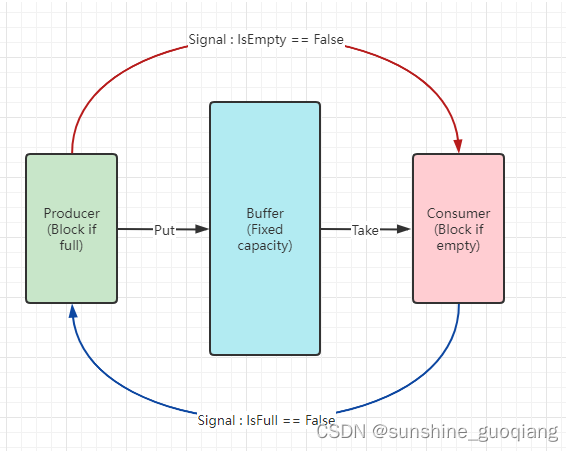
# PV操作解决生产者消费者问题项目文档

2151601 邹昕恺

## 项目描述

1. 基本问题描述：生产者消费者问题是操作系统中的经典问题：系统中有一组生产者进程和一组消费者进程，生产者进程每次生产一个产品放入缓冲区，消费者进程每次从缓冲区中取出一个产品并使用。（注：这里的“产品”理解为某种数据）生产者、消费者共享一个初始为空、大小为n的缓冲区。只有缓冲区没满时，生产者才能把产品放入缓冲区，否则必须等待。(缓冲区没满 ——>生产者生产)只有缓冲区不空时，消费者才能从中取出产品，否则必须等待。(缓冲区没空——>消费者消费)缓冲区是临界资源，各进程必须互斥地访问。(互斥关系)
2. 本题要求：有界缓冲区的容量大小为20，存取1-20这20个整型数。生产者和消费者各有两个以上。当生产者和消费者对有界缓冲区进行操作后,即时显示有界缓冲区的全部内容、当前位置、生产者/消费者进程的标识符。
3. 项目描述：为达到生产者和消费者各有两个以上的题目要求，项目设有五个生产者进程（P1-P5）,五个消费者进程（C1-C5），且一个生产者进程仅对应一个消费者进程，即只有消费者进程C1能消费生产者进程P1进程的生产结果（P2-P5类似）。
4. 最经典的生产者消费者问题为1个生产者，1个消费者，缓冲区大小为1。在经典生产者消费者问题模型的基础上逐渐衍变出了“儿子只能吃妈妈放的橘子，女儿只能吃爸爸生产的苹果”以及“哲学家进餐”问题，本项目在完成经典模型的基础上，还完成了前者变体的方案求解。



## 开发环境

Windows 10

Microsoft Visual Studio 2022

编程语言：c++

## PV操作

1. 信号量

信号量机制是一种功能较强的机制，可用来解决互斥与同步问题，它只能被两个标准的原语wait(S)和signal(S)访问，也可记为“P操作”和“V操作”。

信号量的数据结构为一个值和一个指针，指针指向等待该信号的下一个进程。信号量的值与相应资源的使用情况有关。当信号量的值大于0时，表示当前可用资源的数量；当信号量的值小于0时，其绝对值表示当前阻塞等待使用该资源的进程个数。（信号量值只能用PV操作来改变。）

一般的，当信号量S>=0时，S表示可用资源的数量。执行一次P操作意味着请求分配一个单位资源，因此S的值减1；当S<0时，表示已经没有可用资源，请求者必须阻塞等待别的进程释放该类资源才能继续运行。而执行一个V操作意味着释放一个单位资源，因此S的值加1；若S<=0，表示此刻有进程正在阻塞等待该资源，因此要唤醒一个等待状态的进程，使之运行下去。

本项目中采用建立一个整型值和一个队列的方式来表示每一个信号量。

1. PV操作

PV操作是由P操作原语和V操作原语组成（原语是不可能中断的过程），操作对象是信号量。具体的：

P(S)：① 将信号量S的值减1，即S=S-1；② 如果S>=0，则该进程继续执行；否则进程进入等待队列，置为等待状态。

V(S)：① 将信号量S的值加1，即S=S+1；② 如果S>0，则该进程继续执行；否则释放等待队列中第一个等待信号量的进程。（因为将信号量加1后仍然不大于0，则表示等待队列中有阻塞的进程。）

PV操作的意义：使用PV操作和信号量可以实现进程间的同步和互斥。

1. 信号量

empty是缓冲区中为空的容量大小，初值为20，即buffer的大小

full是缓冲区已被填充的容量大小，初值为0

mutex用于控制互斥访问缓冲区，初值为1

S1-S5用于控制生产者P1-P5与消费者C1-C5之间的同步关系，初值为0

1. PV机制算法框架（以S1信号量为例）：

//生产者进程

while(TRUE)

{

生产一个产品；

P(empty);

P(mutex1);

将一个产品送入缓冲区in；

in = (in+1) mod n;

V(mutex1);

V(S1)

V(full);

}

//消费者进程

while(TRUE)

{

P(full);

P(S1)

P(mutex2);

从缓冲区(out)取出一个产品；

out = (out+1) mod n;

V(empty);

V(mutex2);

消费取出的产品；

}

1. 各信号量算法分析：

① empty的初值为20。每次将一个进程放入缓冲区，就执行一次P(empty)，empty—执行一次当缓冲区被填满后，即P(empty)执行了20次,empty-1被执行了20次，缓冲区中空格子数目为0，empty=0。此时当再有一个就绪队列中的生产者进程试图加入缓冲区，触发P(empty)同时empty--，此时empty<0，则调用P(empty)的该进程被阻塞，并把它插入到等待信号量empty的阻塞队列中，实现生产者进程无法加入至已被填满的buffer。当出现一个消费者进程触发V(empty),empty++被执行，信号量empty的阻塞队列中的第一个进程会被唤醒到就绪队列中，然后调用V(empty)的进程继续运行，当V（empty）执行完成后，表示消费者进程已成功执行，缓冲区获得新空间，这样阻塞队列中被唤醒的生产者进程可以继续执行。

② full的初值为0。当缓冲区有生产者进程并且与该消费者进程相匹配时，则会触发V(full)，执行full++。当缓冲区为空时，若有一个就绪队列中的消费者进程试图执行，则触发P(full),full-1<0,则调用P(full)的进程被阻塞，并把它插入到等待信号量full的阻塞队列中，从而实现当缓冲区为空时不执行消费者进程的作用。直到出现一个生产者进程触发V(full),full+1被执行，等待信号量full的阻塞队列中第一个进程会被唤醒到就绪队列中，然后调用V(full)的进程继续运行。由于生产者PV机制中V(full)原语位于末尾，一旦触发完成就表明存在一个生产者进程执行完毕，也就是说缓冲区中产生了可以被消费者消费的进程，阻塞队列中被唤醒的消费者进程可以继续执行。

③ mutex的初值为1。当有一个生产者进程或者消费者进程正在执行时，执行一次P(mutex)，mutex的值为0。若此时又存在一个进程试图进入到执行队列时，则再一次触发P(mutex),因为mutex-1<0,因此该进程被阻塞，并将其插入到等待信号量mutex的阻塞队列中。当上一个进程执行完毕后，执行V(mutex),此时mutex阻塞队列中被唤醒的第一个进程会进入到执行队列中开始执行，这样就可以保证缓冲区只会有一个进程在执行过程。

④ S进程（以S1为例）,S的初值为0

当缓冲区中没有P1生产者产生的进程时并且此时有一个消费者C1正在尝试进入执行队列，则触发P(S1)原语，S1-1<0，因此该C1被阻塞并且并将C1插入到S1的阻塞队列。当缓冲区有一个P1进程执行完毕时触发V(S1),S1+1<=0。此时，在阻塞队列的C1进程被唤醒，下一步执行C1消费P1。

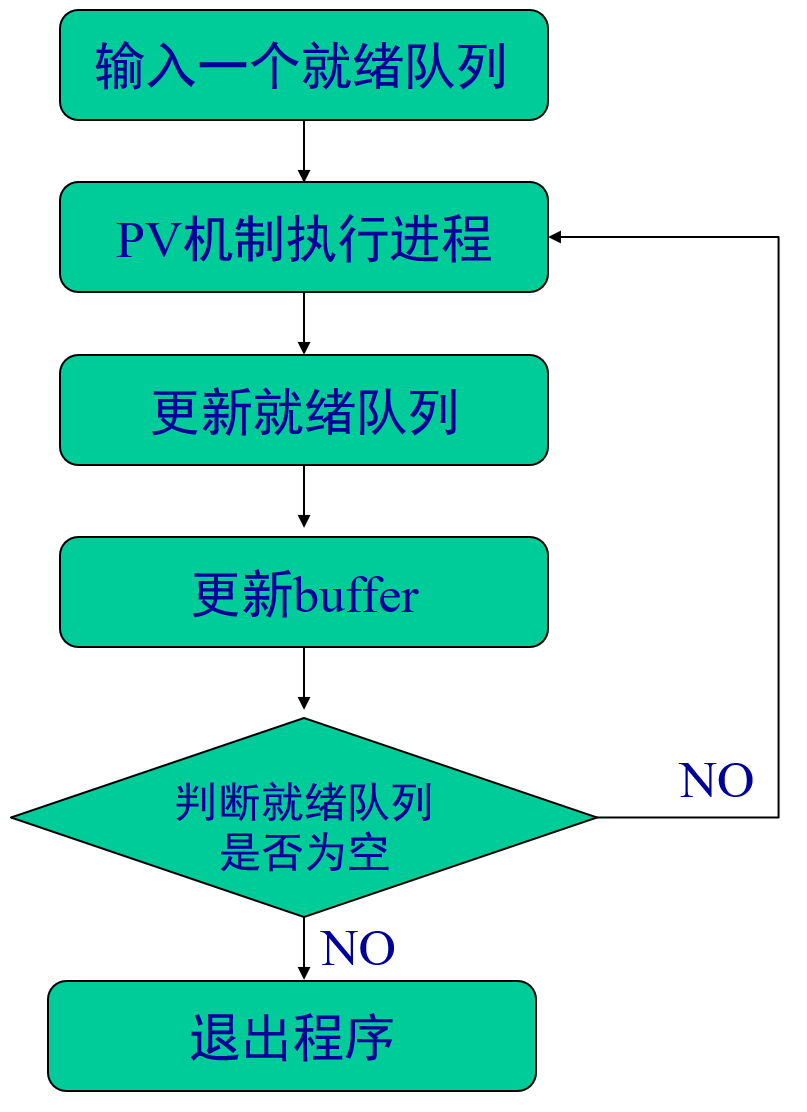
1. 各信号量判断顺序：

对于生产者进程以及消费者进程，优先判断缓冲区是否满足进程的执行情况，对于生产者进程判断缓冲区是否已经被填满，对于消费者进程判断缓冲区是否为空，即根据full和empty的值判断进程是否可以进入缓冲区。

第二步判断S信号量，判断消费者进程是否与生产者进程匹配，即C1是否与P1匹配（C2-C5,P2-P5类似）。

第三步判断mutex信号量，因为一个时刻只能有一个进程在操作缓冲区中。

## 项目程序说明：

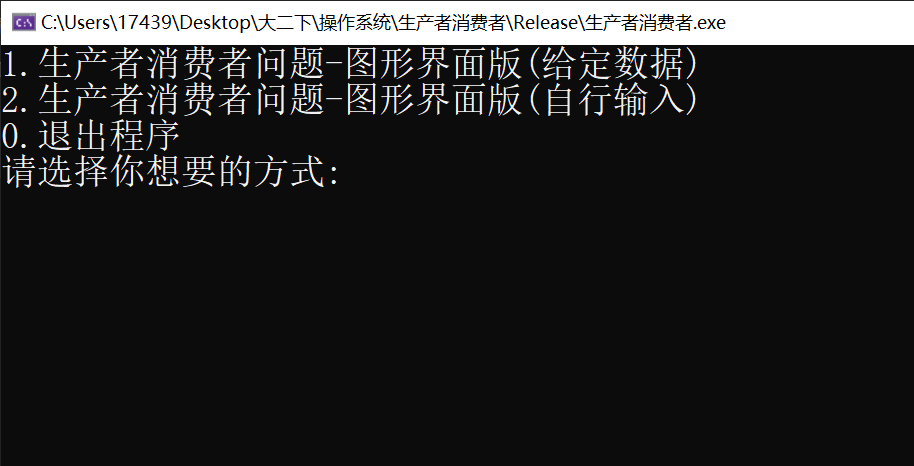
1. 程序流程说明图
2. 程序实现阻塞进程执行

为了使得一个阻塞进程被唤醒后可以继续从来被阻塞的阶段继续执行，因此采用以下的方式：

① 当有一个阻塞队列中的进程被唤醒到就绪队列时，它不会按照正常的安排方式将该进程插入到就绪队列的末尾，而是放到就绪队列的第一位，这样就可以实现PV机制中被唤醒的进程在下一步执行的效果。

② 为了使被唤醒的进程从原来被阻塞的地方接着执行，而不是将该进程从头重新执行，因此本程序通过在函数中设置一个level参数来比较严格性等级作为判断的标记。e.g.当其中一个生产者进程P2执行完P(empty),被阻塞在empty信号量的阻塞队列中，此时empty对应的level为0.当V(empty)语句被执行时，该进程的对应的level变为1大于empty对应的level，因此当P2进程从被阻塞的地方被释放进入就绪队列时，该进程下一步执行的应该是P(mutex)，从而实现一个阻塞进程被唤醒后可以继续从来被阻塞的阶段继续执行。

1. 程序执行文件演示及说明
   1. 菜单界面



输入1，则进入生产者消费者问题-图形界面版（给定数据）界面，该界面无需手动输入生产者进程或消费者进程，根据已给的就绪队列观察并且更好地PV机制如何处理一系列的进程。

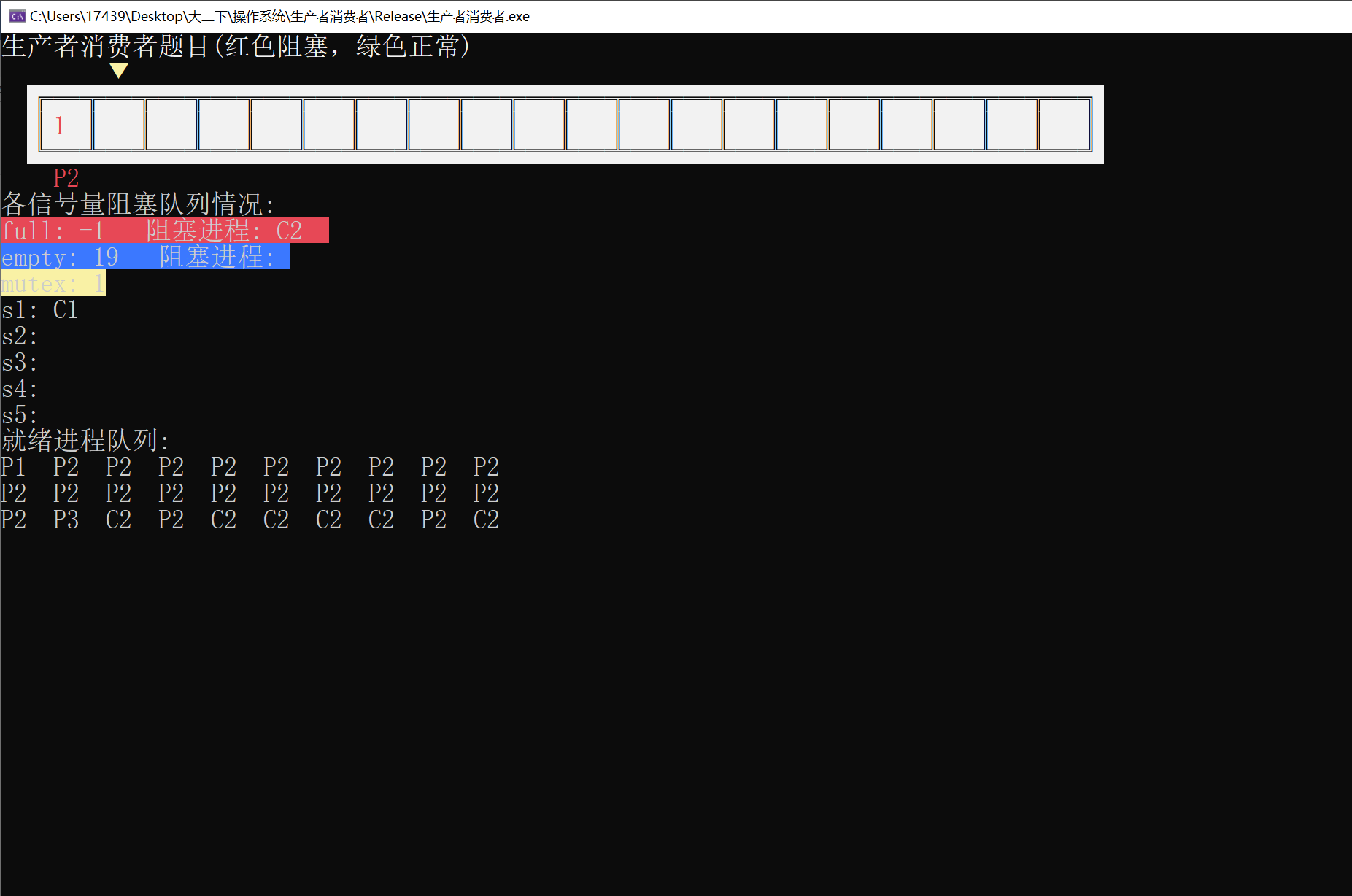
输入2，则进入生产者消费者问题-图形界面版（自行输入）界面，进入该界面用户可以手动输入想要执行的生产者进程或者消费者进程，一次仅能输入一个进程并且大小写敏感。同时在就绪队列中已存在如菜单项1的数据。当不输入新的进程时，按一次回车会执行一次操作。

输入0，退出程序。

如果输入除以上之外的其他选项，程序将不做反应。

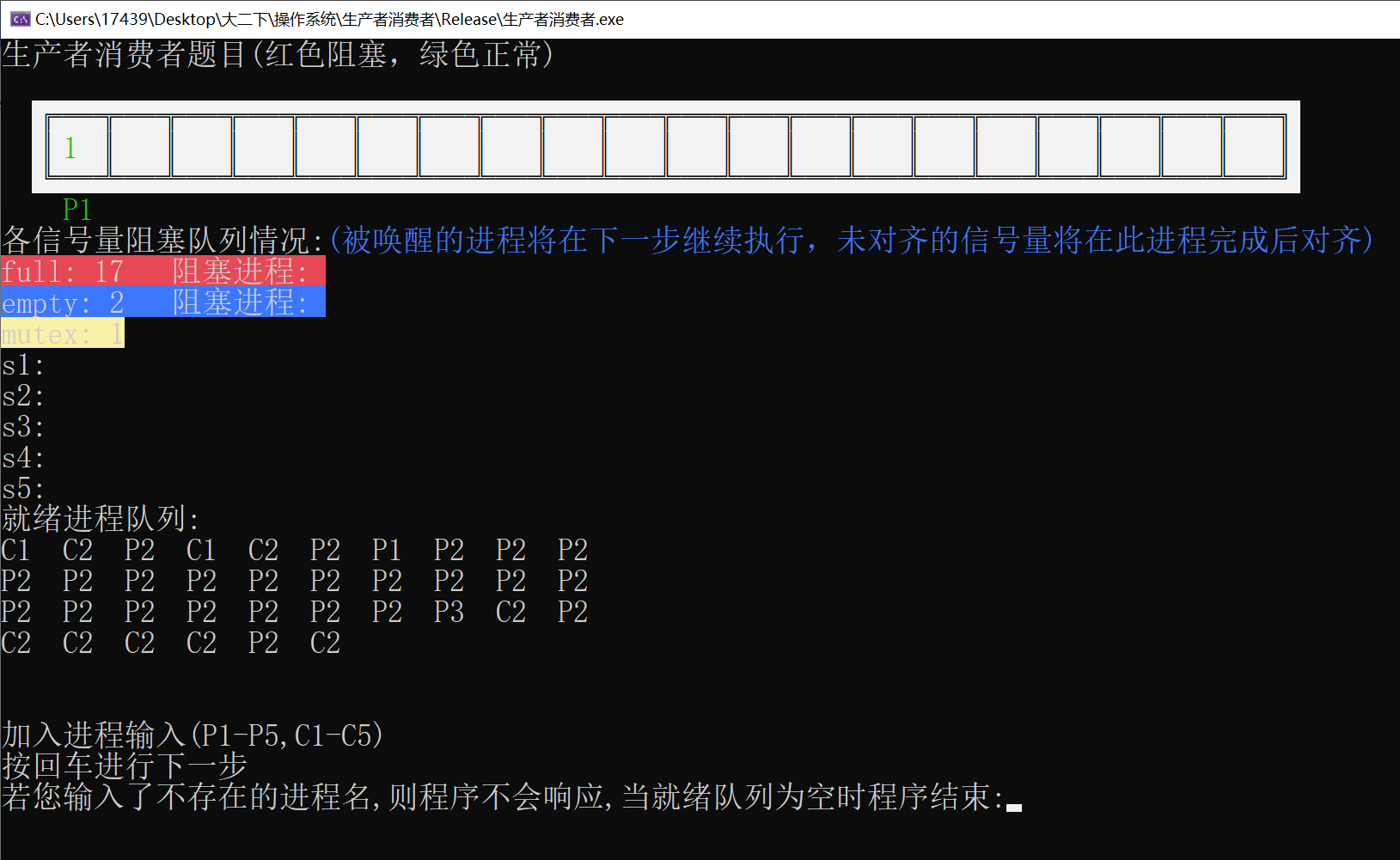
* 1. 菜单项1

1. 图中的白色框代表缓冲区，缓冲区大小为20
2. 缓冲区单元格上方的小三角形表示即将对该单元进行操作
3. 缓冲区下方的字符代表该进程有P1-P5哪个生产者生产的
4. 缓冲区框中的数字代表该缓冲区单元格中被填入的数字
5. 如果一个进程进入操作缓冲区后被阻塞，则该进程会显示在相应的阻塞队列，同时缓冲区的进程会显示为红色以表示正处于阻塞状态。正常运行情况下，缓冲区的字体颜色为绿色。



* 1. 菜单项2

该菜单选项，用户既可以观察以给定就绪队列的进程情况，也可以自行选择加入新的进程标识名，生产者进程P1-P5,消费者进程C1-C5，大小写敏感且每次只能输入一个进程。如果输入的不符合规范则输入无效，用户可以重新输入。



## 项目心得

本次通过对生产者消费者问题的c++代码实现以及简略的图形化界面实现，更有效地帮助了我理解同步、互斥的操作，同时对PV原语的操作以及其代码实现也有了更深的了解，巩固了老师在课堂上的授课知识。同时在基础生产者消费者的问题上，考虑到对进程同步互斥问题的探索以及程序复杂度的考量，本项目完成了“儿子只能吃妈妈放的橘子，女儿只能吃爸爸生产的苹果”的变体，通过先修课程高级语言程序设计的授课老师沈坚老师以及图形化实现的相关代码，在exe文件中简略地实现了生产者消费者问题的图形化界面，之后可用于课后复习或者预习教案。