当我们使用花括号时，{...}，我们是说，如果可能的话，我们希望列表初始化对象。

当我们使用括号时，我们说我们提供的值将用于构造对象。.

const\_iterator的行为类似于const指针（第2.4.2节）。像const指针一样，const\_iterator可以读取但不能写入它所表示的元素; iterator类型的对象既可以读取也可以写入

开始和结束返回的类型取决于它们操作符所针对的对象是否是const。如果对象是const，则开始和结束返回const\_iterator;如果对象不是const，则返回iterator：

**箭头运算符（->）** 是一个简化表达式的重要工具，特别是在使用指针访问对象的成员时。它的作用是将\*\*解引用（dereference）**和**成员访问（member access）\*\*合并为一个操作

由于数组的大小是固定的，因此它们有时可以为专门的应用程序提供更好的运行时性能。然而，这种运行时优势是以失去灵活性为代价的

如果你不知道你需要多少个元素，可以使用vector。

最常见的安全问题来源是缓冲区溢出错误。当程序未能检查下标并错误地使用数组或类似数据结构范围之外的内存时，就会发生此类错误。

数组有一个特殊的属性-在大多数地方，当我们使用数组时，编译器会自动替换第一个元素的指针

在大多数表达式中，当我们使用数组类型的对象时，我们实际上是在使用指向该数组中第一个元素的指针。

数组上的操作实际上常常是指针上的操作，这一事实有多种含义。其中一个含义是，当我们使用数组作为使用auto定义的变量的初始化器时（第2.5.2节），推导出的类型是指针，而不是数组：



对于这个元素，我们唯一能做的就是获取它的地址，我们这样做是为了初始化e。与末端迭代器类似（见3.4.1节），末端指针并不指向元素。

一个指针“一过去”的内置数组的结束行为相同的方式作为迭代器返回的结束操作的一个向量。特别是，我们不能解引用或增加一个结束指针。

与vector和string的下标不同，内置下标运算符的索引不是无符号类型。.

我们可以使用strcat和strcpy。然而，要使用这些函数，我们必须传递一个数组来保存结果字符串。

对于大多数应用程序来说，除了更安全之外，使用库字符串而不是C风格的字符串也更有效。

现代C++程序应该使用向量和迭代器，而不是内置的数组和指针，并使用字符串，而不是C风格的基于数组的字符串。

严格地说，C++中没有多维数组。通常所说的多维数组实际上是数组的数组。当您使用看起来像是多维数组的东西时，记住这一点可能会很有帮助。

当你定义一个指向多维数组的指针时，记住多维数组实际上是一个数组的数组。

我们通常可以通过使用auto或decltype来避免将指针的类型写入数组

文本

描述已自动生成.其中最重要的库类型是vector和string。字符串是一个可变长度的字符序列，向量是一个单一类型对象的容器。

迭代器允许间接访问存储在容器中的对象。迭代器用于访问和导航字符串和向量中的元素。

数组和指向数组元素的指针提供了向量和字符串库的低级类似物。一般来说，库类应该优先于内置在语言中的低级数组和指针替代品。