



中國人民大學
RENMIN UNIVERSITY OF CHINA

第5讲 数组

余力

buaayuli@ruc.edu.cn

输入格式

输入样例↵

3-4↵

3-8-9-10↵

2-5--3-5↵

7-0--1-4↵

```
int main()↵
{↵
    int n, m, sum = 0, x;↵
    scanf("%d%d", &n, &m);↵
    for (int i = 0; i < n; i++)↵
        for (int j = 0; j < m; j++)↵
            {↵
                scanf("%d", &x);↵
                ....↵
            }↵
    return 0;↵
}↵
```

输入样式↵

3-5↵

2017101000-10-2.1-1.2-4.3-2.4-2.5-5.1-5.2-1.6-4.2-4.4↵

2017101001-9-2.1-2.2-2.5-3.2-1.1-1.3-4.3-4.4-5.2↵

2017101002-10-1.4-1.5-2.1-2.2-3.4-3.4-4.1-4.6-5.4-5.5↵

```
---- scanf("%d%d", &n, &k);↵
---- for (s = 0; i = 0; i < n; i++)↵
---- {---- scanf("%s", &id[i]);↵
----- scanf("%d", &num[i]);↵
----- for (j = 0; j < num[i]; j++)↵
----- {---- scanf("%d.%d", &date[s], &class1[s]);↵
----- s += 1;----}↵
---- }↵
```

#293. 外侧元素求和

```
#include <stdio.h>
int main() {
    int n,m,x,s,i,a=0;
    scanf("%d %d",&n,&m);
    for(i=1;i<=n*m;i++){
        if(i<=m){
            scanf("%d",&x);
            s=s+x;}
        else if(i>=m*n-m){
            scanf("%d",&x);
            s=s+x;}
        else if(i%m==1||i%m==0){
            scanf("%d",&x);
            s=s+x;}
        else scanf("%d",&x);
    }
    printf("%d",s);
    return 0;
}
```

原因：**s没有初始化**，但在本地机上，没有初始化就默认为0，友学网上不会

在本地运行可以，但在友学网上不通过

#462 统计字符

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

```
int main() {
```

```
    char c;
```

```
    int i;
```

```
    int cnt[26] = {0};
```

```
    while ((c = getchar()) != '\n') {
```

```
        if (c >= 'A' && c <= 'Z') {
```

```
            cnt[c - 'A']++;
```

```
        }
```

```
    }
```

```
    for (i = 0; i < 26; i++) {
```

```
        char s;
```

```
        s = 'A' + i;
```

```
        printf("%c:%d\n", s, cnt[i]);
```

```
    }
```

```
    return 0;
```

```
}
```

#379149

#462 统计字符

Time Limit Exceeded

0

3068 ms

384 KB

cpp / 288 B

余力(yuli)

2021/10/22 下午7:05

测试点 #0

2

得分: 0

用时: 1007 ms

内存: 384 KiB

测试点 #1

2

得分: 0

用时: 1009 ms

内存: 384 KiB

测试点 #2

2

得分: 0

用时: 1052 ms

内存: 264 KiB

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

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

```
int main() {
```

```
    char s[10000];
```

```
    int a[26] = {0};
```

```
    gets(s);
```

```
    for (int i = 0; i < strlen(s); i++)
```

```
        if (s[i] >= 65 && s[i] <= 90)
```

```
            a[s[i] - 'A']++;
```

```
    for (int i = 0; i < 26; i++)
```

```
        printf("%c:%d\n", 'A' + i, a[i]);
```

```
    return 0;
```

```
}
```

内容提要

- 5.1 数组的概念、定义和初始化
- 5.2 二维数组
- 5.3 数组的排序问题
- 5.4 筛法求素数



中國人民大學
RENMIN UNIVERSITY OF CHINA



01. 数组概念定义初始化

数组

- 一组类型相同的顺序存储的数据（变量）
- 数组名、下标、元素
- 方便对一组数据进行命名和访问
 - 数组名+下标 唯一确定数组中的一个元素
 - 通过数组名+下标可以访问数组中的任意元素
- 应用：
 - 对一组数求最值、平均值
 - 对一组数据排序

一维数组的定义

■ 定义形式

- 类型说明符 数组名[常量]
- 例： `float sheep[10]; int a2001[1000];`

■ 数组的命名规则

- 数组名的第一个字符应为英文字母;
- 用方括号将常量表达式括起;
- 常量表达式定义了数组元素的个数;
- 数组的下标从0开始，如果定义了5个元素，是从第0个元素到第4个元素
- 常量表达式中不允许含有变量

一维数组的数组组织方式

`int boo[4]` (note: 2 bytes per int)

1980	46	4816	3
<code>boo[0]</code>	<code>boo[1]</code>	<code>boo[2]</code>	<code>boo[3]</code>

`char foo[4]` (note: 1-byte char)

h	e	l	p
<code>foo[0]</code>	<code>foo[1]</code>	<code>foo[2]</code>	<code>foo[3]</code>

数组初始化

■ 直接声明时初始化

➤ 例如: `int a[5] = { 3, 5, 4, 1, 2 };`

➤ 效果

a	3	5	4	1	2
下标	0	1	2	3	4

■ 思考：在声明之后，这样初始化可以吗？

`a[0] = 3; a[1] = 5; a[2] = 4; a[3] = 1; a[4] = 2;`

数组元素的访问

- 访问一维数组中元素的形式：

数组名[下标]

- 例如：

➤ $a[0] = a[1] + a[2];$

- 其中：

- 下标写在一个方括号中；
- 下标是整型表达式，如果为浮点型数据，C截去小数部分，自动取整。
- 引用时下标不能超界，否则编译程序检查不出错误，但执行时出现不可知结果。

一维数组的访问

// List A of n integer elements has already been set

int i;

for (i=0;i<n;i++)

printf("%d ", A[i]);

printf("\n");

思考：如何输出每月天数

```
/* prints the days for each month */  
#include <stdio.h>  
#define MONTHS 12  
int main() {  
    int days[MONTHS] = {31,28,31,30,31,30,31,31,30,31,30,31};  
    int index;  
    for (index = 0; index < MONTHS; index++)  
        printf("Month %d has %2d days.\n", index + 1, days[index]);  
    return 0;  
}
```

Month_days.cpp

思考：如果数组没初始化呢？

```
/* no_data.c -- uninitialized array */
#include <stdio.h>
#define SIZE 4
int main(void) {
    int no_data[SIZE];
    /* uninitialized array */
    int i;
    printf("%2s%14s\n", "i", "no_data[i]");
    for (i = 0; i < SIZE; i++)
        printf("%2d%14d\n", i, no_data[i]);
    return 0;
}
```

No_data.cpp

字符数组与字符串

character array but not a string

y	o	u		c	a	n		s	e	e		i	t	.
---	---	---	--	---	---	---	--	---	---	---	--	---	---	---

character array and a string

y	o	u		c	a	n		s	e	e		i	t	.	\0
---	---	---	--	---	---	---	--	---	---	---	--	---	---	---	----

▲
null character

程序举例：哪只羊最重？

- 中秋佳节，有贵客来到草原，主人要从羊群中选一只肥羊宴请宾客，当然要选最重者。
 - 要记录每只羊的重量，如果有成千上万只羊，不可能用一般变量来记录。可以用带有下标的变量，即数组
 - 将羊的重量读入存放到数组中
 - 声明一个变量保存最大的重量，不断更新之

程序框图

bigsheep = 0.0f; 将记录最重的羊的重量置 0
bigsheepNo = 0; 记录最重的羊的编号

for (i=0; i<10; i=i+1)

提示输入第 i 只羊的重量;
键入第 i 只羊的重量 sheep[i];

bigsheep < sheep[i]

是

否

bigsheep = sheep[i];
bigsheepNo = i;
存重者, 记录第 i 只。

输出 bigsheep (最重的羊的重量)

输出 bigsheepNo (最重的羊的编号)

```

#include <stdio.h> // 预编译命令
int main()          // 主函数
{
    float sheep[10] = {0}; // 用于存10只羊每一只的重量
    float bigsheep=0.0;    // 浮点类型变量, 存放最肥羊的重量
    int i=0, bigsheepNo=0; // 整型变量, i 用于计数循环,
                           // bigsheepNo用于记录最肥羊的号

    for ( i=0; i<10; i=i+1 )
    {
        printf("请输入羊的重量sheep[%d]=", i); // 提示用
        scanf("%f", &sheep[i]);                // 输入第i只羊的重量
        if ( bigsheep < sheep[i] ) // 如果第i只羊比当前最肥羊大
        {
            bigsheep = sheep[i]; // 让第i只羊为当前最肥羊
            bigsheepNo = i;      // 纪录第i只羊的编号
        }
    }
    // 循环结束

    printf("最肥羊的重量为%f\n", bigsheep);
    printf("最肥羊的编号为%d\n", bigsheepNo);
    return 0;
}

```

思考

- 1#include <stdio.h>
int main()
{
 int a[4]; // 声明项
 int i=0;
 for(i=0; i<4; i++)
 printf("%d\n", a[i]);
 return 0;
}
- 2.其他不变, 改变声明项为
 int a[4] = { 0, 1, 2, 3 };

Array_initial.cpp

- 3.其他不变, 改变声明项为
int a[4] = { 3, 8 };
- 4.其他不变, 改变声明项为
int a[4] = { 2, 4, 6, 8, 10 };
- 5.其他不变, 改变声明项为
int a[4] = { 2, 4, 6, d };
- 6.其他不变, 改变声明项为
int d;
int a[4] = { 2, 4, 6, d };
- 7.其他不变, 改变声明项为
int n=4;
int a[n] = { 0, 1, 2, 3 };

练习

- 1. 给定一组整数，找到其中最小的整数，并按照输入的逆序输出这组整数
- 2. 用数组的方式来处理Fibonacci数列问题



中國人民大學
RENMIN UNIVERSITY OF CHINA



02. 二维数组

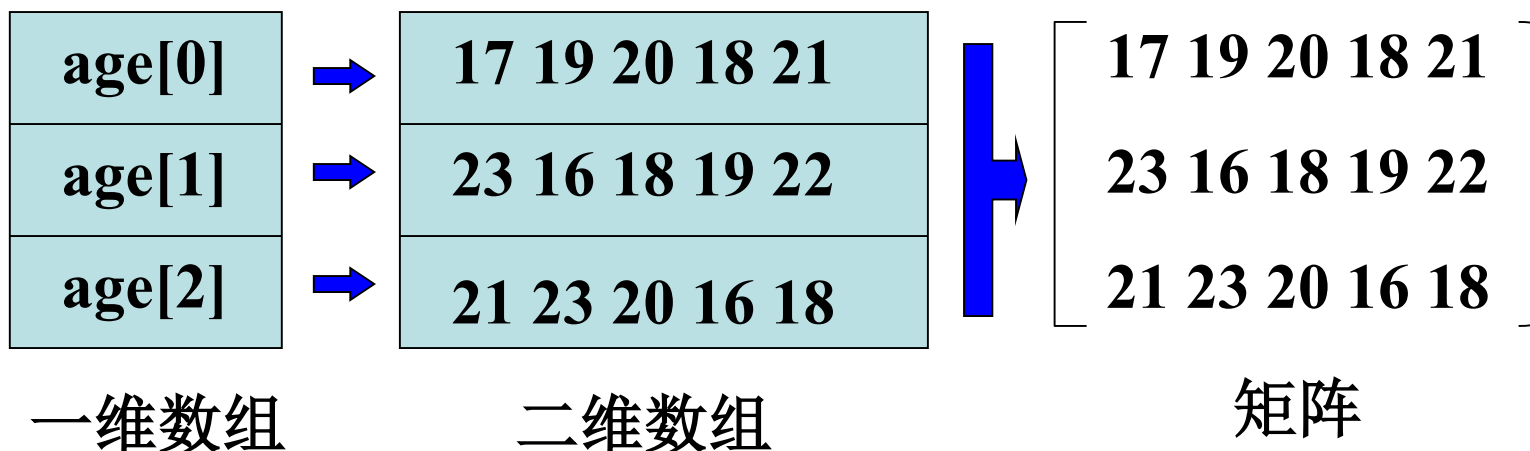
练习题

- 有 n 个学生，每个学生学 m 门课，已知所有学生的各门课的成绩，分别求每门课的平均成绩和每个学生的平均成绩。设各学生成绩如下：

课程 姓名	课程 1	课程 2	课程 3
学生 1	89	78	56
学生 2	88	99	100
学生 3	72	80	61
学生 4	60	70	75

二维数组的概念及其定义

- 当一维数组的每个元素是一个一维数组时，就构成了二维数组。
- 二维数组与数学中的矩阵概念相对应。



二维数组的定义

■ 定义方式

- 类型标识符 数组名[常量表达式1] [常量表达式2]
- 其中：
 - 类型标识符：数组中每个元素的数据类型。可以是C语言中所有的数据类型。
 - 数组名：合法的标识符，数组名就是变量名。
 - 常量表达式1：又称行下标，指出二维数组中一维数组元素的个数。
 - 常量表达式2：又称行列标，表明每个一维数组中的元素个数。

多维数组的定义

- 二维数组的每一个元素又是相同的类型的一维数，就构成了三维数组，……依此类推，就可构成四维或更高维数组
- 定义一个n维数组：

类型标识符 数组名[常量表达式1] [常量表达式2][常量表达式n]

三维数组的排列顺序

- 说明一个三维数组:

```
int a[2][3][2];
```

- 12个元素在内存中排列顺序如右图:

a[0][0][0]
a[0][0][1]
a[0][1][0]
a[0][1][1]
a[0][2][0]
a[0][2][1]
a[1][0][0]
a[1][0][1]
a[1][1][0]
a[1][1][1]
a[1][2][0]
a[1][2][1]

计算多维数组中元素位置的公式

- 一个 $m \times n$ 的二维数组 a ，其中 $a[i][j]$ 在数组中的位置为：

$$i \times n + j + 1$$

- 一个 $m \times n \times u$ 的三维数组 a ，其中 $a[i][j][k]$ 在数组中的位置为：

$$i \times n \times u + j \times n + k + 1$$

2.2 访问二维数组和多维数组

- 访问二维数组中元素的形式：

数组名[下标][下标]

- 其中：

- 每一个下标写在一个方括号中；
- 下标是整型表达式，如果为浮点型数据，C截去小数部分，自动取整。
- 引用时下标不能超界，否则编译程序检查不出错误，但执行时出现不可知结果。

多维数组的遍历

- 遍历多维数组元素的最好算法就是利用嵌套循环
- 一般的结构是：

```
for( i = 0; i <?; i++ )
```

```
    for( j = 0; j <?; j++ )
```

```
        for( k = 0; k <?; k++ )
```

```
            ⋮
```

二维数组和 multidimensional arrays 的初始化

- 二维数组和 multidimensional arrays 都可以初始化，与一维数组初始化的差别是由于维数增多，初始化时特别注意元素的排列顺序。

➤ 例:

二维数组的初始化

```
int i[2][3]={{1,2,3},{4,5,6}};
```

或写成

```
int i[2][3]={1,2,3,4,5,6};
```



中國人民大學
RENMIN UNIVERSITY OF CHINA

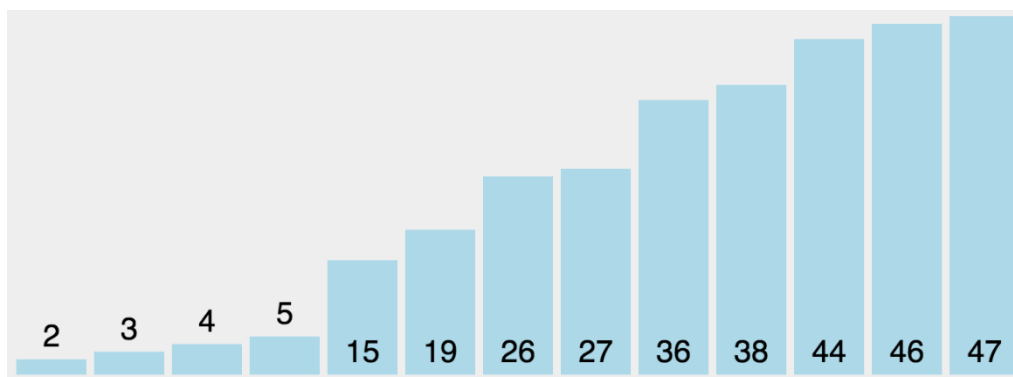
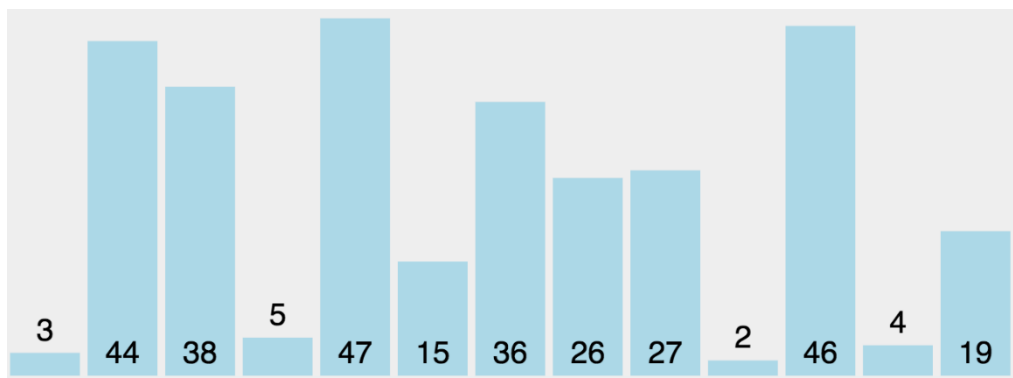


03. 数组排序问题

排序问题 (1)

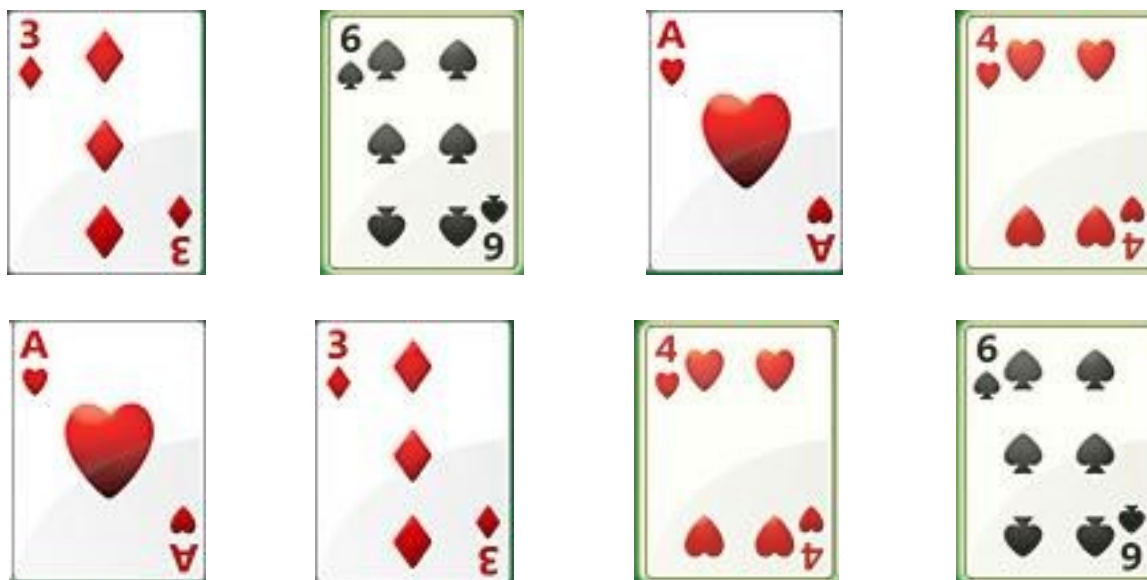
■ 问题定义

- 将数组中的元素按照一定顺序重新排列



排序问题 (2)

- 排序问题例子



- 还有哪些排序的例子？

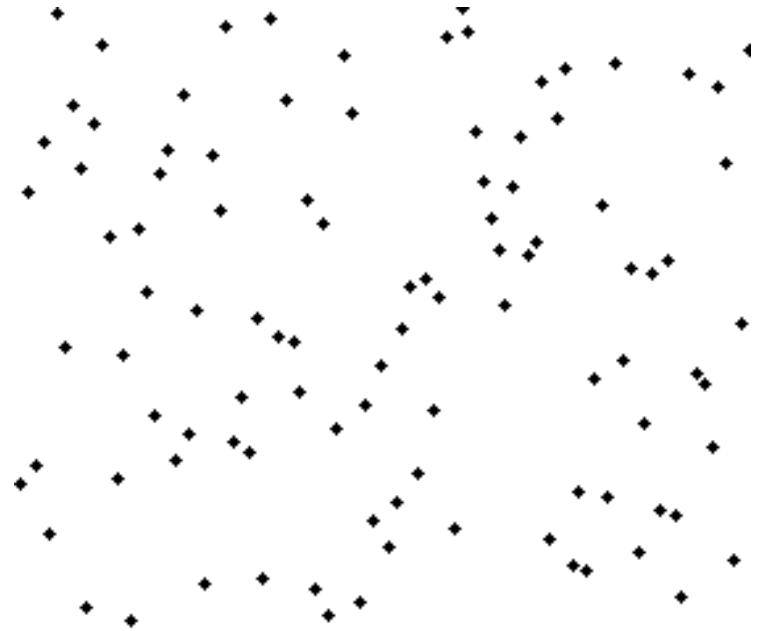
排序算法 – 冒泡排序 (1)

■ 基本思路

- 从头到尾依次访问数组
- 如果发现顺序错误就交换过来
- 直到没有再需要交换的元素

算法运行详解

<https://visualgo.net/sorting>



排序算法 – 冒泡排序 (2)

- 如何将上述思想用程序实现?
- 定义三个变量
 - n – 数组中待排序元素的个数
 - i – 当前是第几趟扫描 $i = 0, 1, 2, \dots, n - 2$
 - j -- 第 i 遍扫描待比较元素的下标 $j = 1, 2, \dots, n - i$

排序算法 – 冒泡排序 (3)

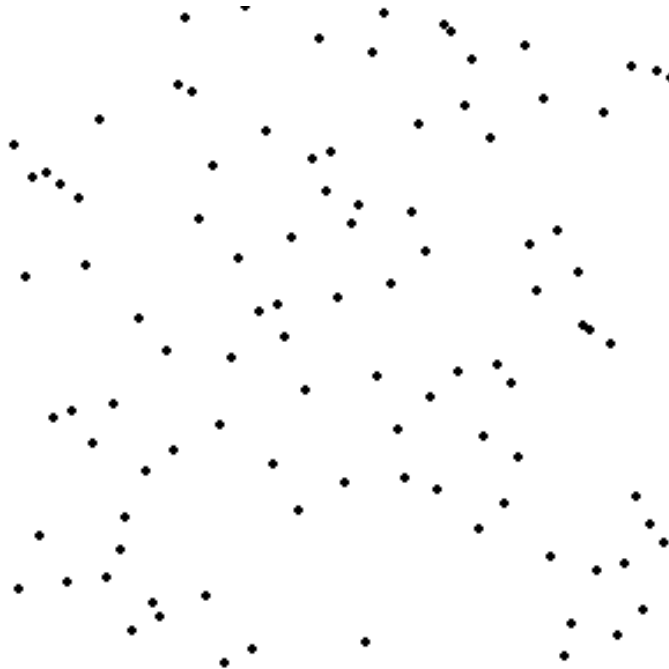
- 练习：画出冒泡排序程序的N-S图
- 核心代码

```
26     for (int i = 0; i < count - 1; i ++ ) {  
27         int swap = 0;  
28         for (int j = 0; j < count - 1 - i; j ++ ) {  
29             if (arr[j] > arr[j+1]) {  
30                 int tmp = arr[j];  
31                 arr[j] = arr[j+1];  
32                 arr[j+1] = tmp;  
33                 swap = 1;  
34             }  
35         }  
36         if (swap == 0) break;  
37     }
```

排序算法 – 选择排序 (1)

■ 基本思路

- 把数组分为已排序和未排序两部分
- 每次遍历未排序部分，选择出其中最小元素，放到已排序部分
- 直到所有元素都位于已排序部分



算法运行详解

<https://visualgo.net/sorting>

排序算法 – 选择排序 (2)

- 如何将上述思想用程序实现?
- 定义三个变量
 - n – 数组中待排序元素的个数
 - i – 已/未排序分界 $i = 0, 1, 2, \dots, n-2$
 - j – 当前访问未排序元素 $j = i+1, i+2, \dots, n-1$
 - min_index – 未排序中最小元素的下标

排序算法 – 选择排序 (3)

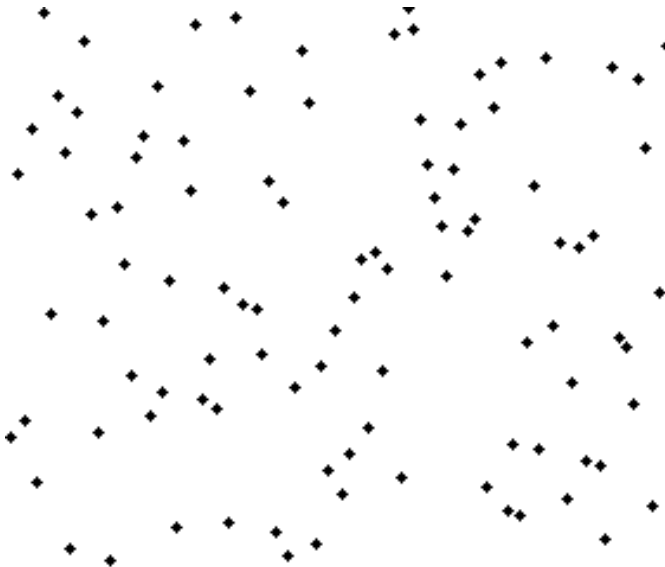
- 练习：画出选择排序程序的N-S图
- 核心代码

```
24     for (int i = 0; i < count - 1; i ++) {  
25         int min_index = i;  
26         for (int j = i + 1; j < count; j ++) {  
27             if (arr[min_index] > arr[j]) {  
28                 min_index = j;  
29             }  
30         }  
31         if (min_index != i) {  
32             int tmp = arr[min_index];  
33             arr[min_index] = arr[i];  
34             arr[i] = tmp;  
35         }  
36     }
```


排序算法 – 插入排序 (1)

■ 基本思路

- 把数组分为已排序和未排序两部分
- 依次访问未排序元素，将它插入到已排序部分的相应位置
- 为了支持插入，需要将元素向后移位
- 直到所有元素都位于已排序部分



算法运行详解

<https://visualgo.net/sorting>

排序算法 – 插入排序 (2)

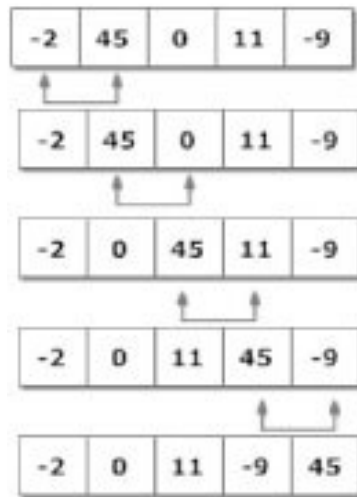
- 如何将上述思想用程序实现?
- 定义三个变量
 - n – 数组中待排序元素的个数
 - i – 已/未排序分界 $i = 0, 1, 2, \dots, n-1$
 - j – 当前访问以排序元素 $j = i-1, i-2, \dots, 0$
 - $\text{arr}[j+1] = \text{arr}[j]$ – 数组元素移位操作

排序算法 – 插入排序 (3)

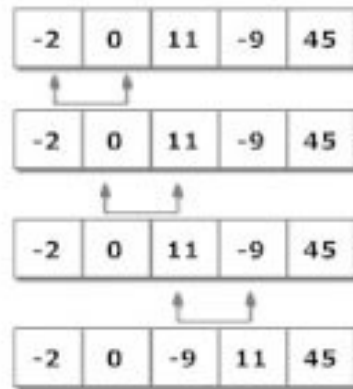
- 练习：画出插入排序程序的N-S图
- 核心代码

```
28     for (int i = 0; i < count; i++) {
29         int tmp = arr[i];
30         int j = i - 1;
31         for (; j >= 0; j--) {
32             if (arr[j] <= tmp) break;
33             arr[j+1] = arr[j];
34         }
35         arr[j+1] = tmp;
36     }
```

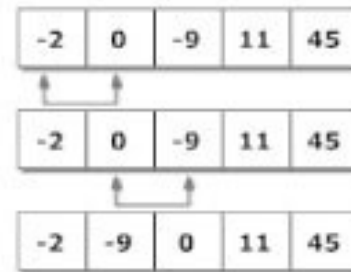
排序算法 – 冒泡排序



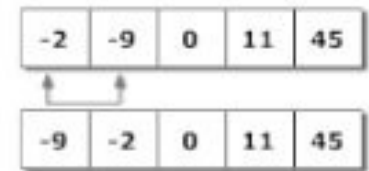
Step 1



Step 2

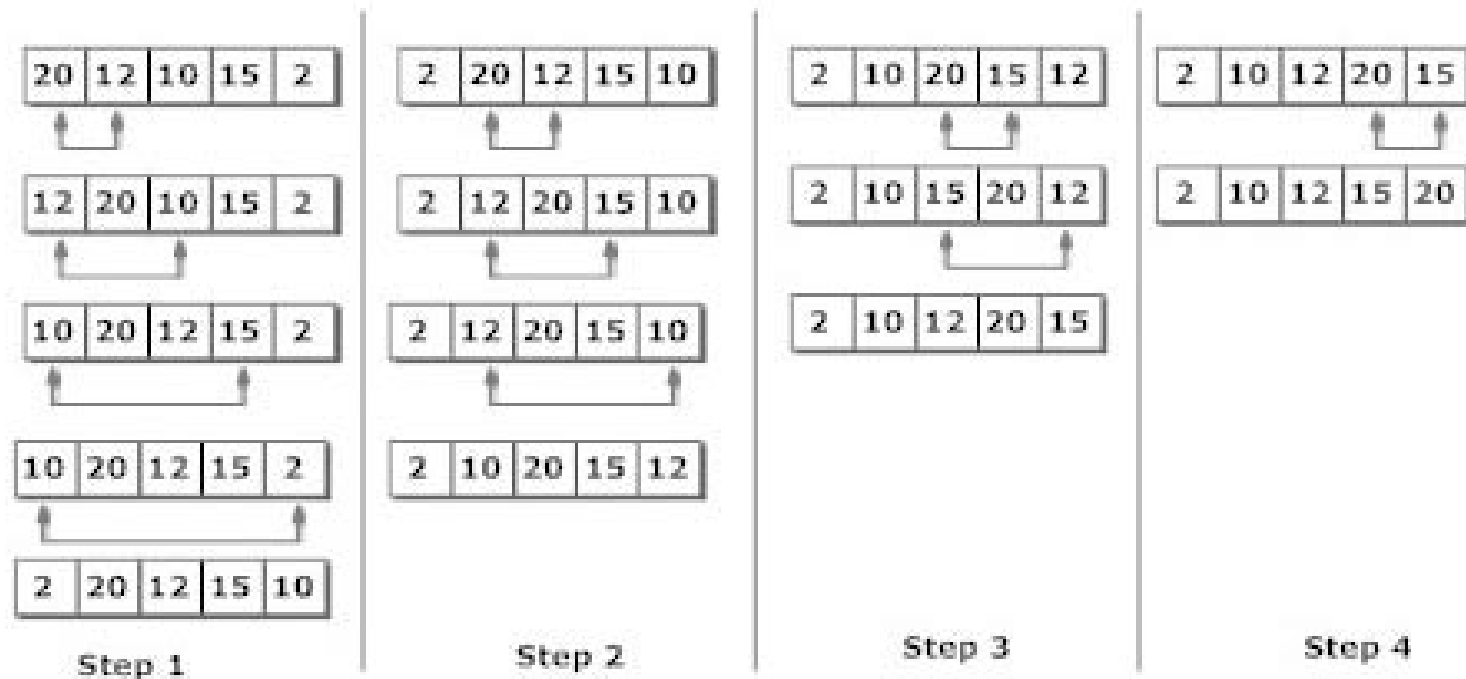


Step 3

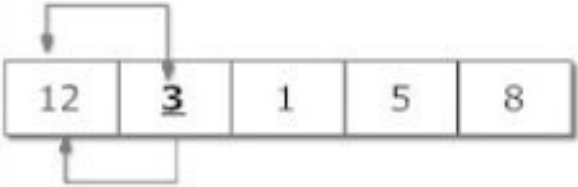
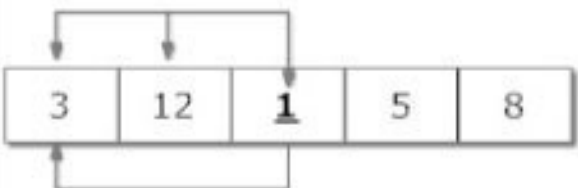
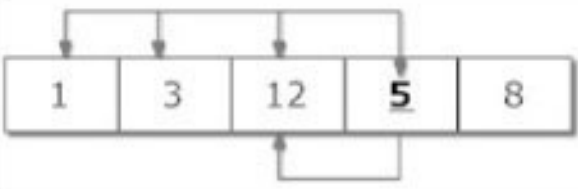
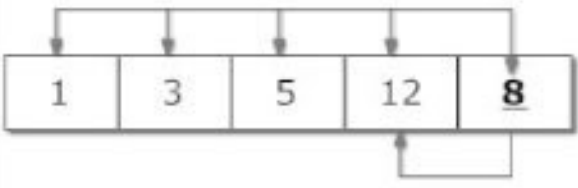



Step 4

排序算法 – 选择排序



排序算法 – 插入排序

Step 1		Checking second element of array with element before it and inserting it in proper position. In this case, 3 is inserted in position of 12.
Step 2		Checking third element of array with elements before it and inserting it in proper position. In this case, 1 is inserted in position of 3.
Step 3		Checking fourth element of array with elements before it and inserting it in proper position. In this case, 5 is inserted in position of 12.
Step 4		Checking fifth element of array with elements before it and inserting it in proper position. In this case, 8 is inserted in position of 12.
		Sorted Array in Ascending Order

排序算法 – 代码比较

```
for (int i = 0; i < count - 1; i++) {
    int swap = 0;
    for (int j = 0; j < count - 1 - i; j++) {
        if (arr[j] > arr[j+1]) {
            int tmp = arr[j];
            arr[j] = arr[j+1];
            arr[j+1] = tmp;
            swap = 1;
        }
    }
    if (swap == 0)
        break;
}

// Selection Sort
for (int i = 0; i < count; i++) {
    int min_index = i;
    for (int j = i + 1; j < count; j++) {
        if (arr[min_index] > arr[j])
            min_index = j;
    }
    if (min_index != i) {
        int tmp = arr[i];
        arr[i] = arr[min_index];
        arr[min_index] = tmp;
    }
}

// Insertion Sort
for (int i = 1; i < count; i++) {
    int tmp = arr[i];
    int j = i - 1;
    for (; j >= 0; j--) {
        if (arr[j] > tmp)
            break;
    }
    arr[j+1] = tmp;
}
```

回文判断

- 回文
 - Able was I ere I saw Elba
 - 落败孤岛孤败落
- 写一个程序，让用户任意输入一个字符串，判断是否是回文

回文判断

■ 分析思路

- 判断位置 i 的字符与 $n-1-i$ 的字符是否相等
- 循环开始和终止的边界?

■ 代码示例

```
int palindrome = 1;
for (int i = 0; i < len / 2; i++) {
    if (array[i] != array[len-1-i]) {
        palindrome = 0;
        break;
    }
}
```

回文判断

- 字符串的读入
 - 使用gets读入字符串
 - 使用scanf读入字符串
 - 使用getchar读入字符串
 - 什么区别?
- 上述字符串长度len怎么确定?

搜索问题

■ 问题的提出

➤ 如何“快速地”在数组中搜索一个数？

➤ 例子：友学1032题目

- 输入

10

24

42 24 10 29 27 12 58 31 8 16

输出：查找元素的位置；找不到，输出-1

二分查找

If searching for 23 in the 10-element array:

2	5	8	12	16	23	38	56	72	91
---	---	---	----	----	----	----	----	----	----

23 > 16, take 2 nd half	L				H				
	2	5	8	12	16	23	38	56	72

23 < 56, take 1 st half	L					H			
	2	5	8	12	16	23	38	56	72

Found 23, Return 5						L	H			
	2	5	8	12	16	23	38	56	72	91

二分查找

```
int l = 0, r = n-1;  
int cmp = 0, found = -1;
```

```
    int mid = (l + r) / 2;  
    cmp ++;
```

```
        found = mid;  
        break;
```

```
    }
```

```
}
```

回顾：数组的排序

■ 一维数组的排序

➤ 排序算法框架：冒泡、选择、插入

➤ 数组排序问题解题三要素

- 排序算法框架的选择
- 元素两两比较的选择分支语句块
- 元素两两交换的语句块

■ 错误思路：使用多次排序过程

■ 正确思路：使用一次排序过程，根据排序标准修改两两比较的语句块

练习：选择排序跟踪


3	1	5	2
---	---	---	---

i



j

因为 $a[1] < a[0]$,
更新min_index为1

```
24 for (int i = 0; i < count - 1; i ++) {  
25     int min_index = i;  
26      for (int j = i + 1; j < count; j ++) {  
27         if (arr[min_index] > arr[j]) {  
28             min_index = j;  
29         }  
30     }  
31     if (min_index != i) {  
32         int tmp = arr[min_index];  
33         arr[min_index] = arr[i];  
34         arr[i] = tmp;  
35     }  
36 }
```

练习：选择排序跟踪

3	1	5	2
---	---	---	---

i



j

因为 $a[2] > a[1]$
不必更新min_index

```
24     for (int i = 0; i < count - 1; i++) {
25         int min_index = i;
26         ➡ for (int j = i + 1; j < count; j++) {
27             if (arr[min_index] > arr[j]) {
28                 min_index = j;
29             }
30         }
31         if (min_index != i) {
32             int tmp = arr[min_index];
33             arr[min_index] = arr[i];
34             arr[i] = tmp;
35         }
36     }
```


练习：选择排序跟踪

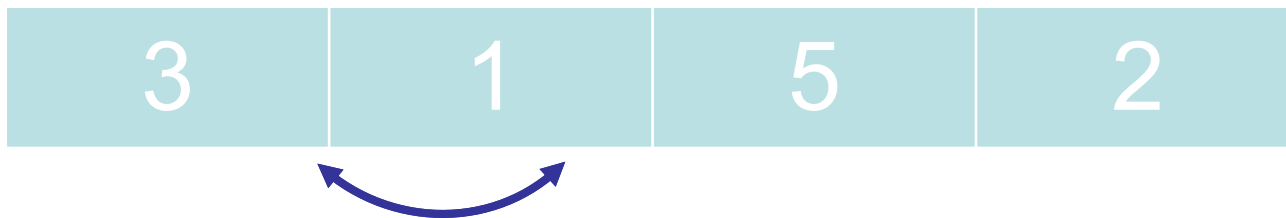
3	1	5	2
---	---	---	---

i

因为 $a[3] > a[1]$
不必更新min_index *j*

```
24     for (int i = 0; i < count - 1; i++) {  
25         int min_index = i;  
26         ➡ for (int j = i + 1; j < count; j++) {  
27             if (arr[min_index] > arr[j]) {  
28                 min_index = j;  
29             }  
30         }  
31         if (min_index != i) {  
32             int tmp = arr[min_index];  
33             arr[min_index] = arr[i];  
34             arr[i] = tmp;  
35     }  
36 }
```

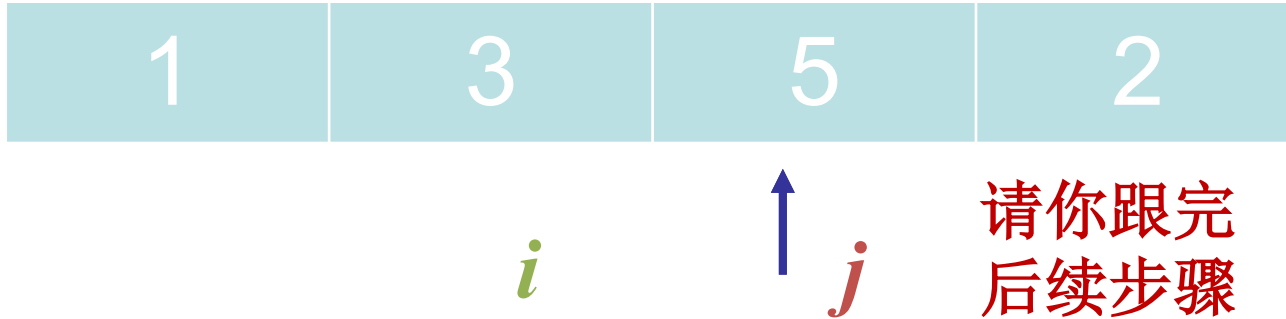
练习：选择排序跟踪



元素交换

```
24     for (int i = 0; i < count - 1; i++) {  
25         int min_index = i;  
26         for (int j = i + 1; j < count; j++) {  
27             if (arr[min_index] > arr[j]) {  
28                 min_index = j;  
29             }  
30         }  
31         if (min_index != i) {  
32             int tmp = arr[min_index];  
33             arr[min_index] = arr[i];  
34             arr[i] = tmp;  
35         }  
36     }
```

练习：选择排序跟踪



```
24     for (int i = 0; i < count - 1; i++) {  
25         int min_index = i;  
26         ➡ for (int j = i + 1; j < count; j++) {  
27             if (arr[min_index] > arr[j]) {  
28                 min_index = j;  
29             }  
30         }  
31         if (min_index != i) {  
32             int tmp = arr[min_index];  
33             arr[min_index] = arr[i];  
34             arr[i] = tmp;  
35         }  
36     }
```

练习：插入排序跟踪

3	1	5	2
---	---	---	---

↑
j

i

此时无需操作
注：可以从*i*=1开始循环

```
28     for (int i = 0; i < count; i ++){
29         int tmp = arr[i];
30         int j = i - 1;
31         ➡ for (; j >= 0; j --){
32             if (arr[j] <= tmp) break;
33             arr[j+1] = arr[j];
34         }
35         arr[j+1] = tmp;
36     }
```

练习：插入排序跟踪

tmp

1

3

1

5

2



j

i

```
28     for (int i = 0; i < count; i++) {  
29         int tmp = arr[i];  
30         int j = i - 1;  
31         → for (; j >= 0; j--) {  
32             if (arr[j] <= tmp) break;  
33             arr[j+1] = arr[j];  
34         }  
35         arr[j+1] = tmp;  
36     }
```

练习：插入排序跟踪

tmp

1

3

3

5

2



j

i

```
28     for (int i = 0; i < count; i ++){
29         int tmp = arr[i];
30         int j = i - 1;
31         for (; j >= 0; j --){
32             if (arr[j] <= tmp) break;
33             arr[j+1] = arr[j];
34         }
35         arr[j+1] = tmp;
36     }
```

练习：插入排序跟踪

tmp

1

1

3

5

2



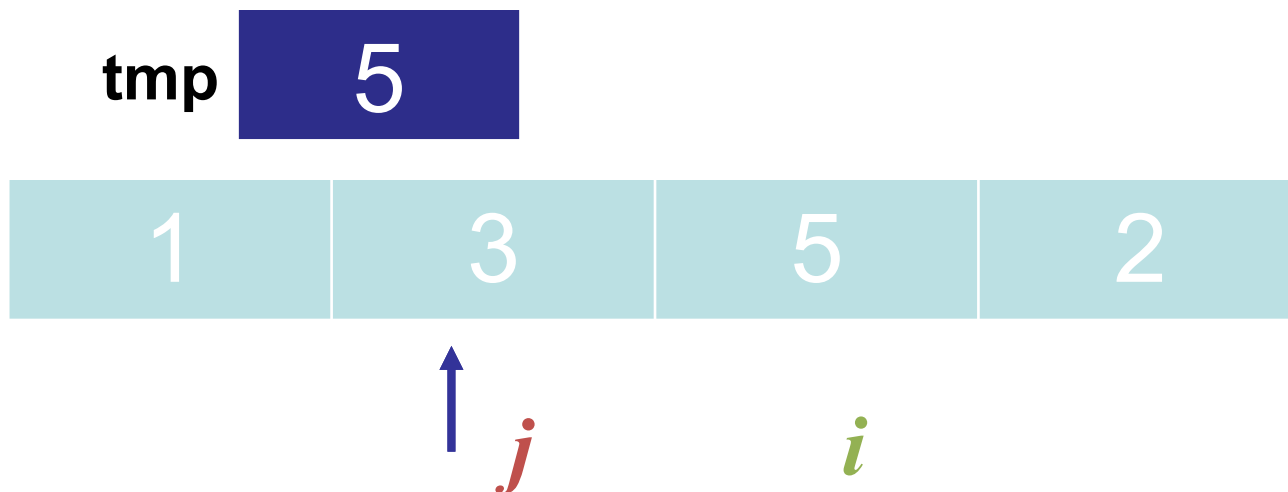
j

i

```
28     for (int i = 0; i < count; i++) {
29         int tmp = arr[i];
30         int j = i - 1;
31         for (; j >= 0; j--) {
32             if (arr[j] <= tmp) break;
33             arr[j+1] = arr[j];
34         }
35         arr[j+1] = tmp;
36     }
```

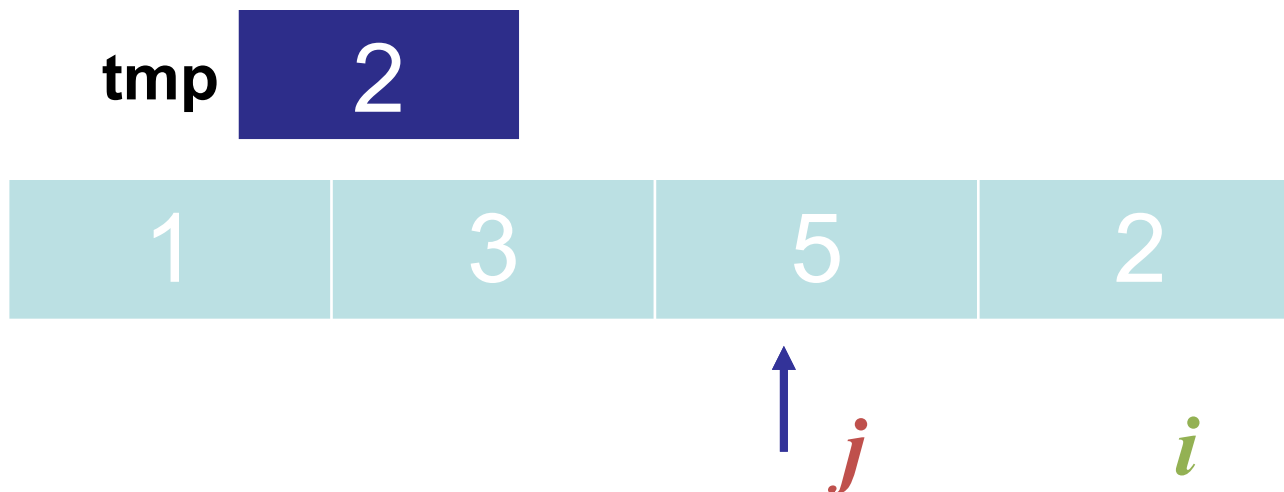


练习：插入排序跟踪



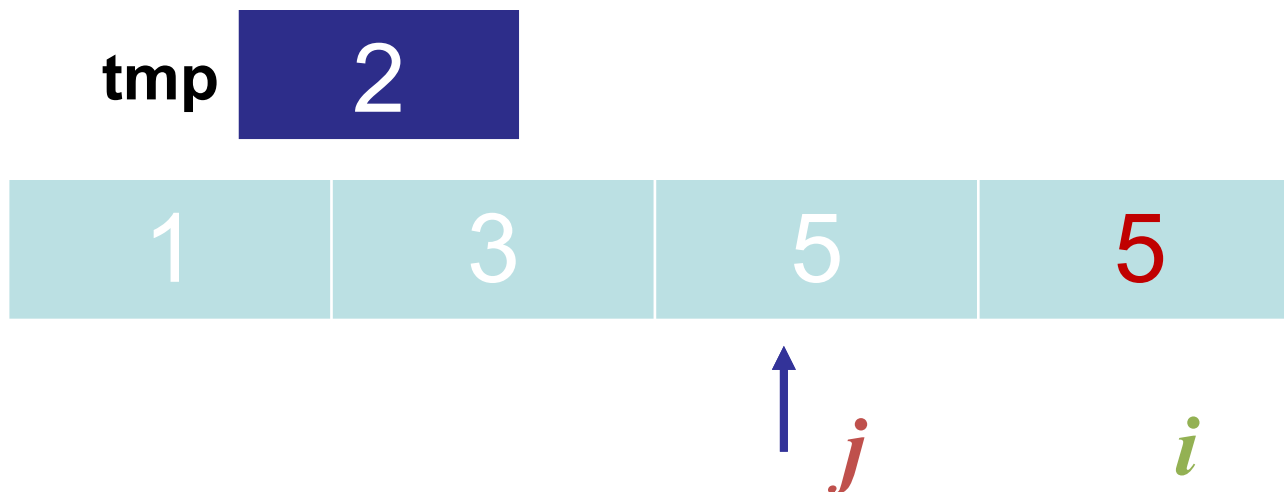
```
28     for (int i = 0; i < count; i++) {  
29         int tmp = arr[i];  
30     →   int j = i - 1;  
31         for (; j >= 0; j--) {  
32             if (arr[j] <= tmp) break;  
33             arr[j+1] = arr[j];  
34         }  
35         arr[j+1] = tmp;  
36     }
```


练习：插入排序跟踪



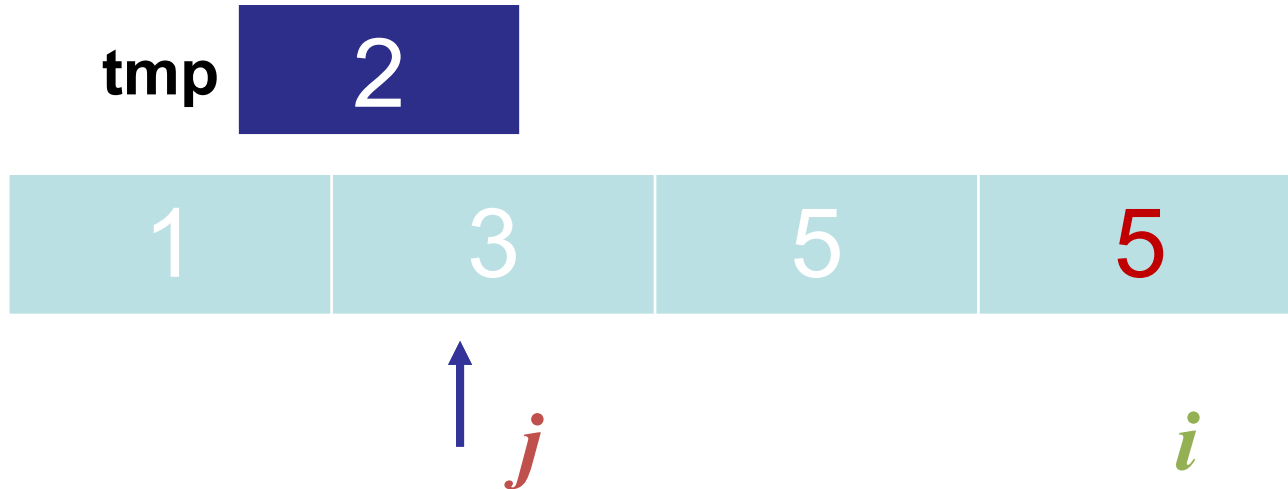
```
28     for (int i = 0; i < count; i++) {  
29         int tmp = arr[i];  
30     ➔   int j = i - 1;  
31         for (; j >= 0; j--) {  
32             if (arr[j] <= tmp) break;  
33             arr[j+1] = arr[j];  
34         }  
35         arr[j+1] = tmp;  
36     }
```

练习：插入排序跟踪



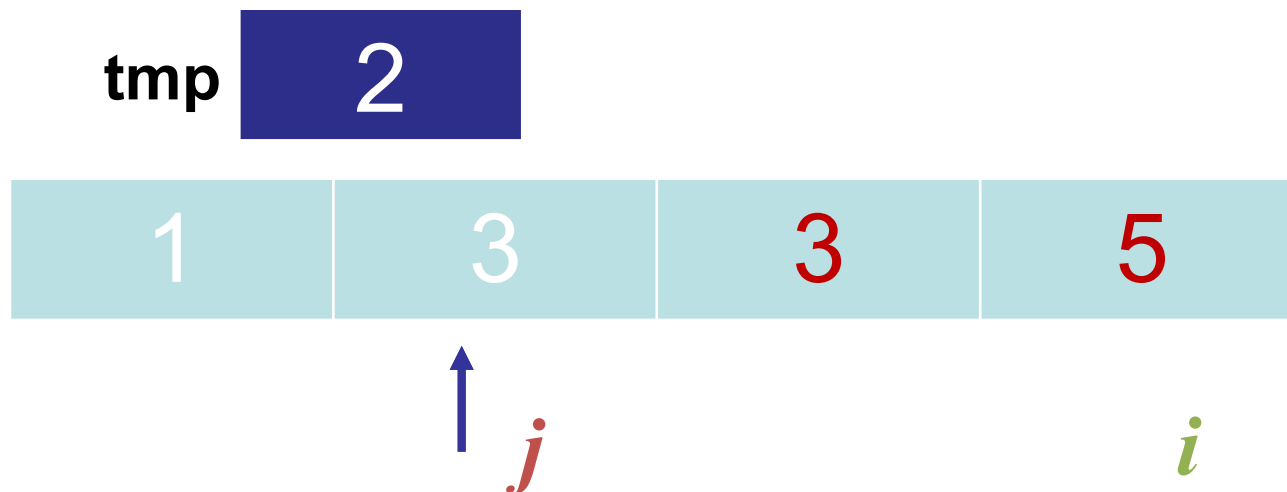
```
28     for (int i = 0; i < count; i++) {  
29         int tmp = arr[i];  
30         int j = i - 1;  
31         for (; j >= 0; j--) {  
32             if (arr[j] <= tmp) break;  
33             arr[j+1] = arr[j];  
34         }  
35         arr[j+1] = tmp;  
36     }
```

练习：插入排序跟踪



```
28     for (int i = 0; i < count; i++) {  
29         int tmp = arr[i];  
30     ➔   int j = i - 1;  
31         for (; j >= 0; j--) {  
32             if (arr[j] <= tmp) break;  
33             arr[j+1] = arr[j];  
34         }  
35         arr[j+1] = tmp;  
36     }
```

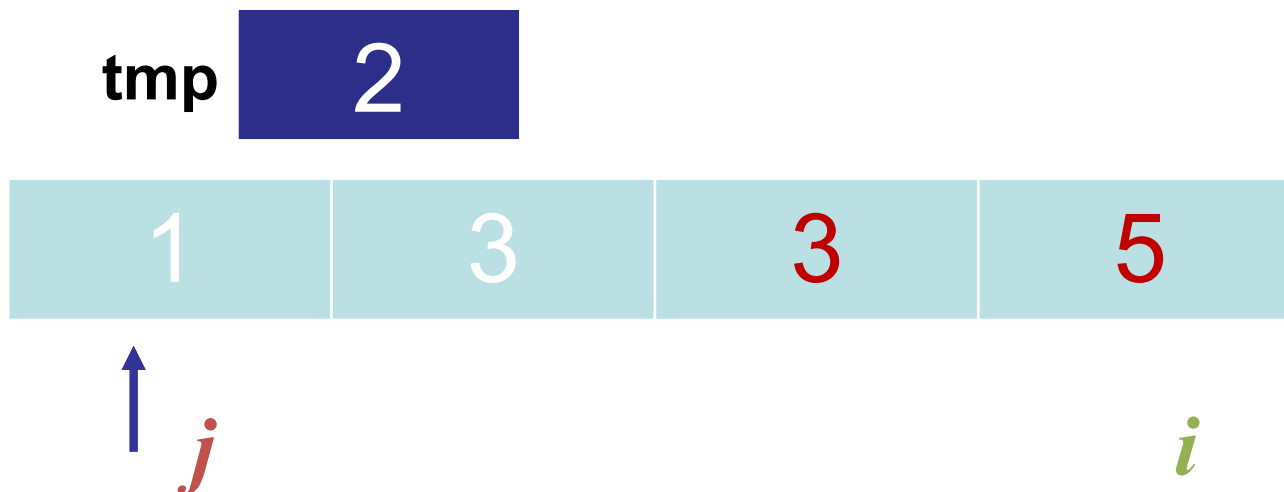
练习：插入排序跟踪



```
28     for (int i = 0; i < count; i++) {
29         int tmp = arr[i];
30         int j = i - 1;
31         for (; j >= 0; j--) {
32             if (arr[j] <= tmp) break;
33             arr[j+1] = arr[j];
34         }
35         arr[j+1] = tmp;
36     }
```

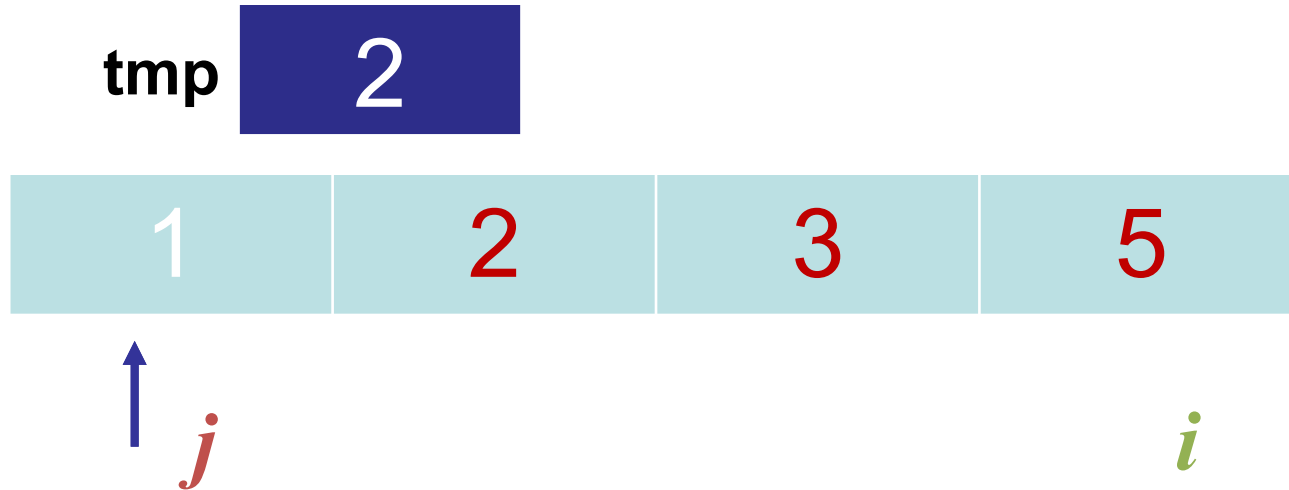
➡

练习：插入排序跟踪



```
28     for (int i = 0; i < count; i++) {  
29         int tmp = arr[i];  
30         ➡ int j = i - 1;  
31         for (; j >= 0; j--) {  
32             if (arr[j] <= tmp) break;  
33             arr[j+1] = arr[j];  
34         }  
35         arr[j+1] = tmp;  
36     }
```

练习：插入排序跟踪



```
28     for (int i = 0; i < count; i++) {  
29         int tmp = arr[i];  
30         int j = i - 1;  
31         for (; j >= 0; j--) {  
32             if (arr[j] <= tmp) break;  
33             arr[j+1] = arr[j];  
34         }  
35         arr[j+1] = tmp;  
36     }
```

思考

- 选择排序和插入排序中的计数器*i*和*j*分别代表了什么含义
 - 选择排序中的 *i*: 已排序部分最后一个元素
 - 选择排序中的 *j*: 未排序部分的每个元素
 - 插入排序中的 *i*: 未排序的第一个元素
 - 插入排序中的 *j*: 已排序部分的每个元素

回顾：二分查找

If searching for 23 in the 10-element array:

2	5	8	12	16	23	38	56	72	91
---	---	---	----	----	----	----	----	----	----

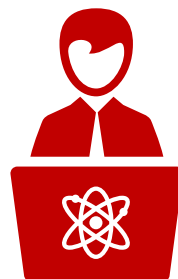
23 > 16, take 2 nd half	L				H				
	2	5	8	12	16	23	38	56	72

23 < 56, take 1 st half						L					H
	2	5	8	12	16	23	38	56	72	91	

Found 23, Return 5	L					H				
	2	5	8	12	16	23	38	56	72	91



中國人民大學
RENMIN UNIVERSITY OF CHINA



谢谢大家!

