1，命名空间的using声明

有了using声明就无须专门的前缀也能使用所需的名字了。using声明具有如下的形式：using namespace name;

头文件不应包含using声明。

2，标准库类型string

标准库类型string表示可变长的字符序列。

参考文档：初始化string对象的方式。

如果使用=号初始化一个变量，实际上执行的是拷贝初始化，编译器把等号右侧的初始值拷贝到新创建的对象中去。

与之相反，如果不使用等号，则执行的是直接初始化。

参考文档：string的操作

string s;cin >> s;读取时string对象会自动忽略开头的空白，并遇到下一处空白停止。如果输入的是" Hello World! "，读取的将是"Hello"。

使用getline读取一整行，getline函数的参数是一个输入流和一个string对象。

编程：每次读入一整行，直至到达文件末尾。

string.size()返回的类型为string::size\_type。这是一个无符号数，因此切忌在表达式中混用了带符号数将可能产生意想不到的结果。

如果一条表达式中已经有了size()函数就不要再使用int了，这样可以避免混用int和unsigned可能带来的问题。

当把string对象和字符字面值及字符串字面值混在一条语句中使用时，必须确保每个加法运算符的两侧的运算对象至少有一个string。

string = "Hello " + " World"是非法的。

切记，字符串字面值与string是不同的类型。

参考文档：字符处理的函数

可以使用for循环来遍历string。for(char c : str)，每次迭代都会拷贝。

如果想要改变string对象中字符的值，必须把循环变量定义成引用类型。

编程：把字符串改写为大写字母的形式。

可以使用下标访问string对象。也可以进行赋值。

3，标准库类型vector

vector能容纳绝大多数类型的对象作为其元素，但是因为引用不是对象，所以不存在包含引用的Vector。

参考文档：初始化vector对象的方法。

范围for语句体内不应改变其所遍历序列的大小。

编程：使用vector按分段统计输入的成绩。

4，迭代器

有迭代器的类型同时拥有返回迭代器的成员，这些类型都拥有名为begin和end的成员。end表示尾元素的下一个位置。如果容器为空，begin和end返回相同。

编程：使用迭代器把字符串改成大写形式。

begin和end返回的具体类型由对象是否是常量决定，如果对象是常量，begin和end返回const\_iterator；如果对象不是常量，返回iterator。

如果对象只需读操作而无须写操作的话最好使用常量类型，为了便于得到const\_iterator类型，可以使用cbegin和cend函数。

迭代器解引用：it->mem和(\*it).mem表达的意思相同。

但凡是使用了迭代器的循环体，都不要向迭代器所属的容器添加元素。

编程：使用迭代器完成二分搜索

5，数组

数组的大小是固定的，如果不清楚元素的确切个数，请使用vector。

维度说明了数组中元素的个数，编译的时候维度应该是已知的。维度必须是一个常量表达式。

和vector一样，数组的元素应为对象，因此不存在引用的数组。

可以用字符串字面值对数组进行初始化，但是要注意字符串字面值的结尾处有一个空字符。

不能将数组的内容拷贝给其他数组作为其初始值，也不能用数组为其他数组赋值。

含有10个整型指针的数组：int \*ptrs[10]。

指向一个含有10个整数的数组的指针：int (\*Parry)[10]。

指向一个含有10个整数的数组的引用：和int (&arrRef)[10]。

要想理解数组声明的含义，最好的办法是从数组的名字开始按照由内向外的顺序阅读。

可以使用for循环来遍历数组。

使用数组时，编译器一般会把它转换成指针。

很多用到数组名字的地方，编译器都会自动地将其替换为一个指向数组首元素的指针。string \*p = nums 等价于 string \*p = &nums[0];

当使用数组作为一个auto变量的初始值时，推断得到的类型是指针而不是数组。

当使用decltype关键字时，上述转换不会发生。

数组内置的下标运算符所用的索引值不是无符号类型，这一点与vector和string不一样。

获取数组的首尾：begin(arr)和end(arr)。

尽管C++支持c风格字符串，但在C++程序中最好还是不要使用它们。这是因为C风格字符串不仅使用起来不太方便，而且极易引发程序漏洞。

C风格字符串char ca[] = "A string example"。

string s;char \*str = s 是错误的， const char \*str = s.c\_str()是正确的。

严格来说，C++语言没有多维数组，多维数组其实是数组的数组。