C++ 标准库 <deque>

在 C++中，<deque> 是标准模板库（STL）的一部分，它提供了双端队列（double-ended queue）的实现。

双端队列是一种允许在两端进行插入和删除操作的线性数据结构。

<deque> 的全称是 "double-ended queue"，它在C++中以模板类的形式存在，允许存储任意类型的数据。

<deque> 是一个动态数组，它提供了快速的随机访问能力，同时允许在两端进行高效的插入和删除操作。这使得 <deque> 成为处理需要频繁插入和删除元素的场景的理想选择。

语法

在 C++ 中，使用 <deque> 需要包含头文件 #include <deque>。以下是 <deque> 的基本语法：

```
#include <iostream>
#include <deque>

int main() {
    std::deque<int> myDeque; // 创建一个整数类型的双端队列
    // 接下来可以进行插入、删除等操作
    return 0;
}
```

常用操作

下面是 std::deque 容器的一些常用成员函数：

函数名称	功能描述
deque()	默认构造函数，创建一个空的 deque 容器。
deque(size_type n)	创建一个包含 n 个默认值元素的 deque 容器。
deque(size_type n, const T& value)	创建一个包含 n 个值为 value 的 deque 容器。
deque(initializer_list<T> il)	使用初始化列表 il 构造 deque 容器。
operator=	赋值操作符，赋值给 deque 容器。
assign()	用新值替换 deque 容器中的所有元素。
at(size_type pos)	返回 pos 位置的元素，并进行范围检查。
operator[](size_type pos)	返回 pos 位置的元素，不进行范围检查。
front()	返回第一个元素的引用。
back()	返回最后一个元素的引用。

函数名称	功能描述
begin()	返回指向第一个元素的迭代器。
end()	返回指向末尾元素后一位置的迭代器。
rbegin()	返回指向最后一个元素的逆向迭代器。
rend()	返回指向第一个元素之前位置的逆向迭代器。
empty()	检查容器是否为空。
size()	返回容器中的元素个数。
max_size()	返回容器可容纳的最大元素个数。
clear()	清除容器中的所有元素。
insert(iterator pos, const T& value)	在 pos 位置插入 value 元素。
erase(iterator pos)	移除 pos 位置的元素。
push_back(const T& value)	在容器末尾添加 value 元素。
pop_back()	移除容器末尾的元素。
push_front(const T& value)	在容器前端添加 value 元素。
pop_front()	移除容器前端的元素。
resize(size_type count)	调整容器大小为 count，多出部分用默认值填充。
swap(deque& other)	交换两个 deque 容器的内容。
get_allocator()	返回一个用于构造双端队列的分配器对象的副本。

实例

下面是一个使用 <deque> 的简单示例，包括元素的插入、访问和删除操作。

实例

```
#include <iostream>
#include <deque>

int main() {
    std::deque<int> myDeque;

    // 插入元素
    myDeque.push_back(10);
```

```
    myDeque.push_back(20);
    myDeque.push_front(5);

    // 访问元素
    std::cout << "Deque contains: ";
    for (int i = 0; i < myDeque.size(); ++i) {
        std::cout << myDeque[i] << " ";
    }
    std::cout << std::endl;

    // 删除元素
    myDeque.pop_back();
    myDeque.pop_front();

    // 再次访问元素
    std::cout << "Deque after popping: ";
    for (int i = 0; i < myDeque.size(); ++i) {
        std::cout << myDeque[i] << " ";
    }
    std::cout << std::endl;

    return 0;
}
```

输出结果:

```
Deque contains: 5 10 20
Deque after popping: 10
```

在不知道 deque 长度的时候，可以使用 deque.front() 与 deque.back() 来访问头尾元素：

实例

```
#include <iostream>
#include <deque>

int main() {
    std::deque<int> d;

    // 向双端队列中添加元素
    d.push_back(10);
    d.push_back(20);
    d.push_front(5);

    // 访问前端元素
    std::cout << "Front element: " << d.front() << std::endl;

    // 访问后端元素
    std::cout << "Back element: " << d.back() << std::endl;

    // 修改前端元素
    d.front() = 15;

    // 修改后端元素
    d.back() = 25;
```

```
    // 再次访问元素
    std::cout << "Modified front element: " << d.front() << std::endl;
    std::cout << "Modified back element: " << d.back() << std::endl;

    return 0;
}
```

输出结果为：

```
Front element: 5
Back element: 20
Modified front element: 15
Modified back element: 25
```

注意：在使用 front() 或 back() 之前，确保双端队列不为空，否则会引发未定义的行为。如果需要检查双端队列是否为空，可以使用 empty() 成员函数。