以下是对项目中三个队列（LockedQueue、MPSCQueue 和 ProducerConsumerQueue）的移植和使用步骤及代码案例：

### **1. LockedQueue**

#### **移植步骤**

1. ****复制头文件****：将 LockedQueue.h 文件复制到你的项目目录中。
2. ****包含头文件****：在需要使用 LockedQueue 的源文件中包含该头文件。

#### **使用步骤**

1. ****创建队列对象****：实例化一个 LockedQueue 对象，指定队列中元素的类型。
2. ****添加元素****：使用 add 方法向队列中添加元素。
3. ****取出元素****：使用 next 方法从队列中取出元素。

#### **代码案例**

cpp

#include "LockedQueue.h"#include <iostream>#include <thread>

int main() {

// 创建一个存储 int 类型元素的 LockedQueue 对象

LockedQueue<int> queue;

// 生产者线程

std::thread producer([&]() {

for (int i = 0; i < 5; ++i) {

queue.add(i);

std::cout << "Produced: " << i << std::endl;

}

});

// 消费者线程

std::thread consumer([&]() {

int value;

while (queue.next(value)) {

std::cout << "Consumed: " << value << std::endl;

}

});

// 等待生产者和消费者线程完成

producer.join();

consumer.join();

return 0;}

### **2. MPSCQueue**

#### **移植步骤**

1. ****复制头文件****：将 MPSCQueue.h 文件复制到你的项目目录中。
2. ****包含头文件****：在需要使用 MPSCQueue 的源文件中包含该头文件。

#### **使用步骤**

1. ****创建队列对象****：实例化一个 MPSCQueue 对象，指定队列中元素的类型。
2. ****添加元素****：使用 Enqueue 方法向队列中添加元素。
3. ****取出元素****：使用 Dequeue 方法从队列中取出元素。

#### **代码案例**

cpp

#include "MPSCQueue.h"#include <iostream>#include <thread>

struct Count {

Count(int \_v) : v(\_v) {}

int v;};

int main() {

// 创建一个存储 Count 类型元素的 MPSCQueue 对象

MPSCQueue<Count> queue;

// 生产者线程 1

std::thread producer1([&]() {

for (int i = 0; i < 5; ++i) {

queue.Enqueue(new Count(i));

std::cout << "Produced by producer1: " << i << std::endl;

}

});

// 生产者线程 2

std::thread producer2([&]() {

for (int i = 5; i < 10; ++i) {

queue.Enqueue(new Count(i));

std::cout << "Produced by producer2: " << i << std::endl;

}

});

// 消费者线程

std::thread consumer([&]() {

Count\* ele;

while (queue.Dequeue(ele)) {

std::cout << "Consumed: " << ele->v << std::endl;

delete ele;

}

});

// 等待生产者和消费者线程完成

producer1.join();

producer2.join();

consumer.join();

return 0;}

### **3. ProducerConsumerQueue**

#### **移植步骤**

1. ****复制头文件****：将 ProducerConsumerQueue.h 文件复制到你的项目目录中。
2. ****包含头文件****：在需要使用 ProducerConsumerQueue 的源文件中包含该头文件。

#### **使用步骤**

1. ****创建队列对象****：实例化一个 ProducerConsumerQueue 对象，指定队列中元素的类型。
2. ****添加元素****：使用 Push 方法向队列中添加元素。
3. ****取出元素****：使用 WaitAndPop 方法从队列中取出元素。

#### **代码案例**

cpp

#include "ProducerConsumerQueue.h"#include <iostream>#include <thread>

int main() {

// 创建一个存储 int 类型元素的 ProducerConsumerQueue 对象

ProducerConsumerQueue<int> queue;

// 生产者线程

std::thread producer([&]() {

for (int i = 0; i < 5; ++i) {

queue.Push(i);

std::cout << "Produced: " << i << std::endl;

}

});

// 消费者线程

std::thread consumer([&]() {

int value;

for (int i = 0; i < 5; ++i) {

queue.WaitAndPop(value);

std::cout << "Consumed: " << value << std::endl;

}

});

// 等待生产者和消费者线程完成

producer.join();

consumer.join();

return 0;}

通过以上步骤和代码案例，你可以将这三个队列移植到你的项目中，并在多线程环境中使用它们进行数据的生产和消费。