

# 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，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。

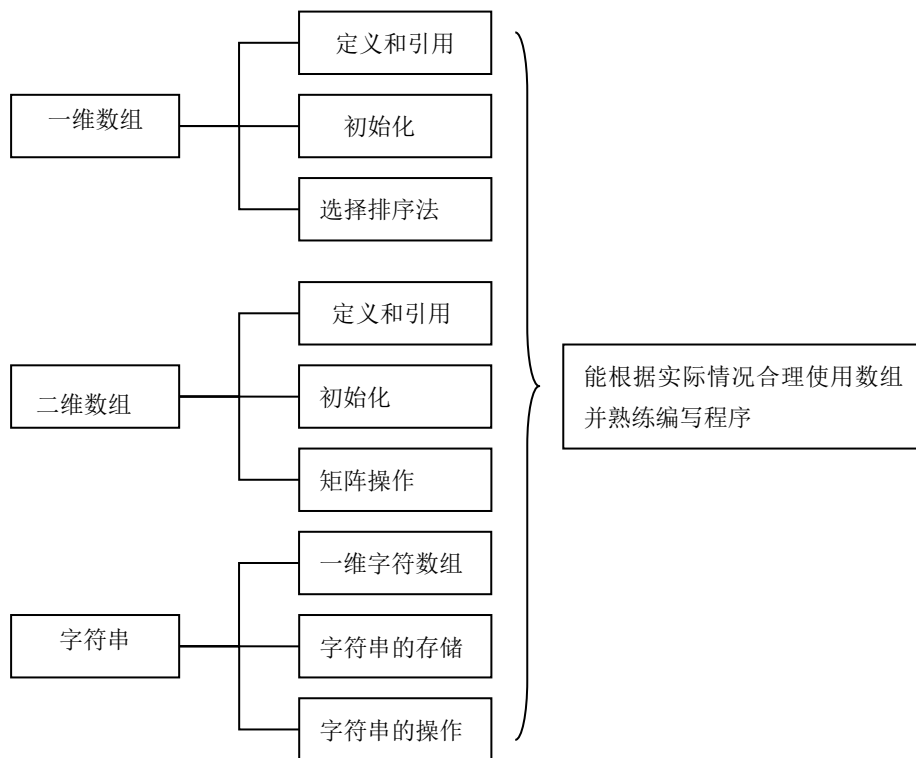





图 7.1 知识能力结构图

## 7.2 讲稿

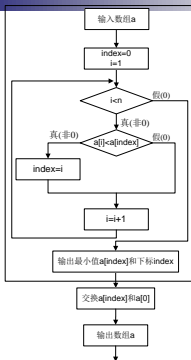
1	 <h3>Chap 7 数 组</h3> <p>7.1 输出所有大于平均值的数</p> <p>7.2 找出矩阵中最大值所在的位置</p> <p>7.3 判断回文</p>	本章分 3 节。
2	 <h3>本章要点</h3> <ul style="list-style-type: none"> <li>■ 什么是数组？为什么要使用数组？如何定义数组？</li> <li>■ 如何引用数组元素？</li> <li>■ 二维数组的元素在内存中按什么方式存放？</li> <li>■ 什么是字符串？字符串结束符的作用是什么？</li> <li>■ 如何实现字符串的存储和操作，包括字符串的输入和输出？</li> <li>■ 怎样理解C语言将字符串作为一个特殊的一维字符数组？</li> </ul>	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>计算平均值后要对数据重新遍历，需要将所有数据保存并方便按顺序访问，引出一维数组。</p>
4	<h4>7.1.1 程序解析—输出大于平均值的数</h4> <pre>int main(void) { int i; double average, sum;   int a[10]; /* 定义1个数组a，它有10个整型元素 */   printf("Enter n: "); scanf("%d", &amp;n);   printf("Enter %d integers: ", n);   for(i = 0; i &lt; 10; i++){     scanf("%d", &amp;a[i]);     sum = 0;     for(i = 0; i &lt; 10; i++){       sum = sum + a[i];     }     average = sum / n;     printf("average = %.2f\n", average);     printf("&gt;average:");     for(i = 0; i &lt; 10; i++){       if(a[i] &gt; average){         printf("%d ", a[i]);       }     }     printf("\n");     return 0;   }</pre> <div> <p>Enter n: 10</p> <p>Enter 10 integers: 55 23 8 11 22 89 0 -1 78 186</p> <p>average = 47.10</p> <p>&gt;average: 55 89 78 186</p> </div>	<p>展示、运行例 7-1 程序。</p> <p>解读程序，可分以下几点说明：定义、输入、处理、输出几部分。</p>

5	<div><div><div><div><div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div></div></div><div><div>数组</div><div><div><div>for(i = 0; i &lt; 10; i++){ sum = sum + a[i]; }</div><div><div>0123456789</div><div>a552381122890-178186</div><div>a[0]a[1]a[9]</div></div></div></div><div><div>■ 数组：相同类型数据的有序集合，在内存中连续存放。<div><div>□ 由数组名和下标唯一地确定每个数组元素</div><div>□ 每个元素都属于同一类型</div></div></div><div>■ 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。<div>□ 优点：表述简洁，可读性高；便于使用循环结构</div></div></div></div></div></div>	<p>展示说明一维数组 a 的组织方式。说明以下问题：</p> <p>什么是数组？</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><div><div><div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div></div></div><div><div>7.1.2 一维数组的定义和引用</div><div>1、定义<div><div>类型名 数组名[数组长度]</div><div>数组长度为常量</div><div>类型名：数组元素的类型</div><div>数组名：数组（变量）的名称，标识符</div><div>数组长度：常量表达式，给定数组的大小</div></div><div>int a[10]; 定义一个含有10个整型元素的数组 a</div><div>char c[200]; 定义一个含有200个字符元素的数组 c</div><div>float f[5]; 定义一个含有5个浮点型元素的数组 f</div></div></div></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><div><div><div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div></div></div><div><div>2. 数组的内存结构</div><div><div>int a[10];</div><div>假设系统规定 int 类型占用2个字节，数组a的内存分配形式</div><div>只要知道了数组首元素的地址以及每个元素所需的字节数，其余各个元素的存储地址均可计算得到。</div><div><div>内存地址 下标 值</div><div>4028 9</div><div>4026 8</div><div>4024 7</div><div>4022 6</div><div>4020 5</div><div>4018 4</div><div>4016 3</div><div>4014 2</div><div>4012 1</div><div>a 4010 0</div></div><div>数组名是一个地址常量，存放数组内存空间的首地址。</div></div></div></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><div><div><div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div></div></div><div><div>3. 引用</div><div><div>■ 先定义，后使用</div><div>■ 只能引用单个的数组元素，不能一次引用整个数组</div><div>int a[10]; 10个数组元素：a[0]、a[1]、..... a[9]</div><div>数组元素：数组名[下标]</div><div>下标：整型表达式</div><div>下标取值范围：[0，数组长度-1]</div><div>■ 数组元素的使用方法与同类型的变量相同</div><div>scanf("%d", &amp;a[i]); sum = sum + a[i]; printf("%d ", a[i]);</div></div></div></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>一次只能引用单个元素。</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>	再次详细说明数组定义和引用的区别。
10	<div> <div></div> <div> <h2>7.1.3 一维数组的初始化</h2> <ul style="list-style-type: none"> <li>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> </li> <li>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];</pre> 等价与 <code>int c[5];</code> </li> </ul> </div> </div>	和简单变量定义时初始化做对比，说明如何在定义数组时进行初始化。 分为全部元素初始化和部分元素初始化两种方式。
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"> <li>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> </li> </ul> </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 &lt; 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><b>例 7-2</b> 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><b>例 7-3</b> 顺序查找法。输入正整数n和整数x，再输入n个整数并存入数组a中，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><b>例 7-4</b> 输入n，再输入n个整数 (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><b>例 7-5</b> 选择法排序。输入n，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><b>例 7-6</b> 调查电视节目欢迎程度。某电视台要进行一次对该台8个栏目的受欢迎情况，共调查了n位观众，现要求编写程序，输入观众的投票，统计输出各栏目的得票情况。</p><p><b>例 7-7</b> 二分查找法。设已有一个n个元素的整形数组a，且按值从小到大有序排列。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。</p> <p>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>										
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n个数(1≤n≤46)，并按每行打印5个数的格式输出，如果最后一行的输出少于5个数，也需要换行。</p><p>1, 1, 2, 3, 5, 8, 13, .....</p><p>用数组计算并存放fibonacci数列的前n个数</p><p>f[0] = f[1] = 1</p><p>f[i] = f[i-1] + f[i-2]     2 ≤ i ≤ 45</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，即 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 &lt;stdio.h&gt; # define MAXN 46 /* 符号常量 */ int main(void) { int i, n;   int fib[MAXN] = {1, 1}; /* 数组初始化 */   printf("Enter n: "); scanf ("%d", &amp;n);    for ( i = 2; i &lt; n; i++) {     fib[i] = fib[i - 1] + fib[i - 2];   }   for ( i = 0; i &lt; n; i++) {     printf ("%11d", fib[i]);     if ((i + 1) % 5 == 0) { /* 5个数换行 */       printf("\n"); }   }   if(n % 5 != 0) { printf("\n"); } /* 总数不是5的倍数换行 */   return 0; }</pre><div>Enter n: 10</div><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]。</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 查找满足条件的所有整数</h3><p>输入正整数n (1≤n≤10)和整数x，再输入n个整数并存入数组a中，然后在数组中查找x，如果找到，输出所有满足条件的元素的下标，否则，输出“Not Found”。</p><p>Enter n, x: 5 9                      Enter n, x: 4 101</p><p>Enter 5 integers: 2 9 8 1 9        Enter 5 integers: 9 8 -101 10</p><p>Index is 1                            Not Found</p><p>Index is 4</p></div></div>	<p>引导学生分析题目要求并思考运行过程，考虑当有不只一个相同元素时如何处理</p>										

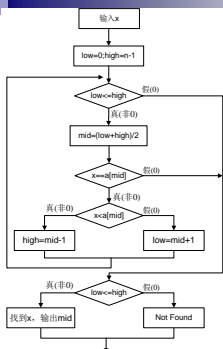
17	<div data-bbox="316 212 558 593"> <pre>#include &lt;stdio.h&gt; #define MAXN 10 int main(void) {     int i, flag, n, x; int a[MAXN];     printf("Enter n, x: ");     scanf("%d%d", &amp;n, &amp;x);     printf("Enter %d integers: ", n);     for (i = 0; i &lt; n; i++) {         scanf("%d", &amp;a[i]);     }     flag = 0;     for (i = 0; i &lt; n; i++) {         if (a[i] == x) {             printf("Index is %d\n", i);             flag = 1;         }     }     if (flag == 0) {         printf("Not Found\n");     }     return 0; }</pre> </div> <div data-bbox="582 212 766 257">例 7-3 源程序</div> <div data-bbox="582 324 790 414"> Enter n, x: 5 9  Enter 5 integers: 2 9 8 1 9  Index is 1  Index is 4 </div> <div data-bbox="582 459 790 526"> Enter n, x: 4 101  Enter 4 integers: 9 8 -101 10  Not Found </div>	<p>分析程序设计方式与方法，重点分析元素的输入过程、输出过程及找元素的过程。</p> <p>特别说明：</p> <p>变量 flag 的作用，即是否找到元素的标志。</p>
18	<div data-bbox="316 629 646 996"> <pre>#include &lt;stdio.h&gt; #define MAXN 10 int main(void) {     int i, flag, n, x; int a[MAXN];     printf("Enter n, x: "); scanf("%d%d", &amp;n, &amp;x);     printf("Enter %d integers: ", n);     for (i = 0; i &lt; n; i++) {         scanf("%d", &amp;a[i]);     }     flag = 0;     for (i = 0; i &lt; n; i++) {         if (a[i] == x) {             printf("Index is %d\n", i);             flag = 1;             break;         }     }     if (flag == 0) {         printf("Not Found\n");     }     return 0; }</pre> </div> <div data-bbox="598 629 790 683">例 7-3 思考(1)</div> <div data-bbox="630 750 750 772">加break语句</div> <div data-bbox="582 817 790 896"> Enter n, x: 5 9  Enter 5 integers: 2 9 8 1 9  Index is 1 </div>	<p>详细分析 break 语句的作用，即一旦找到后就终止循环。</p>
19	<div data-bbox="316 1046 638 1422"> <pre>#include &lt;stdio.h&gt; #define MAXN 10 int main(void) {     int i, flag, n, x;     int a[MAXN];     printf("Enter n, x: ");     scanf("%d%d", &amp;n, &amp;x);     printf("Enter %d integers: ", n);     for (i = 0; i &lt; n; i++) {         scanf("%d", &amp;a[i]);     }     index = -1;     for (i = 0; i &lt; n; i++) {         if (a[i] == x) {             index = i;         }     }     if (index != -1) { printf("Index is %d\n", index); }     else { printf("Not Found\n"); }     return 0; }</pre> </div> <div data-bbox="574 1064 766 1108">例 7-3 思考(2)</div> <div data-bbox="574 1153 782 1220"> Enter n, x: 5 9  Enter 5 integers: 2 9 8 1 9  Index is 4 </div>	<p>改变程序设计方式，使用变量 index 实现，解读程序，说明：</p> <p>index 的作用：</p> <p>找元素的区别，即找最后一个值相同的元素。</p>
20	<div data-bbox="316 1462 598 1836"> <pre>#include &lt;stdio.h&gt; #define MAXN 10 int main(void) {     int i, min, n;     int a[MAXN];     printf("Enter n: "); scanf("%d", &amp;n);     printf("Enter %d integers: ", n);     for (i = 0; i &lt; n; i++) {         scanf("%d", &amp;a[i]);     }     min = a[0];     for (i = 1; i &lt; n; i++) {         if (a[i] &lt; min) { min = a[i]; }     }     printf("min is %d\n", min);     return 0; }</pre> </div> <div data-bbox="534 1462 766 1512">例 7-4 求最小值</div> <div data-bbox="534 1523 790 1590"> Enter n: 6  Enter 6 integers: 2 9 -1 8 1 6  min is -1 </div> <div data-bbox="606 1736 805 1780"> 虽得到了最小值，但不能确定最小值所在下标。 </div>	<p>例 7-4 解决如何在数组中找最小（大）值的方法。特别说明 min 变量的作用。</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。引出下面的程序。</p>

21	<p><b>例 7-4(1) 求最小值及其下标</b></p> <p>输入n(n&lt;10), 再输入n个数, 输出最小值和它所对应的下标。</p> <p>用index记录最小值对应的下标 a[index]就是最小值</p>	<p>由于在数组中只要知道了元素的下标, 则该值也就确定, 故在找最小值时不用上面 min 变量保存最小的值, 而用变量 index 保存最小值所在的下标, 则 a[index]就是最小值。</p> <p>此处可让学生解答如何根据上面程序给出流程图。</p>
22	<p><b>流程图</b></p> 	<p>分析流程图, 并可让学生按照流程图给出源程序。</p>
23	<p><b>求最小值及下标</b></p> <pre> #include &lt;stdio.h&gt; #define MAXN 10 int main(void) {     int i, index, n;     int a[MAXN];     printf("Enter n: ");     scanf("%d", &amp;n);     printf("Enter %d integers: ", n);     for(i = 0; i &lt; n; i++){         scanf("%d", &amp;a[i]);     }     index = 0;     for(i = 1; i &lt; n; i++) {         if(a[i] &lt; a[index]) { index = i; }     }     printf("min is %d\tsub is %d\n", a[index], index);     return 0; } </pre> <p>Enter n: 6 Enter 6 integers: 2 9 -1 8 1 6 min is -1 sub is 2</p> <p>index: 最小值对应的下标 a[index]: 最小值</p>	<p>给出源程序并运行。</p> <p>重点说明语句 index=0 的作用。</p>
24	<p><b>例 7-4(2) 交换最小值</b></p> <p>输入n(n&lt;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。</p> <p>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] &lt;==&gt; a[0]</p>	<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形变量 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: 前两个实例是选择排序法的基础, 必须理解。</p>

25	<div>例 7-5 选择法排序</div> <div>输入<math>n(n&lt;10)</math>，再输入<math>n</math>个数，用选择法将它们从小到大排序后输出。</div> <div>设 <math>n = 5</math>，输入为：3 5 2 8 1</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>	下标	0	1	2	3	4	值	3	5	2	8	1	<div>介绍选择排序法。</div> <div>从原理上说明如何利用前两个功能进行排序。</div> <div>提醒：</div> <div>这是典型的排序算法。</div> <div>什么叫一趟选择排序，即找到最小值并与相应元素交换的过程。</div> <div><math>n</math> 个元素共需进行 <math>n-1</math> 趟选择排序才能完成。</div>
下标	0	1	2	3	4									
值	3	5	2	8	1									
26	<div>选择法分析(1)</div> <div>3 5 2 8 1 (<math>n = 5</math>)</div> <div>5个数(<math>a[0] \sim a[4]</math>)中找最小数，与<math>a[0]</math>交换</div> <div>(0) 1 5 2 8 3 <math>a[4] \rightleftharpoons a[0]</math></div> <div>4个数(<math>a[1] \sim a[4]</math>)中找最小数，与<math>a[1]</math>交换</div> <div>(1) 1 2 5 8 3 <math>a[2] \rightleftharpoons a[1]</math></div> <div>3个数(<math>a[2] \sim a[4]</math>)中找最小数，与<math>a[2]</math>交换</div> <div>(2) 1 2 3 8 5 <math>a[4] \rightleftharpoons a[2]</math></div> <div>2个数(<math>a[3] \sim a[4]</math>)中找最小数，与<math>a[3]</math>交换</div> <div>(3) 1 2 3 5 8 <math>a[4] \rightleftharpoons a[3]</math></div>	<div>分析 5 个元素进行选择排序时每一趟的过程。</div>												
27	<div>n个数重复n-1次 选择法分析(2)</div> <div>(0) <math>n</math> 个数 (<math>a[0] \sim a[n-1]</math>) 中找最小数，与<math>a[0]</math>交换</div> <div>(1) <math>n-1</math>个数 (<math>a[1] \sim a[n-1]</math>) 中找最小数，与<math>a[1]</math>交换</div> <div>.....</div> <div>(k) <math>n-k</math>个数 (<math>a[k] \sim a[n-1]</math>) 中找最小数，与<math>a[k]</math>交换</div> <div>.....</div> <div>(n-2) 2个数 (<math>a[n-2] \sim a[n-1]</math>) 中找最小数，与<math>a[n-2]</math>交换</div> <div>(0) 5个数 (<math>a[0] \sim a[4]</math>) 中找最小数，与<math>a[0]</math> 交换</div> <div>(1) 4个数 (<math>a[1] \sim a[4]</math>) 中找最小数，与<math>a[1]</math> 交换</div> <div>(2) 3个数 (<math>a[2] \sim a[4]</math>) 中找最小数，与<math>a[2]</math> 交换</div> <div>(3) 2个数 (<math>a[3] \sim a[4]</math>) 中找最小数，与<math>a[3]</math> 交换</div>	<div>根据前页 5 个元素的过程，扩展到 <math>n</math> 个元素。并找出规律，即：</div> <div>第 <math>i</math> 趟时，在剩下未排序的 <math>n-i+1</math> 个数 (<math>a[i-1] \sim a[n-1]</math>) 中找到最小值，与 <math>a[i-1]</math> 交换。</div>												
28	<div>流程图</div> <div>外循环控制： <math>n</math> 个数选择排序共需要 <math>n-1</math>趟 <math>k: 0 \sim n-2</math></div> <div>内循环控制： 在下标范围 <math>[k, n-1]</math>内找最小值所在位置index</div> <pre>graph TD     Start([输入n个数]) --&gt; K0[k=0]     K0 --&gt; KltN1{k &lt; n-1}     KltN1 -- 是 Y --&gt; IndexK[index=k]     IndexK --&gt; Jk1[j=k+1]     Jk1 --&gt; JltN{j &lt; n}     JltN -- 否 N --&gt; Swap[交换a[index]和a[j]]     JltN -- 是 Y --&gt; AjltAi[a[j] &lt; a[index]]     AjltAi -- 是 Y --&gt; Indexj[index=j]     Indexj --&gt; JltN     AjltAi -- 否 N --&gt; Swap     Swap --&gt; Jincj[j=j+1]     Jincj --&gt; JltN     JltN -- 否 N --&gt; IncK[k=k+1]     IncK --&gt; KltN1     Swap --&gt; End([输出并返回])</pre>	<div>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。程序框架如下：</div> <div>for(k=0;k&lt;n-1;k++) { //共需 <math>n-1</math> 趟</div> <div>//在剩下的下标区间<math>[k,n-1]</math>内找最小值</div> <div>所在的位置 index;</div> <div>//把 <math>a[index]</math>与 <math>a[k]</math>交换;</div> <div>}</div>												



29	<p>选择法排序 (程序段)</p> <pre> for(k = 0; k &lt; n-1; k++){     index = k;     for(i = k + 1; i &lt; n; i++){         if(a[i] &lt; 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>	根据流程图给出排序过程的程序。
30	<p>例 7-6 投票情况统计</p> <p>某电视台要调查观众对8个栏目（相应栏目编号为1~8）的受欢迎情况，共调查了n（<math>1 \leq n \leq 1000</math>）位观众。输入每一位观众的投票情况（每位观众只能投1个栏目），统计各栏目的得票情况。</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>	本例重点分析如何实现投票统计，数组 count 的作用
31	<p>例7-6 源程序段</p> <pre> #define MAXN 8 static int count[MAXN+1];  for ( i = 1; i &lt;= n; i++) {     printf( "Enter your response: " );     scanf ( "%d", &amp;response );     if (response &gt;= 1 &amp;&amp; response &lt;= MAXN)         count[response]++;     else{         printf ( "invalid: %d\n", response );     } } printf ( "result:\n" ); for ( i = 1; i &lt;= MAXN; i++){     printf ( "%4d%4d\n", i, count[i] ); } </pre> <div> Enter n: 6  Enter your response: 3  Enter your response: 1  Enter your response: 6  Enter your response: 9  invalid: 9  Enter your response: 8  Enter your response: 1  result:  1 2  3 1  6 1  8 1 </div>	运行展示示例程序。 分析数组 count 元素的引用方式，以及为什么不用 count[0]
32	<p>例7-7 二分法查找</p> <p>设已有一个n（<math>1 \leq n \leq 10</math>）个元素的整型数组a，且按值从小到大有序排列。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。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>	本例是否讲解可自行选择。 结合前例 7-3，说明也是查找算法，重点说明： 二分查找的优势。 实现二分查找的数组的条件。 重点说明二分查找的原理与方法。特别是查找区间的确定，以及查找失败时的条件。

33	 <p>二分查找流程图</p> <pre> graph TD     Start([开始]) --&gt; Input[输入x]     Input --&gt; Init[low=0; high=n-1]     Init --&gt; LoopStart(( ))     LoopStart --&gt; Cond1{low &lt;= high}     Cond1 -- 否(0) --&gt; End([结束])     Cond1 -- 是 --&gt; CalcMid[ mid = (low + high) / 2 ]     CalcMid --&gt; Cond2{ x == a[mid] }     Cond2 -- 是 --&gt; End     Cond2 -- 否 --&gt; Cond3{ x &lt; a[mid] }     Cond3 -- 是 --&gt; UpdateHigh[ high = mid - 1 ]     Cond3 -- 否 --&gt; UpdateLow[ low = mid + 1 ]     UpdateHigh --&gt; LoopStart     UpdateLow --&gt; LoopStart     LoopStart --&gt; Cond4{low &lt;= high}     Cond4 -- 否 --&gt; End     Cond4 -- 是 --&gt; Output[找到x, 输出mid]     </pre>	<p>根据二分查找方法，解释算法流程图。重点说明循环开始前的初始情况、循环执行条件、以及循环结束后的判断。</p>
34	<p>二分法查找 (源程序段)</p> <pre> low = 0; high = n - 1;      /* 开始时查找区间为整个数组 */ while ( low &lt;= 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 &lt; 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 &lt;= 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>
35	<p>7.2 找出矩阵中最大值所在的位置</p> <p>将1个<math>m \times n</math>的矩阵存入1个<math>m \times n</math>的二维数组中，找出最大值以及它的行下标和列下标。</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>如何扫描矩阵来找出矩阵中的最大值？</p> <p>从而引出二维数组。</p>
36	<p>7.2.1 程序解析—求矩阵的最大值</p> <p>例 7-8 输入两个正整数<math>m</math>和<math>n</math> (<math>1 \leq m, n \leq 6</math>)，再输入一个1个<math>m \times n</math>的矩阵，找出最大值以及它的行下标和列下标。假设最大值唯一。</p> <p>row: 记录最大值的行下标</p> <p>col: 记录最大值的列下标</p> <p>a[row][col]: 最大值</p>	<p>确定记录矩阵中最大值的方法。</p>

37	<div data-bbox="564 219 799 255" data-label="Section-Header"> <h3>例7-8 源程序段</h3> </div> <pre data-bbox="316 219 783 573"> #define MAXM 6 #define MAXN 6 int a[MAXM][MAXN]; printf("Enter m, n: "); scanf("%d%d", &amp;m, &amp;n); printf("Enter %d integers: \n", m*n); for ( i = 0; i &lt; m; i++) {     for ( j = 0; j &lt; n; j++) {         scanf ("%d", &amp;a[i][j]);     } } row = col = 0; for ( i = 0; i &lt; m; i++) {     for ( j = 0; j &lt; n; j++) {         if ( a[i][j] &gt; a[row][col]) {             row = i;             col = j;         }     } } printf ( "max = a[%d][%d] = %d\n", row, col, a[row][col]); </pre> <div data-bbox="636 322 796 454" data-label="Text"> <pre> Enter m, n: 3 2 Enter 6 integers: 6 3 10 -9 5 -1 max = a[1][0] = 10 </pre> </div>	<p>展示、运行例 7-7 程序。</p> <p>解读程序，可分块方式说明：定义、输入、处理、输出几部分。略过具体的细节，可重点点点一下二维数组 a，以及和矩阵的对应关系。</p>
38	<div data-bbox="622 649 753 683" data-label="Section-Header"> <h3>二维数组</h3> </div> <p data-bbox="335 685 545 712">多维数组的空间想象</p> <p data-bbox="322 725 614 752">一维数组：一列表或一个向量</p> <p data-bbox="322 754 654 781">二维数组：一个表格或一个平面矩阵</p> <p data-bbox="322 784 614 810">三维数组：三维空间的一个方阵</p> <p data-bbox="322 815 654 842">多维数组：多维空间的一个数据列阵</p> <div data-bbox="341 869 718 978" data-label="Image"> </div>	<p>多维数组的空间想象，说明可推广。</p>
39	<div data-bbox="335 1070 727 1104" data-label="Section-Header"> <h3>7.2.2 二维数组的定义和引用</h3> </div> <p data-bbox="352 1128 437 1155">1、定义</p> <p data-bbox="376 1160 660 1187">类型名 数组名[行长度][列长度]</p> <pre data-bbox="376 1223 695 1272"> int a[3][2];     定义1个二维数组a，3行2列，6个元素 </pre> <pre data-bbox="376 1301 738 1350"> int b[5][10];     定义1个二维数组a，5行10列，50个元素 </pre>	<p>重点说明二维数组的定义方法。</p> <p>提醒：</p> <p>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。</p> <p>给出数组的元素个数，即行长度×列长度说明数组 a 所占的字节空间：若 int 类型占 2 个字节，则数组 a 占 <math>3 \times 2 \times 2 = 12</math> 个字节，同理可得到数组 b 所占字节数。</p>
40	<div data-bbox="654 1480 769 1514" data-label="Section-Header"> <h3>2、引用</h3> </div> <p data-bbox="347 1494 493 1520">先定义，后使用</p> <p data-bbox="347 1523 501 1550">数组元素的引用：</p> <p data-bbox="347 1552 564 1579">数组名[行下标][列下标]</p> <p data-bbox="371 1581 601 1608">行下标和列下标：整型表达式</p> <p data-bbox="371 1610 638 1637">行下标的取值范围是[0, 行长度-1]</p> <p data-bbox="371 1639 638 1666">列下标的取值范围是[0, 列长度-1]</p> <pre data-bbox="347 1693 643 1794"> int a[3][2]; 3行2列，6个元素 a[0][0] a[0][1] a[1][0] a[1][1] a[2][0] a[2][1] </pre> <div data-bbox="660 1715 788 1742" data-label="Text"> <p>下标不要越界</p> </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>

41	<div><div><div></div><div></div><div></div></div><div><h2>二维数组在内存中的存放方式</h2><p><code>int a[3][2];</code> 3 行 2 列，6 个元素 表示1个3行2列的矩阵</p><p>二维数组的元素在内存中按行/列方式存放</p><table><tr><td><code>a[0][0]</code></td><td><code>a[0][1]</code></td><td><code>a[0][0]</code></td><td></td></tr><tr><td><code>a[1][0]</code></td><td><code>a[1][1]</code></td><td><code>a[0][1]</code></td><td></td></tr><tr><td><code>a[2][0]</code></td><td><code>a[2][1]</code></td><td><code>a[1][0]</code></td><td></td></tr><tr><td></td><td></td><td><code>a[1][1]</code></td><td></td></tr><tr><td></td><td></td><td><code>a[2][0]</code></td><td></td></tr><tr><td></td><td></td><td><code>a[2][1]</code></td><td></td></tr></table></div></div>	<code>a[0][0]</code>	<code>a[0][1]</code>	<code>a[0][0]</code>		<code>a[1][0]</code>	<code>a[1][1]</code>	<code>a[0][1]</code>		<code>a[2][0]</code>	<code>a[2][1]</code>	<code>a[1][0]</code>				<code>a[1][1]</code>				<code>a[2][0]</code>				<code>a[2][1]</code>		<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>
<code>a[0][0]</code>	<code>a[0][1]</code>	<code>a[0][0]</code>																								
<code>a[1][0]</code>	<code>a[1][1]</code>	<code>a[0][1]</code>																								
<code>a[2][0]</code>	<code>a[2][1]</code>	<code>a[1][0]</code>																								
		<code>a[1][1]</code>																								
		<code>a[2][0]</code>																								
		<code>a[2][1]</code>																								
42	<div><div><div></div><div></div><div></div></div><div><h2>7.2.3 二维数组的初始化</h2><h3>1、分行赋初值</h3><p><code>int a[3][3] = {{1,2,3},{4,5,6},{7,8,9}};</code> <code>static int b[4][3] = {{1,2,3},{},{4,5}};</code></p><table><tr><td>数组a</td><td>数组b</td></tr><tr><td>1 2 3</td><td>1 2 3</td></tr><tr><td>4 5 6</td><td>0 0 0</td></tr><tr><td>7 8 9</td><td>4 5 0</td></tr><tr><td></td><td>0 0 0</td></tr></table><h3>2、顺序赋初值</h3><p><code>int a[3][3] = {1,2,3,4,5,6,7,8,9};</code> <code>static int b[4][3] = {1,2,3,0,0,0,4,5};</code></p></div></div>	数组a	数组b	1 2 3	1 2 3	4 5 6	0 0 0	7 8 9	4 5 0		0 0 0	<p>说明如何在定义二维数组时进行初始化。分为分行赋初值和顺序赋初值两种。</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>														
数组a	数组b																									
1 2 3	1 2 3																									
4 5 6	0 0 0																									
7 8 9	4 5 0																									
	0 0 0																									
43	<div><div><div></div><div></div><div></div></div><div><h2>省略行长度</h2><p>对全部元素都赋了初值</p><p><code>int a[ ][3] = { 1, 2, 3, 4, 5, 6, 7, 8, 9 };</code></p><p>或分行赋初值时，在初值表中列出了全部行</p><p><code>static int b[ ][3] = { {1, 2, 3}, {}, {4, 5}, {} }</code></p><table><tr><td>数组a</td><td>数组b</td></tr><tr><td>1 2 3</td><td>1 2 3</td></tr><tr><td>4 5 6</td><td>0 0 0</td></tr><tr><td>7 8 9</td><td>4 5 0</td></tr><tr><td></td><td>0 0 0</td></tr></table></div></div>	数组a	数组b	1 2 3	1 2 3	4 5 6	0 0 0	7 8 9	4 5 0		0 0 0	<p>说明省略行长度的方式，但建议不要省略。</p>														
数组a	数组b																									
1 2 3	1 2 3																									
4 5 6	0 0 0																									
7 8 9	4 5 0																									
	0 0 0																									
44	<div><div><div></div><div></div><div></div></div><div><h2>7.2.4 使用二维数组编程</h2><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 i 做为外循环的循环变量 列下标 j 内循环</p><table><tr><td>j = 0</td><td>j = 1</td><td>j = 2</td></tr><tr><td>i = 0</td><td><code>a[0][0]</code></td><td><code>a[0][1]</code></td><td><code>a[0][2]</code></td></tr><tr><td>i = 1</td><td><code>a[1][0]</code></td><td><code>a[1][1]</code></td><td><code>a[1][2]</code></td></tr><tr><td>i = 2</td><td><code>a[2][0]</code></td><td><code>a[2][1]</code></td><td><code>a[2][2]</code></td></tr></table></div></div>	j = 0	j = 1	j = 2	i = 0	<code>a[0][0]</code>	<code>a[0][1]</code>	<code>a[0][2]</code>	i = 1	<code>a[1][0]</code>	<code>a[1][1]</code>	<code>a[1][2]</code>	i = 2	<code>a[2][0]</code>	<code>a[2][1]</code>	<code>a[2][2]</code>	<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，但要注意下标不能越界。</p>									
j = 0	j = 1	j = 2																								
i = 0	<code>a[0][0]</code>	<code>a[0][1]</code>	<code>a[0][2]</code>																							
i = 1	<code>a[1][0]</code>	<code>a[1][1]</code>	<code>a[1][2]</code>																							
i = 2	<code>a[2][0]</code>	<code>a[2][1]</code>	<code>a[2][2]</code>																							

45	<div><div><div>生成一个矩阵并输出</div><div>矩阵的运算通常使用二维数组实现</div><div>输入两个正整数m和n（1≤m,n≤3），生成并输出一个m*n的矩阵，其元素的值由下式给出。 a[i][j] = i + j （0 ≤ i ≤ m-1, 0 ≤ j ≤ n-1）</div><div>输入：m=3, n=2</div><div>输出：<table><tr><td></td><td></td><td>j = 0</td><td>j = 1</td><td>j = 2</td></tr><tr><td>0</td><td>1</td><td>i = 0</td><td>a[0][0]</td><td>a[0][1]</td><td>a[0][2]</td></tr><tr><td>1</td><td>2</td><td>i = 1</td><td>a[1][0]</td><td>a[1][1]</td><td>a[1][2]</td></tr><tr><td>2</td><td>3</td><td>i = 2</td><td>a[2][0]</td><td>a[2][1]</td><td>a[2][2]</td></tr></table></div></div></div>			j = 0	j = 1	j = 2	0	1	i = 0	a[0][0]	a[0][1]	a[0][2]	1	2	i = 1	a[1][0]	a[1][1]	a[1][2]	2	3	i = 2	a[2][0]	a[2][1]	a[2][2]	说明基本的处理方法。					
		j = 0	j = 1	j = 2																										
0	1	i = 0	a[0][0]	a[0][1]	a[0][2]																									
1	2	i = 1	a[1][0]	a[1][1]	a[1][2]																									
2	3	i = 2	a[2][0]	a[2][1]	a[2][2]																									
46	<div><div><div><div><div>源程序段</div><div><pre># define MAXM 3 # define MAXN 3 int a[MAXM][MAXN]; scanf ("%d%d", &amp;m, &amp;n); for ( i = 0; i &lt; m; i++){     for ( j = 0; j &lt; n; j++){         a[i][j] = i + j;     } } for ( i = 0; i &lt; m; i++){     for ( j = 0; j &lt; n; j++){         printf ( "%4d", a[i][j] );     }     printf ( "\n" ); }</pre></div></div></div><div><table><tr><td></td><td>i</td><td>j</td><td></td></tr><tr><td>第1次</td><td>0</td><td>0</td><td>a[0][0] = 0</td></tr><tr><td>第2次</td><td>0</td><td>1</td><td>a[0][1] = 1</td></tr><tr><td>第3次</td><td>1</td><td>0</td><td>a[1][0] = 1</td></tr><tr><td>第4次</td><td>1</td><td>1</td><td>a[1][1] = 2</td></tr><tr><td>第5次</td><td>2</td><td>0</td><td>a[2][0] = 2</td></tr><tr><td>第6次</td><td>2</td><td>1</td><td>a[2][1] = 3</td></tr></table></div></div></div>		i	j		第1次	0	0	a[0][0] = 0	第2次	0	1	a[0][1] = 1	第3次	1	0	a[1][0] = 1	第4次	1	1	a[1][1] = 2	第5次	2	0	a[2][0] = 2	第6次	2	1	a[2][1] = 3	展示并运行程序，并模拟单步执行，以查看各下标的变化情况。
	i	j																												
第1次	0	0	a[0][0] = 0																											
第2次	0	1	a[0][1] = 1																											
第3次	1	0	a[1][0] = 1																											
第4次	1	1	a[1][1] = 2																											
第5次	2	0	a[2][0] = 2																											
第6次	2	1	a[2][1] = 3																											
47	<div><div><div><div><div>矩阵的输入</div><div><pre>int a[MAXM][MAXN]; printf ("Enter m, n: "); scanf ("%d%d", &amp;m, &amp;n); printf ("Enter %d integers: \n", m*n); for ( i = 0; i &lt; m; i++){     for ( j = 0; j &lt; n; j++){         scanf ("%d", &amp;a[i][j]);     } }  for (j = 0; j &lt; n; j++){     for (i = 0; i &lt; m; i++){         scanf ("%d", &amp;a[i][j]);     } }</pre></div></div></div><div><div>Enter m, n: 3 2 Enter 6 integers: 6 3 10 -9 5 -1 max = a[1][0] = 10</div><div><table><tr><td></td><td></td><td>j = 0</td><td>j = 1</td><td>j = 2</td></tr><tr><td>i = 0</td><td>a[0][0]</td><td>a[0][1]</td><td>a[0][2]</td><td></td></tr><tr><td>i = 1</td><td>a[1][0]</td><td>a[1][1]</td><td>a[1][2]</td><td></td></tr><tr><td>i = 2</td><td>a[2][0]</td><td>a[2][1]</td><td>a[2][2]</td><td></td></tr></table></div></div></div></div>			j = 0	j = 1	j = 2	i = 0	a[0][0]	a[0][1]	a[0][2]		i = 1	a[1][0]	a[1][1]	a[1][2]		i = 2	a[2][0]	a[2][1]	a[2][2]		若内外循环控制交换，查看变化情况。								
		j = 0	j = 1	j = 2																										
i = 0	a[0][0]	a[0][1]	a[0][2]																											
i = 1	a[1][0]	a[1][1]	a[1][2]																											
i = 2	a[2][0]	a[2][1]	a[2][2]																											
48	<div><div><div>矩阵与二维数组</div><div><pre>int a[N][N];      N是正整数 a[i][j]: i、j的取值范围 [0, N-1]</pre></div><div>用二维数组a表示N*N方阵时，对应关系： <table><tr><td>a[0][0]</td><td>a[0][1]</td><td>a[0][2]</td><td>副对角线</td><td>i+j == N-1</td></tr><tr><td>a[1][0]</td><td>a[1][1]</td><td>a[1][2]</td><td>上三角</td><td>i &lt;= j</td></tr><tr><td>a[2][0]</td><td>a[2][1]</td><td>a[2][2]</td><td>下三角</td><td>i &gt;= j</td></tr><tr><td></td><td></td><td></td><td>主对角线</td><td>i == j</td></tr></table></div></div></div>	a[0][0]	a[0][1]	a[0][2]	副对角线	i+j == N-1	a[1][0]	a[1][1]	a[1][2]	上三角	i <= j	a[2][0]	a[2][1]	a[2][2]	下三角	i >= j				主对角线	i == j	说明二维数组的下标与矩阵（方阵）的对应关系。 提醒：各对应关系时处理矩阵的基础。								
a[0][0]	a[0][1]	a[0][2]	副对角线	i+j == N-1																										
a[1][0]	a[1][1]	a[1][2]	上三角	i <= j																										
a[2][0]	a[2][1]	a[2][2]	下三角	i >= j																										
			主对角线	i == j																										

49

### 例7-9 方阵转置

输入一个正整数 $n$  ( $1 \leq n \leq 6$ ), 根据下式生成一个 $n \times n$ 的方阵, 然后将该方阵转置 (行列互换) 后输出。

$a[i][j] = i * n + j + 1$  ( $0 \leq i, j \leq n-1$ )

分析: `int a[6][6]; n=3时`

1	2	3		1	4	7
4	5	6	→	2	5	8
7	8	9		3	6	9

$a[i][j]$	↔	$a[j][i]$
$a[0][1]$	↔	$a[1][0]$
$a[0][2]$	↔	$a[2][0]$
$a[1][2]$	↔	$a[2][1]$

通过例 7-9 方阵转置说明使用二维数组的方法。

注意:

确定哪部分元素需要交换? 上三角或下三角元素, 不能全部。

转置前后的下标对应关系。

50

### 例7-9 源程序段

```
/* 行列互换*/
for (i = 0; i < n; i++) {
    for (j = 0; j < n; j++) {
        if (i <= j) {
            temp = a[i][j];
            a[i][j] = a[j][i];
            a[j][i] = temp;
        }
    }
}
```

主对角线:  $i == j$   
上三角:  $i <= j$   
下三角:  $i > j$

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

给出程序段, 重点说明行列互换。

考虑:

为什么交换时必须满足  $i <= j$ ?

注意只遍历上三角的方法, 可推广到遍历下三角。

只遍历上三角的方法二。

若遍历下角如何改?

51

```
/* 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;
    }
}
```

### 例7-9 思考

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

让学生思考为什么这个程序不能实现转置?

若对所有元素进行遍历交换, 则每个元素均被交换两次, 相当于没有交换。

52

### 例7-10 计算天数

定义函数 `day_of_year (year, month, day)`, 计算并返回年 `year`、月 `month` 和日 `day` 对应的是该年的第几天。

`day_of_year (2000, 3, 1)` 返回61

`day_of_year (1981, 3, 1)` 返回60

月	0	1	2	3	4	5	6	7	8	9	10	11	12
非闰年	0	31	28	30	30	31	30	31	31	30	31	30	31
闰年	0	31	29	30	30	31	30	31	31	30	31	30	31

```
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 }
}
```

例 7-10 说明如何在实际问题中使用二维数组编写程序。

分析问题, 得出表格数据与二维数组的关系。

53	<div>例7-10 函数</div> <pre>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 = (year4==0 &amp;&amp; year%100!=0)    (year%400==0);     for ( k = 1; k &lt; month; k++){         day = day + tab[leap][k];     }      return day; }</pre>	<p>展示运行例 7-10 程序，说明：</p> <p>为了使得列下标与月分对应，如何不用下标为 0 的单元？</p> <p>如何使用行下标与是否闰年对应？</p>						
54	<div>7.3 判断回文</div> <p>例7-11 输入一个以回车符为结束标志的字符串（少于80个字符），判断该字符串是否为回文。</p> <p>□回文：字符串中心对称</p> <ul style="list-style-type: none"><li>■ "noon" √</li><li>■ "radar" √</li><li>■ "reader" ×</li></ul> <div>7.3.1 程序解析</div> <div>7.3.2 一维字符数组</div> <div>7.3.3 字符串</div> <div>7.3.4 使用字符串编程</div>	<p>引导学生分析题目：</p> <p>这是一个有关字符串的问题，首先需解释什么叫字符串？</p> <p>先考虑：</p> <p>如何存储一个字符串？</p> <p>如何处理字符串？</p>						
55	<div>7.3.1 程序解析-判断回文</div> <pre>#define MAXLINE 80 int main ( void ) {     int i, k;     char line[MAXLINE];     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 &lt; k ) {         if ( line[i] != line[k] ) {             break;         }         i++; k--;     }     if ( i &gt;= k ) { printf("It is a plindrome\n"); }     else { printf("It is not a plindrome\n"); }     return 0; }</pre> <div>Enter a string: radar It is a plindrome</div> <div>Enter a string: reader It is not a plindrome</div>	<p>展示、运行例 7-11 程序。</p> <p>解读程序，可分以下几点说明：定义、输入字符串、处理方法、输出几部分。略过具体的细节，可重点点一下字符数组 s 和 '\0' 的概念。</p>						
56	<div>7.3.2 一维字符数组</div> <ul style="list-style-type: none"><li>■ 字符串的存储和运算可以用一维字符数组实现</li><li>■ 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。</li></ul> <p>char str[80];</p> <p>定义一个含有80个字符型元素的数组str</p> <p>char t[5] = {'H', 'a', 'p', 'p', 'y'};</p> <p>初始化数组 t</p> <div><table><tr><td>t</td><td>H</td><td>a</td><td>p</td><td>p</td><td>y</td></tr></table><p>t[0] t[1]      t[4]</p></div> <p>输出数组 t 的所有元素</p> <pre>for ( i = 0; i &lt; 5; i++ ){     putchar ( t[i] ); }</pre>	t	H	a	p	p	y	<p>说明什么是一维字符数组？它与前面所讲的一维数组的区别。</p>
t	H	a	p	p	y			

57	<div><div></div><h3>一维字符数组</h3><pre>char t[5] = {'H', 'a', 'p', 'p', 'y'}; static char s[6] = {'H', 'a', 'p', 'p', 'y'};</pre><p>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’};</p><table><tr><td>t</td><td>H</td><td>a</td><td>p</td><td>p</td><td>y</td><td>s</td><td>H</td><td>a</td><td>p</td><td>p</td><td>y</td><td>\0</td></tr><tr><td></td><td>t[0]</td><td>t[1]</td><td></td><td>t[4]</td><td></td><td>s[0]</td><td>s[1]</td><td></td><td></td><td></td><td>s[5]</td><td></td></tr></table></div>	t	H	a	p	p	y	s	H	a	p	p	y	\0		t[0]	t[1]		t[4]		s[0]	s[1]				s[5]		说明“\0”的特征。
t	H	a	p	p	y	s	H	a	p	p	y	\0																
	t[0]	t[1]		t[4]		s[0]	s[1]				s[5]																	
58	<div><div></div><h3>7.3.3 字符串</h3><p>字符串常量 用一对双引号括起来的字符序列 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<b>有效长度</b>：有效字符的个数</p></div>	重点说明什么是字符串？以及字符串的特征，即结束标志“\0”。 提醒： “\0”不是字符串有效字符； 字符串的长度是有效字符的个数； 字符串常量的表示法，即用双引号括起来。																										
59	<div><div></div><h3>字符串与一维字符数组</h3><p>字符串：一个特殊的一维字符数组</p><ul style="list-style-type: none"><li>把字符串放入一维字符数组（存储）</li><li>对字符串的操作 ==&gt; 对字符数组的操作</li></ul></div>	说明字符串与一维字符数组的区别。 提醒： C 语言将字符串作为一个特殊的一维字符数组来处理。																										
60	<div><div></div><h3>1. 字符串的存储—数组初始化</h3><p>字符串可以存放在一维字符数组中 static char s[6] = { ‘H’, ‘a’, ‘p’, ‘p’, ‘y’, ‘\0’ }; 字符数组初始化：用字符串常量 static char s[6] = { “Happy” }; static char s[6] = “Happy”; 数组长度 ≥ 字符串的有效长度 + 1 char t[5];    “Happy” 能存入 t 吗？</p><table><tr><td>s</td><td>H</td><td>a</td><td>p</td><td>p</td><td>y</td><td>\0</td></tr><tr><td></td><td>s[0]</td><td>s[1]</td><td></td><td></td><td></td><td>s[5]</td></tr></table></div>	s	H	a	p	p	y	\0		s[0]	s[1]				s[5]	说明定义字符串以及初始化的方法。 提醒： 一维字符数组的长度至少是字符串长度加 1。 字符串必须以“\0”作为结束标志。												
s	H	a	p	p	y	\0																						
	s[0]	s[1]				s[5]																						



61	<div> <div>字符串的存储</div> <div> <pre>auto char s[80] = "Happy";</pre> <p>字符串遇 <code>'\0'</code> 结束</p> <p>第一个 <code>'\0'</code> 前面的所有字符和 <code>'\0'</code> 一起构成了字符串 <code>"Happy"</code></p> <p><code>'\0'</code> 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 <code>'\0'</code> 组成</p> <div> <div>s</div> <div>H a p p y \0 ? ?</div> <div>s[0] s[1] s[5]</div> </div> </div> </div>	字符串由有效字符和 <code>'\0'</code> 组成。
62	<div> <div>2. 对字符串的操作</div> <div> <ul style="list-style-type: none"> <li>把字符串放入一维字符数组（存储）</li> <li>对字符串的操作 ==&gt; 对字符数组的操作</li> </ul> <p>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</p> <p>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 <code>'\0'</code> 之前的字符都是字符串的有效字符，一般用结束符 <code>'\0'</code> 来控制循环</p> <p>循环条件：<code>s[i] != '\0'</code></p> </div> </div>	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。
63	<div> <div>输出字符串</div> <div> <pre>for ( i = 0; s[i] != '\0'; i++){     putchar ( s[i] ); } for ( i = 0; i &lt; 80; i++){     putchar ( s[i] ); } for ( i = 0; i &lt; 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> </div> </div>	给出三种不同的字符串输出方式，主要区别在于循环条件不同，分析每个循环输出的内容。
64	<div> <div>3. 字符串的存储—赋值和输入</div> <div> <ul style="list-style-type: none"> <li>把字符串放入一维字符数组（存储）</li> <li>对字符串的操作 ==&gt; 对字符数组的操作</li> </ul> <p>存储</p> <div> <div> <ul style="list-style-type: none"> <li>数组初始化</li> <li>赋值</li> <li>输入</li> </ul> </div> <div> <p>static char s[6] = "a";</p> <p>s[0] = 'a'; s[1] = '\0';</p> </div> </div> <div> <div>区分"a"和'a'</div> <div> <p>"a" 2个字符'a'和'\0'</p> <p>'a' 1个字符常量</p> </div> </div> <p><code>'\0'</code> 代表空操作，无法输入</p> <p>输入时，设定一个输入结束符</p> <p>将输入结束符转换为字符串结束符 <code>'\0'</code></p> </div> </div>	提醒： “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。 当键盘输入时，由于 <code>'\0'</code> 无法输入，故必须设定一个输入结束符，通常用回车（ <code>'\n'</code> ）作为输入结束符，但需转换成 <code>'\0'</code> 放入。

65	<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"> <li>■ 存储：把字符串放入一维字符数组 数组初始化、赋值、输入</li> </ul> <p>对字符串的操作 ==&gt; 对字符数组的操作</p> <ul style="list-style-type: none"> <li>■ 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 检测字符串结束符 '\0'</li> </ul>	<p>说明字符串和一维数组一样，字符串的编程离不开循环，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>
66	<h3>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>说明输入方式方法，特别注意：定义字符串长度以及用回车 '\n' 结束。</p>
67	<h3>源程序段</h3> <pre># define MAXLINE 80  char str[MAXLINE]; printf ( "Enter a string: " ); i = 0; while ( (str[i] = getchar()) != '\n' ){     i++; } str[i] = '\0'; /* 输入结束符=&gt;字符串结束符 */ count = 0; for ( i = 0; str[i] != '\0'; i++ ){     if ( str[i] &lt;= '9' &amp;&amp; str[i] &gt;= '0' ){         count++;     } } printf ( "count = %d\n", count );</pre> <p>如何改变输入结束符？</p> <p>字符串的输入</p> <p>能省略 str[i] = '\0' 吗？</p> <p>Enter a string: It's 512 count = 3</p> <p>str    i   t   '   s      5   1   2   \0   ?   ?       0 1 2 3 4 5 6 7 8</p>	<p>展示运行程序，详细说明：</p> <p>虚线框内程序段，即输入方法以及如何放到字符数组中，包括 '\0'。</p> <p>说明如何操作字符串来统计数字字符。</p> <p>考虑：</p> <p>能省略 str[i] = '\0' 语句吗？为什么？</p>
68	<h3>例7-12 凯撒密码</h3> <p>输入一个以回车符为结束标志的字符串，再输入一个正整数 offset，用凯撒密码将其加密后输出。</p> <ul style="list-style-type: none"> <li>□ 将明文中的所有字母都在字母表上向后偏移 offset 位后被替换成密文</li> <li>□ 当偏移量 offset 是 2 时，表示所有的字母被向后移动 2 位后的字母替换             <ul style="list-style-type: none"> <li>■ 所有的字母 A 将被替换成 C，字母 B 将变为 D，...，字母 X 变成 Z，字母 Y 则变为 A，字母 Z 变为 B。</li> </ul> </li> </ul>	<p>例 7-12 重点讲解加密的规则以及字符变换的方法</p>

69

# define M 26

例7-12 源程序段

```
scanf("%d",&offset);
if(offset>=M)
    offset=offset%M; /* 移位效果相当于取其余数 */
/* 加密 */
for(i=0; str[i]!='\0'; i++){
    if(str[i]>='A' && str[i]<='Z'){
        if((str[i]-'A'+offset)<M){
            str[i]=str[i]+offset;
        }else{
            str[i]=str[i]-(M-offset); /* 如果向后越界 */
        }
    }else if(str[i]>='a' && str[i]<='z'){
        if((str[i]-'a'+offset)<M){
            str[i]=str[i]+offset;
        }else{
            str[i]=str[i]-(M-offset);
        }
    }
}
```

Enter a string: Hangzhou

Enter offset: 2

After being encrypted:Jcpibjqw

运行示例程序

分析不同字符的处理方法

70

例7-13 字符转换

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

解读题目要求，分析程序的输入与输出结果

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'		

重点介绍将数字字符转换为十进制数字的方法

72

# include <stdio.h>

# define MAXLINE 10

例7-13 源程序

int main(void)

{ int i, n; char s[MAXLINE];

printf ( "Enter a string: " );

i = 0;

while ( (s[i] = getchar()) != '\n' ) {

i++;

s[i] = '\0';

a 1 2 d 3 0 ?

0 1 2 3 4

n = 0;

/\* 将字符串转换为整数 \*/

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: a12d3

digit = 123

运行示例程序

73

a

1

2

d

3

10

?

0

1

2

3

4

例7-13 思考

Enter a string: a12d3  
digit = 123

n = 0;  
for ( i = 0; s[i] != '\0'; i++) {  
if ( s[i] <= '9' && s[i] >= '0' ) {  
n = n \* 10 + (s[i] - '0');  
}  
else {  
break; }  
}  
digit = ?

i	s[i]	s[i]-'0'	n = n*10+(s[i]-'0')
0	'a'		
1	'1'	1	0*10+1 = 1
2	'2'	2	1*10+2 =12
3	'd'		
4	'3'	3	12*10+3 =123
5	'\0'		

思考：如果程序中加入 else break，运行结果有什么变化

74

例7-14 进制转换

输入一个以回车结束的字符串（少于80个字符），滤去所有的非十六进制字符后，组成一个新字符串（十六进制形式），输出该字符串并将其转换为十进制数后输出。

例如：  
输入字符串：zx1?m a0 lkbq  
去掉非十六进制后组成新字符串：1a0b  
转换为十进制为：6667

解读题目要求

75

例7-14 分析

输入字符串：zx1?ma0!kbbq#  
滤去非十六进制后组成新字符串：1a0b  
转换为十进制整数：6667  
分析：  
数组长度取上限10  
以 '#' 做为输入结束符  
"zx1?ma0!kbbq" ==> "1a0b"  
"1a0b" ==> 6667

分析程序的输入和操作过程：  
输入字符串->生产十六进制字符串->转换为十进制数->输出


76

生成十六进制字符串

"zx1?ma0!kbbq" ==> "1a0b"  
str hexad  
k = 0; /\* k: 新字符串hexad的下标 \*/  
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]; /\* 放入新字符串 \*/  
k++;  
}  
}  
hexad[k] = '\0'; /\* 新字符串结束标记 \*/

重点分析生成十六进制字符串的方法

77	<div><div>转换为十进制整数</div><pre>"1a0b" ==&gt; 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] &gt;= '0' &amp;&amp; hexad[i] &lt;= '9' ){         number = number * 16 + hexad[i] - '0';     }else if ( hexad[i] &gt;= 'A' &amp;&amp; hexad[i] &lt;= 'F' ){         number = number * 16 + hexad[i] - 'A' + 10;     }else if ( hexad[i] &gt;= 'a' &amp;&amp; hexad[i] &lt;= 'f' ){         number = number * 16 + hexad[i] - 'a' + 10;     } }</pre></div>	重点分析将十六进制字符串转换为十进制数的方法																		
78	<div><div><div>输入原字符串 str</div><div>滤去非16进制 字符后生成新 字符串hexad</div><div>把字符串hexad 转换成十进制 整数number</div></div><div><pre>i = 0; while ( (str[i] = getchar()) != '#' ){     i++; } str[i] = '\0';  k = 0; for(i = 0; str[i] != '\0'; i++){     if(str[i]&gt;='0'&amp;&amp;str[i]&lt;='9'   str[i]&gt;='a'&amp;&amp;str[i]&lt;='f'  str[i]&gt;='A'&amp;&amp;str[i]&lt;='F'){         hexad[k] = str[i];         k++;     } } hexad[k] = '\0';  number = 0; for(i = 0; hexad[i] !='\0'; i++){     if(hexad[i] &gt;= '0' &amp;&amp;hexad[i] &lt;= '9'){         number = number * 16 + hexad[i] - '0';     }else if(hexad[i] &gt;= 'A' &amp;&amp;hexad[i] &lt;= 'F'){         number = number * 16 + hexad[i] - 'A' + 10;     }else if(hexad[i] &gt;= 'a' &amp;&amp;hexad[i] &lt;= 'f'){         number = number * 16 + hexad[i] - 'a' + 10;     } }</pre></div><div>程序段</div></div>	总结本题的编程思路和方法																		
79	<div><div>字符串小结</div><p>字符串：一个特殊的一维字符数组 <b>'\0'</b></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"><li>■ 字符数组初始化： <code>static char s[80] = "Happy";</code></li><li>■ 赋值： <code>s[0] = 'a'; s[1] = '\0';</code></li><li>■ 输入： 输入结束符 ==&gt; 字符串结束符 <b>'\0'</b></li></ul><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><tr><td></td><td>s[0]</td><td>s[1]</td><td></td><td></td><td></td><td>s[5]</td><td></td><td></td></tr></table></div>	s	H	a	p	p	y	\0	?	?		s[0]	s[1]				s[5]			总结字符串的相关知识。 强调字符串在程序设计中的重要性。
s	H	a	p	p	y	\0	?	?												
	s[0]	s[1]				s[5]														
80	<div><p>■ 把字符串放入一维字符数组（存储）</p><p>■ 对字符串的操作 ==&gt; 对字符数组的操作</p><p>只针对字符串的有效字符和字符串结束符 <b>'\0'</b></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><tr><td></td><td>s[0]</td><td>s[1]</td><td></td><td></td><td></td><td>s[5]</td><td></td><td></td></tr></table></div>	s	H	a	p	p	y	\0	?	?		s[0]	s[1]				s[5]			
s	H	a	p	p	y	\0	?	?												
	s[0]	s[1]				s[5]														

81	<div> <div>  <h2>本章总结</h2> <ul style="list-style-type: none"> <li>■ 一维数组： <ul style="list-style-type: none"> <li>□ 定义、初始化、引用</li> <li>□ 使用一维数组：选择排序、二分查找</li> </ul> </li> <li>■ 二维数组 <ul style="list-style-type: none"> <li>□ 定义、初始化、引用</li> <li>□ 使用二维数组：矩阵</li> </ul> </li> <li>■ 字符串 <ul style="list-style-type: none"> <li>□ 字符串数组与字符串</li> <li>□ 字符串的存储</li> <li>□ 字符串的操作</li> </ul> </li> <li>■ 使用数组进行程序设计</li> </ul> </div> <div> <ul style="list-style-type: none"> <li>• 理解数组的基本概念及在内存中的存放方式</li> <li>• 理解字符串的概念</li> <li>• 能使用一维数组编程</li> <li>• 能使用二维数组编程</li> <li>• 能使用字符串编程</li> <li>• 掌握常用算法</li> </ul> </div> </div>	归纳总结本章的各重要知识点及能力要求。
----	---	---------------------

## 7.3 练习与习题参考答案

### 7.3.1 练习参考答案

练习 7-1 如果将例 7-3 程序中 for 循环中的代码做如下修改，输出结果有变化吗？假设输入数据不变，输出什么？

```
if ( a[i] == x ){
    printf ( "Index is %d\n", i );
    flag = 1;
    break;
}
```

解答：

加入 break 语句后，一旦找到第一个与 x 相同的值就跳出循环。当输入数据仍为 2 9 8 1 9 时，输出将是 index is 1。

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(void)
{
    int i, index, n, t;
    int a[10];
    printf("Input n :") ;
    scanf("%d", &n);
    printf("Input %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("Max = %d , index = %d \n",a[index],index);
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：

```

#include <stdio.h>
int main(void)
{
    int i, n, temp;
    int a[10];
    printf("Input n: ");
    scanf("%d", &n);
    printf("Input %d Integers: ",n);
    for(i=0; i<n; i++)
        scanf("%d", &a[i]);
    for(i=0; i<n/2; i++){
        temp=a[i];
        a[i]=a[n-1-i];
        a[n-1-i]=temp;
    }
    printf("After reversed: ");
    for(i=0; i<n; i++)
        printf("%d ", a[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>
int main(void)
{
    int n, m, i, j, t, f1, f2;
    int a[10], b[10], c[20];
    t=0;
    scanf("%d", &n);
    for(i=0; i<n; i++)
        scanf("%d", &a[i]);

```

```

scanf("%d", &m);
for(i=0; i<m; i++)
scanf("%d", &b[i]);
for(i=0; i<n; i++){
    f1=1;
    for(j=0;j<t;j++)
        if(c[j]==a[i]){
            f1=0;
            break;
        }
    if(f1==1){
        f2=1;
        for(j=0;j<m;j++)
            if(a[i]==b[j]){
                f2=0;
                break;
            }
        if(f2==1) c[t++]=a[i];
    }
}
for(i=0; i<m; i++){
    f1=1;
    for(j=0;j<t;j++)
        if(c[j]==b[i]){
            f1=0;
            break;
        }
    if(f1==1){
        f2=1;
        for(j=0;j<n;j++)
            if(b[i]==a[j]){
                f2=0;
                break;
            }
        if(f2==1) c[t++]=b[i];
    }
}
for(i=0;i<t;i++)
    printf("%d ",c[i]);
return 0;
}

```

练习 7-5 用下列程序段替换例 7-8 中的对应程序段，假设输入数据不变，输出什么？与例 7-8 的输出结果一样吗？为什么？



```

/* 输入矩阵—按照先列后行的顺序 */
for (j = 0; j < n; j++){
    for (i = 0; i < m; 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 7-8 不一样,因为当用上述方式输入是,二维数组中存放值如下:

```

6    -9
3     5
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 1 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>
int main(void)
{
    int a[6][6],i,j,n,sum;
    printf("Input n : ");
    scanf("%d",&n);

```

```

printf("Input array:\n ");
for (i=0;i<n;i++)
    for(j=0;j<n;j++)
        scanf("%d",&a[i][j]);
sum=0;
for (i=0;i<n;i++)
    for (j=0;j<n;j++)
        if( i!=n-1 && j!=n-1 && i+j!=n-1)    sum+=a[i][j];
printf("sum = %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。

```

#include <stdio.h>
int main(void)
{
    int n, m, i, j, k;
    int a[6][6], b[6][6];
    scanf("%d%d", &m, &n);
    for(i=0; i<n; i++)
        for(j=0;j<n;j++)
            scanf("%d", &a[i][j]);
    m=m%6;
    for(j=0;j<n;j++)
        for(i=0;i<n;i++)
            b[i][(n-m+j)%n]=a[i][j];
    for(i=0; i<n; i++){
        for(j=0;j<n;j++)
            printf("%d ", b[i][j]);
        putchar('\n');
    }
    return 0;
}

```

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

解答：

```

#include "stdio.h"
int main(void)
{
    int year, month, day, day_year;
    int day_of_year(int year, int month, int day);
}

```

```

    printf("Input year,month,day: ");
    scanf("%d%d%d",&year, &month, &day);
    day_year=day_of_year(year, month, day);
    printf("Days of year: %d\n", day_year);
    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>
int main(void)
{
    int i,flag,index;
    char ch,s[80];
    scanf("%c", &ch);
    getchar();
    gets(s);
    flag=0;
    for(i=0;s[i]!='\0';i++)
        if(s[i]==ch){
            index=i;
            flag=1;
        }
    if(flag==1)
        printf("%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>
int main(void)
{
    int i,j;
    char s[80],t;
    gets(s);
    for(j=0;s[j]!='\0';j++);
    j--;

```

```

for(i=0;i<j;i++,j--)
{
    t=s[i];
    s[i]=s[j];
    s[j]=t;
}
puts(s);
return 0;
}

```

## 7.3.2 习题参考答案

### 一. 选择题

1	2	3	4	5	6
A	D	D	C	C	B

### 二. 填空题

1	输入 4, 输出 2 输入 5, 输出 3 输入 12, 输出 5 输入 -5, 输出 0	2	(1) i=1 (2) x[i-1]
3	x < a[i] j = n-1; j >= i; j-- a[i] = x	4	(1) 7 (2) 5
5	(A) 1#2#3#4#5#6# (B) 1#4#2#5#3#6#	6	(1) a[i][j] != a[j][i] (2) found = 0; (3) found == 0;
7	str1[i] != '\0' str2[i] = str1[i]; str2[i] = '\0';	8	str[i] != '\0' str[i] != '' j++; str[j] = '\0';

### 三. 程序设计题

1. 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(void)
{
    int i, index, k, n, temp;
    int a[10];
    printf("Input n: ");
    scanf("%d", &n);
    printf("Input %d integers: ");

```

```

for(i=0; i<n; i++)
    scanf("%d", &a[i]);
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;
}
printf("After sorted:");
for(i=0; i<n; i++)
    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, 求出现次数最多的各位数字。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(void)
{
    int i, n, m, max;
    static int a[10];
    printf("Enter n:");
    scanf("%d", &n);
    printf("Enter %d integers:",n);
    for(i=0; i<n; i++)
    {
        scanf("%d", &m);
        if(m<0) m=-m;
        do{ a[m%10]++;
            m=m/10;
        }while(m!=0);
    }
    max=a[0];
    for(i=1; i<10; i++)
        if(a[i]>max) max=a[i];
    printf("%d:",max);
    for(i=0; i<10; i++)
        if(a[i]==max) printf("%d ", i);
}

```

```

    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>
int main(void)
{
    int a[6][6], flag, i, j, n;
    printf("Input n: ");
    scanf("%d", &n);
    printf("Input array: \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)
        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  $n$  ( $1 \leq m, n \leq 6$ )，然后输入该  $m$  行  $n$  列矩阵  $a$  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：

```

#include <stdio.h>
int main(void)
{
    int a[6][6], i, j, m, n, sum;
    printf("Input m,n: ");
    scanf("%d%d", &m, &n);
    printf("Input array: \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("sum of row %d is %d\n",i,sum);
    }
    return 0;
}

```

5. 找鞍点。输入一个正整数  $n(1 \leq n \leq 6)$  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，就输出它的下标，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>
int main(void)
{
    int flag,i,j,k,row,col,n,a[6][6];
    printf("Input n: ");
    scanf("%d",&n);
    printf("Input array:\n ");
    for(i=0; i<n; i++)
        for(j=0; j<n; j++)
            scanf("%d",&a[i][j]);
    for(i=0;i<n;i++){
        flag=1; col=0;
        for(j=0;j<n;j++)
            if (a[i][col]<a[i][j])    col=j;
        for (k=0; k<n; k++)
            if(a[i][col]>a[k][col] ){
                flag=0; break;
            }
        if(flag){
            row=i; break;
        }
    }
    if(flag)
        printf("a[%d][%d]=%d\n", row, col,a[row][col]);
    else
        printf("NO\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。大写辅音字母是指除'A','E','I','O','U'以外的大写字母。试编写相应程序。

解答：

```

#include <stdio.h>

```

```

int main(void)
{
    int count,i;
    char ch,str[80];
    printf("Input a string: ");
    i=0;
    while((ch=getchar())!='\n'){
        str[i++]=ch;
    }
    str[i]='\0';
    count=0;
    for(i=0;str[i]!='\0';i++)
        if(str[i]<='Z'&&str[i]>'A'&&str[i]!='E'&&str[i]!='T'&&str[i]!='O'&&str[i]!='U')
            count++;
    printf("count = %d\n",count);
    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。

原字母    对应字母

A → Z

B → Y

C → X

D → W

.....

X → C

Y → B

Z → A

解答：

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

```
#include "string.h"
```

```
int main(void)
```

```

{
    int i;
    char ch,str[80];
    printf("Input a string: ");
    i=0;
    while((ch=getchar())!='\n'){
        str[i++]=ch;
    }
    str[i]='\0';
    for(i=0;str[i]!='\0';i++)
        if(str[i]<='Z'&&str[i]>='A')

```



```

        str[i]='A'+'Z'-str[i];
    printf("After replaced:");
    for(i=0;str[i]!='\0';i++)
        putchar(str[i]);
    putchar('\n');
    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>
int main(void)
{
    int i,k,number,flag;
    char str[80],s[80];
    i = 0;
    while ( (str[i] = getchar( )) != '#' ){
        i++;
    }
    str[i] = '\0';
    k= 0;
    flag=1;
    for(i = 0; str[i] != '\0'; i++){
        if(str[i]=='-' && k==0) flag=-1;
        if(str[i]>='0'&&str[i]<='9'
        ||str[i]>='a'&&str[i]<='f'||str[i]>='A'&&str[i]<='F'){
            s[k] = str[i];
            k++;
        }
    }
    s[k] = '\0';

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

```

```

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

return 0;
}

```

## 7.4 实验指导教材参考答案

### 7.1 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  $n$  个从小到大排好顺序的整数，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：略

#### 二. 基础编程题

(1) 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

(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：参见练习 7-2

(3) 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：参见习题程序设计题第 1 题

(4) 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：

```

#include <stdio.h>
int main(void)
{
    int i, index, n, t;
    int a[10];
    scanf("%d", &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;

```

```

t=a[0];
a[0]=a[index];
a[index]=t;
index=0;
for(i=1; i<n; i++)
    if(a[i] > a[index])
        index=i;
t=a[n-1];
a[n-1]=a[index];
a[index]=t;
printf("After swap:");
for(i=0; i<n; i++)
    printf("%d ", a[i]);
return 0;
}

```

(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，求出现次数最多的各位数字。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：参见习题程序设计题第 2 题

### 三. 改错题

查找整数：输入正整数  $n$  ( $1 \leq n \leq 20$ ) 和整数  $x$ ，再输入  $n$  个整数并存放在数组  $a$  中，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  $a$  中的最小下标；如果没有找到，输出 “Not Found”。

改错汇总：

错误行号： 12	正确语句： <u>scanf("%d", &amp;a[i]);</u>
错误行号： 16	正确语句： <u>if(a[i]==x) {</u>
错误行号： 21	正确语句： <u>if(flag==0) {</u>

### 四、拓展编程题

(1) 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

(2) 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>
int main(void)
{
    int i,j,n,m,t,flag,index;
    static int a[100][2];

```

```

scanf("%d",&n);
t=0;
for(i=1;i<=n;i++){
    scanf("%d",&m);
    flag=0;
    for(j=0;j<t;j++){
        if(a[j][0]==m){
            flag=1;
            a[j][1]++;
        }
    }
    if(flag==0){
        a[t][1]=1;
        a[t++][0]=m;
    }
}
index=0;
for(i=1;i<t;i++){
    if(a[i][1]>a[index][1]) index=i;
}
printf("%d %d",a[index][0],a[index][1]);
return 0;
}

```

(3) 组个最小数：给定数字 0-9 各若干个，你可以按照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>
int main(void)
{
    int i,k;
    int a[10];
    for(i=0;i<10;i++){
        scanf("%d",&a[i]);
    }
    k=1;
    while(a[k]==0 && k<10) k++;
    if(k<10) {
        printf("%d",k);
        a[k]--;
    }
    if(a[0]>0){
        while(a[0]>0){
            printf("0");
            a[0]--;
        }
    }
}

```

```

while(k<10){
    if(a[k]>0){
        printf("%d",k);
        a[k]--;
    }
    else k++;
}
return 0;
}

```

(4) 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>
int main(void)
{
    int n,i,k;
    int a[100][2],box[100];
    scanf("%d", &n);
    for(i=0;i<n;i++)
        scanf("%d",&a[i][0]);
    for(i=0;i<n;i++)
        box[i]=100;

    for(i=0;i<n;i++){
        for(k=0;k<n;k++){
            if(box[k]>=a[i][0]){
                box[k]-=a[i][0];
                a[i][1]=k+1;
                break;
            }
        }
    }
    for(i=0;i<n;i++)
        printf("%d %d\n",a[i][0], a[i][1]);
    for(k=0;box[k]<100;k++);
    printf("%d\n",k);
    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  $n$  ( $1 \leq m, n \leq 6$ )，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：略

## 二. 基础编程题

(1) 矩阵运算：读入一个正整数  $n$  ( $1 \leq n \leq 10$ )，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 ( $3 \leq m, n \leq 20$ ) 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>

int main(void)
{
    int m,n,i,j;
    int a[21][21];
    scanf("%d%d", &m, &n);
    for(i=1;i<=m;i++)
        for(j=1;j<=n;j++)
            scanf("%d",&a[i][j]);
    for(i=2;i<=m;i++)
        for(j=2;j<=n;j++)
            if(a[i][j]>a[i-1][j]&&a[i][j]>a[i][j+1]&&a[i][j]>a[i+1][j]&&a[i][j]>a[i][j-1])
                printf("%d %d %d\n",a[i][j],i,j);
    return 0;
}
```

(3) 计算天数：按照格式“yyyy/mm/dd”（即“年/月/日”）输入日期，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)` 计算并返回年 `year`、月 `month` 和日 `day` 对应的是该年的第几天。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  $n$  ( $1 \leq n \leq 10$ ) 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：参见习题程序设计题第 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：

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

```

int main(void)
{
    int n,i,j;
    int a[10][10];
    scanf("%d",&n);
    for(i=0;i<n;i++)
        for(j=0;j<=i;j++)
            if(j==0) a[i][j]=1;
            else if(i==j) a[i][j]=1;
            else a[i][j]=a[i-1][j-1]+a[i-1][j];
    for(i=0;i<n;i++){
        for(j=1;j<n-i;j++)
            printf(" ");
        for(j=0;j<=i;j++)
            printf("%4d",a[i][j]);
        putchar('\n');
    }
    return 0;
}

```

### 三. 改错题

方阵循环右移：输入两个正整数  $m$  和  $n$  ( $m \geq 1, 1 \leq n \leq 6$ )，然后输入  $n$  阶方阵  $a$  中的元素，将该方阵中的每个元素循环向右移  $m$  个位置。

改错汇总：

错误行号： 11    正确语句： for (i = 0; i < n; i++) {  
 错误行号： 19    正确语句： b[i][j] = a[i][j];

### 四、拓展编程题

(1) 找鞍点：一个矩阵元素的“鞍点”是指该位置上的元素值在该行上最大、在该列上最小。输入 1 个正整数  $n$  ( $1 \leq n \leq 6$ ) 和  $n$  阶方阵  $a$  中的元素，如果找到  $a$  的鞍点，就输出其下标，否则，输出“NONE”。假设方阵  $a$  至多存在 1 个鞍点。试编写相应程序。

解答：参见习题程序设计题第 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  $n \times n$  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  $n \times n$  的方阵里。输入一个正整数  $n$  ( $n < 10$ )，输出  $n$  阶的螺旋方阵，每个数字占 3 位。试编写相应程序。

解答：

```

#include <stdio.h>

int main()
{
    int N,i,j,n,num=1;
    int a[10][10]={0};
    scanf("%d",&N);

    for(n=0;n<=N/2;n++)

```

```

{
    for(j=n;j<=N-n-1;j++)
        a[n][j]=num++;
    for(i=n+1;i<N-n-1;i++)
        a[i][N-n-1]=num++;
    for(j=N-n-1;j>n;j--)
        a[N-n-1][j]=num++;
    for(i=N-n-1;i>n;i--)
        a[i][n]=num++;
}
//输出螺旋矩阵
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>
int main()
{
    int n,i,j,k,x,x1,x2,y1,y2,s=0,t=0;
    char a[10][10];
    scanf("%d",&n);
    getchar();
    for(i=0;i<2*n;i++){
        for(j=0;j<2*n;j++){
            {
                a[i][j]=getchar();
                getchar();
            }
        }
    }
    scanf("%d",&x);
    for(k=1;k<=x;k++)
    {
        scanf("%d%d%d%d",&x1,&y1,&x2,&y2);
    }
}

```



```

        if(a[x1-1][y1-1]==a[x2-1][y2-1] && a[x1-1][y1-1]!='*')
        {
            a[x1-1][y1-1]=a[x2-1][y2-1]='*';
            t++;
            if(t==2*n*n)
            {
                printf("Congratulations!\n");
                break;
            }
        }
        for(i=0;i<2*n;i++)
        {
            for(j=0;j<2*n;j++)
            {
                if(j==0)
                    printf("%c",a[i][j]);
                else
                    printf(" %c",a[i][j]);
            }
            putchar('\n');
        }
    }
    else {
        printf("Uh-oh\n");
        s++;
        if(s==3)
        {
            printf("Game Over\n");
            break;
        }
    }
}
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），对该字符串进行逆序，输出逆序后的字符串。

解答：略

### 二. 基础编程题

(1) 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。大写辅音字母是指除'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

(2) 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

(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。

原字母    对应字母

A → Z

B → Y

C → X

D → W

.....

X → C

Y → B

Z → A

解答: 参见习题程序设计第 7 题

(4) 凯撒密码: 为了防止信息被别人轻易窃取, 需要把电码明文通过加密方式变换成为密文。输入一个以回车符为结束标志的字符串 (少于 80 个字符), 再输入一个整数 offset, 用凯撒密码将其加密后输出。恺撒密码是一种简单的替换加密技术, 将明文中的所有字母都在字母表上偏移 offset 位后被替换成密文, 当 offset 大于零时, 表示向后偏移; 当 offset 小于零时, 表示向前偏移。例如, 当偏移量 offset 是 2 时, 表示所有的字母被向后移动 2 位后的字母替换, 即所有的字母 A 将被替换成 C, 字母 B 将变为 D, ..., 字母 X 变成 Z, 字母 Y 则变为 A, 字母 Z 变为 B; 当偏移量 offset 是 -1 时, 表示所有的字母被向前移动 1 位后的字母替换, 即所有的字母 A 将被替换成 Z, 字母 B 将变为 A, ..., 字母 Y 则变为 X, 字母 Z 变为 Y。

解答: 参见例 12

(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。如果在第一个十六进制字符之前存在字符“-”, 则代表该数是负数。试编写相应程序。

解答: 参见习题程序设计第 8 题

### 三. 改错题

数字字符转换: 输入一个以回车结束的字符串 (少于 80 个字符), 将其中第一次出现的连续的数字字符 ('0'~'9') 转换为整数, 遇到非数字字符则停止。例如, 将字符串 “x+y=35+z+9” 转换为整数是 35。

改错汇总:

错误行号: 9    正确语句: while((str[i] = getchar()) != '\n') {

错误行号: 12	正确语句: str[i]='\0';
错误行号: 15	正确语句: if(str[i] >= '0' && str[i] <= '9'){
错误行号: 24	正确语句: }
错误行号: 25	正确语句: else break;

#### 四、拓展编程题

(1) 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), 按照输入的顺序输出该字符串中所出现过的大写英文字母, 每个字母只输出一遍; 若无大写英文字母则输出“Not Found”。

解答:

```
#include<stdio.h>
int main(void){
    char str[80],newstr[80];
    int i=0,j,k;
    printf("Input a string: ");
    while((str[i]=getchar( ))!='\n')
        i++;
    str[i]='\0';
    k=0;
    for(i=0;str[i]!='\0';i++){
        if(ch>='A'&&ch<='Z')
        {
            for(j=0;j<k;j++)
                if(newstr[j]==str[i]) break;
            if(j>=k){
                newstr[k]=str[i];
                k++;
            }
        }
    }
    newstr[k]='\0';
    if(k==0)
        printf("Not Found");
    else
        for(i=0; newstr[i]!='\0'; i++)
            putchar(newstr[i]);
    putchar('\n');
    return 0;
}
```

(2) 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>
```

```

int main(void){
    char str[80],ch;
    int i=0,j,k;
    printf("Input a string: ");
    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;
    }
    for(i = 0; str[i] != '\0'; i++)
        putchar(str[i]);
    putchar('\n');
    return 0;
}

```