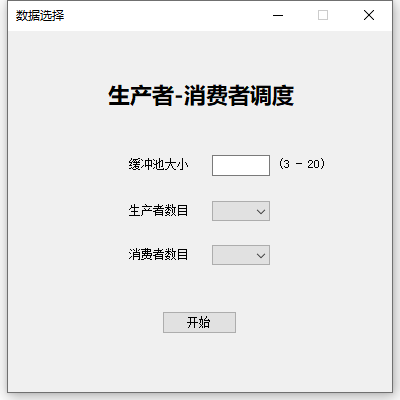
**生产者-消费者调度程序**

**使用方法：**

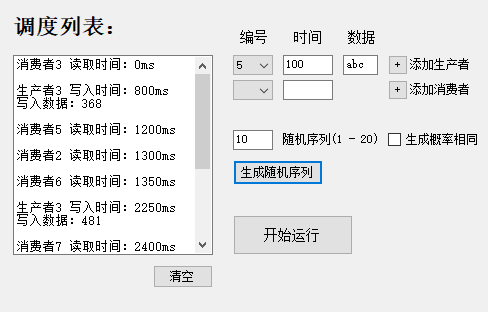
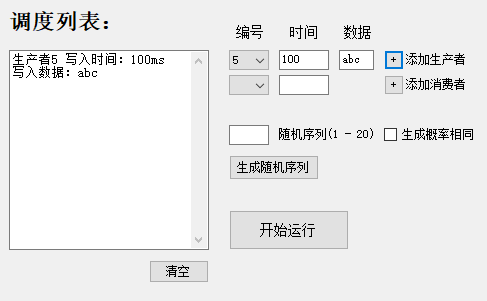
1、运行程序。输入缓冲池数目，选择生产者，消费者数目后进入主界面。



2、右下方为添加进程界面，可选择生产者、消费者，输入该进程开始调度的时间，和写入的数据（仅生产者），输入的进程开始调度时间要不早于最后一个加入调度列表的进程的开始调度时间。单击右侧的添加按钮即可添加该进程进入调度列表。最大添加量为20。

可以输入列表长度，随机生成进程列表。可选择生成生产者，消费者的概率是否相同（否则两者生成概率之比为两者数量之比）。

可点击清空按钮清空列表重新输入。随机生成进程列表时默认先清空列表。



3、输入调度列表后，点击“开始运行”按钮即可观察到生产者，消费者的运行情况，左下侧信息栏实时显示运行信息。在运行期间，大部分按钮禁止操作。



4、运行完毕后，可选择清空缓冲池，清空运行信息，更改调度列表，再次运行等操作。

**规则：**

**生产者-消费者调度：**

1、若干进程通过有限的共享缓冲区交换数据；

2、“生产者”进程不断放入数据，而“消费者”进程不断取走数据；

3、共享缓冲区共有N个；任何时刻只能有一个进程可对共享缓冲区进行操作。

4、生产者/消费者通过等待获取缓冲区容量信号量（是否空满）和缓冲区执行信号量（是否有其他进程正在访问缓冲区）来获得读写权限

5、当缓冲区满时，生产者必须等待直到缓冲区出现可写空位；当缓冲区空时，消费者必须等待直到缓冲区出现可读数据。

**本程序运行规则：**

1、本程序运用程序内线程来模拟进程，主调度线程指调度分发生产者/消费者的线程，执行线程指生产者/消费者线程

2、 执行线程被调度指该线程开始运行，申请信号量，并非指开始执行读写操作。

3、 每个生产者消耗650ms进行写操作，每个消费者消耗350ms进行读操作

4、 当生产者(消费者)在等待缓冲区容量信号量时，若之后已无消费者(生产者)等待执行，则退出等待，输出无法正常执行信息

5、 开始运行时间指生产者/消费者开始对缓冲区执行读写操作时间，非开始调度时间

6、 某个执行线程在运行时可存在被主调度线程尝试再次调度的情况，此时主调度线程创建一个子线程来监测等待执行线程运行完毕，然后再度调用

7、 生产者指针指向下一个写入的位置，消费者指针指向要读取的位置

8、 多个执行线程等待信号量时，选择哪个线程获取信号量来执行读写动作随机

**代码实现：**

本程序通过C#语言运用winform框架来实现。控件及其操作均来自winform框架。

**类：**

**StartForm类**：开始页面类，包含输入缓冲池大小，生产者消费者数目控件，及唤醒主页面按钮

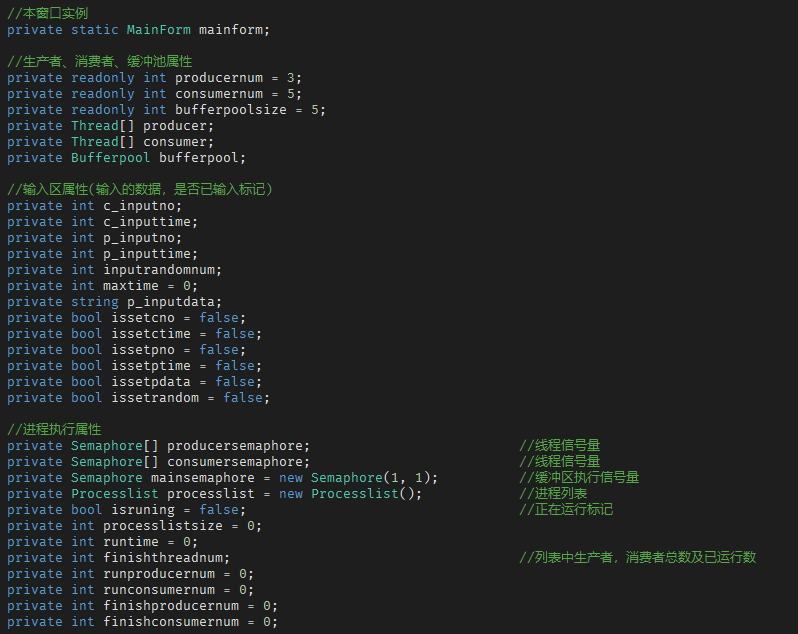
**MainForm类**：主页面类，包含大部分控件及其操作方法，与生产者、消费者调度线程执行方法

**MyProcess类**：为MainForm内部类，存储一个线程的信息，及其相应的执行方法

**Processlist类**：为MainForm内部类，存储进程列表，有添加，清除操作

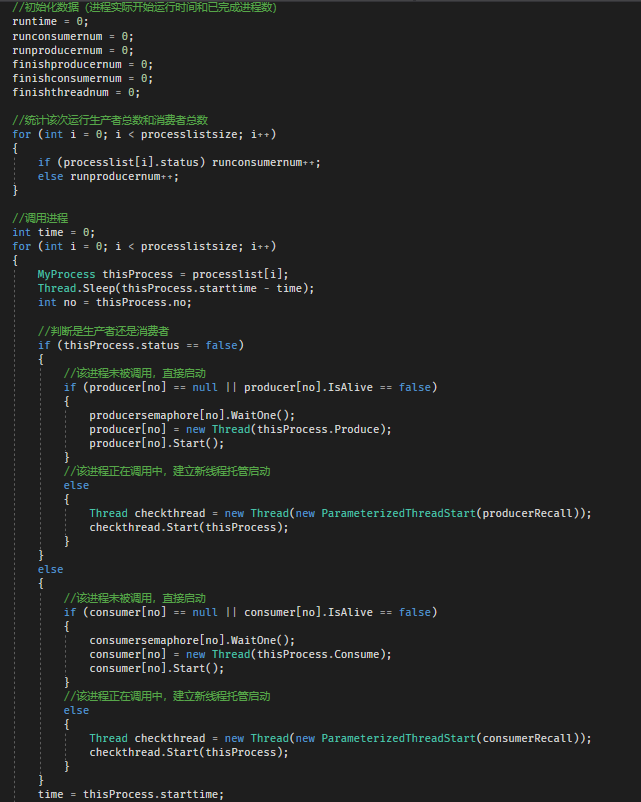
**Bufferpool类**：为MainForm内部类，包含缓冲池的所有控件与往缓冲池进行读写，重置缓冲池的方法，包含缓冲池empty，fill信号量及其Wait，Release，isEmpty等相关操作

**主要属性：**



**主要运行逻辑：**

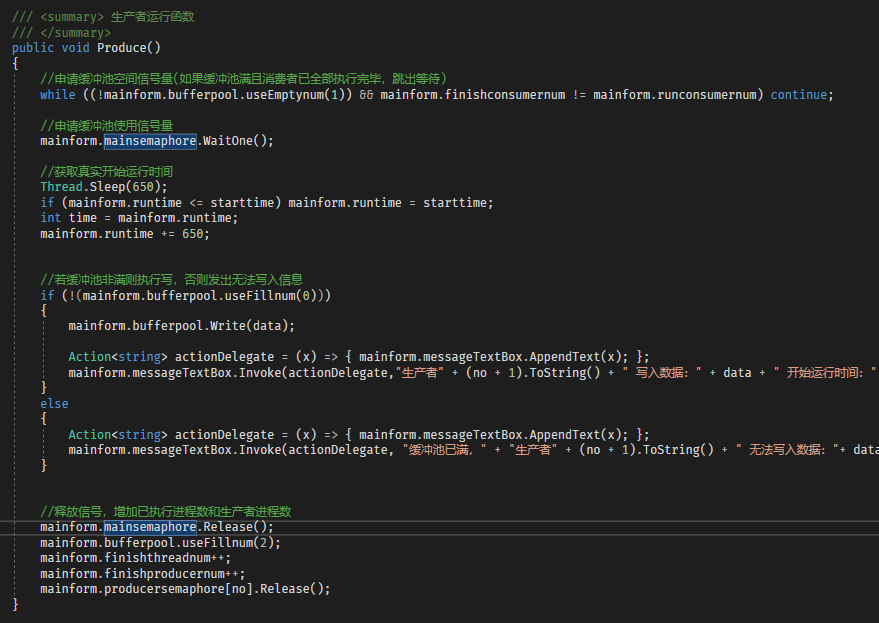
点击开始运行按钮，创建调度线程来调度分发调度列表中的生产者、消费者。首先初始化数据，统计要执行的生产者、消费者数。然后通过每个进程的开始时间来唤醒线程。如果通过检查得到某个线程已被唤醒正在执行，则新建立一个子线程来监测等待该线程执行完成后再度调度执行。检查线程的执行情况通过该线程自己的信号量来实现。



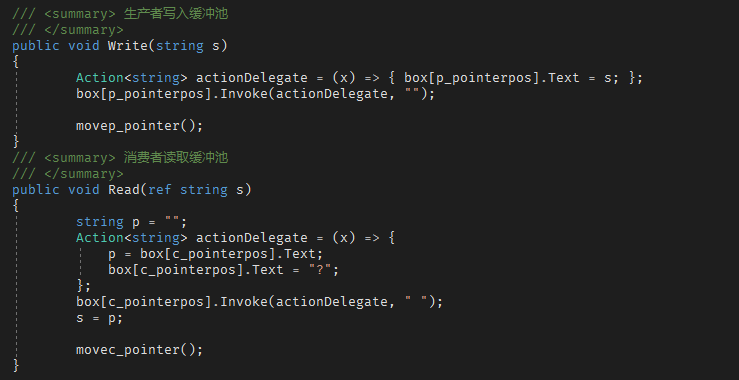


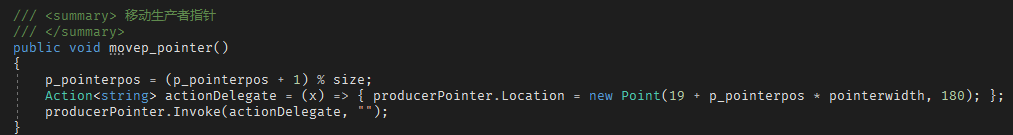
生产者（消费者）线程被唤醒后，申请缓冲区容量(empty，full)信号量。后再申请缓冲区执行信号量。均得到后计算实际开始运行时间，并进行写（读）操作，更新运行信息。然后释放缓冲区执行信号量，释放缓冲区容量(full，empty)信号量，统计生产者，消费者进程执行数目信息，释放本线程信号量。

若申请缓冲区容量(empty，full)信号量时通过统计得知信号量已无法申请，则退出申请等待，输出执行失败信息



缓冲区读/写即在相应指针位置通过委托来跨线程获取/更新相应缓冲区信息，然后移动指针至下一位置，若越界则回到零点





主调度线程尾端等待所有进程执行完毕，然后退出线程，运行完毕。

