Project 2:多线程编程实践

代码设计逻辑

该代码实现了一个多线程的生产者-消费者模型，涉及四个线程：线程A是生产者，线程B、C、D是消费者。这些线程通过共享变量进行通信，其中包括缓冲区（buffer）、缓冲区中未被消费的数据个数（buffer_count）、生产者结束标志（flag）以及三个互斥量（mutex）和两个条件变量（full、empty）。

线程A的作用是不断往缓冲区中插入数据。它首先对缓冲区进行初始化，将所有位置设为0，表示缓冲区为空。然后，它使用互斥锁（mutex）对共享变量进行保护，在缓冲区未满的情况下，往缓冲区的空位插入数据。每次插入后，它会更新缓冲区中未被消费的数据个数（buffer_count），并通过条件变量（empty）通知消费者缓冲区非空。当生产者线程结束后，将生产者结束标志（flag）设置为1，释放互斥锁。

线程B和线程C的作用是从缓冲区中取出能被给定整数（pb或pc）整除的数字，并统计个数和求和。它们使用互斥锁（mutex）对共享变量进行保护，在缓冲区非空的情况下，从缓冲区中取出符合条件的数字，更新统计值，并通过条件变量（full）通知生产者缓冲区非满。线程B处理能被pb整除的数字，线程C处理能被pc整除的数字。

线程D的作用是从缓冲区中取出不能被给定整数（pb和pc）整除的数字，并统计个数和求和。它使用互斥锁（mutex）对共享变量进行保护，在缓冲区非空的情况下，从缓冲区中取出不符合条件的数字，更新统计值，并通过条件变量（full）通知生产者缓冲区非满。

在主函数中，首先检查命令行参数是否符合要求，然后读取输入的三个整数，作为全局变量pa、pb和pc的值。然后创建四个线程，并等待它们执行完毕。

通过互斥锁和条件变量的使用，实现了线程之间的同步和互斥，保证了对共享变量的安全访问，避免了数据竞争的问题。

测试示例

使用给定的五组测试用例

```
./main 10 5 3
./main 100 5 3
./main 200 5 3
./main 200 10 5
./main 200 10 3
```

分别得到如下的输出结果

```
./main 10 5 3
Thread C, 3, 18
Thread D, 5, 22
Thread B, 2, 15
```

```
./main 100 5 3
Thread D, 53, 2632
Thread C, 32, 1638
Thread B, 15, 780
```

```
./main 200 5 3
Thread D, 107, 10732
Thread C, 63, 6483
Thread B, 30, 2885
```

```
./main 200 10 5
Thread C, 35, 3750
Thread D, 160, 16000
Thread B, 5, 350
```

```
./main 200 10 3
Thread C, 64, 6363
Thread B, 16, 1740
Thread D, 120, 11997
```