类型相配
    memset(str,'0',8);
    while(data)
    {
        str[i]=(char)(data%2+'0');
        data=data/2;
        i++;
    }
    while(i<=7)
    {
        str[i]='0';
        i++;
    }
    for(i=0;i<4;i++)
    {
        t=str[i];
        str[i]=str[7-i];
        str[7-i]=t;
    }
    return str;
}
void main()
{
    int a,i;
    char *s;//这里分不分配存储空间都行，但一定要是指针，和局部函数返回的类型一致

    printf("请输入一个整数:");
    scanf("%d",&a);


    printf("化为二进制为：");
    s=Intostr(a);
    for(i=0;i<8;i++)
    {
        printf("%c",s[i]);
    }

    system("pause");
}
```

写法2：结果作为局部函数指针类型参数

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

    void Intostr(int data,char *str)
    {
        char t;
        int i=0;//对于参数str，不能再为其分配存储空间
        memset(str,'0',8);
        while(data)
        {
            str[i]=(char)(data%2+'0');
            data=data/2;
            i++;
        }
        while(i<=7)
        {
            str[i]='0';
            i++;
        }
```

```
        for(i=0;i<4;i++)
        {
            t=str[i];
            str[i]=str[7-i];
            str[7-i]=t;
        }

    }
    void main()
    {
        int a,i;
        char s[8];

        printf("请输入一个整数:");
        scanf("%d",&a);


        printf("化为二进制为：");
        Intostr(a,s);
        for(i=0;i<8;i++)
        {
            printf("%c",s[i]);
        }

        system("pause");
    }
```

2.将整数化为字符串

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#define N 10  //整数化为字符串，最大int型才65536,故字符串长度为10足矣

void Intostr(int data,char *str)
{
    char t;
    int i=0,n;//对于参数str，不能再为其分配存储空间
    memset(str,'0',N);
    while(data)
    {
        str[i]=(char)(data%10+'0');
        data=data/10;
        i++;
    }

    n=i;//此时i就为字符串长度，记为n
    str[n]='\0';//截断字符串
    for(i=0;i<n/2;i++)//将该字符串数组反过来，即得到正确的值
    {
        t=str[i];
        str[i]=str[n-1-i];
        str[n-1-i]=t;
    }

}
void main()
{
    int a,i;
    char s[N];

    printf("请输入一个整数:");
    scanf("%d",&a);


    printf("化为二进制为：");
```

```
    Intostr(a,s);

    puts(s);
    system("pause");
}
```

3.将字符串化为整数

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#define N 10

void Strtoin(char *str,long *data)
{
    int i=0,l;
    long d=0;

    l=strlen(str);
    while(i<l)
    {
        d=d*10;
        d=d+(int)(str[i]-'0');
        i++;
    }
    *data=d;
}

void main()
{
    int i;
    char s[N];
    long data=0;

    printf("请输入一个字符串:");
    gets(s);


    printf("化为整数为: ");
    Strtoin(s,&data);

    printf("%ld",data);
    system("pause");
}
```

4.编程实现字符串的strcpy方法

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#define N 10

char *Mystrcpy(char *strDest,const char *strSrc)
{
    int i=0;
    while(strSrc[i])
    {
        strDest[i]=strSrc[i];
        i++;
    }
    strDest[i]='\0';
    return strDest;
}
void main()
{
```

```
    char strSrc[N],strDest[N],*rel;
    printf("请输入将被复制的源字符串:\n");
    gets(strSrc);
    printf("请输入赋值字符串的目的字符串:\n");
    gets(strDest);
    rel=Mystrcpy(strDest,strSrc);

    puts(rel);
    puts(strDest);
    system("pause");

}
```

5.编程实现字符串循环右移n位

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#define N 10

void LoopMove(char *str,int n)
{
    int l=strlen(str),i=n,j;
    char t;
    while(i)
    {
        j=l-1;
        t=str[j];
        while(j>=1)
        {
            str[j]=str[j-1];
            j--;
        }//最后j为0
        str[j]=t;
        i--;
    }
}
void main()
{
    char str[N];
    int n;
    printf("请输入将被移位的字符串:\n");
    gets(str);
    printf("请输入循环右移的步数:\n");
    scanf("%d",&n);

    LoopMove(str,n);
    printf("循环右移后的字符串为:\n");
    puts(str);

    system("pause");

}
```

6.给定一个字符串，求其最长的子字符串

思路：

若字符串为abcdefbcd，长度为l,则其最长子字符串为bcd。

使用两层循环：

- 外循环用i遍历，i>=0,i<l,依次取出字符串abcdefbcd,bcdefbcd,cdefbcd……d,即取(l-1)次，每次取出原字符串的第i最后的字符。
- 内循环用j遍历，j>=i+1,j<l。若外层是abcdefbcd，则内层就依次取出bcdefbcd,cdefbcd...;即每次取出原字符串的第（i+j)到最后的字符。那样每次外循环就可以和后面的字符串比较l-i-1次。每次比较前驱字符，遇到相等的字符就存入r字符串，直到不相等，这时候记下相同的子串位数；下一

个比较如果位数比较大，就取而代之。

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#define max 100

void Findmaxsub(char *str,char *substr)
{
    int l=strlen(str);
    char *s=(char *)malloc(l);
    char *t=(char *)malloc(l);


    int i,j,h,n=0;

    for(i=0;i<=l-1;i++)
    {
        memcpy(s,str+i,l-i);//一次比较中的前一个子串
        for(j=i+1;j<=l-1;j++)
        {
            memcpy(t,str+j,l-j);//一次比较中的后一个子串
            h=0;
            while(s[h]==t[h]&&h<l-j)
            {
                h++;
            }
            if(h>n)
            {
                n=h;
                memcpy(substr,s,n);
            }
        }

    }

    substr[n]='\0';
}

void main()
{
    char str[max],substr[max/2];
    printf("请输入字符串:\n");
    gets(str);
    Findmaxsub(str,substr);
    printf("最长的重复子串为:\n");
    puts(substr);

    system("pause");
}
```

7.请写一个函数模拟C++的strstr()函数，把主串中子串及子串以后的字符全部返回。如strstr("12345678","234")，即返回"2345678“

思路：

依次从主串的第1，2，3....个字符开始作为比较字符串和子串比较。依次将比较字符串和子串的对应序号的字符比较，如果可以全部相同则找到该子串，返回该子串和其后面的字符。

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#define max 100

char *Mystrstr(char *str,char *substr)
```

```
{
    int l=strlen(str);
    int lsub=strlen(substr);
    char *rel=(char *)malloc(l);
    int i,j,k;
    for(i=0;i<l;i++)
    {
        k=i;
        j=0;
        while(str[k]==substr[j]&&j<lsub)
        {
            k++;
            j++;
        }
        if(j==lsub)
        {
            memcpy(rel,&str[i],l-i);
            rel[l-i]='\0';
            return rel;
        }
    }
}

void main()
{
    char str[max],substr[max];
    int n;
    char *mystrstr;
    puts("请输入一个字符串:\n");
    gets(str);
    puts("请输入子串:\n");
    gets(substr);

    mystrstr=Mystrstr(str,substr);
    puts(mystrstr);

    system("pause");
}
```

8.求一个字符串中连续出现次数最多的子串。

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#define max 100

char *Mostsub(char *str)
{
    int l=strlen(str);
    int i,j,k,h,n,num=1;
    char *s=(char *)malloc(l);
    char *r=(char *)malloc(l);
    char *p=str;
    for(i=0;i<l;i++)// i记录子串从主原串哪个字符开始取，如原串abcbcbcabc，则子串分别从a、b、c、b开始取
    {
        for(j=1;j<=(l-i)/2;j++)//j记录从某个字符开始取的位数。因为重复次数要想最多，那至少为2，所以子串长度最多取剩余长度的一半
        {
            memcpy(s,p+i,j);
            k=0;
            h=i+j;
            n=1;//记录连续子串个数，初始为1
            while(h<l&&s[k]==str[h])
            {
                k++;
                h++;
                if(k==j)//下一个子串和该子串相同，再往下比较，连续子串个数加1
```

```
                {
                    k=0;
                    n++;
                }
            }
            if(n>num)
            {
                num=n;
                s[j]='\0';
                memcpy(r,s,j+1);//或strcpy(r,s)
            }

        }
    }
    return r;
}

void main()
{
    char str[max];
    char *mostsub;
    puts("请输入一个字符串:\n");
    gets(str);


    mostsub=Mostsub(str);
    puts(mostsub);

    system("pause");

}
```