默认情况下，“数组”指我们在第 5 章中学习的数组，不包括动态数组（“动态数组”是用操作符 new 分配内存的数组，我们会在第 7 章介绍）。

1、定义数组

定义数组时，**必须用常量规定数组的大小**。**绝对禁止**用变量。

int n = 10; int a[n]; // 这是错的！因为 n 是变量【赋了值的变量也是变量!!!】
const int n = 10; int a[n]; // 这是对的，因为 n 是符号常量
const int n; cin >> n; int a[n]; // 这是错的【常量只能初始化，不可以赋值！不可以赋值!】

定义数组时如果**没有初始化**，那么所有元素的初始值都是**随机值**。
如果给**一部分元素初始化**，那么**其余元素的初始值都是零**。

数组≠数组元素。对于一维数组而言，**数组元素**是单个**变量**。

2、数组名

不带中括号的数组名表示该数组的起始地址。
用 cout<<输出地址分两种情况：
①如果是字符类型的地址，那么无论该地址是字符数组名、字符指针还是字符变量的地址，都输出字符串；
②如果不是字符类型的地址，那么输出地址。

不能给数组名赋值。
例如 int x[10]; cin>>x; // 错误，无法编译。
虽然 char y[10]; cin>>y; // 可以编译，但这不是给 y 赋值，而是给 y 的元素赋值。

3、数组元素的下标

对于元素**个数**为常量 **N** 的数组，下标的合法范围是从 **0 到 N-1**。**不能用浮点数**作为下标。
【很多同学把元素个数和下标最大值搞混了，**从第 0 个开始数，你也是数了一个数啊】**

使用 **for 循环**处理数组元素时要**注意下标的取值**。
如果在循环过程中会让下标超出合法范围，很可能产生错误结果。
尤其要注意 **a[i+1]、b[j-1]**之类的情况，避免 i+1 可取到 N、j-1 可取到-1。

4、基于数组的排序

二分查找的前提是数组中的元素已经按数值顺序排列好。

选择排序：详见课本上代码清单 6-10。
重点注意：
1、别漏了 rh=lh;,
2、if 语句没有大括号。

冒泡排序：详见课本上代码清单 6-11。
重点注意：
1、别漏了 flag=false;,
2、外层 for 循环是从 i=1 开始的，而内层 for 循环是从 j=0 开始的。

5、二维数组

二维数组的**第 i 行的末尾**和**第 i+1 行的开头**在**内存中是连接在一起**的。
二维数组的每一个元素都是一个一维数组。
如有 int a[4][5];，那么 a[i]是一维 int 数组。
这意味着 a[i]具备数组名的特征：**可以输出，不可被赋值。**

6、字符串的输入输出

当使用【cin>>字符数组名】来输入**字符串**时，若输入**字符的数量大于数组长度**，则会发生下标越界的 bug。

输出字符串是通过**【cout<<字符串的起始地址】**来实现的，**输出到‘\0’为止**。
如果字符数组中**没有‘\0’**，那么它就没有储存一个完整的字符串，输出该数组名**会出错**。

```
char a[10];
cin >> a; // a 存储了以‘\0’结尾的字符串，但输入 10 个以上的字符时会越界
cin.getline(a, 10); // a 存储了以‘\0’结尾的字符串，且不会越界
for(int i=0; i<10; ++i) cin.get(a[i]); // 可能会读到‘\n’，但不会变成‘\0’
```

用**循环**处理字符串时，要注意根据**‘\0’来判断字符串是否结束**，并**及时结束循环**，否则可能产生错误。
【是空字符‘\0’，不是空格字符，也不是‘0’（‘\0’的 ASCII 码是 0)】

cin.getline 的**第一个参数**类型是一维字符数组，所以可以用**一维字符数组的数组名**（也就是一级字符指针），但**不能用二维字符数组的数组名**。
比如定义了 char a[4][5]; 后不能用 a 作为 cin.getline 的第一个参数，但可以用 a[0]、a[1]等作为 cin.getline 的第一个参数。

```
char a[4][5];
cin.getline(a[0], 5); // 正确
cin.getline(a, 5); // 语法错误
```

cin.getline 的**第二个参数**规定了读取字符的数量，**该数量是包含 '\0' 的**。例如第二个参数为 10，而输入了 10 个字符，那么只有前 9 个字符会被存入数组，并在末尾添加 '\0'。

```
char a[10];
cin.getline(a, 10); // 输入 0123456789
cout << a; // 输出 012345678
```

用 cin.getline 读取的一行字符串是不包括它的第三个参数的（即结束字符），该字符会被'\0'取代，因此后续在**判断字符串结尾时应以'\0'为标志**。

```
char a[100];
cin.getline(a, 20, '.'); // 这可以
cin.getline(a, 20); // 这也可以
cin.getline(a, '.'); // 这不可以!
```

7、字符串处理函数

strcpy 和 strcat 都有两个参数，第一个参数是 dst（目的地），第二个参数是 src（来源）。
src 字符串包含从 src 地址开始直到第一个'\0'为止的所有字符，与字符数组的大小无关。
使用 strcpy 时，要在 dst 数组中留够足以存下 src 字符串的空间。
使用 strcat 时，要在 dst 数组中留够足以存下拼接后的整个字符串的空间。
如果 dst 数组的空间不足，strcpy 和 strcat 会修改位于数组之后的内存空间，造成 bug。

要将字符串按长度排序，可用 strlen 获取字符串长度，用 strcpy 来交换两个字符串（需要另外定义一个数组作为中转平台）。
要比较两个字符串是否相同，可使用 strcmp 函数，其返回值为 0 时表示相同。

8、关于一些常见的但是很容易混淆的基础知识：

关于"字符"、"字符数组"和"字符串"的概念对比：

	概念	注意事项
字符	单个 char 类型的数据，常量有单引号	可以转换成 ASCII 表中的编码
字符数组	数组，每个元素都是 char 类型的变量	数组名表示第 0 个元素的地址
字符串	一串连续的字符，常量有双引号	必须以空字符结尾

关于"空字符"、"空格字符"、"空白字符"、"字符零"的概念对比：

	代码	ASCII 码	含义
空字符	'\0'	0	可表示字符串结尾
空格字符	' '	32	空格这个字符，可简称为空格符
空白字符	N/A 注	N/A 注	空格符、回车符和制表符的统称， 不包括空字符
字符零	'0'	48	数字零这个字符

注：N/A (Not applicable)，表示内容并不适用，在填表时多见。

关于“cin>>”、“cin.getline”和“cin.get”的使用对比：

	每次读取的内容	对字符串的处理	适用场景
cin>>	空白字符（空格符、回车符、制表符）前的内容（不包括空白字符）	把读取的字符都存储在字符数组里，并在读取的最后一个字符之后添加一个空字符，表示字符串结尾	读取不含空白字符的字符串（常表现为一个单词）；还可以读取其它类型的数据，如整数或实数、单个字符
cin.get	一个任何字符（包括空白与非空白字符）	可通过每次读取一个字符、循环（到输入回车符停止）读取的方式来存储字符串中的每一个字符，但用户输入的回车依然是回车符，并不会被处理成空字符，因此严格意义上讲，这并不是一个字符串。	适用于需要读取并存储空白字符的场景；适用于根据每个字符来依次处理的场景
cin.getline	结束符（默认是回车符）前的内容（通常是一行），包括结束符在内	把读取的字符都存储在字符数组里，末尾的结束符（默认是回车符）会被替换为空字符，表示字符串结尾	适用于一次读取一句话（包含空格符的多个单词）的场景