Lesson11---deque(了解)

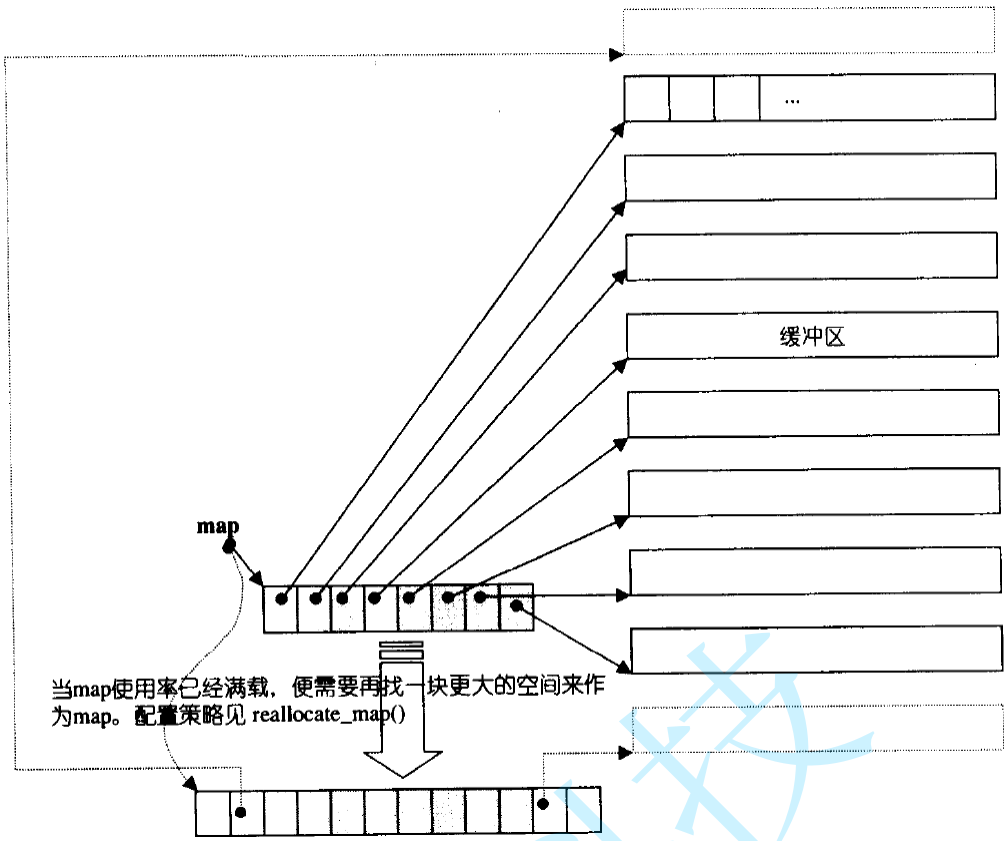
【本节目标】

- **1. deque的介绍**
- **2. deque的使用**
- **3. deque的应用**

1. deque的介绍及使用

1.1 deque的介绍

> deque在线文档说明
>
> 翻译：
>
> 1. deque(发音类似“deck”)，是双端队列不规则的首字母缩写，双端队列是动态大小的序列式容器，其可以像两端进行伸缩。
> 2. 特定的库可以以不同的方式实现deque，但通常都是一种动态数组。不论在何种情况下，它都允许通过随机访问迭代器直接访问单个元素，可以根据需要动态的伸缩。
> 3. 因此，deque提供了一些与vector相似的功能，但deque在头部和尾部进行数据插入和删除操作更加高效。与vector不同的是，deque不能保证所有的元素存储在连续的空间中，在deque中通过指针加偏移量方式访问元素可能会导致非法的操作。
> 4. vector与list提供了相似的接口，因此其具有类似的用途，但是内部的实现原理不同：vector使用使用了动态数组，该数组通常需要动态增长；deque中的元素可能分散在不同的存储块中，在deque中保存了一些必要的信息，通常用来在常数范围内直接访问deque中的任何一个元素，所以deque的内部实现比vector复杂，但是这些额外信息使得dque在某些情况下增长更加的高效，特别是在序列比较大，重新分配成本比较高的情况下。
> 5. 除了在频繁在头部或尾部进行插入和删除操作外，deque比list和forward_list的性能更差。



2. deque的使用

2.1 deque的构造

函数声明	接口说明
deque()	构造空的双端队列
deque(n, val = value_type())	用n个值为val的元素构造双端队列
deque(InputIterator first, InputIterator last)	用[first, last)的区间构造双端队列
deque(const deque& x)	双端队列的拷贝构造函数

2.2 deque的迭代器

双端队列底层是一段假象的连续空间，实际是分段连续的，为了维护其“整体连续”的假象，落在了deque的迭代器身上。下图为deque的原理图：

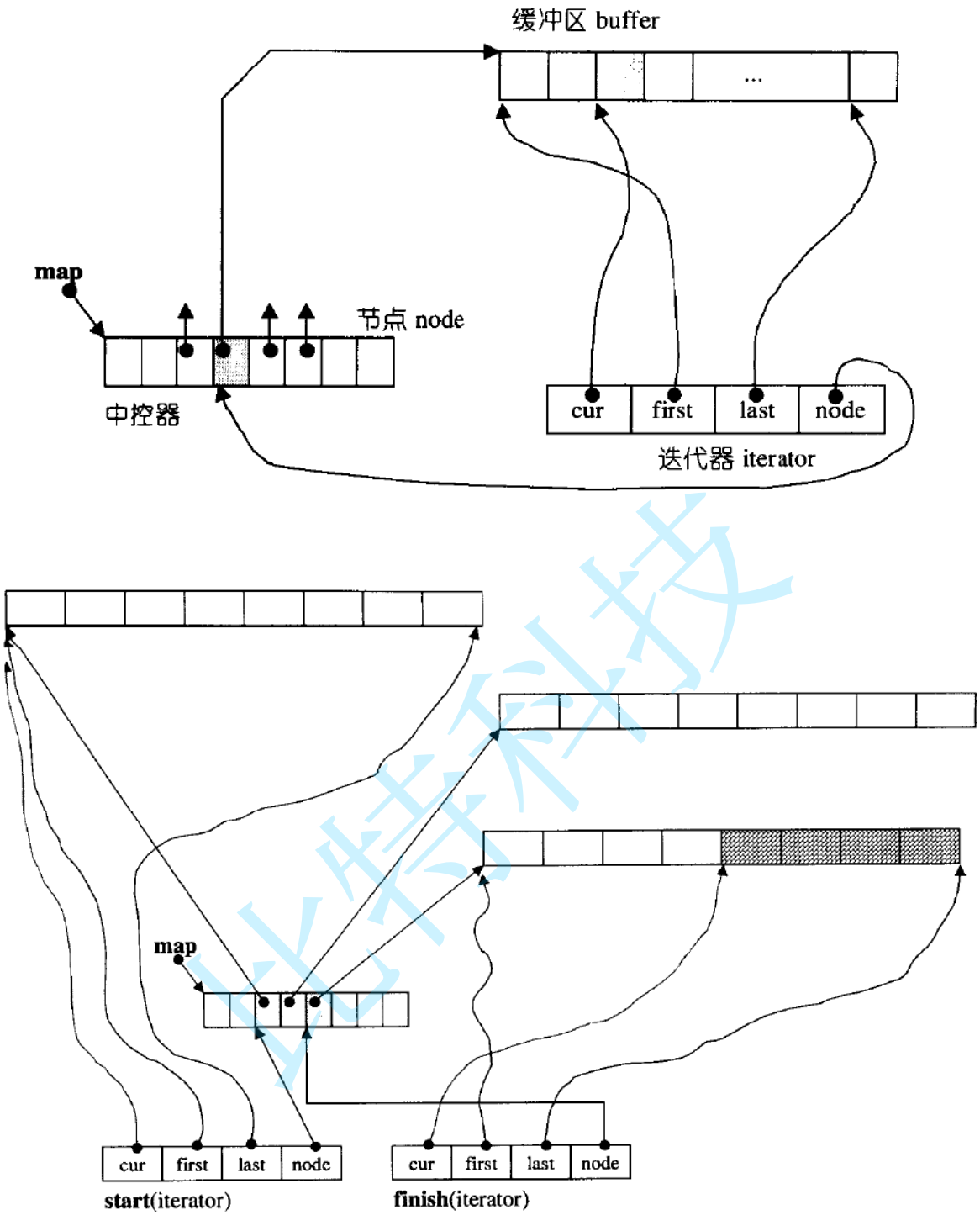


图 **4-12** deque::begin() 传回迭代器 start，deque::end() 传回迭代器 finish。这两个迭代器都是 deque 的 data members。图中所示的这个 deque 拥有 20 个 int 元素，以 3 个缓冲区储存之。每个缓冲区 32 bytes，可储存 8 个 int 元素。map 大小为 8（起始值），目前用了 3 个节点。

函数声明	接口说明
[begin(), end()())	begin:容器起始位置 end最后一个元素下一个位置
[rbegin(), rend())	反向迭代器rbegin在end位置，rend在begin
[cbegin(), cend())	const迭代器，与begin和end位置相同，但不能修改其空间内容
[crbegin(), crend())	const反向迭代器，与crbegin在cend位置，crend在cbegin位置

2.3 deque的容量操作

函数声明	接口说明
size()	返回deque中有效元素个数
empty()	检测deque是否为空，是返回true，否则返回false
resize(sz, value)	将deque中的元素改变到sz，多出的空间用value填充

2.4 deque的元素访问操作

函数声明	接口说明
operator[]	返回deque中n位置上元素的引用
front()	返回deque中首元素的引用
back()	返回deque中最后一个元素的引用

2.4 deque中修改操作

函数声明	接口说明
push_back() 和 pop_back()	deque的尾插和尾删
push_front() 和 pop_front()	deque任意位置插入和删除
insert(pos, value) 和 erase(pos)	删除deque头部元素
swap()	交换两个deque中的内容
clear()	将deque中的元素清空

2.5 deque中接口应用实例

```cpp
#include<deque>

void PrintDeque(const std::deque<int>& d)
{
    for (const auto& e : d)
        cout << e << " ";
    cout << endl;
}

// 测试deque的构造函数
void TestDeque1()
{
    // 构造空的双端队列
    std::deque<int> d1;

    // 用10个值为5的元素构造双端队列
    std::deque<int> d2(10, 5);
    PrintDeque(d2);

    // 用数组的区间构造双端队列
    int array[] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 0 };
    std::deque<int> d3(array, array+sizeof(array)/sizeof(array[0]));
    PrintDeque(d3);

    // 用d3拷贝构造d4
    std::deque<int> d4(d3);
    PrintDeque(d4);
}

// 测试deque中的迭代器
void TestDeque2()
{
    int array[] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 0 };
    std::deque<int> d(array, array+sizeof(array)/sizeof(array[0]));

    // 利用正向迭代器打印deque中的元素
    for (auto it = d.cbegin(); it != d.cend(); ++it)
        cout << *it << " ";
    cout << endl;

    auto cit = d.cbegin();
    //*it = 100;   该条语句编译失败，it为const迭代器，其指向空间元素值不能修改


    // 利用反向迭代器逆向打印deque中的元素
    for (auto it = d.crbegin(); it != d.crend(); ++it)
        cout << *it << " ";
    cout << endl;
}

void TestDeque3()
{
     int array[] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 0 };
```

```
    std::deque<int> d1(array, array+sizeof(array)/sizeof(array[0]));

    // 在deque的尾部插入5，头部插入1，并打印
    d1.push_back(6);
    d1.push_front(2);
    PrintDeque(d1);
    cout << d1.size() << endl;
    cout << d1.front() << endl;
    cout << d1.back() << endl;

    // 在deque的begin位置插入元素0
    d1.insert(d1.begin(), 0);
    PrintDeque(d1);

    // 删除deque首部与尾部元素
    d1.pop_front();
    d1.pop_back();
    d1.erase(d1.begin());
    PrintDeque(d1);

    // 将deque中的元素清空
    d1.clear();
    cout << d1.size() << endl;
}


// 问题：如果要对deque中的元素进行排序，以下的效率高吗?
#include <algorithm>
#include <deque>
void TestDequeSort()
{
    int array[] = { 5, 2, 1, 9, 6, 3, 8, 7, 4, 0 };
    deque<int> d(array, array + sizeof(array) / sizeof(array[0]));
    PrintDeque(d);

    // 利用标准库中的算法对deque中的元素进行升序排序
    sort(d.begin(), d.end());
    PrintDeque(d);
}

/*
   上述对deque中排序操作的效率是非常低的，当对deque排序时，需要多次对deque中的元素进行整体遍历，而deque中的元素整体遍历时效率比较低，这是因为deque底层的空间不连续，如果要进行整体遍历，在某段空间的默认或首部时，必须要计算下一段或者前一段空间的位置，导致deque的效率比较底下。
*/
```

3. deque的应用场景

> deque在序列式容器中比较鸡肋，因为**如果只是简单的存储元素，使用vector即可，如果对元素任意位置进行插入或者删除操作比较多，使用list即可**，所以一般很少去使用deque。**deque最大的应用，就是用其作为标准库中stack和queue的底层结构**(见下节)。

4. 总结

4.1 本章小结

知识块	知识点	分类	掌握程度
deque介绍	deque原理了解	概念型	了解
deque接口	deque接口原理了解	概念型	掌握
	deque接口的使用	应用型	掌握

4.2 本章作业

1. 熟悉deque的原理
2. 掌握deque的常规接口使用