**C++中string类型的erase()函数详解**

# 首先理解：

# 什么是迭代器？迭代器有什么作用？

## ****什么是迭代器？（迭代器VS指针）****

**指针：一般用来访问连续的内存空间地址元素，有序存放的数据序列（**当然指针也可以访问不连续的：如链表**）；**

**迭代器：可以用来管理不连续存放的数据序列元素地址，比指针有更多的操作更强更多的管理能力。**

先说说迭代器它是干嘛的吧！迭代器的作用是用来访问容器（用来保存元素的数据结构）中的元素，所以使用迭代器，我们就可以访问容器中里面的元素。没错！这和访问数组这个序列的指针一样，因为数组范围内的指针就是迭代器的一种。

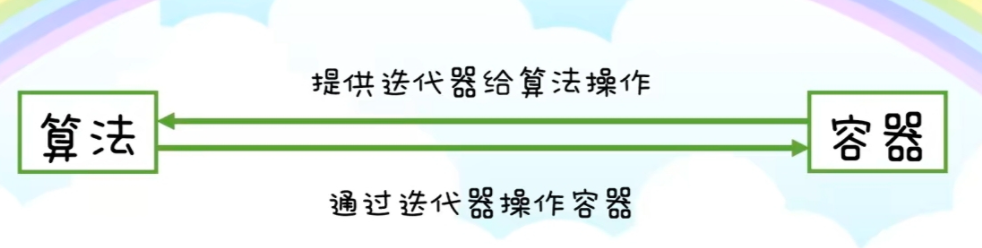
指针和迭代器有什么区别呢？

1.指针是C语言里面就有的东西，而迭代器是C++里面才有的

2.指针用起来灵活，效率高。迭代器功能更丰富一些

3.指针通常用来访问的是**序列**的元素，但不是所有的容器都会在连续的内存空间上保存数据。所以，对于这些容器我们不能单纯地使用指针作为迭代器，而是针对每一种容器，都会有专门配对的迭代器。

4.指针用起来灵活，效率高。迭代器功能更丰富一些，c++的stl里面很多算法都是基于迭代器的，一部分算法的参数可以传递指针作为迭代器使用。



迭代器怎么使用？

对于所有的迭代器，它们的使用方法和指针一样，比如自增（++），解引用（\*）。除了数组以外，在大部分的容器中都会提供成员函数begin(）（在类中创建，类是C++中对于C语言中的结构体的延伸），用来获取容器开始位置的迭代器，会提供成员函数end（），用来获取容器结束位置的迭代器。

如何理解begin（）和end（）这两个返回值为迭代器的函数？

begin（）相当于数组中首元素地址，end（）相当于**末元素后一位置的地址**（首元素地址+元素数量）

迭代器五大种类及功能解析

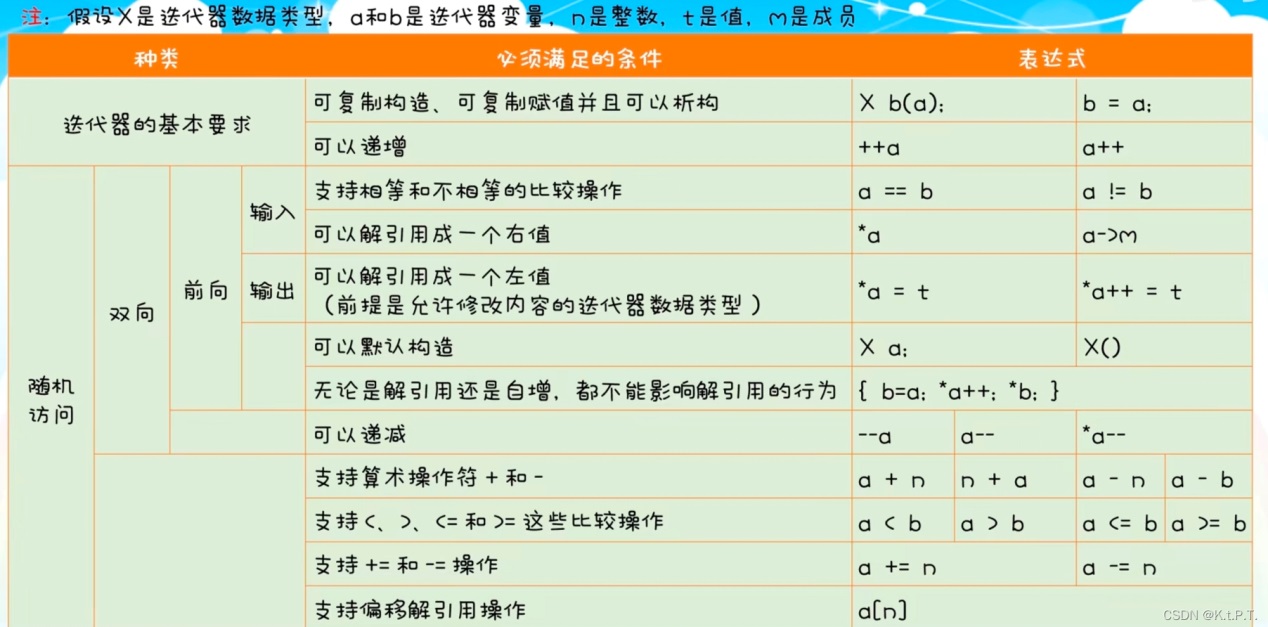
InputIterator（输入迭代器）：只要求支持拷贝、自增和解引访问。

OutputIterator（输出迭代器）：只要求支持拷贝、自增和解引赋值。

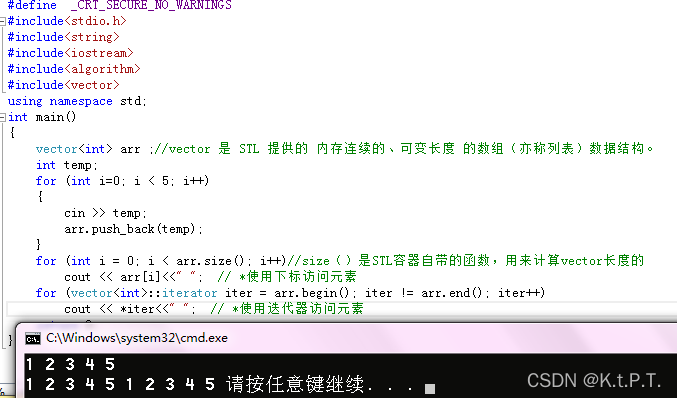
ForwardIterator（向前迭代器）：同时满足 InputIterator 和 OutputIterator 的要求。

BidirectionalIterator（双向迭代器）：在 ForwardIterator 的基础上支持自减（即反向访问）。

RandomAccessIterator（随机访问迭代器）：在 BidirectionalIterator 的基础上支持加减运算和比较运算（即随机访问）。



## ****使用迭代器\*样例展示：****



vector<int>是一种数据结构—我们熟悉的数组（亦称列表）

**用vector<int>::iterator定义迭代器iter**

（迭代器的类型就为container::iterator）（container是某个STL容器）

arr.begin()是返回值指向首元素的为迭代器的函数：

常用四大返回值为迭代器的函数

1.begin()/cbegin()

返回指向首元素的迭代器，其中 \*begin = front。

2.end()/cend()

返回指向数组尾端占位符的迭代器，注意尾端占位符是没有元素的，**不能进行解引用**。

3.rbegin()/crbegin()

返回指向逆向数组的首元素的逆向迭代器，可以理解为**正向容器的末元素**。

4.rend()/crend()

**返回指向逆向数组末元素后一位置的迭代器，对应容器首的前一个位置，没有元素**。

使用返回值为迭代器的函数\*样例展示：



**sort()和reverse()都是STL提供的算法函数**

**可以大大提高写代码的效率哦!**

**C++中string类型的erase()函数详解**

标准库类型string表示可变长的字符序列。可以通过string类的erase()函数来对该字符序列进行删除操作。erase()函数共有3种格式，分别用来删除指定位置的字符、删除指定长度的字符串和删除指定范围的字符串。

1 删除指定位置的字符

erase()函数用于删除指定位置字符的格式为：

iterator erase (iterator p);

其中，参数[iterator](https://so.csdn.net/so/search?q=iterator&spm=1001.2101.3001.7020)是迭代器（可以理解为指针、容器中的迭代器），即删除迭代器p指向的字符。返回值为删除后的字符串。

# 2 删除指定长度的[字符串](https://so.csdn.net/so/search?q=%E5%AD%97%E7%AC%A6%E4%B8%B2&spm=1001.2101.3001.7020)

erase()函数用于删除指定长度字符串的格式为

string& erase(size\_t pos=0, size\_t len = npos);

其中，参数pos表示要删除字符串的起始位置，其默认值是0；len表示要删除字符串的长度，其默认值是string::npos。返回值是删除后的字符串。

相关链接：

string::npos，是string类型的静态常量成员，表示size\_t类型的最大值。在用于表示string长度时，实际上就是指定了string类型对象的长度。所以，对于erase()函数的第二种格式，其默认参数的含义就是将string中的所有内容都删除。

3 删除指定范围的字符串

erase()函数用于删除指定范围字符串的格式为：

iterator erase (iterator first, iterator last);

其中，first和last分别表示范围的起点和终点。需要注意的是，该范围包含first，但是不包含last，即该范围是[first, last)。erase()函数的返回值是删除后的字符串。

# 4 相关代码

std::string str("This is an example sentence.");

str.erase(10, 8);

str.erase(str.begin() + 9);

str.erase(str.begin() + 5, str.end() - 9);

str是string类型的变量，首先调用erase()函数在str中删除指定长度的字符串，即删除从第10位开始，长度为8的字符串。需要注意的是，str的起始位置是从0开始的。删除的str变为“This is an sentence.”接下来调用erase()函数删除指定位置的字符，str.begin()是获取str起始位置处的迭代器，删除str.begin()+9迭代器对应的字符，即字符“n”。此时，str的值变为“This is a sentence.”。最后，调用erase()函数删除指定范围的字符串。最终，str的值变为“This sentence.”。