实验九 任务要求

1. 建立一个一维数组a，并完成以下任务，并输入输出格式参见下图。

(1) 利用rand()产生10个10~100之间的整数，存入a中并输出；

(2) 使用冒泡法，将数组a按降序排列（从大到小），并输出；

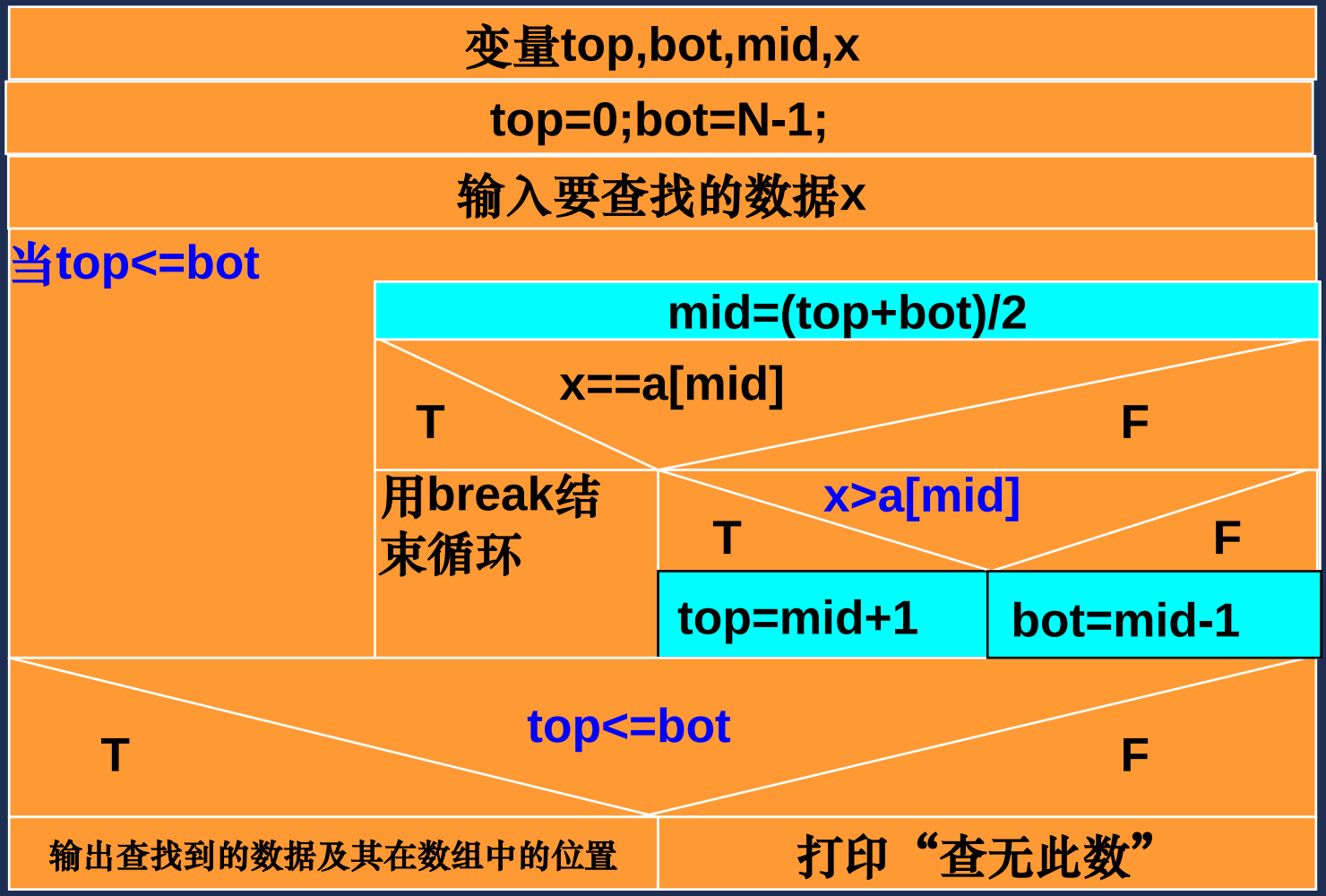
(3) 使用折半查找完成指定数据（从键盘输入）的查找，并输出是否查找到的信息。

```
const int N=10;
int a[N];
int i;
srand(time(NULL)); //初始化，置于循环外
for(i=0;i<N;i++)
    a[i]=10+rand()%(100-10+1);
```

折半查找算法的N-S流程图

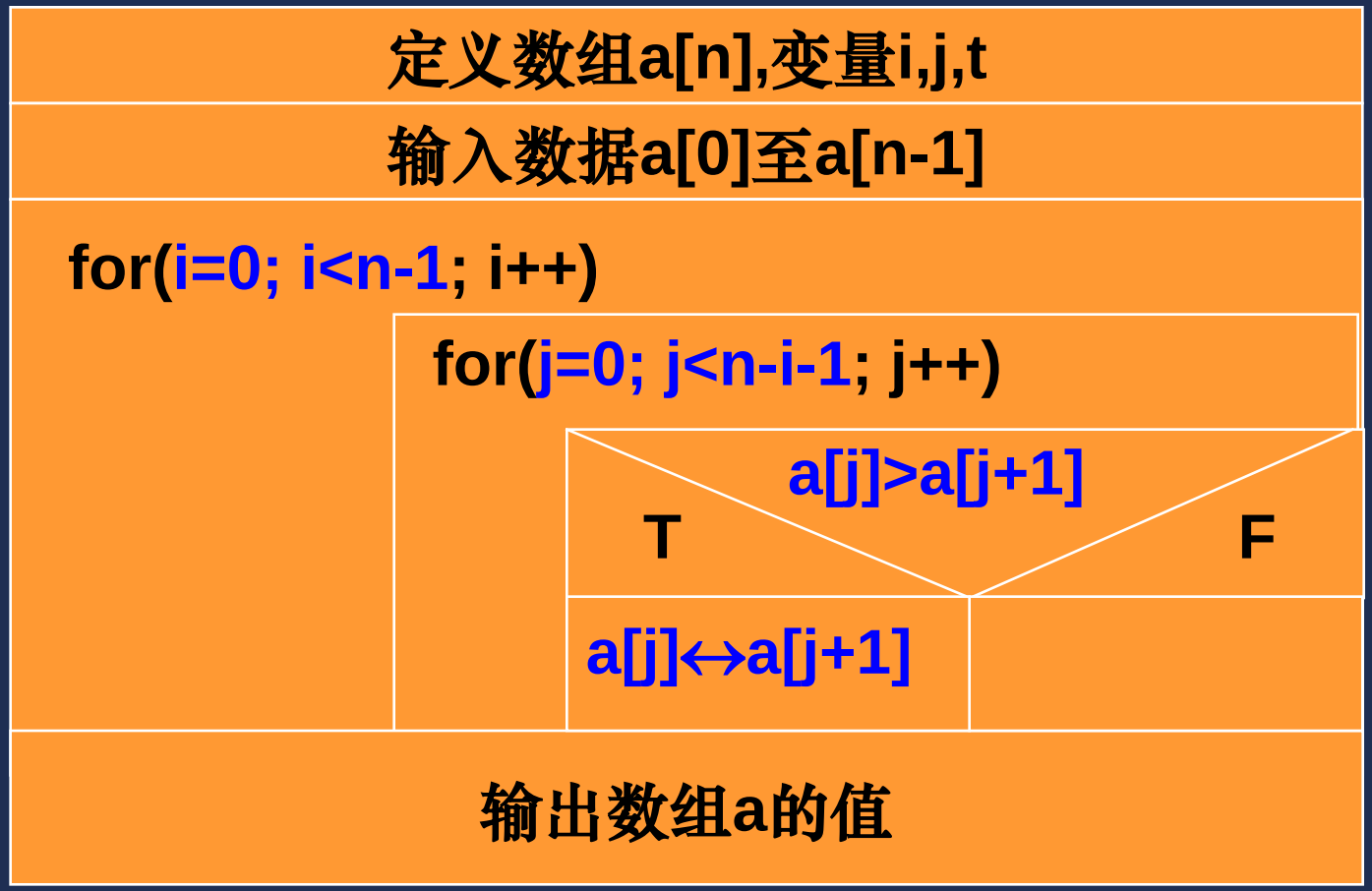
下图所示算法适用于：从小到大排序的数列！！！

如果数据从大到小排序，如何修改算法？？？

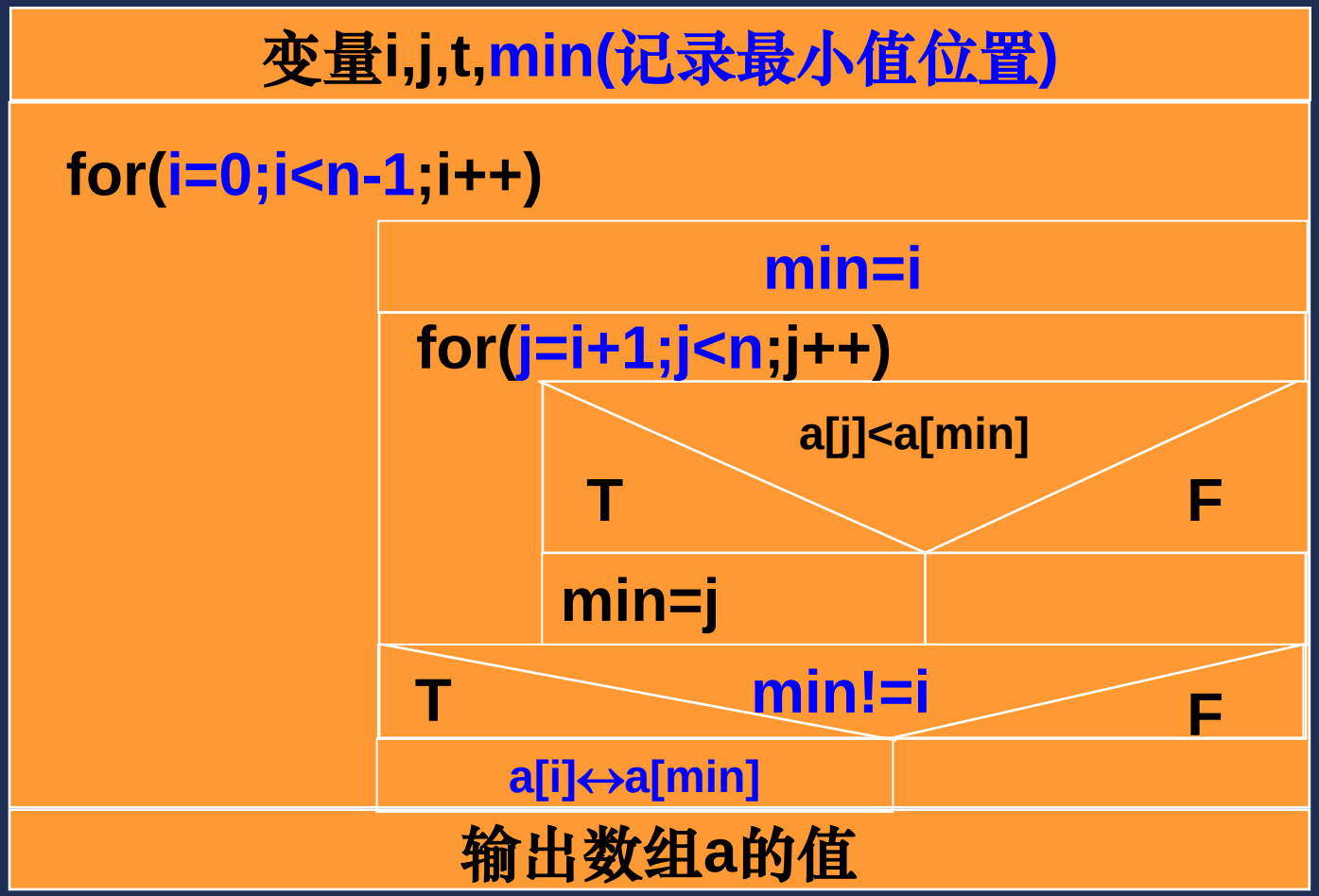


冒泡法N-S流程图

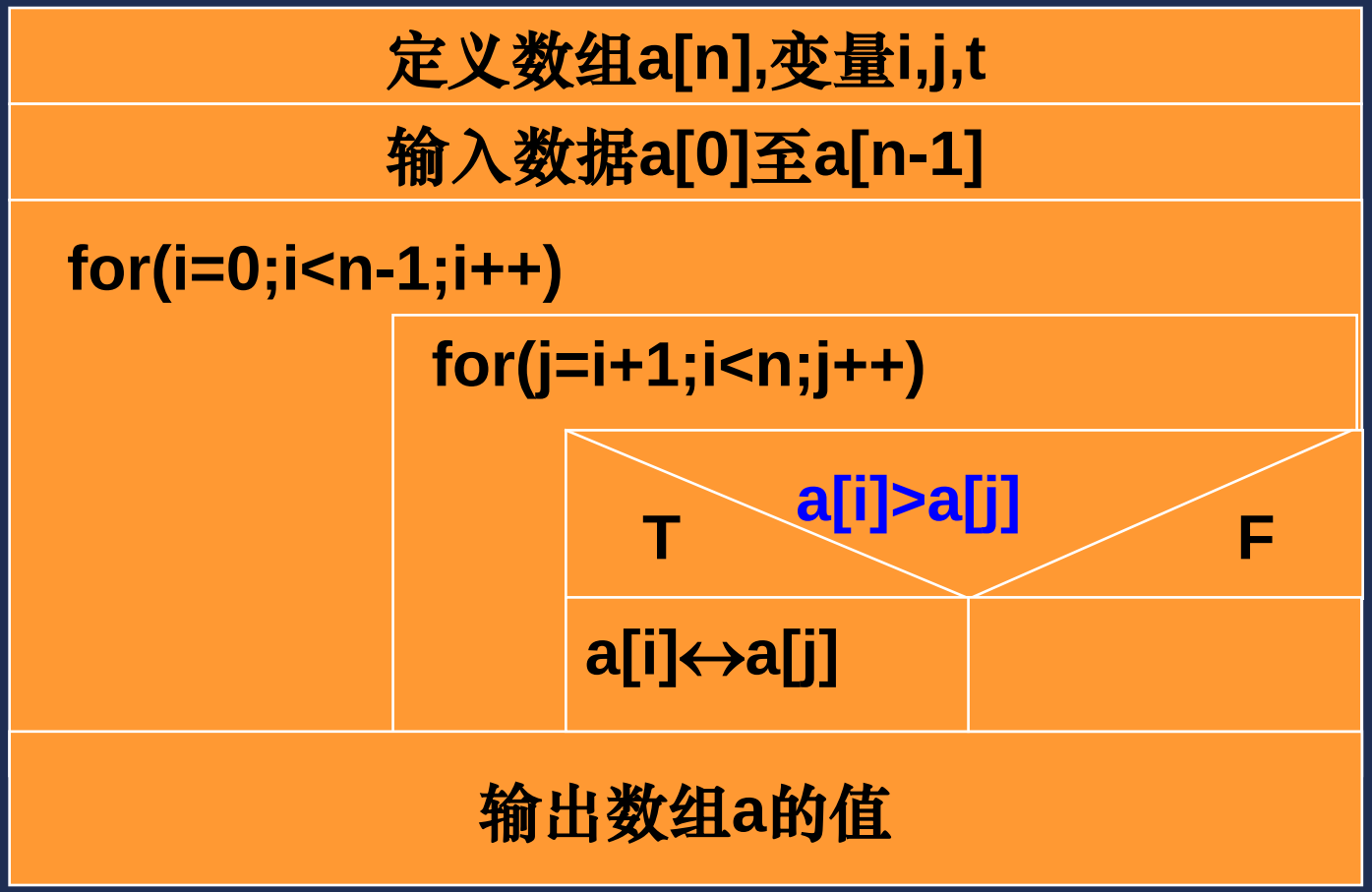
降序排列（从大到小）如何修改下列算法



选择法算法的N-S流程图



改进的选择法算法


定义数组a[n],变量i,j,t
输入数据a[0]至a[n-1]
for(i=0;i<n-1;i++)
for(j=i+1;i<n;j++)
a[i]>a[j]
T
F
a[i]↔a[j]
输出数组a的值

实验九 任务要求

2. 建立一个一维数组a，并完成以下任务，并输入输出格式参见下图。

（1） 以自动生成的斐波那契数列前10项（前两项为1、1）为数组元素赋初值，并输出；

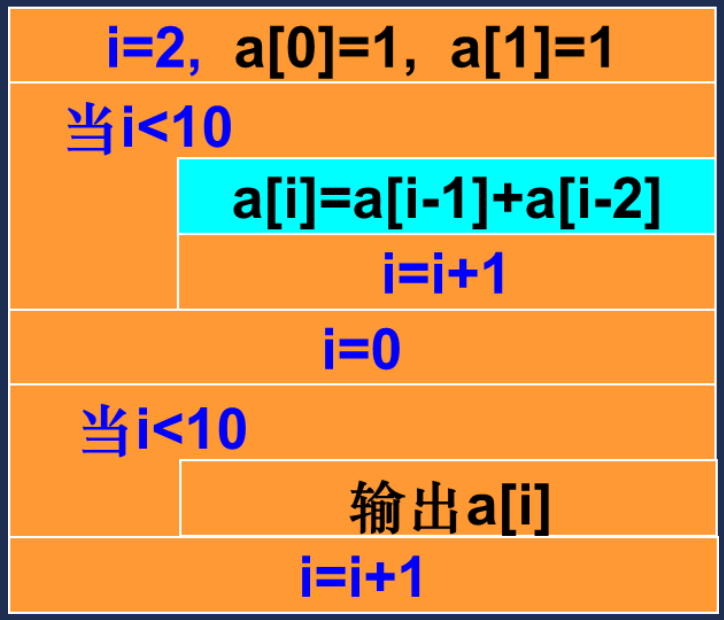
（2）将键盘输入的一个数据插入，并数组元素值仍保持有序；

（3） 把数组中相同的数据删除到只剩下一个，并输出；

（4） 求出数组所有元素的平均值，并输出；

（5） 删除数组中所有低于平均值的那些元素。

```
自动生成的10个整数:
 1  1  2  3  5  8  13  21  34  55
输入待插入的数据: 21
插入后数组序列为:
1  1  2  3  5  8  13  21  21  34  55
去重后数组序列为:
1  2  3  5  8  13  21  34  55
数组所有元素平均值为:15.78
数组中高于平均值的元素:
21  34  55
请按任意键继续. . .
```

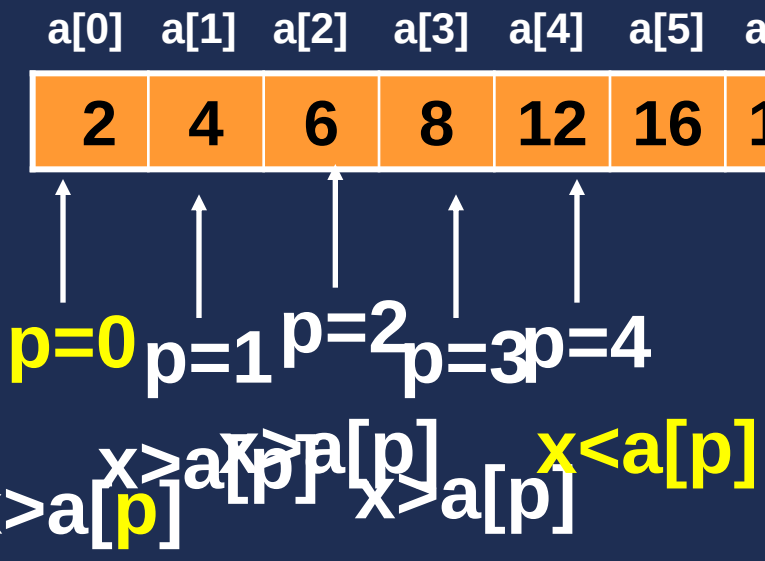


插入

把一个整数插入到一个由小到大的有序数列中，并仍然保持由小到大的顺序。

const int N=11;　int a[N],n ;　//n为数组实际长度，n<=N-1，若n=10

x=11

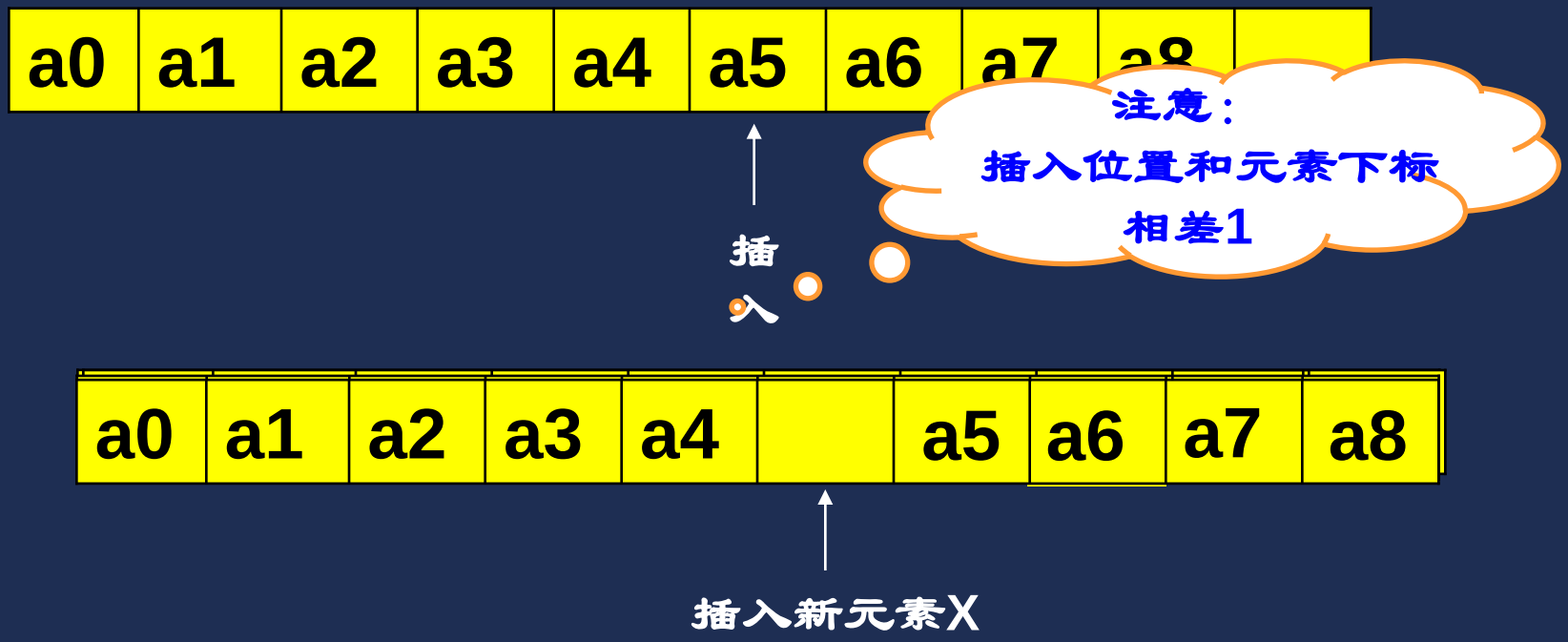



```
while(x>a[p]&&p<n)
    p++;
```

```
for(i=n-1;i>=p;i--)
    a[i+1]=a[i];
```


```
a[p]=x;
```

在指定位置插入新元素



数组长度为n，插入位置为S，插入的元素值为X（X，S均从键盘输入值），则插入操作：

```
for (j=n-1;j>=S-1;j--)  a[j+1]=a[j];    //元素后移，空出插入位置
a[S-1]=X;       //插入新元素，注意插入位置和元素下标的关系
```

补充案例

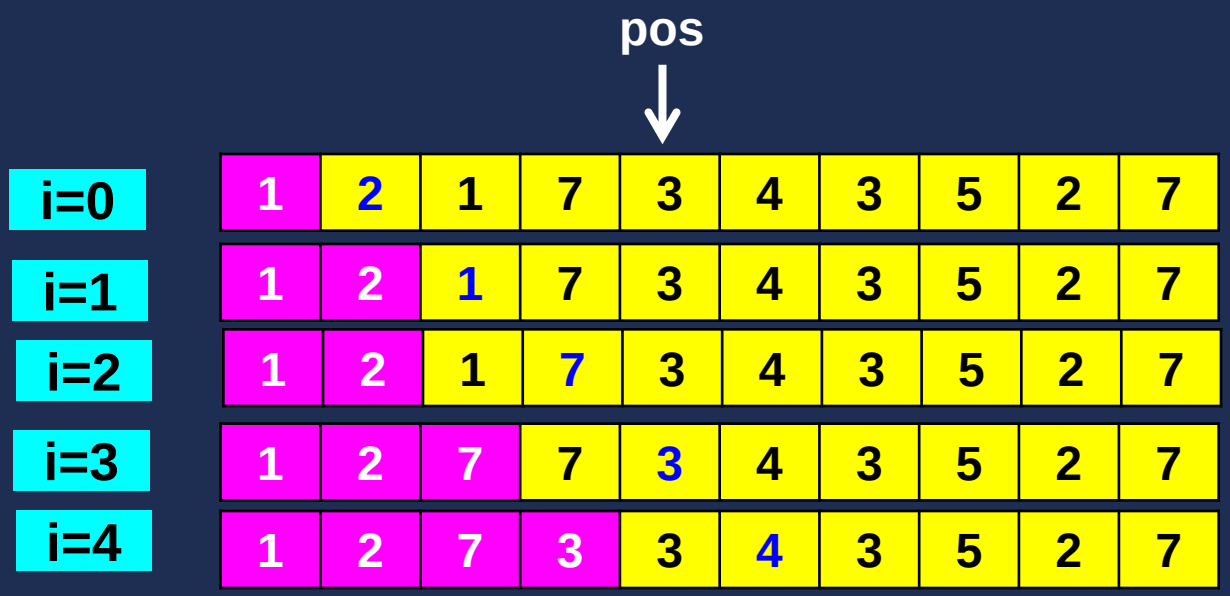
编写程序：只用一个数组把所有相同的数删到只剩下一个

1	2	1	7	3	4	3	5	2	7

1	2	7	3	4	5

处理流程：把一个数组分成两部分，前面一部分为已经处理好的数据；后面为待处理的数据。

把第1个数组元素默认视为已经处理好的结果，则从第2个到最后一个（下标1~n-1）为剩下的需处理的元素。同时用变量pos（初值为1）记录处理好的元素的存放位置。



补充案例

编写程序：把一个数列中所有系统的数删到只剩下一个

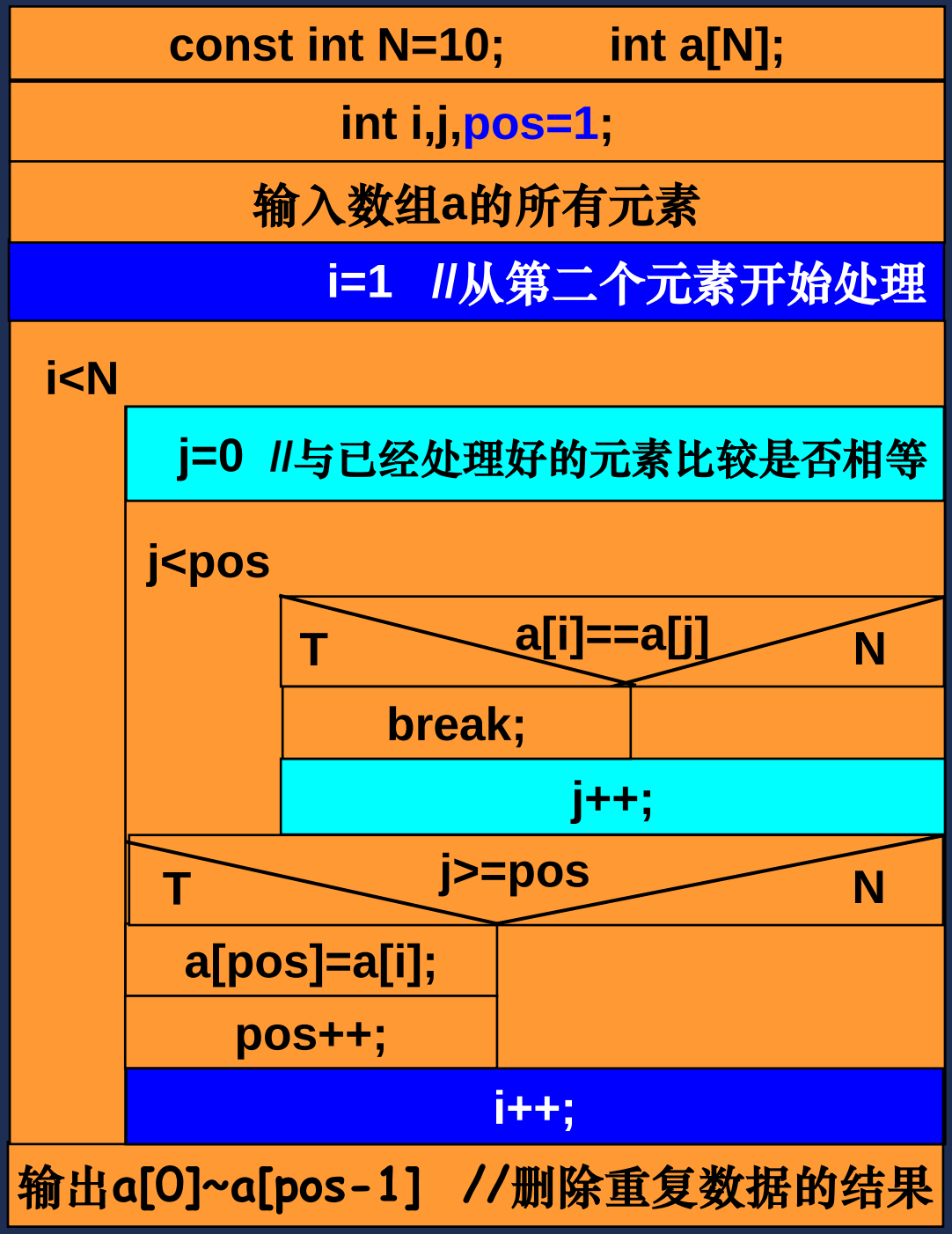
1	2	1	7	3	4	3	5	2	7

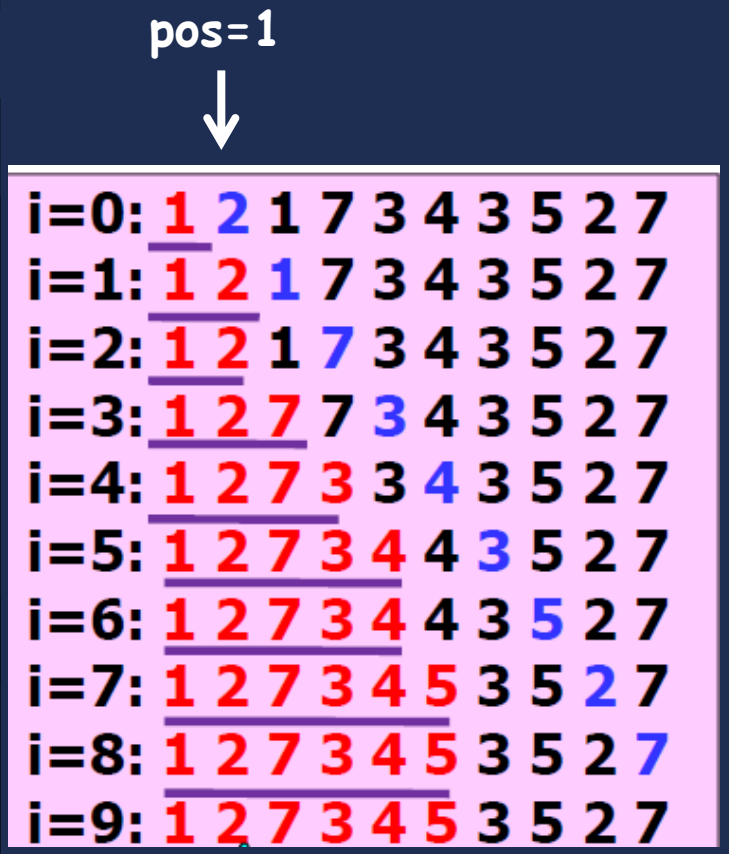
1	2	7	3	4	5

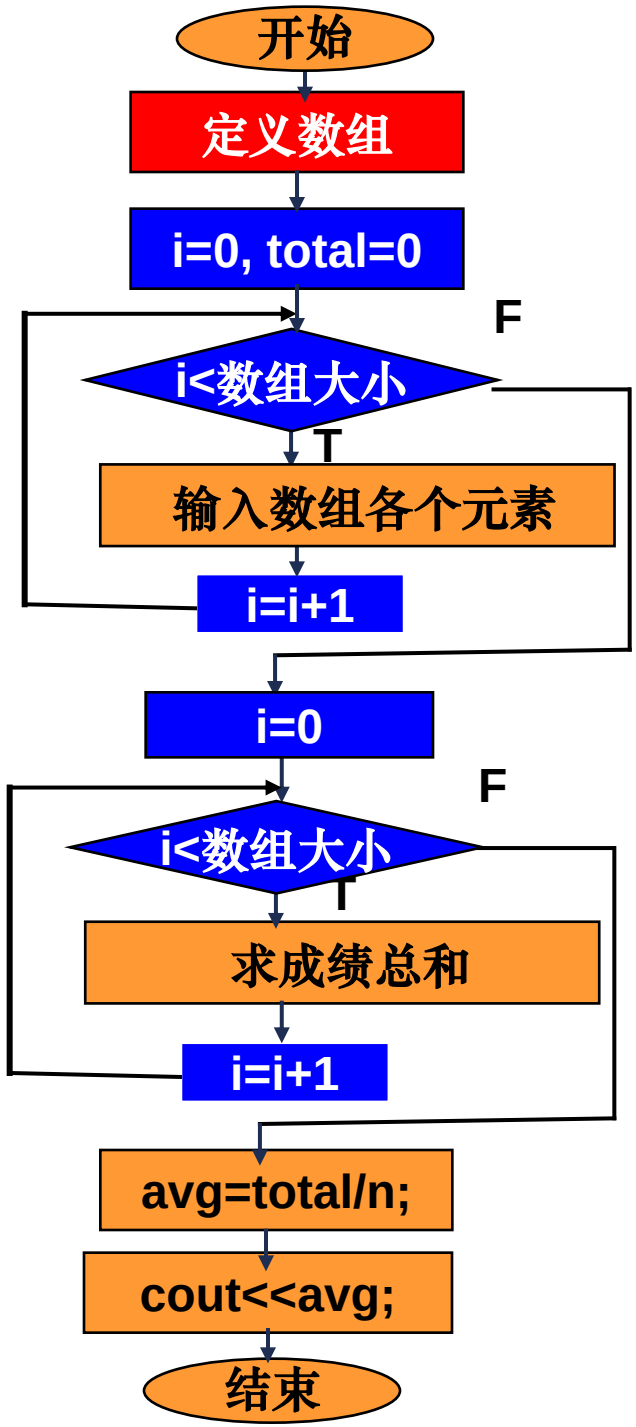
算法思想：

第1个元素默认为已经处理好的结果，从第2个元素开始，依次和前面已经处理好的元素作比较，判断是否相同，若不相同，则将其存入pos指定的位置，同时将pos指向下一个存放位置(pos+1)，之后继续处理数组中下一个未处理的元素。若相同，则直接跳过该元素，继续处理下一个未处理的元素，直到所有元素都处理完。

```
i=0: 1 2 1 7 3 4 3 5 2 7
i=1: 1 2 1 7 3 4 3 5 2 7
i=2: 1 2 1 7 3 4 3 5 2 7
i=3: 1 2 7 7 3 4 3 5 2 7
i=4: 1 2 7 3 3 4 3 5 2 7
i=5: 1 2 7 3 4 4 3 5 2 7
i=6: 1 2 7 3 4 4 3 5 2 7
i=7: 1 2 7 3 4 5 3 5 2 7
i=8: 1 2 7 3 4 5 3 5 2 7
i=9: 1 2 7 3 4 5 3 5 2 7
```


const int N=10; int a[N];
int i,j,pos=1;
输入数组a的所有元素
i=1 //从第二个元素开始处理
i<N
j=0 //与已经处理好的元素比较是否相等
j<pos
T
a[i]==a[j]
N
break;
j++;
T
j>=pos
N
a[pos]=a[i];
pos++;
i++;
输出a[0]~a[pos-1] //删除重复数据的结果


pos=1
i=0: 1 2 1 7 3 4 3 5 2 7
i=1: 1 2 1 7 3 4 3 5 2 7
i=2: 1 2 1 7 3 4 3 5 2 7
i=3: 1 2 7 7 3 4 3 5 2 7
i=4: 1 2 7 3 3 4 3 5 2 7
i=5: 1 2 7 3 4 4 3 5 2 7
i=6: 1 2 7 3 4 4 3 5 2 7
i=7: 1 2 7 3 4 5 3 5 2 7
i=8: 1 2 7 3 4 5 3 5 2 7
i=9: 1 2 7 3 4 5 3 5 2 7

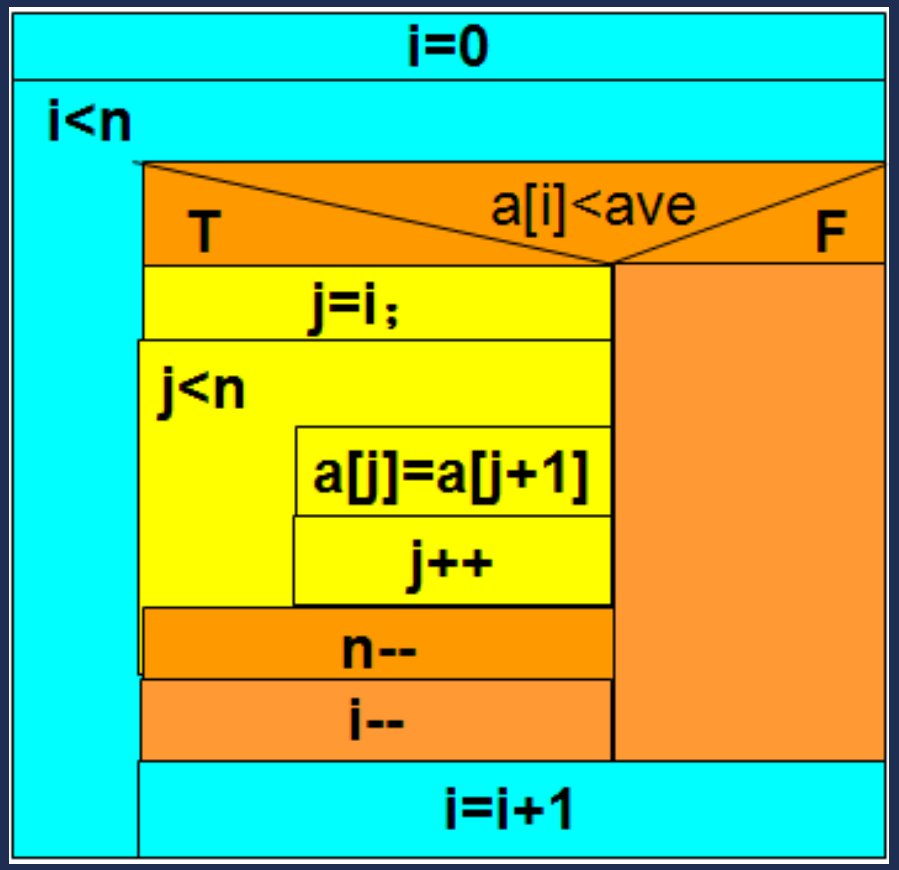


利用一维数组计算平均值

```
#include <iostream>
using namespace std;
void main( )
{  const int  N=10;
    int  scores[N];     // 定义数组
     int n,i,total=0;            // n为数组实际长度
    float avg=0.0;
   cout<<“请输入数组长度：”;
   cin>>n;                       // 输入数组实际长度
   for(i=0;i<n;i++)
    {
        cin>>scores[i];  //利用循环逐个输入数组元素
      total=total+scores[i];  //对数组元素求和
     }
   avg=float(total)/n;     //计算平均成绩
   cout<<“Average:”<<avg<<endl;
}
```

一维数组基本操作：删除数组中所有小于平均值的元素

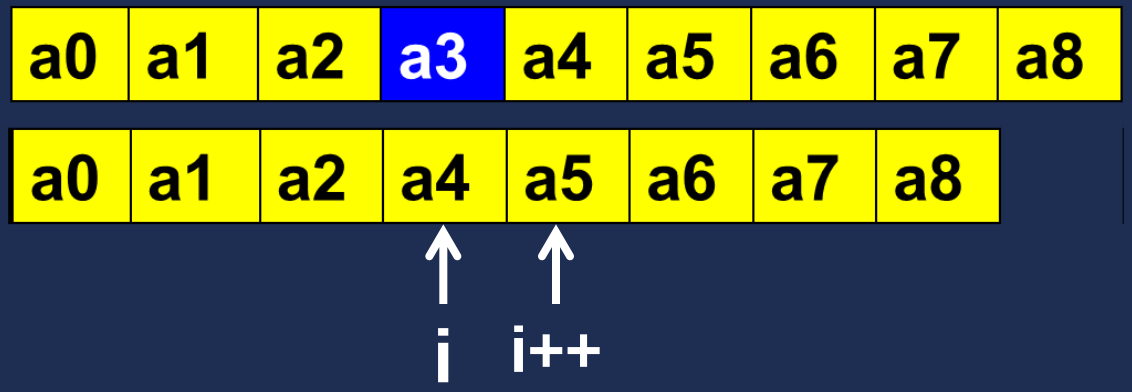
删除元素a[i],即将a[i+1]移至a[i];a[i+2]移至a[i+1];以此类推，直到将a[n-1]移至a[n-2]为止。

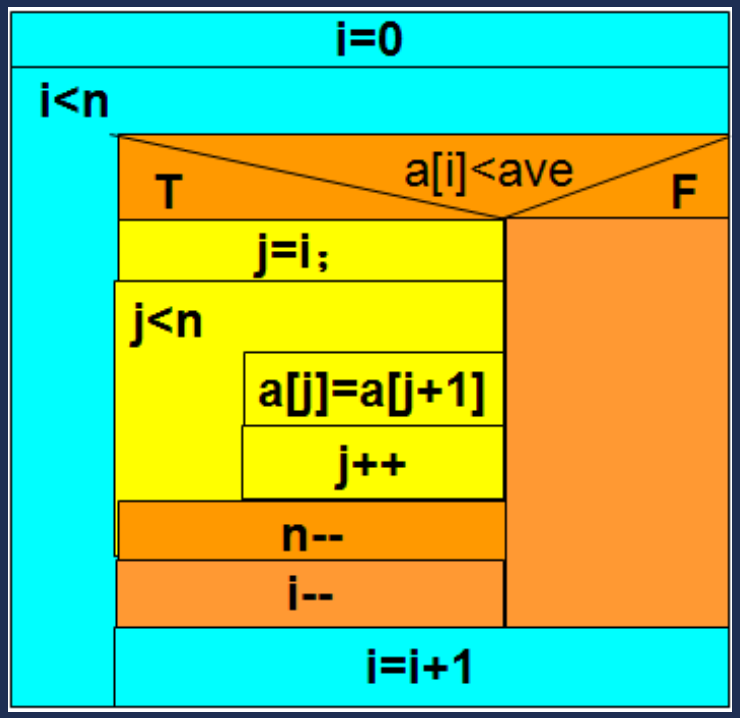


注意：

- 算法描述中，n--，表示删除小于平均值的元素后，数组的实际长度。
- 为什么还需要i--？

删除数组中所有小于平均值的元素





- 一次循环处理完成后会删除比平均值小的元素a[i](a[3])，然后进行下一次处理(i++)，则下一次处理会从a[5]开始。
- 实际上，原来的a[4]已经移动至a[i](a[3])的位置，为了不漏掉元素，需要“回溯”下标(i--)，则在进行i++操作后，刚好是正确的处理位置。