第13条：更喜欢vector和string而不是动态分配的数组。

任何时候你发现自己准备动态分配一个数组（即，绘制“新T[...]”），你应该考虑使用向量或者字符串。

项目14：使用准备金避免不必要的重新分配。

1.分配一个新的内存块，它是容器当前容量的几倍。在大多数实现中，向量和字符串容量每次增长1.5到2倍。2.将所有元素从容器的旧内存复制到新内存中。3.销毁旧内存中的对象。4.释放旧内存。

第15条：注意字符串实现的变化。

文本

描述已自动生成.第16条：知道如何将向量和字符串数据传递给遗留API。

第17条：使用“互换技巧”来削减过剩产能。

第18条：避免使用矢量<bool>。

关联容器

条款20：指定指针的关联容器的比较类型。

关联容器默认使用std::less作为比较函数，这适用于大多数数据类型，包括基本数据类型和标准库中的数据类型。然而，当容器存储的是指针时，默认的比较行为是比较指针的地址，而不是指针所指向的对象的内容。

以下是如何理解这句话的几个要点：

1. **指针比较的默认行为**：默认情况下，如果两个指针被放入一个关联容器，它们将根据它们的地址进行比较，而不是它们所指向的值。
2. **指定比较类型**：如果你想根据指针所指向的值来排序或比较指针，你需要提供一个自定义的比较函数，或者使用标准库中提供的比较函数，如std::greater、std::less\_equal等。
3. **自定义比较函数**：你可以编写一个比较函数，该函数接受两个指针参数，并返回一个布尔值，指示第一个参数是否“小于”第二个参数。例如，如果你想根据指针指向的整数值来比较指针，你可以这样定义比较函数：

第21条：总是让比较函数在相等值时返回false。

以下是对这句话的解释：

1. **比较函数的定义**：在C++中，关联容器的比较函数通常是一个函数对象，它定义了两个元素之间的“小于”关系。这个函数对象应该重载了operator()，接受两个参数，并根据参数返回一个布尔值。
2. **相等的含义**：在比较函数中，如果两个元素被认为是“相等”的，则比较函数应该返回false。这是因为关联容器是基于比较函数来维护元素顺序的，它们通常不允许有相等的元素（multiset和multimap是例外，它们可以包含多个相等的元素）。
3. **为什么相等时返回false**：对于set和map等容器，如果比较函数在元素相等时返回true，则容器将无法正确处理这种情况，可能会导致未定义的行为，比如无法插入元素或者无法正确地删除元素。
4. **示例**：以下是一个简单的比较函数示例，它确保在两个元素相等时返回false

Item 22：避免在set和multiset中就地修改键。

请原谅我这么说，但关键是要记住，对于集合（set）和多重集合（multiset），如果你对容器元素进行任何就地修改，你有责任确保容器保持有序。

文本

描述已自动生成

第23条：考虑用排序向量替换关联容器。

文本

描述已自动生成

第24条：当效率很重要时，在map：：operator[]和map：：insert之间小心选择。

第25条：熟悉非标准散列容器。