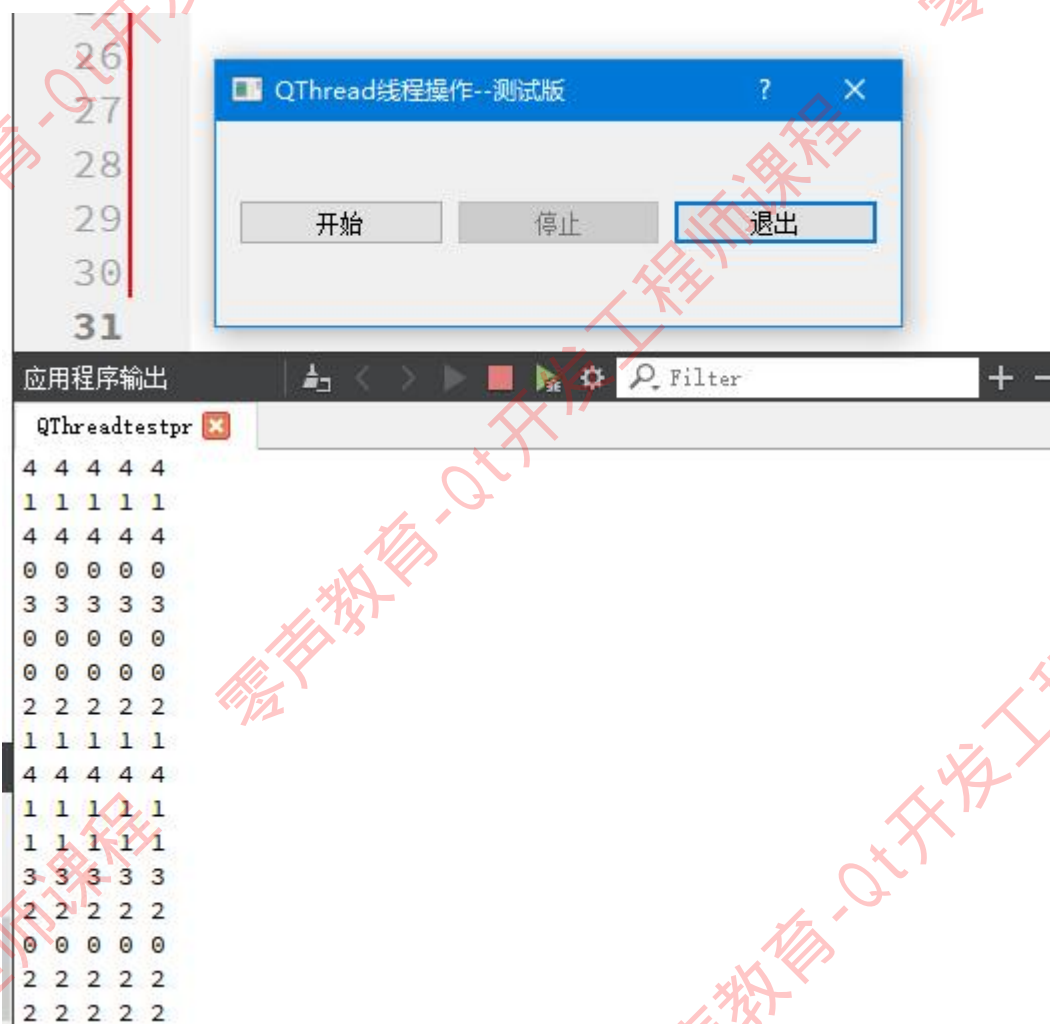


2036-线程编程（互斥量 信号量）--零声教育

一、线程

Qt 提供 QThread 类进行多任务处理，QThread 继承自 QObject 类，并且提供 QMutex 类来实现同步。

互斥量(又称互斥锁)，是一个可以处于两态之一的变量:解锁和加锁。测试程序如下：



二、信号量

生产者/消费者实例中对同步的需求有两处：

- 如果生产者过快地生产数据，将会覆盖消费者还没有读取的数据。
- 如果消费者过快地读取数据，将越过生产者并且读取到一些过期数据。

Qt 中 `QSemaphore` 类代表信号量，主要 API 函数如下：

QSemaphore Class

The `QSemaphore` class provides a general counting semaphore. [More...](#)

Header: `#include <QSemaphore>`

qmake: `QT += core`

- [List of all members, including inherited members](#)

Note: All functions in this class are `thread-safe`.

Public Functions

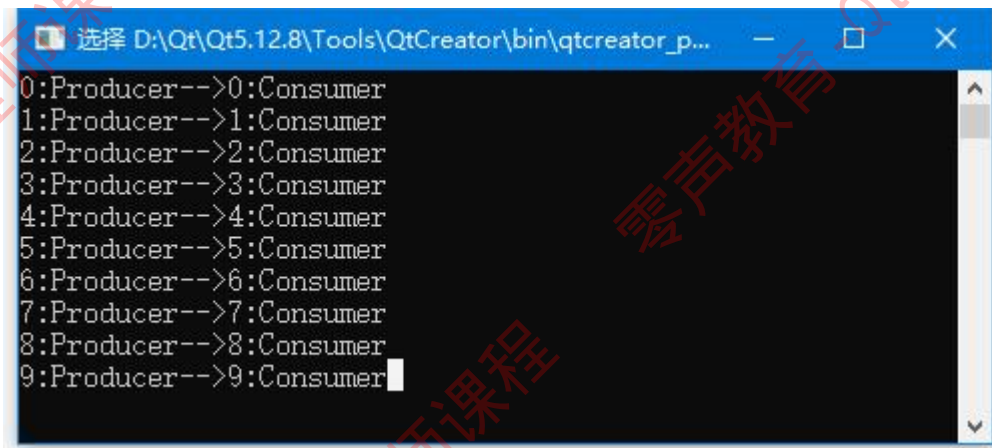
`QSemaphore(int n = 0)`

`~QSemaphore()`

void `acquire(int n = 1)` ✓

int `available() const`

void `release(int n = 1)` ✓



```
选择 D:\Qt\Qt5.12.8\Tools\QtCreator\bin\qtcreator_p...
0:Producer-->0:Consumer
1:Producer-->1:Consumer
2:Producer-->2:Consumer
3:Producer-->3:Consumer
4:Producer-->4:Consumer
5:Producer-->5:Consumer
6:Producer-->6:Consumer
7:Producer-->7:Consumer
8:Producer-->8:Consumer
9:Producer-->9:Consumer
```

三、互斥量（一般指互斥锁）

使用目的：保证共享数据操作的完整性，每个对象对应一个互斥锁标记，此标记用来保证任一时刻只能有一个线程访问对象。

互斥量两态：加锁和解锁。

QMutex 类和 QMutexLocker 类

QMutex 和 QMutexLocker 是基于互斥量的线程同步类。比如 QMutex 定义的实例化就是一个互斥量，主要使用 3 个核心 API 函数：

QMutex Class

The **QMutex** class provides access serialization between threads. [More...](#)

Header: `#include <QMutex>`

qmake: `QT += core`

Inherits: `QBasicMutex`

- [List of all members, including inherited members](#)

Note: All functions in this class are **thread-safe**.

Public Types

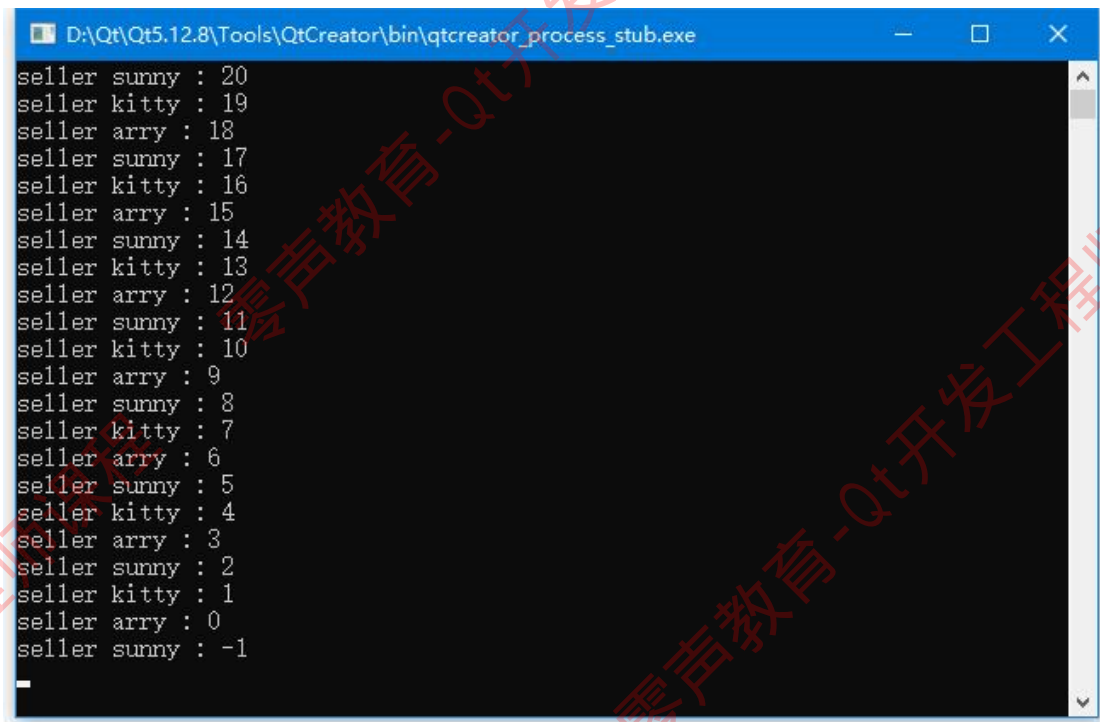
enum **RecursionMode** { Recursive, NonRecursive }

Public Functions

```
QMutex(QMutex::RecursionMode mode = NonRecursive)
~QMutex()
bool isRecursive() const
void lock()
bool tryLock(int timeout = 0)
bool try_lock()
bool try_lock_for(std::chrono::duration<Rep, Period> duration)
bool try_lock_until(std::chrono::time_point<Clock, Duration> timePoint)
void unlock()
```

QMutexLocker 类简化锁定和解锁互斥锁。

【售票窗口模拟程序运行结果】



```
D:\Qt\Qt5.12.8\Tools\QtCreator\bin\qtcreator_process_stub.exe
seller sunny : 20
seller kitty : 19
seller arry : 18
seller sunny : 17
seller kitty : 16
seller arry : 15
seller sunny : 14
seller kitty : 13
seller arry : 12
seller sunny : 11
seller kitty : 10
seller arry : 9
seller sunny : 8
seller kitty : 7
seller arry : 6
seller sunny : 5
seller kitty : 4
seller arry : 3
seller sunny : 2
seller kitty : 1
seller arry : 0
seller sunny : -1
```