C++ 迭代器

列表（std::list）语法

std::list是C++ STL中的双向链表容器，需要包含头文件。

- 声明：`std::list<Type> listName;` 如`std::list<int> numbers;`表示存储整数的列表。
- 常用操作：

函数	描述	示例
`push_front(value)`	在开头插入元素	`numbers.push_front(0)`
`push_back(value)`	在末尾添加元素	`numbers.push_back(4)`
`front()`	返回第一个元素	`std::cout << numbers.front()`
`back()`	返回最后一个元素	`std::cout << numbers.back()`
`pop_front()`	删除第一个元素	`numbers.pop_front()`
`pop_back()`	删除最后一个元素	`numbers.pop_back()`
`insert(iterator, value)`	在指定位置插入元素	`numbers.insert(itr, 0)`
`remove(element)`	删除所有指定值的元素	`numbers.remove(1)`

迭代器语法

- **声明迭代器**
  - 基本形式：`std::list<Type>::iterator it;` 如`std::list<int>::iterator it;`
  - 对于只读访问，可用const_iterator：`std::list<int>::const_iterator it;`
  - 反向迭代器：`std::list<int>::reverse_iterator rit;`用于从后向前遍历。
  - 自C++11起，可用auto简化：`auto it = myList.begin();`
- **常用操作：**
  - 解引用：`*it`获取当前元素值，可写`*it = new_val;`修改。
  - 增减：`++it`向前移动，`--it`向后移动（列表支持双向）。
  - 比较：`it != myList.end()`，`it == anotherIt`等。
- 使用示例：

```
std::list<int> myList = {1, 2, 3};
for (std::list<int>::iterator it = myList.begin(); it != myList.end(); ++it) {
    std::cout << *it << " ";
} // Output: 1 2 3
```

对于自定义类型，如`struct Student { std::string name; }`，可写`std::list<Student>::iterator it;`，访问如`it->name`