

第 7 章 数组

7.1 教学要点

本章通过典型程序解析，主要介绍有关数组的定义、存储方式、引用等基本概念，包括一维数组、二维数组和字符串，并介绍几个典型的算法。使学生理解能综合运用数组编写程序。

7.1 节通过案例“输出所有大于平均值的数”，详细介绍一维数组的基本使用方法，教师在讲授时，应详细介绍一维数组在内存的存储方式、一维数组的定义及引用、一维数组的初始化，并通过若干实例说明一维数组的使用方法，其中应重点分析选择排序法的实现。使学生能正确使用一维数组进行程序设计。

7.2 节通过案例“找出矩阵中最大值所在的位置”，详细介绍二维数组的基本使用方法，教师在讲授时，应详细介绍二维数组在内存的存储方式、二维数组的定义及引用、二维数组的初始化，并通过若干实例说明二维数组的使用方法，其中应重点说明对矩阵的操作。使学生能正确使用二维数组进行程序设计。

7.3 节通过案例“判断回文”，详细介绍字符串的基本概念及使用方法，教师在讲授时，应详细介绍字符串与一维字符数组的区别、字符串的存储以及字符串的操作方法，并通过若干实例说明字符串的正确使用，其中应重点分析进制转换的实现。使学生能正确使用字符串进行程序设计。

讲授学时：6 学时，实验学时同讲授学时。

本章的知识能力结构图见图 7.1。

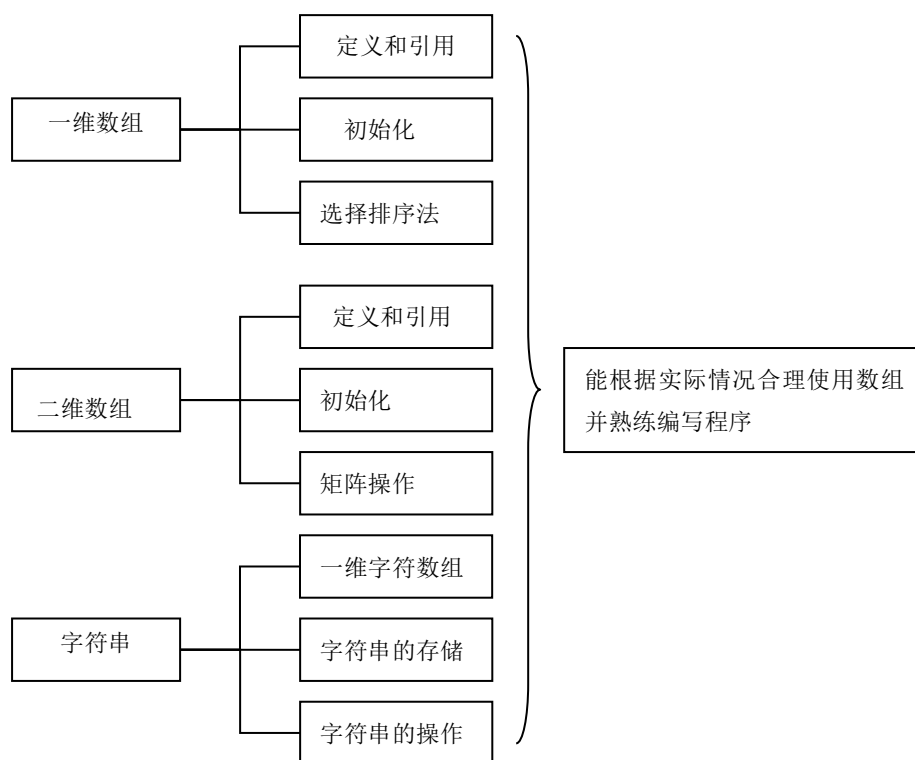






图 7.1 知识能力结构图

7.2 讲稿

1	 <h3>Chap 7 数组</h3> <p>7.1 输出所有大于平均值的数</p> <p>7.2 找出矩阵中最大值所在的位置</p> <p>7.3 判断回文</p>	本章分 3 节。
2	 <h3>本章要点</h3> <ul style="list-style-type: none"> ■ 什么是数组？为什么要使用数组？如何定义数组？ ■ 如何引用数组元素？ ■ 二维数组的元素在内存中按什么方式存放？ ■ 什么是字符串？字符串结束符的作用是什么？ ■ 如何实现字符串的存储和操作，包括字符串的输入和输出？ ■ 怎样理解C语言将字符串作为一个特殊的一维字符数组？ 	提出本章的学习要点。
3	 <h3>7.1 输出所有大于平均值的数</h3> <p>例7-1 输入10个整数，计算这些数的平均值，再输出所有大于平均值的数。</p> <p>7.1.1 程序解析</p> <p>7.1.2 一维数组的定义和引用</p> <p>7.1.3 一维数组的初始化</p> <p>7.1.4 使用一维数组编程</p>	<p>引导学生分析题目：</p> <p>这是一个查找问题，输入 10 个整数后，先计算平均值，再查找并输出大于平均值的数，因此，这些整数在输入后，还要被使用 2 次。</p> <p>需要解决的问题是：</p> <p>怎么保存这 10 个数？定义 10 个变量？</p> <p>怎么保存 100 个数？定义 100 个变量？</p> <p>怎么保存 n 个数？定义 n 个变量？</p> <p>引出一维数组。</p>
4	<h4>7.1.1 程序解析—输出大于均值的数</h4> <pre>int main(void) { int i; double average; int a[10]; printf("Enter 10 integers:"); sum = 0; for(i = 0; i < 10; i++){ scanf("%d", &a[i]); sum = sum + a[i]; } average = sum / 10; printf("average = %.2f\n", average); printf(">average:"); for(i = 0; i < 10; i++){ if(a[i] > average) printf("%d ", a[i]); } printf("\n"); return 0; }</pre> 	<p>展示、运行例 7-1 程序。</p> <p>解读程序，可分块说明数组的定义、输入、处理、输出 4 部分。</p> <p>略过具体的细节，重点说明一维数组 a 的作用。</p>

5	<div data-bbox="300 208 820 387"> </div> <div data-bbox="316 398 807 555"> <ul style="list-style-type: none"> ■ 数组：相同类型数据的有序集合，在内存中连续存放。 <ul style="list-style-type: none"> □ 由数组名和下标唯一地确定每个数组元素 □ 每个元素都属于同一类型 ■ 一批相同类型的变量使用同一个数组变量名，用下标来相互区分。 <ul style="list-style-type: none"> □ 优点：表述简洁，可读性高；便于使用循环结构 </div>	<p>展示说明一维数组 a 的组织方式。</p> <p>说明以下问题：</p> <p>什么是数组？</p> <p>一维数组 a 在内存中的存储方式。</p> <p>数组下标默认从 0 开始。</p> <p>如何确定数组中的一个元素？</p> <p>使用数组的好处。</p>
6	<div data-bbox="300 624 820 965"> </div> <div data-bbox="316 741 751 965"> <p>类型名 数组名 [数组长度] 数组长度为常量</p> <p>类型名：数组元素的类型</p> <p>数组名：数组（变量）的名称，标识符</p> <p>数组长度：常量表达式，给定数组的大小</p> <p>int a[10]; 定义一个含有10个整型元素的数组 a</p> <p>char c[200]; 定义一个含有200个字符元素的数组 c</p> <p>float f[5]; 定义一个含有5个浮点型元素的数组 f</p> </div>	<p>重点说明一维数组的定义方法。</p> <p>提醒：数组长度必须是常量表达式，给出数组的大小。</p> <p>说明数组 a 或 c 的组织方式，特别是定义数组时的数组长度与数组使用时下标的对应关系。如数组 a，对应的是 a[0]~a[9]。所占的字节空间：若 int 类型占 2 个字节，则数组 a 占 10×2=20 个字节。</p>
7	<div data-bbox="300 1041 820 1382"> </div> <div data-bbox="316 1355 783 1382"> <p>数组名是一个地址常量，存放数组内存空间的首地址。</p> </div>	<p>说明数组的内存结构。说明以下几点：</p> <p>数组 a 所占内存字节数的计算。</p> <p>数组中下标连续的单元其内存地址也连续。</p> <p>数组名是常量，存放整个数组空间的起始地址，不允许被修改。</p>
8	<div data-bbox="300 1458 820 1798"> </div>	<p>重点说明如何引用数组元素。</p> <p>提醒：</p> <p>数组的定义和引用的区别，特别是下标。</p> <p>一次只能引用一个元素。</p> <p>引用时下标的合理范围是 0~长度-1，下标不能越界。</p>

9	<div> <div></div> <div> <h2>区分数组的定义和数组元素的引用</h2> <p>定义数组 类型名 数组名[数组长度]</p> <p>引用数组元素 数组名[下标]</p> <pre>int a[10]; a[0] = a[9] = 0; a[i] = i;</pre> <div>数组长度为常量</div> <div>下标不要越界</div> </div> </div>	再次提醒数组的定义和引用的区别。
10	<div> <div></div> <div> <h2>7.1.3 一维数组的初始化</h2> <ul style="list-style-type: none"> 定义数组时，对数组元素赋初值 类型名 数组名[数组长度] = {初值表}; <pre>int a[10] = {1,2,3,4,5,6,7,8,9,10}; a[0]=1, a[1]=2,..... a[9]=10</pre> 静态数组、动态数组的初始化 <pre>static int b[5] = {1, 2, 3, 4, 5};</pre> 静态存储的数组如果没有初始化，所有元素自动赋0 <pre>static int b[5];</pre> 动态存储的数组如果没有初始化，所有元素为随机值 <pre>auto int c[5]; 等价于 int c[5];</pre> </div> </div>	和简单变量的初始化做对比，说明如何在定义数组时进行初始化。 分为以下两种方式： 全部元素初始化； 部分元素初始化。
11	<div> <div></div> <div> <h2>针对部分元素的初始化</h2> <pre>static int b[5] = {1, 2, 3}; b[0] = 1, b[1] = 2, b[2] = 3, b[3] = 0, b[4] = 0</pre> <pre>auto int fib[20] = {0, 1}; fib[0] = 0, fib[1] = 1, 其余元素不确定</pre> <ul style="list-style-type: none"> 如果对全部元素都赋初值，可以省略数组长度 <pre>int a[10] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}</pre> <div>建议不要省略数组长度</div> </div> </div>	说明如何针对部分元素初始化。 提醒： 不建议省略数组长度。
12	<div> <div></div> <div> <h2>7.1.4 使用一维数组编程</h2> <h3>数组和循环</h3> <pre>for (i = 0; i < n; i++) printf ("%d ", a[i]);</pre> <p>数组下标作为循环变量，通过循环，逐个处理数组元素</p> </div> </div>	数组的使用离不开循环，说明如何将下标作为循环变量，通过循环对数组的所有元素逐个处理。 特别说明这是一种常用的处理方法。

13	<div><div><div></div><div></div><div></div></div><div><h3>一维数组示例</h3><p>例 7-2 用数组计算fibonacci数列的前10个数，并按每行打印5个数的格式输出。</p><p>例 7-3 顺序查找法。输入5个整数，将它们存入数组a中，再输入1个数x，然后在数组中查找x，如果找到，输出相应的最小下标，否则，输出“Not Found”。</p><p>例 7-4 输入n(n<10)，再输入n个数 (1) 输出最小值和它所对应的下标 (2) 将最小值与第一个数交换，输出交换后的n个数</p><p>例 7-5 选择排序法。输入一个n(1<n≤10)，再输入n个整数，用选择法将它们从小到大排序后输出。</p><p>例 7-6 调查电视节目欢迎程度。某电视台要进行一次对该台8个栏目（设相应栏目编号为1~8）的受欢迎情况，共调查了1000位观众，现要求编写程序，输每一位观众的投票，每位观众只能选择一个最喜欢的栏目投票，统计输出各栏目的得票情况。</p><p>补充 二分查找法。设已有一个10个元素的整形数组a，且按值从小到大有序。输入一个整数x，然后在数组中查找x，如果找到，输出相应的下标，否则，输出“Not Found”。</p></div></div>	<p>举例使用一维数组编程，可适当让学生编写。需要注意例 7-3 和例 7-4 是为例 7-5 选择排序算法打下基础。</p> <p>补充的二分查找法可根据学生掌握情况，自行选择讲解。</p>										
14	<div><div><div></div><div></div><div></div></div><div><h3>例 7-2 计算fibonacci数列</h3><p>用数组计算fibonacci数列的前10个数，并按每行打印5个数的格式输出。</p><p>1, 1, 2, 3, 5, 8, 13,</p><p>用数组计算并存放fibonacci数列的前10个数</p><p>f[0] = f[1] = 1</p><p>f[i] = f[i-1] + f[i-2] 2 ≤ i ≤ 9</p></div></div>	<p>说明数列的各项值如何与数组元素对应。并根据数列的特点找出数组元素间的关系：f[i] = f[i-1] + f[i-2]，从而利用循环完成程序。</p>										
15	<div><div><div></div><div></div><div></div></div><div><h3>例 7-2 源程序</h3><pre>#include <stdio.h> int main(void) { int i; int fib[10] = {1, 1}; /* 数组初始化 */ for (i = 2; i < 10; i++) fib[i] = fib[i-1] + fib[i-2]; for (i = 0; i < 10; i++){ printf("%6d", fib[i]); if ((i+1) % 5 == 0) /* 5个数换行 */ printf("\n"); } return 0; }</pre><table><tr><td>1</td><td>1</td><td>2</td><td>3</td><td>5</td></tr><tr><td>8</td><td>13</td><td>21</td><td>34</td><td>55</td></tr></table></div></div>	1	1	2	3	5	8	13	21	34	55	<p>根据上页分析，给出源程序，并运行。</p> <p>特别指出：</p> <p>数列的第 i 项与数组下标的对应，即保存在下标为 i-1 的单元 fib[i-1]。并建议学生养成从 0 开始数数的习惯。</p> <p>数列前两项的值怎么确定？</p> <p>如何计算并保存第 3 项后的数列值？</p> <p>如何每行 5 个格式化输出？</p>
1	1	2	3	5								
8	13	21	34	55								
16	<div><div><div></div><div></div><div></div></div><div><h3>例7-3 在数组中查找一个给定的数</h3><p>输入5个整数，将它们存入数组a中，再输入1个数x，然后在数组中查找x，如果找到，输出相应的最小下标，否则，输出“Not Found”。</p><p>输入：2 9 8 1 9</p><p>9</p><p>输出：1</p><p>输入：2 9 8 1 6</p><p>7</p><p>输出：Not Found</p></div></div>	<p>引导学生分析题目要求并思考实现路径。</p> <p>(1)如果题目中明确只有一个满足条件的元素，则将 x 与数组中的元素逐个比较，发现相等的就输出相应的下标；</p> <p>(2)如果有多个满足条件的元素，本例要求找到第一个相等的元素（最小下标），将 x 与数组中的元素逐个比较，发现相等的就输出相应的下标，并停止查找；</p> <p>(3)如果没有找到，输出 Not found。</p> <p>(1)可以作为(2)的特例，针对(2)和(3)如何实现？</p>										


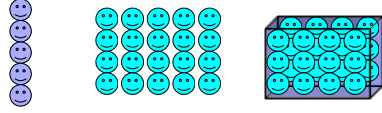
17	<pre> #include <stdio.h> int main(void) { int i, flag, x; int a[5]; printf ("Enter 5 integers: "); for (i = 0; i < 5; i++) scanf ("%d", &a[i]); printf ("Enter x: "); scanf ("%d", &x); flag = 0; for (i = 0; i < 5; i++) if (a[i] == x){ printf ("Index is %d\n", i); flag = 1; break; } if (flag == 0) printf("Not Found\n"); return 0; } </pre> <p>例 7-3 源程序</p> <p>Enter 5 integers: 2 9 8 1 9 Enter x: 9 Index is 1</p> <p>Enter 5 integers: 2 9 8 1 9 Enter x: 7 Not Found</p> <p>flag的作用?</p>	<p>介绍算法，分析程序的实现方式，重点分析并单步演示数组元素的输入过程、找元素的过程、输出过程。</p> <p>特别说明</p> <p>flag 的作用：是否找到相同元素的标志；</p> <p>break 的作用：一旦找到后就终止循环。</p>
18	<pre> #include <stdio.h> int main(void) { int i, flag, x; int a[5]; printf("Enter 5 integers: "); for(i = 0; i < 5; i++) scanf("%d", &a[i]); printf("Enter x: "); scanf("%d", &x); flag = 0; for(i = 0; i < 5; i++) if(a[i] == x){ printf("Index is %d\n", i); flag = 1; break; } if(flag == 0) printf("Not Found\n"); return 0; } </pre> <p>例 7-3 思考(1) 去掉break语句，结果？</p> <p>Enter 5 integers: 2 9 8 1 9 Enter x: 9 Index is 1 Index is 4</p>	<p>详细分析 break 语句的作用。</p> <p>修改程序，去掉 break 语句，则题意不同。再次解读程序，并查看运行结果。</p>
19	<pre> #include <stdio.h> int main(void) { int i, index, x; int a[5]; printf("Enter 5 integers: "); for(i = 0; i < 5; i++) scanf("%d", &a[i]); printf("Enter x: "); scanf("%d", &x); index = -1; for(i = 0; i < 5; i++) if(a[i] == x) index = i; if(index != -1) printf("Index is %d\n", index); else printf("Not Found\n"); return 0; } </pre> <p>例 7-3 思考(2)</p> <p>Enter 5 integers: 2 9 8 1 9 Enter x: 9 Index is 4</p> <p>index的作用?</p>	<p>修改程序，新增变量 index，解读程序。</p> <p>说明：</p> <p>(1)index 的作用：是否找到相同元素的标志与记录，如果找到，则记录下标，否则，值为-1；</p> <p>(2)题意的区别，找最后一个值相同的元素，即输出相应的最大下标。</p> <p>提问：</p> <p>如果要找到第一个值相同的元素，如何修改程序？</p>
20	<pre> #include <stdio.h> int main(void) { int i, min, n; int a[10]; printf("Enter n: "); scanf("%d", &n); printf("Enter %d integers: ", n); for(i = 0; i < n; i++) scanf("%d", &a[i]); min = a[0]; for(i = 1; i < n; i++) if(a[i] < min) min = a[i]; printf("min is %d\n", min); return 0; } </pre> <p>例 7-4 求最小值</p> <p>Enter n: 6 Enter 6 integers: 2 9 -1 8 1 6 min is -1</p> <p>方法</p> <p>虽得到了最小值，但不能确定最小值所在下标。</p>	<p>例 7-4 解决如何在数组中找最小（大）值的方法。</p> <p>特别说明 min 变量的作用：存放最小值，初值为 a[0]，先假设 a[0]为最小值，再与其他元素比较，并不断更新最小值 min。</p> <p>程序缺陷：虽找到了最小值，但不知道该值在几号单元，即下标。从而引出下面的程序。</p>

21	<div> <div>#include <stdio.h> int main(void) { int i, index, min, n; int a[10]; printf("Enter n: "); scanf("%d", &n); printf("Enter %d integers: ", n); for(i = 0; i < n; i++) scanf("%d", &a[i]); min = a[0]; index = 0; for(i = 1; i < n; i++) if(a[i] < min){ min = a[i]; index = i; } printf("min is %d index is %d\n", min, index); return 0; }</div> <div>例7-4 求最小值及下标</div> <div>Enter n: 6 Enter 6 integers: 2 9 -1 8 1 6 min is -1 index is 2</div> <div>min: 最小值 index: 最小值所在下标</div> </div>	找最小值时，用 min 变量保存最小值，同时用变量 index 记录相应的下标。
22	<div> <div>例 7-4(1) 求最小值及其下标</div> <div>输入n(n<10), 再输入n个数, 输出最小值和它所对应的下标。</div> <div>用index记录最小值对应的下标 a[index]就是最小值</div> </div>	继续优化程序。 在数组中只要知道了元素的下标，则元素的值也就确定了。 故在找最小值时，除了可以用上面的方法以外，还可以只用变量 index 保存最小值所在的下标，则 a[index]就是最小值。 此处可让学生解答如何根据上面程序给出流程图。
23	<div> <div>流程图</div> </div>	分析流程图，并可让学生按照流程图给出源程序。
24	<div> <div>求最小值及下标</div> <div>#include <stdio.h> int main(void) { int i, index, n; int a[10]; printf("Enter n: "); scanf("%d", &n); printf("Enter %d integers: ", n); for(i = 0; i < n; i++) scanf("%d", &a[i]); index = 0; for(i = 1; i < n; i++) if(a[i] < a[index]) index = i; printf("min is %d\tindex is %d\n", a[index], index); return 0; }</div> <div>Enter n: 6 Enter 6 integers: 2 9 -1 8 1 6 min is -1 index is 2</div> </div>	给出源程序并运行。 重点说明 index=0 的作用：假设 a[0]是最小值的初值，相应的下标为 0，即假设 0 是最小值的下标；再与其他元素比较，并不断更新最小值的下标。

25	<div><div><div>例 7-4(2) 交换最小值</div><div>输入$n(n<10)$, 再输入n个数, 将最小值与第一个数交换, 输出交换后的n个数。</div><div>用index记录最小值对应的下标 a[index]就是最小值 最小值与第一个数交换 a[index] <==> a[0]</div></div></div>	<p>前一个示例已找到最小值所在下标 index, 现考虑如何与第一个元素 a[0]交换。</p> <p>可先借用如何交换两个简单整型变量 x、y 来说明。</p> <p>建议由学生在前一程序基础上修改完成。</p> <p>提醒:</p> <p>例 7-4 的两个例子是选择排序法的基础。</p>												
26	<div><div><div>例 7-5 选择法排序</div><div>输入$n(n<10)$, 再输入n个数, 用选择法将它们从小到大排序后输出。</div><div>设 $n = 5$, 输入为: 3 5 2 8 1</div><div><table><tr><td>下标</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td></tr><tr><td>值</td><td>3</td><td>5</td><td>2</td><td>8</td><td>1</td></tr></table><div>第0趟: 1 5 2 8 3</div><div>第1趟: 1 2 5 8 3</div><div>第2趟: 1 2 3 8 5</div><div>第3趟: 1 2 3 5 8</div></div></div></div>	下标	0	1	2	3	4	值	3	5	2	8	1	<p>介绍选择排序。</p> <p>首先说明利用前例的两个功能进行排序的思路。</p> <p>提醒: 这是典型的排序算法。</p> <p>一趟选择排序: 找到最小值并与相应元素交换的过程。</p> <p>n 个元素需进行 $n-1$ 趟选择排序才能完成。为了方便和数组元素下标的对应, 在本例的说明中, 排序的趟数从 0 开始计数, 第 $0 \sim n-2$ 趟。</p>
下标	0	1	2	3	4									
值	3	5	2	8	1									
27	<div><div><div>选择法分析(1)</div><div>3 5 2 8 1 ($n = 5$)</div><div>5个数(a[0]~a[4])中找最小数, 与a[0]交换</div><div>(0) 1 5 2 8 3 a[4] <==> a[0]</div><div>4个数(a[1]~a[4])中找最小数, 与a[1]交换</div><div>(1) 1 2 5 8 3 a[2] <==> a[1]</div><div>3个数(a[2]~a[4])中找最小数, 与a[2]交换</div><div>(2) 1 2 3 8 5 a[4] <==> a[2]</div><div>2个数(a[3]~a[4])中找最小数, 与a[3]交换</div><div>(3) 1 2 3 5 8 a[4] <==> a[3]</div></div></div>	<p>以 5 个数排序为例, 具体分析 5 个元素进行选择排序时每一趟的过程。</p>												
28	<div><div><div>n个数重复n-1次</div><div>选择法分析(2)</div><div>(0) n 个数 (a[0] ~ a[n-1]) 中找最小数, 与a[0]交换</div><div>(1) n-1个数 (a[1] ~ a[n-1]) 中找最小数, 与a[1]交换</div><div>.....</div><div>(k) n-k个数 (a[k] ~ a[n-1]) 中找最小数, 与a[k]交换</div><div>.....</div><div>(n-2) 2个数 (a[n-2] ~ a[n-1]) 中找最小数, 与a[n-2]交换</div><div>(0) 5个数 (a[0]~a[4]) 中找最小数, 与 a[0] 交换</div><div>(1) 4个数 (a[1]~a[4]) 中找最小数, 与 a[1] 交换</div><div>(2) 3个数 (a[2]~a[4]) 中找最小数, 与 a[2] 交换</div><div>(3) 4个数 (a[3]~a[4]) 中找最小数, 与 a[3] 交换</div></div></div>	<p>从上述 5 个元素排序的过程, 扩展到 n 个元素(a[0] ~ a[n-1]), 需排序 $n-1$ 趟, 计数从 0 开始, 即第 $0 \sim n-2$ 趟。</p> <p>找规律, 第 k 趟时, 有 k 个数(a[0] ~ a[k-1])已经排好序, 在剩下未排序的($n - k$)个数(a[k] ~ a[n-1])中找到最小值, 与 a[k] 交换。</p>												

29	<p>流程图</p> <p>外循环控制： n 个数选择排序共需要 n-1 次</p> <p>内循环控制： 在下标范围 [k, n-1] 内找最小值所在位置 index</p>	<p>给出选择排序法的流程图，并说明流程结构，分外循环和内循环两部分，显然整个过程是一个二重循环。</p> <p>程序框架如下：</p> <pre>for (k = 0; k < n-1; k++) { // 共需 n-1 趟 // 在下标区间[k, n-1]内找最小值所在的位置 index(内循环实现); // 把 a[index]与 a[k]交换; }</pre>
30	<p>选择法排序（程序段）</p> <pre>for(k = 0; k < n-1; k++){ index = k; for(i = k + 1; i < n; i++) if(a[i] < a[index]) index = i; temp = a[index]; a[index] = a[k]; a[k] = temp; }</pre> <div> Enter n: 5 Enter 10 integers: 3 5 2 8 1 After sorted: 1 2 3 5 8 </div>	<p>根据流程图给出选择排序的程序。</p>
31	<p>例 7-6 投票情况统计</p> <p>某电视台要调查观众对该台8个栏目（设相应栏目编号为1~8）的受欢迎情况，共调查了1000位观众。现要求编写程序，输入每一位观众的投票情况（每位观众只能选择一个最喜欢的栏目投票），统计并输出各栏目的得票数。</p> <p>数组 count 保存各栏目的得票数 count[i]: 记录编号为 i (1~8) 的栏目的得票数 count[i]++: 统计编号为 i (1~8) 的栏目的得票数</p>	<p>引导学生分析题目：</p> <p>这是一个对得票数进行统计的问题。</p> <p>首先考虑：</p> <p>对于 8 个栏目，如何表示每个栏目的得票数？</p> <p>使用一维数组 count，并直接把栏目号作为下标，即用 count[i]记录 i 栏目的得票数。</p>
32	<p>例7-6 源程序</p> <pre>#include <stdio.h> int main(void) { int i, response; static int count[9]; for (i = 1; i <= 1000; i++) { printf("Enter your response: "); scanf ("%d", &response); if (response >= 1 && response <= 8) count[response]++; else printf ("invalid: %d\n", response); } printf ("result:\n"); for (i = 1; i <= 8; i++) printf ("%4d%4d\n", i, count[i]); return 0; }</pre> <div> input your response: 3 input your response: 6 input your response: 9 this is a bad response: 9 input your response: 8 ... result: 1 2 2 0 3 4 ... </div>	<p>说明 2 点：</p> <p>(1)为了程序直观易读，数组长度定义为 9。8 个栏目使用 count[1]~ count[8]共 8 个元素统计得票数，第 0 号元素 count[0]未用。</p> <p>(2)由于数组 count 作为计数器使用，初值应赋 0。</p> <p>如果使用静态数组，则初值自动赋 0；否则，需要写一段代码将数组的初值赋 0。</p>


33	<p>补充 二分法查找</p> <p>设已有一个10个元素的整型数组a，且按值从小到大有序排列。</p> <p>输入一个整数x，然后在数组中查找x，如果找到，输出相应的下标，否则，输出“Not Found”。</p> <p>例7-3顺序查找算法简单明了，其查找过程就是对数组元素从头到尾的遍历过程。但是，当数组很大时，查找的效率不高。</p> <p>二分查找的效率较高，但前提是数组元素必须是有顺序的。</p>	<p>本例是否讲解可自行选择。</p> <p>结合例 7-3，对比介绍查找算法。说明二分查找的优势、数组的条件。</p> <p>重点说明二分查找的算法，查找区间的确定，以及查找失败时的条件判断。</p>
34	<p>二分法查找流程图</p> <pre> graph TD Start([开始]) --> Input[输入 x] Input --> Init[low=0, high=n-1] Init --> LoopCond{low <= high} LoopCond -- 是 --> CalcMid[mid = (low + high) / 2] CalcMid --> CompareMid{x == a[mid]} CompareMid -- 是 --> OutputMid[输出 mid] CompareMid -- 否 --> Branch{ } Branch --> HighAdj[high = mid - 1] Branch --> LowAdj[low = mid + 1] HighAdj --> LoopCond LowAdj --> LoopCond LoopCond -- 否 --> NotFound[Not Found] OutputMid --> End([结束]) </pre>	<p>解释算法流程图。</p> <p>重点说明：</p> <p>循环开始前的初始情况；</p> <p>循环执行条件；</p> <p>循环结束后的判断。</p>
35	<p>二分法查找（程序段）</p> <pre> low = 0; high = n - 1; /* 开始时查找区间为整个数组 */ while (low <= high) { /* 循环条件 */ mid = (low + high) / 2; /* 中间位置 */ if (x == a[mid]) break; /* 查找成功，中止循环 */ else if (x < a[mid]) high = mid - 1; /* 新查找区间为前半段，high前移 */ else low = mid + 1; /* 新查找区间为后半段，low后移 */ } if (low <= high) printf("Index is %d\n", mid); else printf("Not Found\n"); </pre>	<p>根据流程图给出二分查找过程的核心程序段，可让学生课后补充完整。</p>
36	<p>7.2 找出矩阵中最大值所在的位置</p> <p>将1个3*2的矩阵存入1个3*2的二维数组中，找出最大值以及它的行下标和列下标，并输出该矩阵。</p> <p>7.2.1 程序解析</p> <p>7.2.2 二维数组的定义和引用</p> <p>7.2.3 二维数组的初始化</p> <p>7.2.4 使用二维数组编程</p>	<p>引导学生分析题目：</p> <p>这是一个有关矩阵的问题。</p> <p>先考虑：</p> <p>如何存储一个矩阵？</p> <p>如何确定矩阵中的一个元素？</p> <p>如何遍历一个矩阵来找出其中的最大值？从而引出二维数组。</p>

37	<p>7.2.1 程序解析—求矩阵的最大值</p> <p>例 7-7 将1个3*2的矩阵存入1个3*2的二维数组中，找出最大值以及它的行下标和列下标，并输出该矩阵。</p> <p>row 记录最大值的行下标 col 最大值的列下标 a[row][col] 就是最大值</p>	<p>如何记录矩阵中的最大值？ 如何记录矩阵中的最大值及下标？</p>
38	<p>例7-7 源程序</p> <pre> int main(void) { int col, i, j, row; int a[3][2]; printf("Enter 6 integers:\n"); for (i = 0; i < 3; i++) for (j = 0; j < 2; j++) scanf ("%d", &a[i][j]); for (i = 0; i < 3; i++){ for (j = 0; j < 2; j++) printf("%4d", a[i][j]); printf("\n"); } row = col = 0; for (i = 0; i < 3; i++) for (j = 0; j < 2; j++) if (a[i][j] > a[row][col]){ row = i; col = j; } printf ("max = a[%d][%d] = %d\n", row, col, a[row][col]); return 0; } </pre> 	<p>展示、运行例 7-7 程序。 解读程序，可分块说明数组的定义、输入、处理、输出 4 部分。 略过具体的细节，重点说明二维数组 a 的作用，以及和矩阵的对应关系。</p>
39	<p>二维数组</p> <p>多维数组的空间想象</p> <p>一维数组： 一列长表或一个向量 二维数组： 一个表格或一个平面矩阵 三维数组： 三维空间的一个方阵 多维数组： 多维空间的一个数据列阵</p> 	<p>多维数组的空间想象。</p>
40	<p>7.2.2 二维数组的定义和引用</p> <p>1. 定义 类型名 数组名[行长度][列长度]</p> <p>int a[3][2]; 定义1个二维数组 a，3 行 2 列，6 个元素</p> <p>int b[5][10]; 定义1个二维数组 b，5 行 10 列， 50 个元素</p>	<p>重点说明二维数组的定义方法。 提醒： 二维数组必须给出行、列的长度，且长度值也必须是常量表达式。 给出数组的元素个数（即行长度×列长度），计算数组所占的存储空间： 若 int 类型占 2 个字节，则数组 a 占 3×2×2=12 个字节，同理可得到数组 b 所占字节数。</p>

41	<div><div><div></div><div></div><div></div></div><div><div>2. 引用</div><div>先定义，后使用</div><div>数组元素的引用：</div><div>数组名[行下标] [列下标]</div><div>行下标和列下标：整型表达式</div><div>行下标的取值范围是[0，行长度-1]</div><div>列下标的取值范围是[0，列长度-1]</div><div>int a[3][2]; 3 行 2 列， 6 个元素</div><div>a[0][0] a[0][1]</div><div>a[1][0] a[1][1]</div><div>a[2][0] a[2][1]</div><div>下标不要越界</div></div></div>	<p>重点说明如何引用二维数组元素。</p> <p>提醒：</p> <p>引用二维数组的元素必须指定两个下标，即行下标和列下标。</p> <p>行下标和列下标的允许取值范围，不能越界。</p> <p>二维数组元素与矩阵的对应关系。</p>
42	<div><div><div></div><div></div><div></div></div><div><div>二维数组在内存中的存放方式</div><div>int a[3][2];</div><div>3 行 2 列， 6 个元素</div><div>表示1个3行2列的矩阵</div><div>二维数组的元素在内存中按行/列方式存放</div><div>a[0][0] a[0][1]</div><div>a[1][0] a[1][1]</div><div>a[2][0] a[2][1]</div><div>a[0][0]</div><div>a[0][1]</div><div>a[1][0]</div><div>a[1][1]</div><div>a[2][0]</div><div>a[2][1]</div></div></div>	<p>说明二维数组在内存中的存储方式。</p> <p>提醒：</p> <p>二维数组的行、列下标从 0 开始，需要注意与矩阵元素的对应关系。</p>
43	<div><div><div></div><div></div><div></div></div><div><div>7.2.3 二维数组的初始化</div><div>1. 分行赋初值</div><div>int a[3][3] = { { 1, 2, 3 }, { 4, 5, 6 }, { 7, 8, 9 } };</div><div>static int b[4][3] = { { 1, 2, 3 }, { }, { 4, 5 } };</div><div>数组a</div><div>1 2 3</div><div>4 5 6</div><div>7 8 9</div><div>数组b</div><div>1 2 3</div><div>0 0 0</div><div>4 5 0</div><div>0 0 0</div><div>2. 顺序赋初值</div><div>int a[3][3] = { 1, 2, 3, 4, 5, 6, 7, 8, 9 };</div><div>static int b[4][3] = { 1, 2, 3, 0, 0, 0, 4, 5 };</div></div></div>	<p>说明如何在定义二维数组时进行初始化。</p> <p>分行赋初值和顺序赋初值两种方式：</p> <p>根据 ppt 中例子给出一般形式，并说明初始值和二维数组中元素的对应关系。</p> <p>提醒：</p> <p>如果只对部分元素赋初值，要注意初值表中数据的书写顺序，建议采用分行赋初值。</p>
44	<div><div><div></div><div></div><div></div></div><div><div>省略行长度</div><div>对全部元素都赋了初值</div><div>int a[][3] = { 1, 2, 3, 4, 5, 6, 7, 8, 9 };</div><div>或分行赋初值时，在初值表中列出了全部行</div><div>static int b[][3] = { { 1, 2, 3 }, { }, { 4, 5 }, { } }</div><div>数组a</div><div>1 2 3</div><div>4 5 6</div><div>7 8 9</div><div>数组b</div><div>1 2 3</div><div>0 0 0</div><div>4 5 0</div><div>0 0 0</div></div></div>	<p>说明在什么情况下可以省略行长度。</p> <p>建议不要省略。</p>

45	<div><div></div><div>7.2.4 使用二维数组编程</div></div> <p>行下标和列下标分别做为循环变量，通过二重循环，遍历二维数组</p> <p>通常将行下标做为外循环的循环变量 列下标 内循环</p>	<p>在操作二维数组时，如何遍历二维数组的每一个元素？或如何访问二维数组的每一个元素是操作二维数组的关键。</p> <p>说明遍历的方法，即行下标作为外循环控制变量，列下标作为内循环控制变量，用二重循环方式遍历。</p> <p>特别注意下标不能越界。</p>												
46	<div><div></div><div>例7-8 生成一个矩阵并输出</div></div> <p>定义1个 3*2 的二维数组a，数组元素的值由下式给出，按矩阵的形式输出a。</p> <p>$a[i][j] = i + j \quad (0 \leq i \leq 2, 0 \leq j \leq 1)$</p> <pre>int a[3][2]; a[0][0] a[0][1] 0 1 a[1][0] a[1][1] 1 2 a[2][0] a[2][1] 2 3</pre>	<p>说明基本的处理方法。</p>												
47	<div><div></div><div>例7-8 源程序</div></div> <pre>#include <stdio.h> int main(void) { int i, j; int a[3][2]; for (i = 0; i < 3; i++) for (j = 0; j < 2; j++) a[i][j] = i + j; for (i = 0; i < 3; i++){ for (j = 0; j < 2; j++) printf("%4d", a[i][j]); printf("\n"); } return 0; }</pre> <table><tr><td>a[0][0]</td><td>a[0][1]</td><td>0</td><td>1</td></tr><tr><td>a[1][0]</td><td>a[1][1]</td><td>1</td><td>2</td></tr><tr><td>a[2][0]</td><td>a[2][1]</td><td>2</td><td>3</td></tr></table>	a[0][0]	a[0][1]	0	1	a[1][0]	a[1][1]	1	2	a[2][0]	a[2][1]	2	3	<p>展示并运行例 7-8 程序，并模拟单步执行，以查看各下标的变化情况。</p>
a[0][0]	a[0][1]	0	1											
a[1][0]	a[1][1]	1	2											
a[2][0]	a[2][1]	2	3											
48	<div><div></div><div>二维数组的输入</div></div> <p>例7-7中，int a[3][2]; for (i = 0; i < 3; i++) for (j = 0; j < 2; j++) scanf ("%d", &a[i][j]);</p> <div><div>Enter 6 integers: 3 2 10 -9 6 -1 3 2 10 -9 6 -1 max = a[1][0] = 10</div><div><table><tr><td>a[0][0]</td><td>a[0][1]</td></tr><tr><td>a[1][0]</td><td>a[1][1]</td></tr><tr><td>a[2][0]</td><td>a[2][1]</td></tr></table><div>max = a[2][0] = 10</div></div><div><pre>for (j = 0; j < 2; j++) for (i = 0; i < 3; i++) scanf ("%d", &a[i][j]);</pre></div></div>	a[0][0]	a[0][1]	a[1][0]	a[1][1]	a[2][0]	a[2][1]	<p>若内外循环控制交换，查看变化情况。</p>						
a[0][0]	a[0][1]													
a[1][0]	a[1][1]													
a[2][0]	a[2][1]													

49	<div><div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div></div>
----	---

53	<div>  <h3>例7-9 思考</h3> <pre> /* 行列互换 */ for (i = 0; i < n; i++) for (j = 0; j < n; j++){ temp = a[i][j]; a[i][j] = a[j][i]; a[j][i] = temp; } </pre> <div> <div> <div>i=0</div> <div>1 4 7</div> </div> <div> <div>i=1</div> <div>1 2 7</div> </div> <div> <div>i=2</div> <div>1 2 3</div> </div> </div> <div> <div>2 5 6</div> <div>4 5 8</div> <div>4 5 6</div> </div> <div> <div>3 8 9</div> <div>3 6 9</div> <div>7 8 9</div> </div> </div>
----	---

57	<div><div><div>7.3.1 程序解析-判断回文</div></div><pre>int main (void) { int i, k; char line[80]; printf ("Enter a string: "); k = 0; while ((line[k] = getchar()) != '\n') k++; line[k] = '\0'; i = 0; /* i是字符串首字符的下标 */ k = k - 1; /* k是字符串尾字符的下标 */ while (i < k) { if (line[i] != line[k]) break; i++; k--; } if (i >= k) printf("It is a palindrome\n"); else printf("It is not a palindrome\n"); return 0; }</pre><div><div>Enter a string: abcba It is a palindrome</div><div>Enter a string: abcdba It is not a palindrome</div></div></div>	<p>展示、运行例 7-11 程序。</p> <p>解读程序，可分块说明字符数组的定义、输入、处理部分。</p> <p>略过具体的细节，重点说明字符数组 line 的作用、字符数组与字符串的关系、'\0' 的概念。</p>											
58	<div><div><div>7.3.2 一维字符数组</div></div><ul style="list-style-type: none">■ 字符串的存储和运算可以用一维字符数组实现■ 一维字符数组的定义、引用、初始化与其他类型的一维数组一样。<pre>char str[80]; 定义一个含有80个字符型元素的数组str char t[5]={ 'H', 'a', 'p', 'p', 'y' }; 初始化数组 t 输出数组 t 的所有元素 t <table><tr><td>H</td><td>a</td><td>p</td><td>p</td><td>y</td></tr></table> for (i = 0; i < 5; i++) t[0] t[1] t[4] putchar (t[i]);</pre></div>	H	a	p	p	y	<p>说明什么是一维字符数组？</p>						
H	a	p	p	y									
59	<div><div><div>一维字符数组</div></div><pre>char t[5] = { 'H', 'a', 'p', 'p', 'y' }; static char s[6] = { 'H', 'a', 'p', 'p', 'y' }; static char s[6] = { 'H', 'a', 'p', 'p', 'y', 0 }; 0代表字符 '\0'，也就是ASCII码为 0 的字符 static char s[6] = { 'H', 'a', 'p', 'p', 'y', '\0' }; t <table><tr><td>H</td><td>a</td><td>p</td><td>p</td><td>y</td></tr></table> s <table><tr><td>H</td><td>a</td><td>p</td><td>p</td><td>y</td><td>\0</td></tr></table> t[0] t[1] t[4] s[0] s[1] s[5]</pre></div>	H	a	p	p	y	H	a	p	p	y	\0	<p>解释'\0'的特征。</p>
H	a	p	p	y									
H	a	p	p	y	\0								
60	<div><div><div>7.3.3 字符串</div></div><p>字符串常量</p><p>用一对双引号括起来的字符序列</p><p>一个字符串结束符 '\0'</p><p>"Happy"</p><p>字符串结束符</p><p>6个字符 'H' 'a' 'p' 'p' 'y' '\0'</p><p>有效字符</p><p>字符串的有效长度：有效字符的个数</p></div>	<p>重点说明什么是字符串？以及字符串的特征，即结束标志'\0'。</p> <p>提醒：</p> <p>'\0'不是字符串有效字符；</p> <p>字符串的长度是有效字符的个数；</p> <p>字符串常量的表示法，即用双引号括起来。</p>											

61	<div>字符串与一维字符数组</div> <p>字符串：一个特殊的一维字符数组</p> <ul style="list-style-type: none">■ 把字符串放入一维字符数组（存储）■ 对字符串的操作 ==> 对字符数组的操作	<p>说明字符串与一维字符数组的区别。</p> <p>提醒：</p> <p>C 语言将字符串作为一个特殊的一维字符数组来处理。</p>									
62	<div>1. 字符串的存储—数组初始化</div> <p>字符串可以存放在一维字符数组中</p> <pre>static char s[6] = { 'H', 'a', 'p', 'p', 'y', '\0' }; // 字符数组初始化：用字符串常量</pre> <pre>static char s[6] = { "Happy" }; static char s[6] = "Happy";</pre> <p>数组长度 ≥ 字符串的有效长度 + 1</p> <pre>char t[5]; "Happy" 能存入 t 吗？</pre> <div><div>s</div><table><tr><td>H</td><td>a</td><td>p</td><td>p</td><td>y</td><td>\0</td></tr></table><div>s[0] s[1] s[5]</div></div>	H	a	p	p	y	\0	<p>说明定义字符串以及初始化的方法。</p> <p>提醒：</p> <p>一维字符数组的长度至少是字符串长度加 1；</p> <p>字符串必须以'\0'作为结束标志。</p>			
H	a	p	p	y	\0						
63	<div>字符串的存储</div> <pre>auto char s[80] = "Happy";</pre> <p>字符串遇 '\0' 结束</p> <p>第一个 '\0' 前面的所有字符和 '\0' 一起构成了字符串 "Happy"</p> <p>'\0' 之后的其他数组元素与该字符串无关</p> <p>字符串由有效字符和字符串结束符 '\0' 组成</p> <div><div>s</div><table><tr><td>H</td><td>a</td><td>p</td><td>p</td><td>y</td><td>\0</td><td>?</td><td>?</td><td></td></tr></table><div>s[0] s[1] s[5]</div></div>	H	a	p	p	y	\0	?	?		<p>字符串由有效字符和'\0'组成。</p>
H	a	p	p	y	\0	?	?				
64	<div>2. 对字符串的操作</div> <ul style="list-style-type: none">■ 把字符串放入一维字符数组（存储）■ 对字符串的操作 ==> 对字符数组的操作<ul style="list-style-type: none">普通字符数组：数组元素的个数是确定的，一般用下标控制循环字符串：没有显式地给出有效字符的个数，只规定在字符串结束符 '\0' 之前的字符都是字符串的有效字符，一般用结束符 '\0' 来控制循环循环条件：s[i] != '\0'	<p>由于使用一维字符数组来存储字符串，故对字符串的操作就是对字符数组的操作。但有区别，区别在于循环的控制条件，建议重点说明。</p>									

65	<h3>输出字符串</h3> <pre> for (i = 0; s[i] != '\0'; i++) 输出? putchar (s[i]); for (i = 0; i < 80; i++) putchar (s[i]); for (i = 0; i < len; i++) putchar (s[i]); </pre> <div> <div>s</div> <div>H a p p y \0 ? ?</div> <div>s[0] s[1] s[5]</div> </div>	<p>PPT 中列出的三种输出方式，主要区别在于循环条件不同，分析每个循环语句的输出结果，进而得出字符串输出的正确方式。</p>
66	<h3>3. 字符串的存储—赋值和输入</h3> <ul style="list-style-type: none"> ■ 把字符串放入一维字符数组（存储） ■ 对字符串的操作 ==> 对字符数组的操作 <p>存储</p> <ul style="list-style-type: none"> □ 数组初始化 区分"a" 和 'a' static char s[6] = "a"; □ 赋值 s[0] = 'a'; s[1] = '\0'; □ 输入 '\0' 代表空操作，无法输入 输入时，设定一个输入结束符 将输入结束符转换为字符串结束符 '\0' <div> <div>"a" 2 个字符 'a' 和 '\0'</div> <div>'a' 1 个字符常量</div> </div>	<p>字符串的存储可以采用数组初始化、赋值、输入 3 种方法。</p> <p>赋值时必须单个元素进行，不能整体赋值；</p> <p>输入时，由于'\0'无法输入，故须设定一个输入结束符，通常用回车（'\n'）作为输入结束符，再转换成'\0'放入。</p> <p>提醒：</p> <p>“a”和'a'的区别，”a”是字符串常量，包括两个字符，需用字符数组存放；而'a'是字符常量，只有一个字符，用字符变量存放。</p>
67	<h3>7.3.4 使用字符串编程</h3> <p>C语言将字符串作为一个特殊的一维字符数组来处理。</p> <ul style="list-style-type: none"> ■ 存储：把字符串放入一维字符数组 <p>数组初始化、赋值、输入</p> <p>对字符串的操作 ==> 对字符数组的操作</p> <ul style="list-style-type: none"> ■ 对一维字符数组的操作：针对字符串的有效字符和字符串结束符 <p>检测字符串结束符 '\0'</p>	<p>说明字符串和一维数组一样，字符串的处理离不开循环，通常也是用下标作为循环变量，但循环的控制条件将由字符串结束标志'\0'来确定。</p> <p>特别说明这是一种常用的处理方法。</p>
68	<h3>例7-12 统计数字字符个数</h3> <p>输入一个以回车符为结束标志的字符串（少于80个字符），统计其中数字字符'0'.....'9'的个数。</p> <p>分析：</p> <p>数组长度取上限80</p> <p>以 '\n' 做为输入结束符</p>	<p>例 7-12 说明输入的处理方式，特别注意：数组长度以及用回车'\n'作为输入结束符。</p>

69

```
int main(void)
{
    int count, i;
    char str[80];

    printf("Enter a string: ");

    i = 0;
    while ( (str[i] = getchar()) != '\n' )
        i++;

    str[i] = '\0'; /* 如何改变输入结束符?
                   字符串的输入 */

    count = 0;
    for ( i = 0; str[i] != '\0'; i++)
        count++;

    printf("count = %d\n", count); /* 能省略 str[i] != '\0' 吗? */

    return 0;
}
```

Enter a string: It's 512
count = 3

s	i	t	'	s					
0	1	2	3	4	5	6	7	8	9

io ? ?

展示运行例 7-12 程序，详细说明：
虚线框内程序段，即字符串的输入方法以及
如何放到字符数组中，包括'\0'。
能省略语句 `str[i]='\0'` 吗？为什么？
说明如何操作字符串来统计数字字符。

70

例7-13 字符串转换

输入一个以回车符为结束标志的字符串（少于10个字符），提取其中所有的数字字符（'0'.....'9'），将其转换为一个十进制整数输出。

分析:

数组长度取上限10

以 '\n' 做为输入结束符

"123" ==> 123

例 7-13 分析题意，确定数组长度、输入结束符。

71

s 1 2 3 0 ? ? ? ? ? ? ? ? "123" == 123
 0 1 2 3 n = 0;
 for (i = 0; s[i] != '\0'; i++)
 if (s[i] <= '9' && s[i] >= '0')
 n = n * 10 + (s[i] - '0');

i	s[i]	s[i]-'0'	n = n*10+(s[i]-'0')
0	'1'	1	0*10+1 = 1
1	'2'	2	1*10+2 =12
2	'3'	3	12*10+3 =123
3	'\0'		

分析程序段，以字符串 123 的转换为例，解释如何实现将输入的由数字组成的字符串转换成整数的方法及循环过程，注意循环结束条件。

72

```
#include <stdio.h>
int main(void)
{
    int i, number; char s[10];
    printf ("Enter a string: "); // 输入字符串 */
    i = 0;
    while ( (s[i] = getchar()) != '\n' )
        i++;
    s[i] = '\0';

    n = 0;
    for ( i = 0; s[i] != '\0'; i++)
        if ( s[i] <= '9' && s[i] >= '0' )
            n = n * 10 + (s[i] - '0');
    printf ("digit = %d\n", n);
    return 0;
}
```

Enter a string: **a123d**
digit = 123

a	1	2	d	3	?	?
0	1	2	3	4		

展示运行例 7-13 源程序。

先输入字符串：123，查看结果，解释见
上页 PPT；

再输入字符串 a12d3，查看结果，解释见下页 PPT。

73

77	<div><div>转换为十进制整数</div><pre>"1a0b" ==> 6667 hexad number number = 0; /* 存放十进制数, 先清0 */ for (i = 0; hexad[i] !='\0'; i++) { /* 逐个转换 */ if (hexad[i] >= '0' && hexad[i] <= '9') number = number * 16 + hexad[i] - '0'; else if (hexad[i] >= 'A' && hexad[i] <= 'F') number = number * 16 + hexad[i] - 'A' + 10; else if (hexad[i] >= 'a' && hexad[i] <= 'f') number = number * 16 + hexad[i] - 'a' + 10; }</pre></div>	<p>再解决第 2 个难点问题——转换为十进制整数。</p> <p>分析程序段，将十六进制字符串 hexad 转换为十进制整数。</p> <p>其中对数字字符转换的解释见例 7-13；重点解释对大小写英文字母的转换方法。</p>									
78	<div><div><div><div>输入原字符串str</div><div>↓</div><div>滤去非16进制字符后生成新字符串hexad</div><div>↓</div><div>把字符串hexad转换成十进制整数number</div></div><div><pre>printf("Enter a string: "); i = 0; while ((str[i] = getchar()) != '#') i++; str[i] = '\0'; k = 0; for(i = 0; str[i] != '\0'; i++) if(str[i]>='0'&&str[i]<='9' str[i]>='a'&&str[i]<='f' str[i]>='A'&&str[i]<='F'){ hexad[k] = str[i]; k++; } hexad[k] = '\0'; number = 0; for(i = 0; hexad[i] !='\0'; i++){ if(hexad[i] >= '0' &&hexad[i] <= '9') number = number * 16 + hexad[i] - '0'; else if(hexad[i] >= 'A' &&hexad[i] <= 'F') number = number * 16 + hexad[i] - 'A' + 10; else if(hexad[i] >= 'a' &&hexad[i] <= 'f') number = number * 16 + hexad[i] - 'a' + 10; }</pre></div><div>程序段</div></div></div>	<p>对本例的 3 个主要步骤，根据前面的分析和程序段实现，进行汇总。</p> <p>根据程序运行过程的分析，给出运行流程，并分析每一块的功能，从而给出每一块的程序段。</p> <p>说明这是解决复杂问题的常用方法。</p>									
79	<div><div>字符串小结</div><p>字符串：一个特殊的一维字符数组 '\0'</p><p>■ 把字符串放入一维字符数组（存储）</p><p>数组长度足够</p><ul style="list-style-type: none">■ 字符数组初始化： static char s[80] = "Happy";■ 赋值： s[0] = 'a'; s[1] = '\0';■ 输入： 输入结束符 ==> 字符串结束符 '\0'<pre>i = 0; while ((s[i]=getchar()) != '\n') i++; s[i] = '\0';</pre><table><tr><td>s</td><td>H</td><td>a</td><td>p</td><td>p</td><td>y</td><td>\0</td><td>?</td><td>?</td></tr></table><p>s[0] s[1] s[5]</p></div>	s	H	a	p	p	y	\0	?	?	<p>总结字符串的相关知识。</p> <p>强调字符串在程序设计中的重要性。</p>
s	H	a	p	p	y	\0	?	?			
80	<div><ul style="list-style-type: none">■ 把字符串放入一维字符数组（存储）■ 对字符串的操作 ==> 对字符数组的操作<p>只针对字符串的有效字符和字符串结束符 '\0'</p><pre>for (i = 0; s[i] != '\0'; i++) putchar(s[i]);</pre><table><tr><td>s</td><td>H</td><td>a</td><td>p</td><td>p</td><td>y</td><td>\0</td><td>?</td><td>?</td></tr></table><p>s[0] s[1] s[5]</p></div>	s	H	a	p	p	y	\0	?	?	
s	H	a	p	p	y	\0	?	?			

本章总结

- 一维数组：
 - 定义、初始化、引用
 - 使用一维数组：选择排序
- 二维数组
 - 定义、初始化、引用
 - 使用二维数组：矩阵
- 字符串
 - 字符数组与字符串
 - 字符串的存储
 - 字符串的操作
- 使用数组进行程序设计

• 正确理解数组的基本概念及在内存中的存放方式；
 • 掌握使用一维数组编写程序；
 • 掌握使用二维数组编写程序；
 • 正确理解字符串的概念，掌握使用字符串编写程序；
 • 能合理运用数组进行程序设计，熟练掌握几个常用的算法；

7.3 练习与习题参考答案

7.3.1 练习参考答案

练习 7-1 将例 7-3 程序中的 break 语句去掉，输出结果有变化吗？假设输入数据不变，输出什么？

解答：

如果将 break 语句去掉，当在数组 a 中找到与 x 值相同的元素并输出其下标后，继续查找并输出与 x 值相同的元素。即查找并输出与 x 值相同的所有元素的下标。

当输入数据仍为 2 9 8 1 9 时，输出将是

Index is 1

Index is 4

练习 7-2 求最大值及其下标。输入一个正整数 n ($1 < n \leq 10$)，再输入 n 个整数，输出最大值及其对应的最小下标，下标从 0 开始。试编写相应程序。

解答：

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

```
int main()
```

```
{
```

```
    int i, n, max, maxi;
```

```
    int a[10];
```

```
    scanf("%d", &n);
```

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

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

```
    }
```

```
    max = a[0]; maxi = 0;
```

```
    for (i = 1; i < n; i++) {
```

```
        if (a[i] > max) {
```

```
            max = a[i]; maxi = i;
```

```
        }
```

```

    }
    printf("%d %d\n", max, maxi);

    return 0;
}

```

练习 7-3 将数组中的数逆序存放。输入一个正整数 n ($1 < n \leq 10$)，再输入 n 个整数，存入数组 a 中，先将数组 a 中的这 n 个数逆序存放，再按顺序输出数组 a 中的 n 个元素。试编写相应程序。

解答 1:

```

#include <stdio.h>

int main()
{
    int i, high, low, n, temp;
    int num[10];

    scanf("%d", &n);
    for(i = 0; i < n; i++)
        scanf("%d", &num[i]);
    low = 0, high = n - 1;
    while(low <= high) {
        temp = num[high];
        num[high] = num[low];
        num[low] = temp;
        low++;
        high--;
    }
    for(i = 0; i < n; i++) {
        if(i == 0)
            printf("%d", num[i]);
        else
            printf(" %d", num[i]);
    }
    printf("\n");

    return 0;
}

```

解答 2:

```

#include <stdio.h>

int main()
{
    int i, n, temp;
    int num[10];

```

```

scanf("%d", &n);
for(i = 0; i < n; i++)
    scanf("%d", &num[i]);
for(i = 0; i < n/2; i++){
    temp = num[i];
    num[i] = num[n-1-i];
    num[n-1-i] = temp;
}
for(i = 0; i < n; i++) {
    if(i == 0)
        printf("%d", num[i]);
    else
        printf(" %d", num[i]);
}
printf("\n");

return 0;
}

```

练习 7-4 找出不是两个数组共有的元素。输入一个正整数 n ($1 < n \leq 10$)，再输入 n 个整数，存入第 1 个数组中；然后输入一个正整数 m ($1 < m \leq 10$)，再输入 m 个整数，存入第 2 个数组中，找出所有不是这两个数组共有的元素。试编写相应程序。

解答：

```

#include <stdio.h>
#define MAXN 20
int main()
{
    int i, j, flag, n1, n2;
    int a1[MAXN], a2[MAXN];
    static int v1[MAXN], v2[MAXN];

    scanf("%d", &n1);
    for (i = 0; i < n1; i++) {
        scanf("%d", &a1[i]);
    }
    scanf("%d", &n2);
    for (i = 0; i < n2; i++) {
        scanf("%d", &a2[i]);
    }

    for (i = 0; i < n1; i++) {
        if (v1[i] == 0) {
            for (j = i + 1; j < n1; j++) {
                if (a1[j] == a1[i]) {

```



```

        v1[j] = 1;
    }
}
flag = 0;
for (j = 0; j < n2; j++) {
    if (a2[j] == a1[i]) {
        v2[j] = 1;
        flag = 1;
    }
}
if(flag != 0){
    v1[i] = 1;
}
}
for (i = 0; i < n2; i++) {
    if (v2[i] == 0) {
        for (j = i + 1; j < n2; j++) {
            if (a2[j] == a2[i]) {
                v2[j] = 1;
            }
        }
        flag = 0;
        for (j = 0; j < n1; j++) {
            if (a1[j] == a2[i]) {
                v1[j] = 1;
                flag = 1;
            }
        }
        if(flag != 0){
            v2[i] = 1;
        }
    }
}
flag = 0;
for (i = 0; i < n1; i++) {
    if (v1[i] == 0) {
        if (flag == 0)
            printf("%d", a1[i]);
        else {
            printf(" %d", a1[i]);
        }
        flag = 1;
    }
}

```

```

    }

    for (i = 0; i < n2; i++) {
        if (v2[i] == 0) {
            if (flag == 0)
                printf("%d", a2[i]);
            else {
                printf(" %d", a2[i]);
            }
            flag = 1;
        }
    }
    printf("\n");

    return 0;
}

```

练习 7-5 给二维数组赋值时，如果把列下标作为外循环的循环变量，行下标作为内循环的循环变量，输入的数据在二维数组中如何存放？用下列 for 语句替换例 7-7 中的对应语句，将输入的 6 个数存入二维数组中，假设输入数据不变，输出什么？与例 7-7 中的输出结果一样吗？为什么？

```

for( j=0;j<2;j++)
    for( i=0;i<3;i++)
        scanf("%d",&a[i][j]);

```

解答：

当把列下标作为外循环的循环变量，行下标作为内循环的循环变量时，输入的数据将以列优先的方式存放。当用上述 for 循环方式时，输出结果为：max=a[2][0]=10，与原例 7-7 不一样，因为当用上述方式输入是，二维数组中存放值如下：

```

3    -9
2     6
10   -1

```

练习 7-6 在例 7-9 的程序中，如果将遍历上三角矩阵改为遍历下三角矩阵，需要怎样修改程序？运行结果有变化吗？如果改为遍历整个矩阵，需要怎样修改程序？输出是什么？为什么？

解答：

只需按要求修改矩阵的输出部分，方法如下，其运行结果不变。

```

for(i = 0; i < n; i++)
    for(j = 0; j < i; j++) {
        temp = a[i][j];
        a[i][j] = a[j][i];
        a[j][i] = temp;
    }

```

若修改为遍历整个程序，方法如下，则运行结果仍将输出原矩阵，无法达到转置要求，

原因是矩阵中每个元素相应被交换了 2 次。

```
for(i = 0; i < n; i++)
    for(j = 0; j < n; j++) {
        temp = a[i][j];
        a[i][j] = a[j][i];
        a[j][i] = temp;
    }
```

练习 7-7 矩阵运算。读入一个正整数 $n(1 \leq n \leq 6)$ ，再读入 n 阶方阵 a ，计算该矩阵除副对角线、最后一列和最后一行以外的所有元素之和。副对角线为从矩阵的右上角至左下角的连线。试编写相应程序。

解答：

```
#include <stdio.h>
#define MAXN 10
int main()
{
    int i, j, n, sum;
    int a[MAXN][MAXN];

    scanf("%d", &n);
    for (i = 0; i < n; ++i){
        for (j = 0; j < n; ++j){
            scanf("%d", &a[i][j]);
        }
    }
    sum = 0;
    for (i = n-1; i >= 0; --i){
        for (j = n-1; j >= 0; --j){
            if ((i != n-1) && (j != n-1) && (i+j) != n-1) {
                sum += a[i][j];
            }
        }
    }

    printf("%d\n", sum);

    return 0;
}
```

练习 7-8 方阵循环右移。读入 2 个正整数 m 和 $n(1 \leq n \leq 6)$ ，再读入 n 阶方阵 a ，将该方阵中的每个元素循环向右移 m 个位置，即将第 $0、1、\dots、n-1$ 列变换为第 $n-m、n-m+1、\dots、n-1、0、1、\dots、n-m-1$ 列，移动后的方阵可以存到另一个二维数组中。试编写相应程序。

解答 1：

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

```

#define MAXN 6
int main()
{
    int i, j, m, n;
    int a[MAXN][MAXN], b[MAXN][MAXN];

    scanf("%d%d", &m, &n);
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            scanf("%d", &a[i][j]);
        }
    }
    m = m % n;
    for (j = 0; j < n; j++) {
        for (i = 0; i < n; i++) {
            b[i][(j+m) % n] = a[i][j];
        }
    }
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            printf("%d ", b[i][j]);
        }
        printf("\n");
    }

    return 0;
}

```

解答 2：斜体部分也可以写为：

```

    for (j = 0; j < n; j++) {
        for (i = 0; i < n; i++) {
            if (j < m){
                b[i][j] = a[i][n - m + j];
            }
            else{
                b[i][j] = a[i][j - m];
            }
        }
    }
}

```

练习 7-9 计算天数。输入日期（年、月、日），计算其是该年的第几天。试编写相应程序，要求调用例 7-10 中定义的函数 `day_of_year(year, month, day)`。

解答 1：

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

```

int day_of_year(int year, int month, int day);
int main()
{
    int year, month, day, day_year;

    scanf("%d/%d/%d", &year, &month, &day);
    day_year = day_of_year(year, month, day);
    printf("%d\n", day_year);

    return 0;
}
int day_of_year(int year, int month, int day)
{
    int k, leap;
    int tab[2][13] = {
        {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31},
        {0, 31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31}
    };

    leap = (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0);
    for(k = 1; k < month; k++){
        day = day + tab[leap][k];
    }
    return day;
}

```

解答 2:

```

#include <stdio.h>
int main()
{
    int day, leap, month, year;

    scanf("%d/%d/%d", &year, &month, &day);
    if ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0)){
        leap = 1;
    }
    else {
        leap = 0;
    }
    switch (month) {
        case 12: day += 30;
        case 11: day += 31;
        case 10: day += 30;
        case 9: day += 31;
        case 8: day += 31;
    }
}

```

```

        case 7: day += 30;
        case 6: day += 31;
        case 5: day += 30;
        case 4: day += 31;
        case 3: day += (28+leap);
        case 2: day += 31;
        case 1: break;
        default: break;
    }
    printf("%d\n", day);

    return 0;
}

```

练习 7-10 查找指定字符。输入一个字符，再输入一个以回车结束的字符串(少于 80 个字符)，在字符串中查找该字符。如果找到，则输出该字符在字符串中所对应的最大下标，下标从 0 开始；否则输出 “Not Found”。试编写相应程序。

解答：

```

#include <stdio.h>
#define MAXLEN 81
int main(void)
{
    char cc, ch;
    char str[MAXLEN];
    int flag, i, index;

    cc = getchar();
    getchar();
    i = 0;
    while((ch = getchar()) != '\n'){
        str[i] = ch;
        i++;
    }
    str[i] = '\0';

    flag = index = 0;
    for(i = 0; str[i] != '\0'; i++) {
        if(str[i] == cc) {
            flag = 1;
            index = i;
        }
    }

    if(flag != 0)

```

```

        printf("index = %d\n", index);
    else
        printf("Not Found\n");

    return 0;
}

```

练习 7-11 字符串逆序。输入一个以回车结束的字符串（少于 80 个字符），将该字符串逆序存放，输出逆序后的字符串。试编写相应程序。

解答：

```

#include <stdio.h>
#define MAXLEN 81
int main()
{
    char temp;
    char str[MAXLEN];
    int i, j;

    i = 0;
    while ((str[i] = getchar()) != '\n') {
        i++;
    }
    str[i] = '\0';

    j = i - 1;
    i = 0;
    while (i < j) {
        temp = str[i];
        str[i] = str[j];
        str[j] = temp;
        i++;
        j--;
    }

    printf("%s\n", str);

    return 0;
}

```

7.3.2 习题参考答案

一. 选择题

1	2	3	4	5	6	7	8
---	---	---	---	---	---	---	---

D	D	D	C	C	C	C	B
---	---	---	---	---	---	---	---

二. 填空题

- 未知
- (1) $i=1$
(2) $x[i-1]$
- 12
- (1) 2
(2) 3
- (1) $j \geq 1$
(2) i
- (1) $a[i][j] \neq a[j][i]$
(2) $\text{found} = 1;$
(3) $\text{found} \neq 0;$

三. 程序设计题

- 选择法排序。输入一个正整数 n ($1 < n \leq 10$)，再输入 n 个整数，将它们从大到小排序后输出。试编写相应程序。

解答：

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

```
int main()
```

```
{
```

```
    int i, index_max, k, max, n, temp;
```

```
    int a[10];
```

```
    scanf("%d", &n);
```

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

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

```
    for(k = 0; k < n; k++){
```

```
        max = a[k];
```

```
        index_max = k;
```

```
        for(i = k + 1; i < n; i++){
```

```
            if(a[i] > max){
```

```
                max = a[i];
```

```
                index_max = i;
```

```
            }
```

```
        }
```

```
        temp = a[k];
```

```
        a[k] = a[index_max];
```

```
        a[index_max] = temp;
```

```
    }
```

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

```
        if(i == 0)
```

```
            printf("%d", a[i]);
```



```

        else
            printf(" %d", a[i]);
    }
    printf("\n");

    return 0;
}

```

2. 求一批整数中出现最多的数字。输入一个正整数 n ($1 < n \leq 1000$)，再输入 n 个整数，分析每个整数的每一位数字，求出现次数最多的数字。例如输入 3 个整数 1234、2345、3456，其中出现次数最多的数字是 3 和 4，均出现了 3 次。试编写相应程序。

解答：

```

#include <stdio.h>
int main()
{
    int i, max, n, number;
    static int cnt[10];

    scanf("%d", &n);
    while(n > 0){
        scanf("%d", &number);
        do{
            cnt[number%10]++;
            number /= 10;
        }while(number != 0);
        n--;
    }
    max = cnt[0];
    for(i = 1; i < 10; i++){
        if(cnt[i] > max){
            max = cnt[i];
        }
    }
    printf("%d:", max);
    for (i = 0; i < 10; i++){
        if (cnt[i] == max) printf(" %d", i);
    }
    printf("\n");

    return 0;
}

```

3. 判断上三角矩阵。输入一个正整数 n ($1 \leq n \leq 6$) 和 n 阶方阵 a 中的元素，如果 a 是上三角矩阵，输出“YES”，否则，输出“NO”。上三角矩阵指主对角线以下的元素都为 0 的矩阵，主

对角线为从矩阵的左上角至右下角的连线。试编写相应程序。

解答：

```
#include <stdio.h>
#define MAXN 10
int main()
{
    int flag, i, j, k, n, t;
    int a[MAXN][MAXN];

    scanf("%d", &t);
    for (k = 0; k < t; k++) {
        scanf("%d", &n);
        for (i = 0; i < n; i++){
            for (j = 0; j < n; j++){
                scanf("%d", &a[i][j]);
            }
        }
        flag = 1;
        for (i = 0; i < n; i++){
            for (j = 0; j < i; j++){
                if(a[i][j] != 0){
                    flag = 0;
                }
            }
        }
        if(flag != 0) {
            printf("YES\n");
        }
        else {
            printf("NO\n");
        }
    }

    return 0;
}
```

4. 求矩阵各行元素之和。输入 2 个正整数 m 和 n ($1 \leq m \leq 6, 1 \leq n \leq 6$)，然后输入矩阵 a (m 行 n 列) 中的元素，分别求出各行元素之和，并输出。试编写相应程序。

解答 1：

```
#include <stdio.h>
#define MAXN 6
int main()
{
    int i, j, m, n, sum;
```

```

int a[MAXN][MAXN];

scanf("%d %d",&m, &n);
for(i = 0; i < m; i++){
    for(j = 0; j < n; j++){
        scanf("%d", &a[i][j]);
    }
}
for(i = 0; i < m; i++){
    sum = 0;
    for(j = 0; j < n; j++){
        sum = sum + a[i][j];
    }
    printf("%d\n", sum);
}

return 0;
}

```

解答 2:

```

#include <stdio.h>
int main()
{
    int a, i, j, m, n, sum;

    scanf("%d %d", &m, &n);
    for (i = 0; i < m; i++) {
        sum = 0;
        for (j = 0; j < n; j++) {
            scanf("%d", &a);
            sum += a;
        }
        printf("%d\n", sum);
    }

    return 0;
}

```

5. 找鞍点。输入 1 个正整数 n ($1 \leq n \leq 6$) 和 n 阶方阵 a 中的元素，假设方阵 a 最多有 1 个鞍点，如果找到 a 的鞍点，就输出其下标，否则，输出"NO"。鞍点的元素值在该行上最大，在该列上最小。试编写相应程序。

解答:

```

# include <stdio.h>
# define MAXN 6
int main()

```

```

{
    int i, j, flag, n;
    int a[MAXN][MAXN], maxr[MAXN], minc[MAXN];

    scanf("%d", &n);
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            scanf("%d", &a[i][j]);
        }
    }
    for (i = 0; i < n; i++) {
        maxr[i] = a[i][0];
        minc[i] = a[0][i];
    }
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            if (a[i][j] > maxr[i]) maxr[i] = a[i][j];
            if (a[i][j] < minc[j]) minc[j] = a[i][j];
        }
    }
    flag = 0;
    for (i = 0; i < n; i++){
        for (j = 0; j < n; j++){
            if ((a[i][j]==maxr[i]) && (a[i][j]==minc[j])) {
                printf("%d %d\n", i, j);
                flag = 1;
                break;
            }
        }
    }
    if (flag == 0){
        printf("NONE\n");
    }

    return 0;
}

```

6. 统计大写辅音字母。输入一个以回车结束的字符串（少于 80 个字符），统计并输出其中大写辅音字母的个数。大写辅音字母：除 'A', 'E', 'I', 'O', 'U' 以外的大写字母。试编写相应程序。

解答 1:

```

#include <stdio.h>
int check (char ch);
int main()

```

```

{
    char ch;
    int cnt = 0;

    scanf("%c", &ch);
    while (ch != '\n') {
        cnt += check(ch);
        scanf("%c", &ch);
    }
    printf("%d\n", cnt);

    return 0;
}

```

```

int check (char ch)
{
    int ret;

    if ( ch < 'A' || ch > 'Z' ) {
        return 0;
    }
    switch (ch) {
    case 'A':
    case 'E':
    case 'I':
    case 'O':
    case 'U': ret = 0; break;
    default: ret = 1; break;
    }

    return ret;
}

```

解答 2:

```

#include <stdio.h>
# define MAXLEN 81
int main()
{
    char ch;
    char str[MAXLEN];
    int cnt, i;

    i = 0;
    while ( (str[i] = getchar()) != '\n' ){

```

```

        i++;
    }
    str[i] = '\0';

    cnt = 0;
    for( i = 0; str[i] != '\0'; i++ ) {
        if( str[i] >= 'A' && str[i] <= 'Z' ){
            if( (str[i] != 'A') && (str[i] != 'E') && (str[i] != 'T') && (str[i] != 'O') && (str[i] != 'U') ){
                cnt++;
            }
        }
    }
    printf("%d\n", cnt);

    return 0;
}

```

7. 字符串替换。输入一个以回车结束的字符串（少于 80 个字符），将其中的大写字母用下面列出的对应大写字母替换，其余字符不变，输出替换后的字符串。试编写相应程序。

解答 1:

```

#include <stdio.h>
char replace ( char ch );
int main()
{
    char ch;

    scanf("%c", &ch);
    while ( ch != '\n' ) {
        printf("%c", replace(ch));
        scanf("%c", &ch);
    }
    printf("\n");

    return 0;
}
char replace ( char ch )
{
    if ( ch < 'A' || ch > 'Z'){
        return ch;
    }
    else{
        return ( 'A'+ 'Z' - ch );
    }
}

```

解答 2:

```
#include <stdio.h>
# define MAXLEN 81
int main()
{
    char str[MAXLEN];
    int i;

    i = 0;
    while ( (str[i] = getchar()) != '\n' ){
        i++;
    }
    str[i] = '\0';

    for( i = 0; str[i] != '\0'; i++ ) {
        if( str[i] >= 'A' && str[i] <= 'Z' ){
            str[i] = 'A' + 'Z' - str[i];
        }
    }

    printf ( "%s\n", str );

    return 0;
}
```

8. 字符串转换成十进制整数。输入一个以#结束的字符串，滤去所有的非十六进制字符（不分大小写），组成一个新的表示十六进制数字的字符串，然后将其转换为十进制数后输出。如果过滤后字符串的首字符为“-”，代表该数是负数。试编写相应程序。

解答:

```
#include <stdio.h>
# define MAXLEN 81
int main()
{
    char str[MAXLEN];
    int digit, i, j, flag, number;

    i = 0;
    while ( (str[i] = getchar()) != '\n' ){
        i++;
    }
    str[i] = '\0';

    flag = 1;
```

```

j = 0;
for ( i = 0; str[i] != '\0'; i++ ) {
    if ( (str[i] >= '0' && str[i] <= '9') || (str[i] >= 'A' && str[i] <= 'F') || (str[i] >= 'a' && str[i] <= 'f') ) {
        str[j] = str[i];
        j++;
    }
    else if ( str[i] == '-' && j == 0 && flag == 1){
        flag = -1;
    }
}
str[j] = '\0';

number = 0;
for ( i = 0; str[i] != '\0'; i++ ) {
    if ( str[i] >= '0' && str[i] <= '9' ) {
        digit = str[i] - '0';
    }
    else if ( str[i] >= 'A' && str[i] <= 'F' ) {
        digit = str[i] - 'A' + 10;
    }
    else if ( str[i] >= 'a' && str[i] <= 'f' ) {
        digit = str[i] - 'a' + 10;
    }
    number = number * 16 + digit;
}

if ( number == 0 ) {
    printf("%d\n", number);
}
else {
    printf("%d\n", flag * number);
}

return 0;
}

```

7.4 实验指导教材参考答案

7.4.1 一维数组

一. 调试示例

简化的插入排序：输入一个正整数 n ($0 < n < 9$) 和一组 (n 个) 有序的整数，再输入一个整数 x ，把 x 插入到这组数据中，使该组数据仍然有序。

解答 1:


```

#include <stdio.h>
#define MAXN 10
int main()
{
    int i, j, n, x;
    int a[MAXN];

    scanf("%d", &n);
    for (i = 0; i < n; i++){
        scanf("%d", &a[i]);
    }
    scanf("%d", &x);
    for (i = 0; i < n; i++) {
        if(x < a[i]){
            break;
        }
    }
    for(j = n-1; j >= i; j--){
        a[j+1] = a[j];
    }
    a[i] = x;
    for (i = 0; i <= n; i++){
        printf("%d ", a[i]);
    }
    printf("\n");

    return 0;
}

```

解答 2:

```

#include <stdio.h>
#define MAXN 10
int main()
{
    int i, n, x;
    int a[MAXN];

    scanf("%d", &n);
    for (i = 0; i < n; i++){
        scanf("%d", &a[i]);
    }
    scanf("%d", &x);
    for (i = n-1; i >= 0 && x < a[i]; i--) {
        a[i+1] = a[i];
    }
}

```

```

    }
    a[i+1] = x;
    for (i = 0; i <= n; i++){
        printf("%d ", a[i]);
    }
    printf("\n");

    return 0;
}

```

二、基础编程题

(1) 求最大值及其下标：输入一个正整数 n ($1 < n \leq 10$)，再输入 n 个整数，输出最大值及其对应的最小下标，下标从 0 开始。试编写相应程序。

解答：参见练习 7-2。

习题程序设计第 7 题。

(2) 将数组中的数逆序存放：输入一个正整数 n ($1 < n \leq 10$)，再输入 n 个整数，存入数组 a 中，先将数组 a 中的这 n 个数逆序存放，再按顺序输出数组 a 中的 n 个元素。试编写相应程序。

解答：参见练习 7-3。

(3) 找出不是两个数组共有的元素：输入一个正整数 n ($1 < n \leq 10$)，再输入 n 个整数，存入第 1 个数组中；然后输入一个正整数 m ($1 < m \leq 10$)，再输入 m 个整数，存入第 2 个数组中，找出所有不是这两个数组共有的元素。试编写相应程序。

解答：参见练习 7-4。

(4) 选择法排序：输入一个正整数 n ($1 < n \leq 10$)，再输入 n 个整数，将它们从大到小排序后输出。试编写相应程序。

解答：参见习题程序设计第 1 题。

(5) 求一批整数中出现最多的数字：输入一个正整数 n ($1 < n \leq 1000$)，再输入 n 个整数，分析每个整数的每一位数字，求出现次数最多的数字。例如输入 3 个整数 1234、2345、3456，其中出现次数最多的数字是 3 和 4，均出现了 3 次。试编写相应程序。

解答：参见习题程序设计第 2 题。

三、改错题

查找整数：设 a 是一个整型数组， n 和 x 都是整数，数组 a 中各元素的值互异。在数组 a 的元素中查找与 x 相同的元素，如果找到，输出 x 在数组 a 中的下标位置；如果没有找到，输出“没有找到与 x 相同的元素!”。

解答：

改错汇总：

错误行号： 11 正确语句： scanf("%d", &a[i]);

错误行号： 15 正确语句： if(a[i]==x) break

错误行号： 16 正确语句： if(i>=n)

include <stdio.h>

```

#define MAXN 20
int main(void)
{
    int i, flag, n, x;
    int a[MAXN];

    scanf("%d%d", &n, &x);
    for(i = 0; i < n; i++)
        scanf("%d", &a[i]);
    flag = 0;
    for(i = 0; i < n; i++) {
        if(a[i] == x){
            flag = 1;
            break;
        }
    }
    if(flag == 0) {
        printf("Not Found\n");
    }
    else{
        printf("%d\n", i);
    }

    return 0;
}

```

四、拓展编程题

(1) 输出数组元素：输入 1 个正整数 n ($1 < n \leq 10$)，再输入数组 a 的 n 个整数，把 a 中所有的后项减前项之差存入数组 b ，并按每行 3 个元素的格式输出数组元素 b 。

解答：

```

#include <stdio.h>
#define MAXN 10
int main(void)
{
    int i, n;
    int a[MAXN], b[MAXN];

    scanf("%d", &n);
    for(i = 0; i < n; i++){
        scanf("%d", &a[i]);
    }
    for(i = 0; i < n-1; i++){
        b[i] = a[i+1] - a[i];
    }
}

```

```

for(i = 0; i < n-1; i++) {
    if ((i+1) % 3 == 1){
        printf("%d", b[i]);
    }
    else{
        printf(" %d", b[i]);
    }
    if((i+1) % 3 == 0){
        printf("\n");
    }
}
if ((n-1) % 3 != 0){
    printf("\n");
}

return 0;
}

```

(2) 数字加密：输入一个四位数，将该数每一位上的数字加 9，然后除以 10 取余，作为该位上的新数字，最后将千位和十位上的数字互换，百位和个位上的数字互换，组成加密后的新四位数。例如输入 1257，经过加 9 取余后得到新数字 0146，再经过两次换位后得到 4601。试编写相应程序。

解答 1：

```

#include <stdio.h>
# define MAXN 4
int main(void)
{
    int i, newnum, number, temp;
    int digit[MAXN];

    scanf("%d", &number);
    for(i = 0; i < MAXN; i++){
        digit[i] = (number%10 + 9) % 10;
        number = number / 10;
    }
    temp = digit[0];
    digit[0] = digit[2];
    digit[2] = temp;
    temp = digit[1];
    digit[1] = digit[3];
    digit[3] = temp;
    printf("The encrypted number is ");
    for(i = MAXN-1; i >= 0; i--){
        printf("%d", digit[i]);
    }
}

```

```

    }
    printf("\n");

    return 0;
}

```

解答 2:

```

#include <stdio.h>
#define MAXN 4
int main()
{
    int digit[MAXN], i;
    char c;

    for (i = 0; i < MAXN; i++) {
        scanf("%c", &c);
        digit[i] = c - '0';
        digit[i] = (digit[i] + 9) % 10;
    }
    printf("The encrypted number is %d%d%d%d\n", digit[2], digit[3], digit[0], digit[1]);

    return 0;
}

```

(3) 交换最小值和最大值: 输入一个正整数 n ($1 < n \leq 10$), 再输入 n 个整数, 将最小值与第一个数交换, 最大值与最后一个数交换, 然后输出交换后的 n 个数。试编写相应程序。

解答 1:

```

#include <stdio.h>
#define MAXN 10
int main()
{
    int i, min, max, mini, maxi, n, temp;
    int a[MAXN];

    scanf("%d", &n);
    for (i = 0; i < n; i++){
        scanf("%d", &a[i]);
    }
    min = a[0];
    mini = 0;
    for (i = 1; i < n; i++) {
        if (a[i] < min) {
            min = a[i]; mini = i;
        }
    }
}

```

```

temp = a[0]; a[0] = min; a[mini] = temp;
max = a[0];
maxi = 0;
for (i = 1; i < n; i++) {
    if (a[i] > max) {
        max = a[i]; maxi = i;
    }
}
temp = a[n-1]; a[n-1] = max; a[maxi] = temp;
for (i = 0; i < n; i++){
    printf("%d ", a[i]);
}
printf("\n");

return 0;
}

```

解答 2:

```

#include <stdio.h>
#define MAXN 10
int main()
{
    int i, min, max, mini, maxi, n, temp;
    int a[MAXN];

    scanf("%d", &n);
    for (i = 0; i < n; i++){
        scanf("%d", &a[i]);
    }
    min = max = a[0];
    mini = maxi = 0;
    for (i = 1; i < n; i++) {
        if (a[i] < min) {
            min = a[i]; mini = i;
        }
        if (a[i] > max) {
            max = a[i]; maxi = i;
        }
    }
    temp = a[0]; a[0] = min; a[mini] = temp;
    if (maxi == 0) maxi = mini;
    temp = a[n-1]; a[n-1] = max; a[maxi] = temp;
    for (i = 0; i < n; i++){
        printf("%d ", a[i]);
    }
}

```

```

printf("\n");

return 0;
}

```

(4) 求整数序列中出现次数最多的数：要求统计一个整型序列中出现次数最多的整数及其出现次数。试编写相应程序。

解答：

```

#include <stdio.h>
#define MAXN 1000
int main()
{
    int cnt, i, j, max, maxcnt, n;
    int a[MAXN], v[MAXN];

    scanf("%d", &n);
    for (i = 0; i < n; i++) {
        scanf("%d", &a[i]);
        v[i] = 0;
    }
    maxcnt = 1; max = a[0];
    for (i = 0; i < n; i++) {
        if (v[i] == 0) {
            v[i] = 1;
            cnt = 1;
            for (j = i + 1; j < n; j++)
                if (a[j] == a[i]) {
                    cnt++;
                    v[j] = 1;
                }
            if (cnt > maxcnt) {
                maxcnt = cnt;
                max = a[i];
            }
        }
    }
    printf("%d %d\n", max, maxcnt);

    return 0;
}

```

(5) 组个最小数：给定数字 0-9 各若干个，你可以以任意顺序排列这些数字，但必须全部使用。目标是使得最后得到的数尽可能小（注意 0 不能做首位）。试编写相应程序。

解答：

```

#include <stdio.h>
void print(int n, int x) {
    for (; n > 0; --n) {
        printf("%d",x);
    }
}
int main()
{
    int i;
    int a[10];

    for (i = 0; i < 10; ++i) {
        scanf("%d", &a[i]);
    }
    for (i = 1; a[i] == 0; ++i)
        ;
    printf("%d", i);
    --a[i];
    for (i = 0; i < 10; ++i) {
        print(a[i], i);
    }
    printf("");

    return 0;
}

```

(6) 装箱问题: 假设有 n 项物品, 大小分别为 $s_1, s_2, \dots, s_i, \dots, s_n$, 其中 s_i 满足: $1 \leq s_i \leq 100$ 的整数。要把这些物品装入到容量为 100 的一批箱子 (序号 $1 \sim n$) 中。装箱方法是: 对每项物品 s_i , 依次扫描所有这些箱子, 把 s_i 放入足以能够容下它的第一个箱子中 (first-fit 策略)。编写程序模拟这个装箱的过程, 并输出每个物品所在的箱子序号, 以及所需的箱子数目。

解答:

```

#include <stdio.h>
#define MAXC 100
#define MAXN 1000
int main()
{
    int i, j, B[MAXN], N, s, maxb;

    scanf("%d", &N);
    for (i=0; i<N; i++) B[i] = 0;
    maxb = 0;
    for (i=0; i<N; i++) {
        scanf("%d", &s); printf("%d ", s);
        for (j=0; j<maxb; j++) {

```



```

        if (B[j]+s <= MAXC) {
            B[j] += s;
            printf("%d\n", j+1);
            break;
        }
    }
    if (j==maxb) {
        B[j] += s;
        printf("%d\n", j+1);
        maxb ++;
    }
}
printf("%d\n", maxb);

return 0;
}

```

7.2 二维数组

一、调试示例

求矩阵各行元素之和：输入两个正整数 m 和 n ($m \geq 1$, $n \leq 6$)，然后输入该 m 行 n 列二维数组 a 中的元素，分别求出各行元素之和并输出。

解答：参见习题程序设计第 4 题。

二、基础编程题

(1) 矩阵运算：读入一个正整数 n ($1 \leq n \leq 6$)，再读入 n 阶方阵 a ，计算该矩阵除副对角线、最后一列和最后一行以外的所有元素之和。副对角线为从矩阵的右上角至左下角的连线。试编写相应程序。

解答：参见练习 7-7。

(2) 求矩阵的局部极大值：给定 M 行 N 列的整数矩阵 A ，如果 A 的非边界元素 $A[i][j]$ 大于相邻的上下左右 4 个元素，那么就称元素 $A[i][j]$ 是矩阵的局部极大值。要求输出给定矩阵的全部局部极大值及其所在的位置。

解答：

```

#include <stdio.h>
#define MAXN 20
int main()
{
    int i, j, flag, m, n;
    int a[MAXN][MAXN];

    scanf("%d %d", &m, &n);
    for (i = 0; i < m; i++){

```

```

        for (j = 0; j < n; j++){
            scanf("%d", &a[i][j]);
        }
    }
    flag = 0;
    for (i = 1; i < m-1; i++) {
        for (j = 1; j < n-1; j++) {
            if ((a[i][j] > a[i-1][j]) && (a[i][j] > a[i+1][j])
                && (a[i][j] > a[i][j-1]) && (a[i][j] > a[i][j+1])) {
                printf("%d %d %d\n", a[i][j], i+1, j+1);
                flag = 1;
            }
        }
    }
    if (flag == 0){
        printf("None %d %d\n", m, n);
    }

    return 0;
}

```

(3) 计算天数：输入日期（年、月、日），计算其是该年的第几天。要求定义和调用函数 `day_of_year(year, month, day)`（教材例 7-10）。试编写相应程序。

解答：参见练习 7-9。

(4) 判断上三角矩阵：输入一个正整数 n ($1 \leq n \leq 6$) 和 n 阶方阵 a 中的元素，如果 a 是上三角矩阵，输出“YES”，否则，输出“NO”。上三角矩阵指主对角线以下的元素都为 0 的矩阵，主对角线为从矩阵的左上角至右下角的连线。试编写相应程序。

解答：参见习题程序设计第 3 题。

(5) 打印杨辉三角：输入一个整数 N ($1 \leq N \leq 10$)。要求以正三角形的格式输出前 N 行杨辉三角。每个数字占固定 4 位。试编写相应程序。

解答 1：

```

#include <stdio.h>
#define MAXN 10
void PrintBlank ( int n );
int main()
{
    int i, j, n;
    static int a[MAXN][MAXN];

    scanf ( "%d", &n );
    PrintBlank ( n-1 );
    printf ( "%4d\n", 1 );
}

```

```

    for ( i = 1; i < n; i++ ) {
        PrintBlank ( n-1-i );
        a[i][0] = a[i][i] = 1;
        printf ( "%4d", a[i][0] );
        for ( j = 1; j < i; j++ ){
            a[i][j] = a[i-1][j-1] + a[i-1][j];
            printf ( "%4d", a[i][j] );
        }
        printf ( "%4d\n", a[i][i] );
    }

    return 0;
}

```

```

void PrintBlank ( int n )
{
    for ( ; n > 0; --n ) {
        printf ( " ");
    }
}

```

解答 2:

```

#include <stdio.h>
#define MAXN 10
void PrintBlank(int n);
int main()
{
    int i, j, m1, m2, n, t;
    static int a[2][MAXN];

    scanf("%d", &n);
    PrintBlank(n-1);
    printf("%4d\n", 1);
    m1 = 0; m2 = 1;
    for (i = 1; i < n; i++) {
        PrintBlank(n-1-i);
        a[m2][0] = a[m2][i] = 1;
        for ( j = 1; j < i; j++ ){
            a[m2][j] = a[m1][j-1] + a[m1][j];
        }
        for ( j = 0; j <= i; j++ ){
            printf("%4d", a[m2][j]);
        }
        printf("\n");
    }
}

```

```

        t = m1; m1 = m2; m2 = t;
    }

    return 0;
}

void PrintBlank(int n)
{
    for (; n > 0; --n) {
        printf(" ");
    }
}

```

三、改错题

方阵循环右移：输入两个正整数 m 和 n ($m \geq 1, n \leq 6$)，然后输入该 m 行 n 列二维数组 a 中的元素，将该二维数组 a 中的每个元素向右移一列，最后一列换到第一列，移动后的数组存到另一个二维数组 b 中，按矩阵形式输出 b 。

解答：参见练习 7-8。

四、拓展编程题

(1) 找鞍点：输入 1 个正整数 n ($1 \leq n \leq 6$) 和 n 阶方阵 a 中的元素，假设方阵 a 最多有 1 个鞍点，如果找到 a 的鞍点，就输出其下标，否则，输出 "NO"。鞍点的元素值在该行上最大，在该列上最小。试编写相应程序。

解答：参见习题程序设计第 5 题。

(2) 螺旋方阵：所谓“螺旋方阵”，是指对任意给定的 N ，将 1 到 $N*N$ 的数字从左上角第 1 个格子开始，按顺时针螺旋方向顺序填入 $N*N$ 的方阵里。输入一个正整数 n ($n \leq 9$)，输出 n 阶的螺旋方阵。试编写相应程序。

解答：

```

#include <stdio.h>
#define MAXN 9
int main()
{
    int cnt, i, j, k, n, nn;
    int a[MAXN][MAXN];

    scanf("%d", &n);
    nn = n*n;
    k = 0; cnt = 1;
    while (cnt <= nn) {
        for (j = k; j < n-k; j++) {
            a[k][j] = cnt++;
        }
        for (i = k + 1; i < n-k; i++) {

```

```

        a[i][n-1-k] = cnt++;
    }
    for (j = n-2-k; j >= k; j--) {
        a[n-1-k][j] = cnt++;
    }
    for (i = n-2-k; i >= k+1; i--) {
        a[i][k] = cnt++;
    }
    k++;
}
for (i = 0; i < n; i++) {
    for (j = 0; j < n; j++)
        printf("%3d", a[i][j]);
    printf("\n");
}

return 0;
}

```

(3) 简易连连看：给定一个 $2N \times 2N$ 的方阵网格游戏盘面，每个格子中放置一些符号。这些符号一定是成对出现的，同一个符号可能不止一对。程序读入玩家给出的一对位置 $(x1, y1)$ 、 $(x2, y2)$ ，判断这两个位置上的符号是否匹配。如果匹配成功，则将两个符号消为“*”并输出消去后的盘面；否则输出“Uh-oh”。若匹配错误达到 3 次，则输出“Game Over”并结束游戏。或者当全部符号匹配成功，则输出“Congratulations!”，然后结束游戏。试编写相应程序。

解答：

```

#include <stdio.h>
#define MAXN 8
char a[MAXN][MAXN];
void printP( int n )
{
    int i, j;
    for (i=0; i<n; i++) {
        printf("%c", a[i][0]);
        for (j=1; j<n; j++)
            printf(" %c", a[i][j]);
        printf("\n");
    }
}

int main()
{
    int n, k, i, j, l, x1, y1, x2, y2, cnt1, cnt2;
    char c;

```

```

scanf("%d", &n); scanf("%c", &c);
n *= 2;
for (i=0; i<n; i++)
    for (j=0; j<n; j++)
        scanf("%c%c", &a[i][j], &c);
cnt1 = n*n; cnt2 = 0;
scanf("%d", &k);
for (l=0; l<k; l++) {
    scanf("%d %d %d %d", &x1, &y1, &x2, &y2);
    if ((a[x1-1][y1-1]!='*') && (a[x1-1][y1-1] == a[x2-1][y2-1])) {
        a[x1-1][y1-1] = a[x2-1][y2-1] = '*';
        cnt1 -=2;
        if (!cnt1) break;
        printP(n);
    }
    else {
        printf("Uh-oh\n");
        cnt2++;
        if (cnt2==3) break;
    }
}
if (!cnt1) printf("Congratulations!\n");
else if (cnt2==3) printf("Game Over\n");

return 0;
}

```

7.3 字符串

一、调试示例

字符串逆序：输入一个以回车结束的字符串（少于 80 个字符），将它的内容逆序输出。如“ABCD”输出为“DCBA”。

解答：参见练习 7-11。

二、基础编程题

(1) 查找指定字符：输入一个字符，再输入一个以回车结束的字符串（少于 80 个字符），在字符串中查找该字符。如果找到，则输出该字符在字符串中所对应的最大下标，下标从 0 开始；否则输出“Not Found”。试编写相应程序。

解答：参见练习 7-10。

(2) 统计大写辅音字母：输入一个以回车结束的字符串（少于 80 个字符），统计并输出其中大写辅音字母的个数。大写辅音字母：除'A','E','I','O','U'以外的大写字母。试编写相应程序。

解答：参见习题程序设计第 6 题。

(3) 字符串替换：输入一个以回车结束的字符串（少于 80 个字符），将其中的大写字母用下面列出的对应大写字母替换，其余字符不变，输出替换后的字符串。试编写相应程序。

解答：参见习题程序设计第 7 题。

(4) 输出大写英文字母：输入一个以回车结束的字符串（少于 80 个字符），输出其中所出现过的大写英文字母；若无大写英文字母则输出 “Not Found”。

解答 1：

```
#include<stdio.h>
# define MAXLEN 81
# define UPPERLEN 26
int main()
{
    char upper[UPPERLEN], str[MAXLEN];
    int i, j, k, flag;

    i = 0;
    while ( (str[i] = getchar( )) != '\n' ) {
        i++;
    }
    str[i] = '\0';

    k = 0;
    for ( i = 0; str[i] != '\0'; i++) {
        if ( str[i] >= 'A' && str[i] <= 'Z' ) {
            flag = 0;
            for ( j = 0; j < k; j++) {
                if( upper[j] == str[i]) {
                    flag = 1;
                    break;
                }
            }
            if( flag == 0 ) {
                upper[k] = str[i];
                k++;
            }
        }
    }
    upper[k] = '\0';
    if ( k == 0 ) {
        printf ("Not Found\n");
    }
    else {
```

```

        printf ("%s\n", upper);
    }

    return 0;
}

```

(5) 字符串转换成十进制整数：输入一个以#结束的字符串，滤去所有的非十六进制字符（不分大小写），组成一个新的表示十六进制数字的字符串，然后将其转换为十进制数后输出。如果过滤后字符串的首字符为“-”，代表该数是负数。试编写相应程序。

解答：参见习题程序设计第 8 题。

三、改错题

字符串转换：输入一个以回车结束的字符串（少于 80 个字符），把字符串中的所有数字字符（'0'~'9'）转换为整数，去掉其他字符。例如，字符串“3a56bc”转换为整数后是 356。

解答：

```

#include <stdio.h>
# define MAXLEN 81
int main()
{
    char str[MAXLEN];
    int digit, i, number;

    i = 0;
    while ( (str[i] = getchar()) != '\n' ){
        i++;
    }
    str[i] = '\0';

    number = 0;
    for ( i = 0; str[i] != '\0'; i++ ) {
        if ( str[i] >= '0' && str[i] <= '9' ) {
            digit = str[i] - '0';
            number = number * 10 + digit;
        }
    }

    printf("%d\n", number);

    return 0;
}

```

四、拓展编程题

(1) 统计字符出现次数：输入一个以回车符结束的字符串（少于 80 个字符），再输入一个字符，统计并输出该字符在字符串中出现的次数。试编写相应程序。

解答:

```
#include <stdio.h>
#define MAXLEN 81
int main()
{
    char ch;
    char str[MAXLEN];
    int count, i;

    i = 0;
    while ( (str[i] = getchar()) != '\n' ){
        i++;
    }
    str[i] = '\0';

    ch = getchar();
    count = 0;
    for(i = 0; str[i] != '\0'; i++){
        if ( ch == str[i]) {
            count++;
        }
    }

    printf("%d\n", count);
    return 0;
}
```

(2) 字符串字母大小写转换: 输入一个以 # 结束的字符串, 将小写字母全部转换成大写字母, 把大写字母全部转换成小写字母, 其它字符不变输出到屏幕。试编写相应程序。

解答:

```
#include <stdio.h>
#define MAXLEN 31
int main()
{
    char str[MAXLEN];
    int i;

    i = 0;
    while ( (str[i] = getchar()) != '#' ){
        i++;
    }
    str[i] = '\0';

    for( i = 0; str[i] != '\0'; i++ ) {
```

```

        if ( str[i] >= 'A' && str[i] <= 'Z' ){
            str[i] = str[i] - 'A' + 'a';
        }
        else if ( str[i] >= 'a' && str[i] <= 'z' ){
            str[i] = str[i] - 'a' + 'A';
        }
    }

    printf ( "%s\n", str );

    return 0;
}

```

(3) 删除重复字符：输入一个以回车结束的字符串（少于 80 个字符），去掉重复的字符后，按照字符 ASCII 码顺序从小到大排序后输出。试编写相应程序。

解答：

```

#include<stdio.h>
#define MAXLEN 81
int main()
{
    char str[80], ch;
    int i, j, k;

    i = 0;
    while ( (str[i] = getchar( )) != '\n' ) {
        i++;
    }
    str[i] = '\0';

    for ( i = 0; str[i] != '\0'; i++ ) {
        ch = str[i];
        j = i + 1;
        while ( str[j] != '\0' ) {
            if ( str[j] != ch ) {
                j++;
            }
            else {
                for ( k = j; str[k] != '\0'; k++)
                    str[k] = str[k+1];
            }
        }
    }

    for ( i = 0; str[i] != '\0'; i++ ) {

```

```

        k = i;
        for ( j = i+1; str[j] != '\0'; j++ ) {
            if ( str[j] < str[k] ) {
                k = j;
            }
        }
        ch = str[i];
        str[i] = str[k];
        str[k] = ch;
    }

    printf("%s\n", str);

    return 0;
}

```

练习 7-8 编写程序，输出一张九九乘法口诀表。提示：将乘数、被乘数和乘积放入一个二维数组中，再输出该数组。

解答：

```

#include <stdio.h>
int main(void)
{
    int i,j,n,a[10][10];
    for( i=0; i<=9; i++ )
        for( j=0; j<=9; j++ )
            if( i==0)  a[i][j]=j;
            else if( j==0)  a[i][j]=i;
            else  a[i][j]=i*j;
    for( i=0; i<=9; i++ ){
        for( j=0; j<=9; j++ )
            if( i==0&&j==0 ) printf( "%-4c", '*');
            else if( i==0||j<=i ) printf( "%-4d", a[i][j]);
        printf("\n");
    }
    return 0;
}

```

9. 编写程序，输入一个十进制数，再输入一个其他进制的基数（范围在 2 到 16 之间），将十进制数转换成相应的基数进制数。如输入 10（十进制数），再输入 2（要转换成二进制），输出 1010（十进制数 10 转换成二进制是 1010）。

```

#include <stdio.h>
int main(void)
{
    int x,base,i,k,y;

```

```
char s[10];
printf("Input x: ");
scanf("%d",&x);
printf("Input base: ");
scanf("%d",&base);
k=-1;
do{
    y=x%base;
    x=x/base;
    k++;
    if(y>=10)
        s[k]=y-10+'A';
    else
        s[k]=y+'0';
}while(x!=0);
for(i=k;i>=0;i--)
    putchar(s[i]);
putchar('\n');
return 0;
```