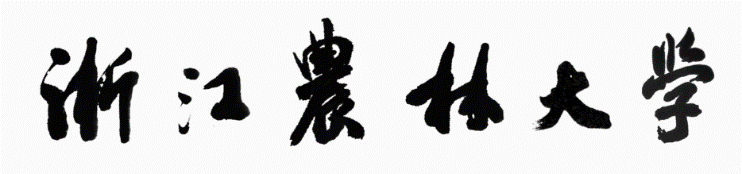
**** 

**《操作系统》期末课程实验报告**

**（2020-2021学年第一学期）**

**Name（姓名）：周炜翔、张金浩**

**Student ID（学号）：201805120532、201805120629**

**Department（学院）：信息工程学院**

**Major（专业）：计算机科学与技术**

**Class（班级）：183班**

**Date（日期）：2021/01/05**

目录

[一、课题设计的目的 3](#_Toc60857321)

[二、课程设计的问题描述 3](#_Toc60857322)

[2.1 问题描述 3](#_Toc60857323)

[2.2 实验要求 3](#_Toc60857324)

[三、详细设计 4](#_Toc60857325)

[3.1 算法分析 4](#_Toc60857326)

[3.2 算法伪代码 4](#_Toc60857327)

[3.3 丈夫算法流程图 6](#_Toc60857328)

[3.4 妻子算法流程图 7](#_Toc60857329)

[3.5 小狗算法流程图 8](#_Toc60857330)

[3.6 小猫算法流程图 9](#_Toc60857331)

[四、源代码 10](#_Toc60857332)

[五、测试结果 15](#_Toc60857333)

[5.1 测试样例表格一览 15](#_Toc60857334)

[5.2 测试样例对应截图 15](#_Toc60857335)

[六、总结 21](#_Toc60857336)

# 一、课题设计的目的

了解、体会、运用、掌握操作系统互斥与同步。

# 二、课程设计的问题描述

## 2.1 问题描述

房间有一只碗，每次只能放入一个巧克力。丈夫专向碗中放黑巧克力，妻子专向碗中放白巧克力，小狗专等吃碗里的黑巧克力，小猫专等吃碗里的白巧克力。只要碗空，则丈夫或妻子都可以向碗放一个巧克力，仅当碗中有自己需要的巧克力时，小狗或小猫可以从碗中取出巧克力。

## 2.2 实验要求

1. 每次只能放进一块；
2. 每次只能拿出一块；
3. 放进和拿出不能同时发生；
4. 在同一时间，只能有一种宠物吃一块；
5. 在同一时间，只能有一个人放进一块；
6. 丈夫放黑色的，妻子放白色的；
7. 小狗只吃黑色的，小猫只吃白色的；

# 三、详细设计

## 3.1 算法分析

丈夫和小狗是相互制约的，丈夫进程执行完，即往盘中放入黑巧克力后，小狗进程才能执行即吃黑巧克力，是同步关系；妻子和小猫是相互制约的，妻子进程执行完，即往盘中放入白巧克力，小猫进程才能执行即吃白巧克力，也是同步关系而丈夫和妻子这两个进程不能同时进行，是互斥关系。

## 3.2 算法伪代码

1. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*伪代码\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
2. **void** father()
3. {
4. **while**(**true**)
5. {
6. wait(empty);//等待盘空，申请临界资源
7. produce black\_chocolate;
8. signal(black\_chocolate);//black\_chocolate++,告知小狗有黑巧克力了
9. }
10. }
12. **void** mother()
13. {
14. **while**(**true**)
15. {
16. wait(empty);//等待盘空，申请临界资源
17. produce white\_chocolate;
18. signal(white\_chocolate);//white\_chocolate++,告知小猫有白巧克力了
19. }
20. }
22. **void** dog()
23. {
24. **while**(**true**)
25. {
26. wait(black\_chocolate);//等待黑巧克力
27. consume black\_chocolate
28. signal(empty);//empty++,告知爸爸盘子空了
29. }
30. }
31. **void** cat()
32. {
33. **while**(**true**)
34. {
35. wait(white\_chocolate);//等待白巧克力
36. consume white\_chocolate
37. signal(empty);//empty++,告知妈妈盘子空了
38. }
39. }

## 3.3 丈夫算法流程图

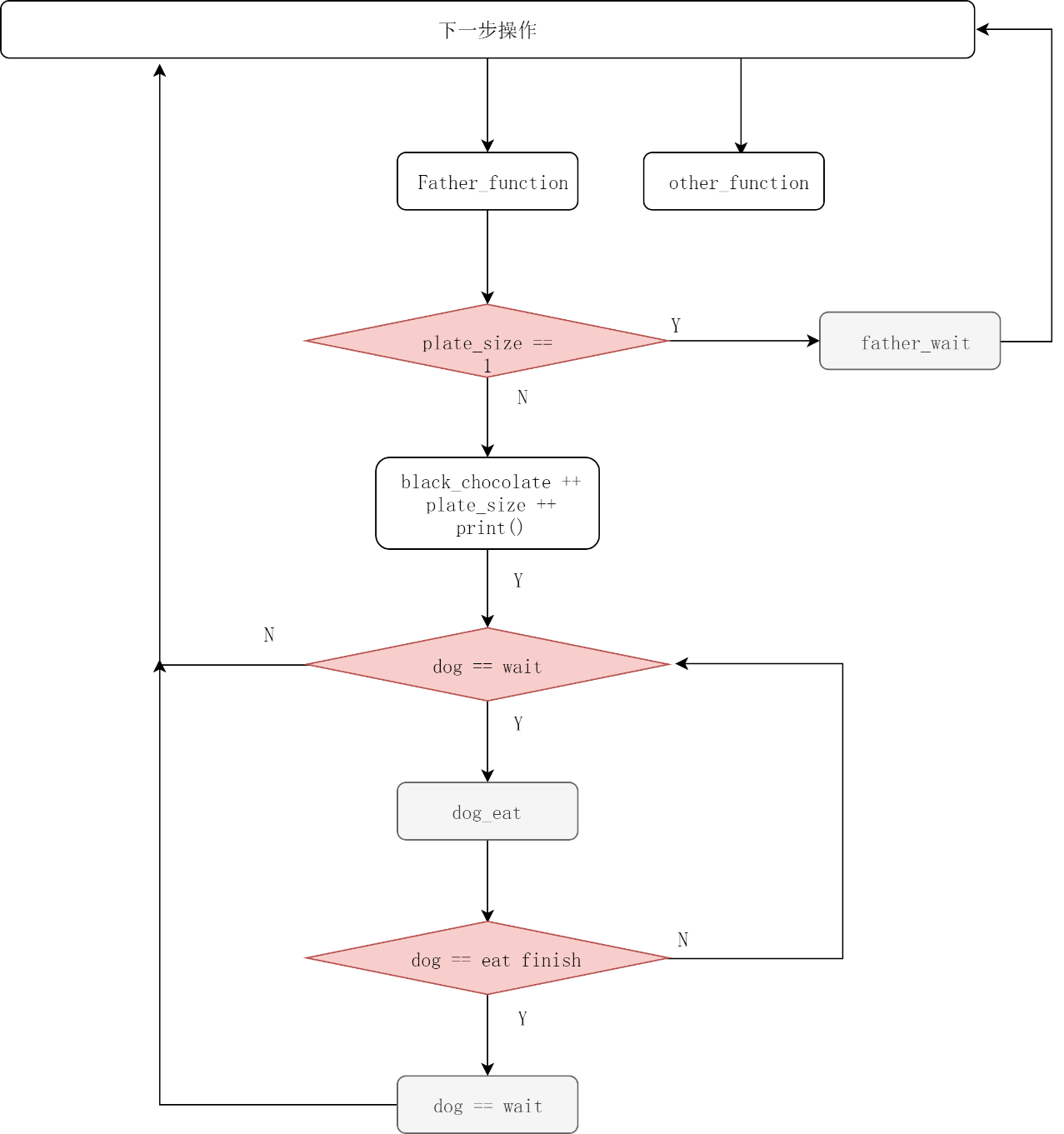


图3-2 父亲算法流程图

## 3.4 妻子算法流程图

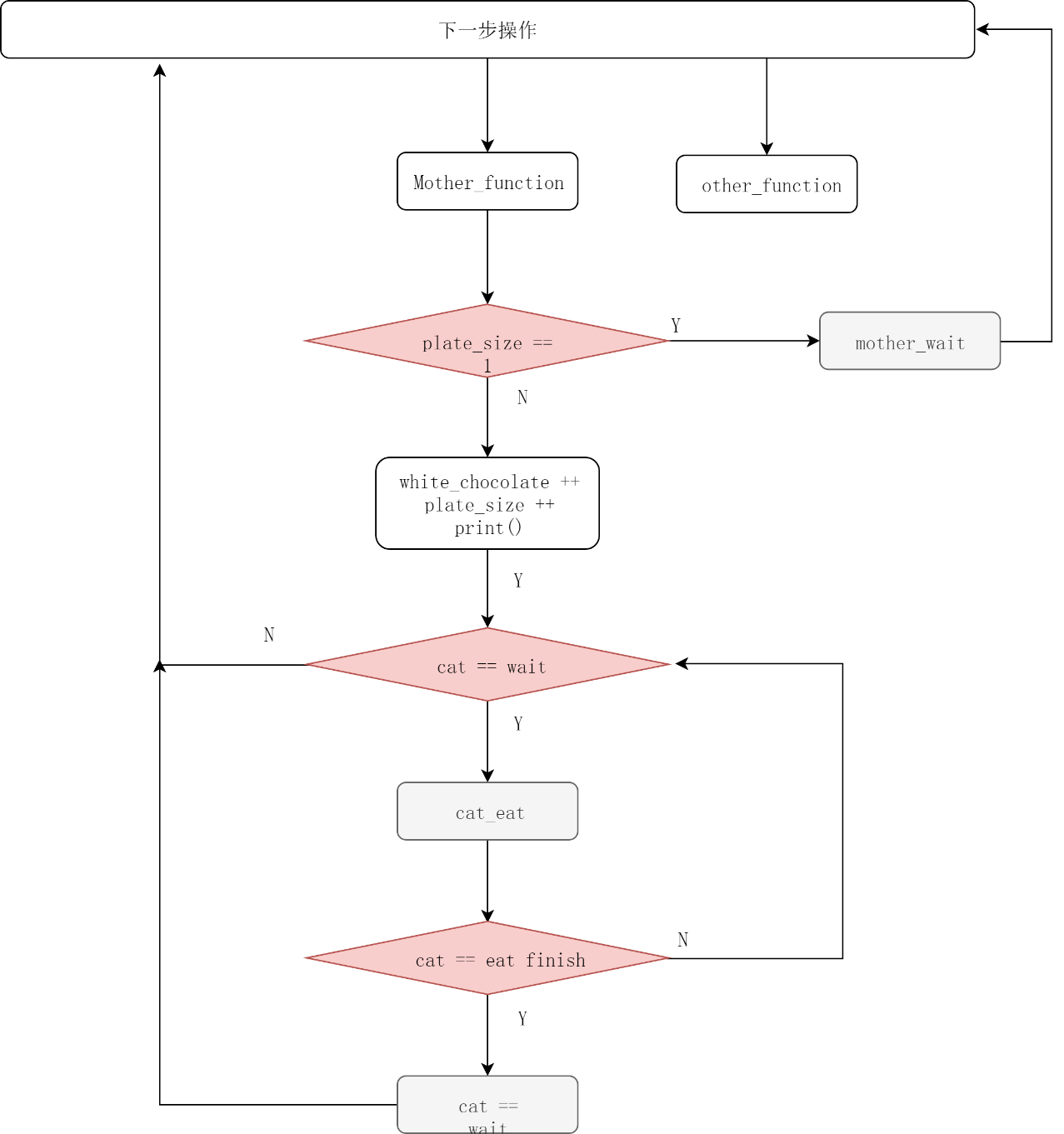


图3-3 母亲算法流程图

## 3.5 小狗算法流程图

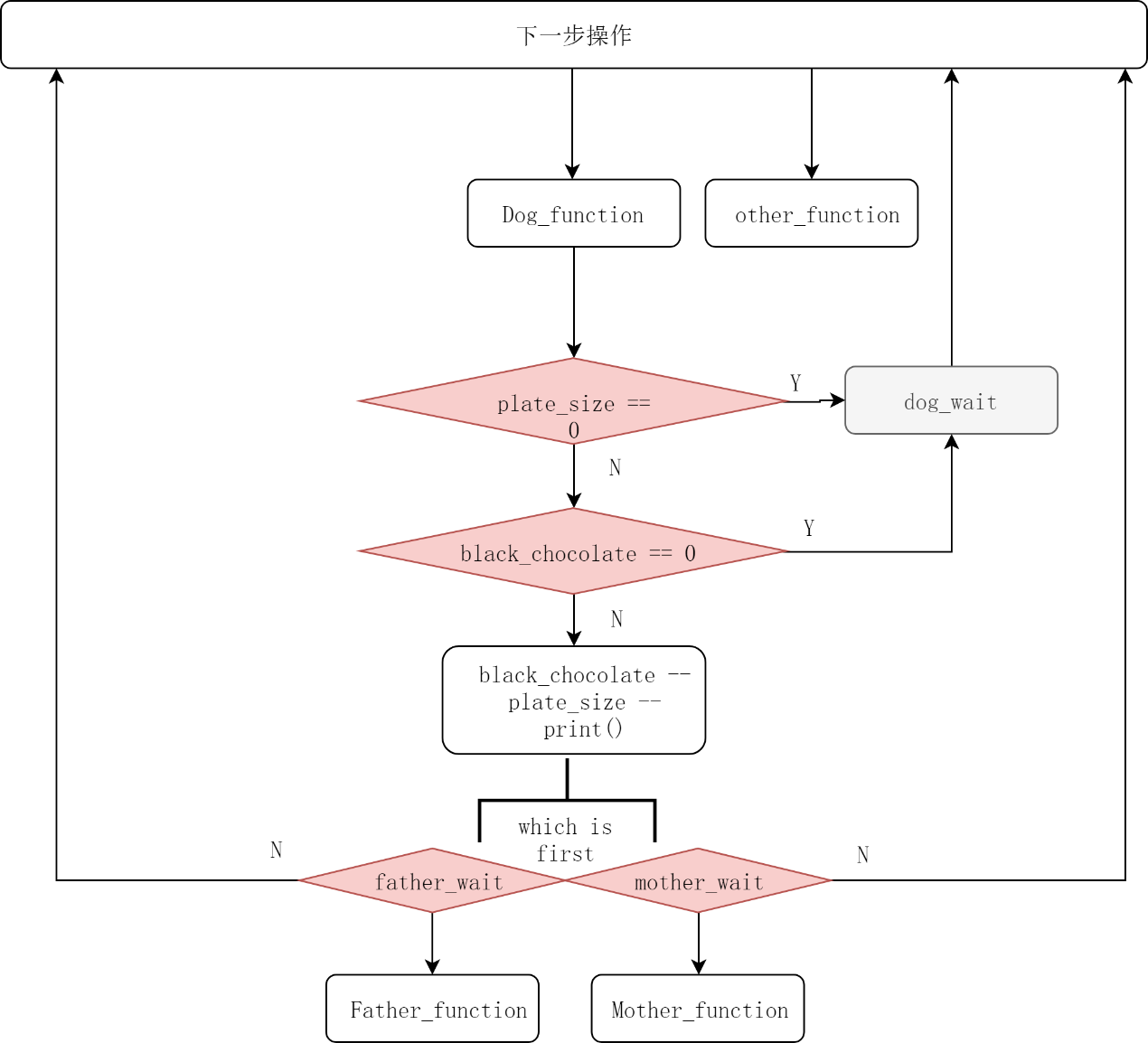


图3-3 小狗算法流程图

## 3.6 小猫算法流程图

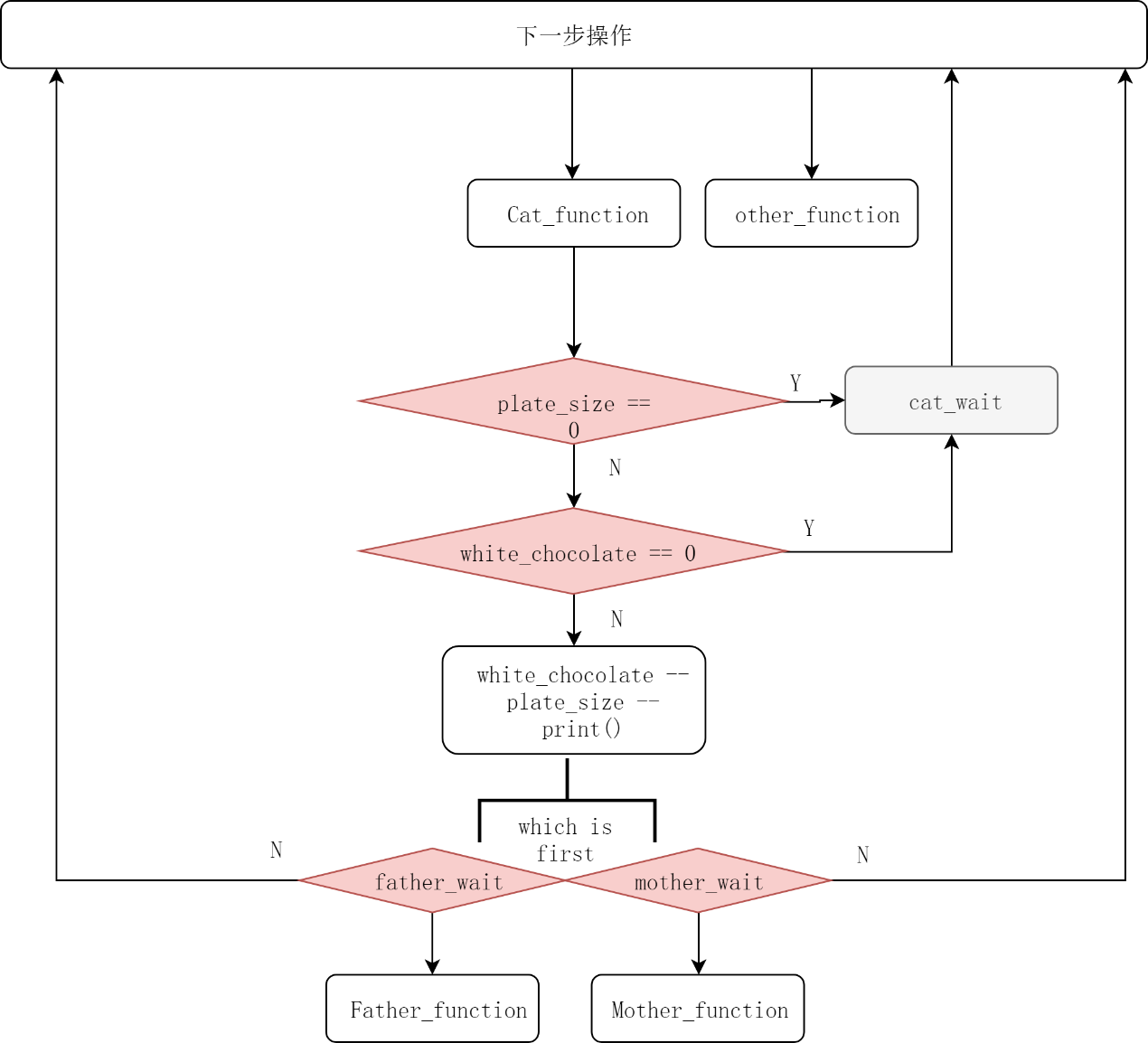


图3-3 小猫算法流程图

# 四、源代码

1. #include <windows.h>
2. #include <iostream>
4. **using** **namespace** std;


8. //声明句柄
9. **HANDLE** EmptyPlate;          //声明空盘子
10. **HANDLE** husbandThread;       //声明丈夫线程
11. **HANDLE** wifeThread;          //声明妻子线程
12. **HANDLE** bCoco;               //声明黑巧克力
13. **HANDLE** wCoco;               //声明白巧克力
14. **HANDLE** dogThread;           //声明小狗线程
15. **HANDLE** catThread;           //声明小猫线程
17. //线程函数声明
18. **DWORD** WINAPI husband(**LPVOID** IpParameter);       //丈夫线程函数声明
19. **DWORD** WINAPI wife(**LPVOID** IpParameter);          //妻子线程函数声明
20. **DWORD** WINAPI dog(**LPVOID** IpParameter);           //小狗线程函数声明
21. **DWORD** WINAPI cat(**LPVOID** IpParameter);           //小猫线程函数声明
23. //初始化黑、白巧克力
24. **int** wConoNum = 0;           //黑巧克力定为0
25. **int** bConoNum = 0;           //白巧克力定为0
26. **int** num = 0;
28. **class** PvTest
29. {
30. **public**:
31. PvTest();
32. ~PvTest();
33. };
35. PvTest::PvTest()
36. {
38. }
40. //析构回收线程以及信号量
41. PvTest::~PvTest()
42. {
44. }
46. **int** main()
47. {
48. cout << " 请输入线程的次数(>=0): " << endl;
49. cin >> num;
50. **while** (num < 0)
51. {
52. cout << " 输入线程的次数不符合要求,请重新输入! " << endl;
53. cout << " 请输入线程的次数(>=0): " << endl;
54. cin >> num;
55. }
57. //创建信号量
58. EmptyPlate = CreateSemaphore(NULL, 1, 1, NULL);
59. bCoco = CreateSemaphore(NULL, 0, 1, NULL);                          //黑巧克力
60. wCoco = CreateSemaphore(NULL, 0, 1, NULL);                          //白巧克力
62. //创建线程
63. // \* 参数 1 内核安全属性 null 为默认
64. // \* 参数 2 线程可用的空间大小
65. // \* 参数 3 表示线程函数的地址
66. // \* 参数 4 传给线程的参数
67. // \* 参数 5 控制线程创建，为 0 表示创建后立即启动，CREATE\_SUSPENDED 表示创建后先暂停
68. // \* 参数 6 表示 lpTHreadId 返回的线程 ID ， 传入 NULL 表示不需要 返回 ID 号
69. husbandThread = CreateThread(NULL, 0, husband, NULL, 0, NULL);      //创建丈夫线程
70. wifeThread = CreateThread(NULL, 0, wife, NULL, 0, NULL);            //创建妻子线程
71. dogThread = CreateThread(NULL, 0, dog, NULL, 0, NULL);              //创建小狗线程
72. catThread = CreateThread(NULL, 0, cat, NULL, 0, NULL);              //创建小猫线程
74. //等线程的结束
75. WaitForSingleObject(husbandThread, INFINITE);                       //等待丈夫线程结束
76. WaitForSingleObject(wifeThread, INFINITE);                          //等待妻子线程结束
77. WaitForSingleObject(dogThread, INFINITE);                           //等待小狗线程结束
78. WaitForSingleObject(catThread, INFINITE);                           //等待小猫线程结束
80. //关闭线程句柄
81. CloseHandle(husbandThread);                                         //关闭丈夫线程
82. CloseHandle(wifeThread);                                            //关闭妻子线程
83. CloseHandle(dogThread);                                             //关闭小狗线程
84. CloseHandle(catThread);                                             //关闭小猫线程
86. //关闭信号量句柄
87. CloseHandle(EmptyPlate);                                            //关闭空盘子句柄
88. CloseHandle(wCoco);                                                 //关闭黑巧克力句柄
89. CloseHandle(bCoco);                                                 //关闭白巧克力句柄
91. **return** 0;
92. }

95. //丈夫线程
96. **DWORD** WINAPI husband(**LPVOID** IpParameter)
97. {
98. **for** (**int** i = 0; i < num; ++i)
99. {
100. WaitForSingleObject(EmptyPlate, INFINITE);              //P操作
101. // 开始临界区
102. bConoNum++;
103. cout << "\n丈夫往盘中放一个黑巧克力\n";
104. cout << "盘子中情况：" << ((bConoNum == 1) ? "○" : "●") << endl;
105. // 结束临界区
106. ReleaseSemaphore(bCoco, 1, NULL);                       //V操作
107. Sleep(1000);
108. }
109. **return** 0;
110. }
112. //妻子线程
113. **DWORD** WINAPI wife(**LPVOID** IpParmeter)
114. {
115. **for** (**int** i = 0; i < num; ++i)
116. {
117. WaitForSingleObject(EmptyPlate, INFINITE);              //P操作
118. // 开始临界区
119. wConoNum++;
120. cout << "\n妻子往盘中放一个白巧克\n";
121. cout << "盘子中情况：" << (( bConoNum == 1) ? "○" : "●") << endl;
122. // 结束临界区
123. ReleaseSemaphore(wCoco, 1, NULL);                       //V操作
124. Sleep(500);
125. }
126. **return** 0;
127. }
129. //小狗线程
130. **DWORD** WINAPI dog(**LPVOID** IpParameter)
131. {
132. **for** (**int** i = 0; i < num; ++i)
133. {
134. WaitForSingleObject(bCoco, INFINITE); //p操作
135. bConoNum--;
136. cout << "小狗吃黑巧克!" << endl;
137. ReleaseSemaphore(EmptyPlate, 1, NULL); //v操作
138. }
139. **return** 0;
140. }
142. //小猫线程
143. **DWORD** WINAPI cat(**LPVOID** IpParameter)
144. {
145. **for** (**int** i = 0; i < num; ++i)
146. {
147. WaitForSingleObject(wCoco, INFINITE); //p操作
148. wConoNum--;
149. cout << "小猫吃白巧克力!" << endl;
150. ReleaseSemaphore(EmptyPlate, 1, NULL); //v操作
151. }
152. **return** 0;
153. }

# 五、测试结果

## 5.1 测试样例表格一览

（1代表黑巧克力，0代表白巧克力）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 实验操作 | 理想结果 | 实验结果 | 是否正确 | 对应图片 |
| 01 | 输入线程次数-1 | 报错并要求重新输入 | 报错并要求重新输入 | 是 | 5-2-01 |
| 02 | 输入线程次数-1 | 退出 | 退出 | 是 | 5-2-02 |
| 03 | 输入线程次数1 | 各执行1次，丈夫先 | 各执行1次，丈夫先 | 是 | 5-2-03 |
| 04 | 输入线程次数1 | 各执行1次，妻子先 | 各执行1次，妻子先 | 是 | 5-2-04 |
| 05 | 输入线程次数2 | 各2次 | 各2次，1001 | 是 | 5-2-05 |
| 06 | 输入线程次数2 | 各2次 | 各2次，0101 | 是 | 5-2-06 |
| 07 | 输入线程次数3 | 各3次 | 各3次，100101 | 是 | 5-2-07 |
| 08 | 输入线程次数3 | 各3次 | 各3次，010101 | 是 | 5-2-08 |
| 09 | 输入线程次数4 | 各4次 | 各4次，10010011 | 是 | 5-2-09 |
| 10 | 输入线程次数4 | 各4次 | 各4次，01010011 | 是 | 5-2-10 |
| 11 | 输入线程次数5 | 各5次 | 各5次，1001001011 | 是 | 5-2-11 |
| 12 | 输入线程次数5 | 各5次 | 各5次，0101001011 | 是 | 5-2-12 |

## 5.2 测试样例对应截图

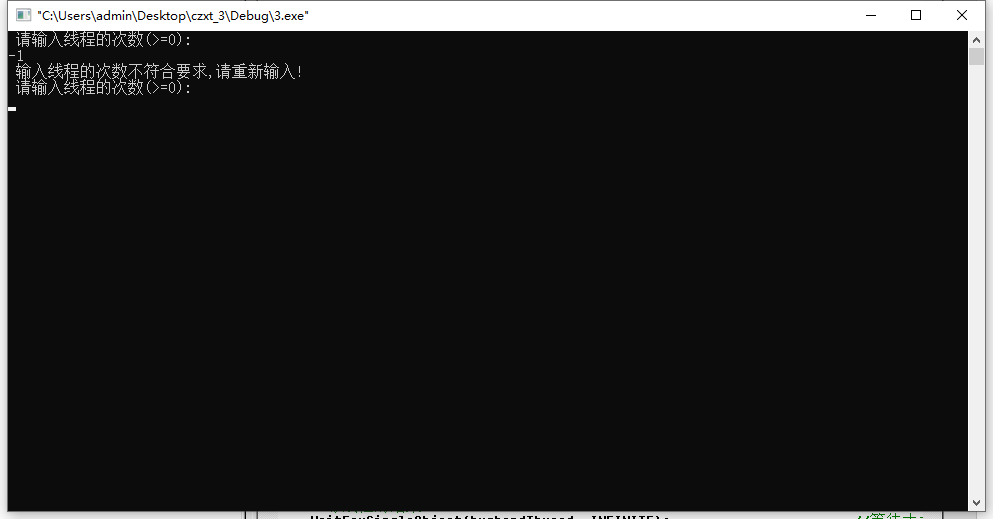


图 5-2-01

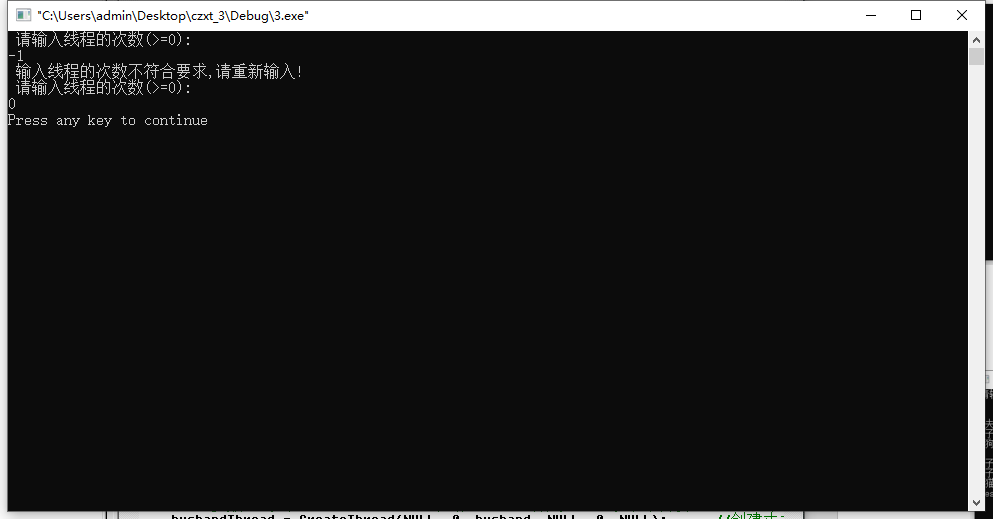


图 5-2-02

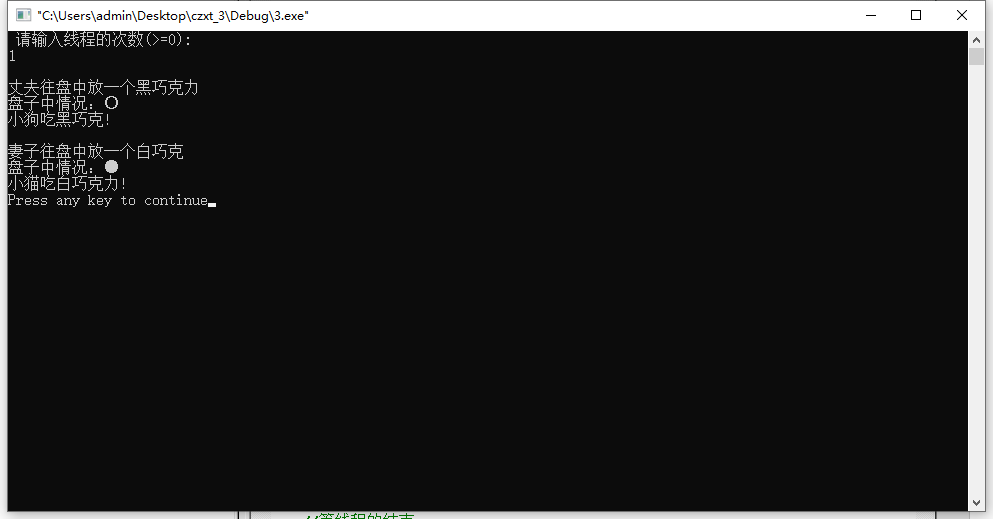


图 5-2-03

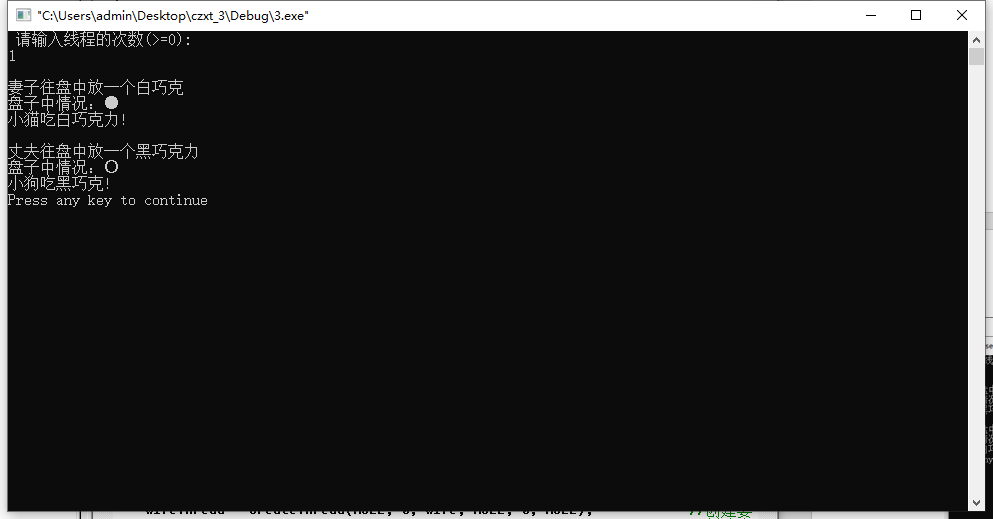


图 5-2-04

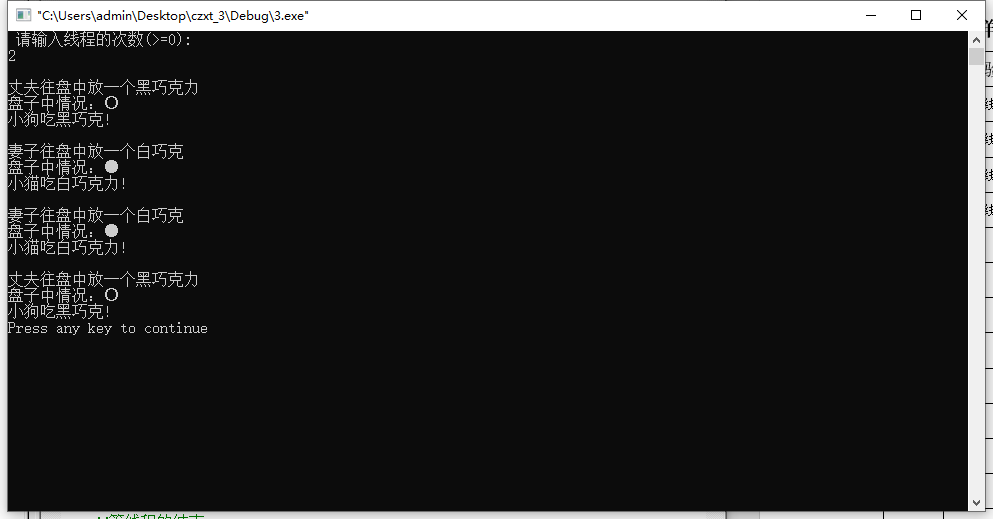


图 5-2-05

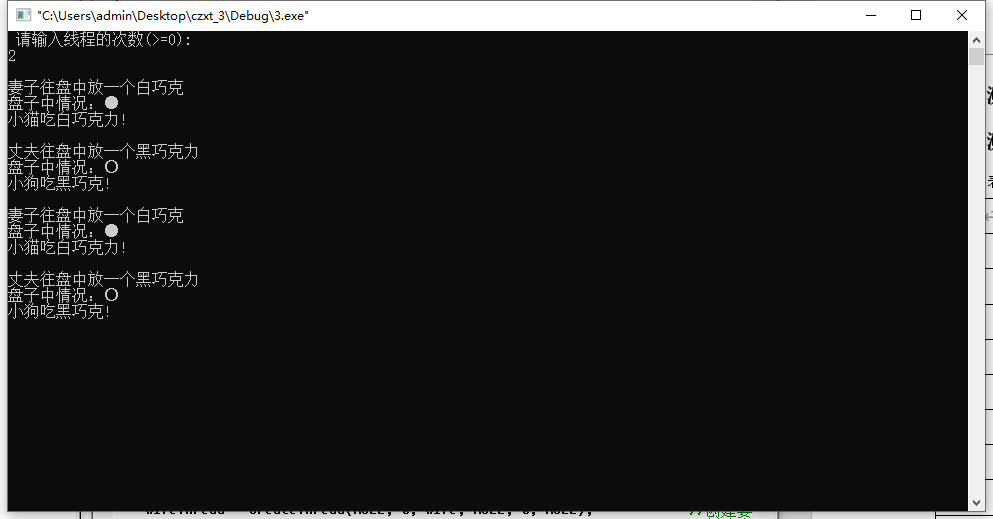


图 5-2-06

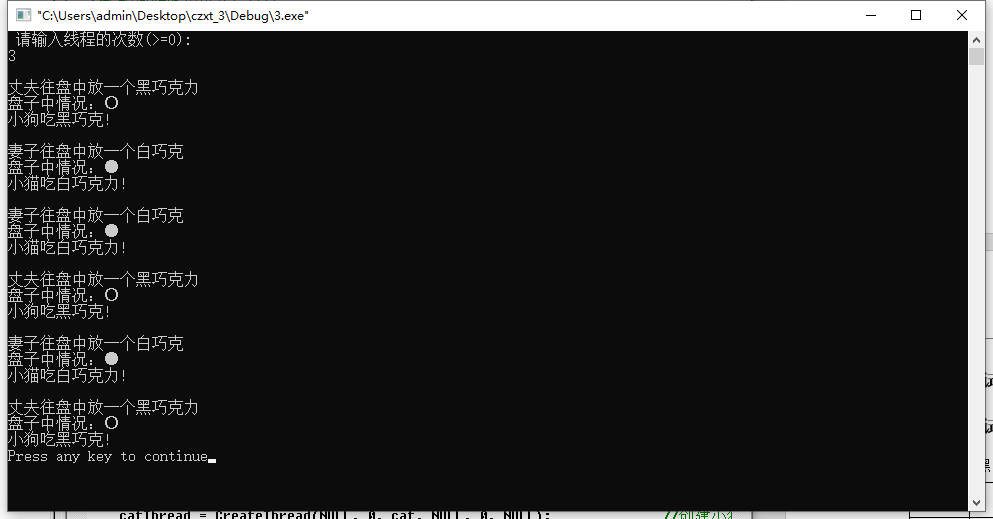


图 5-2-07

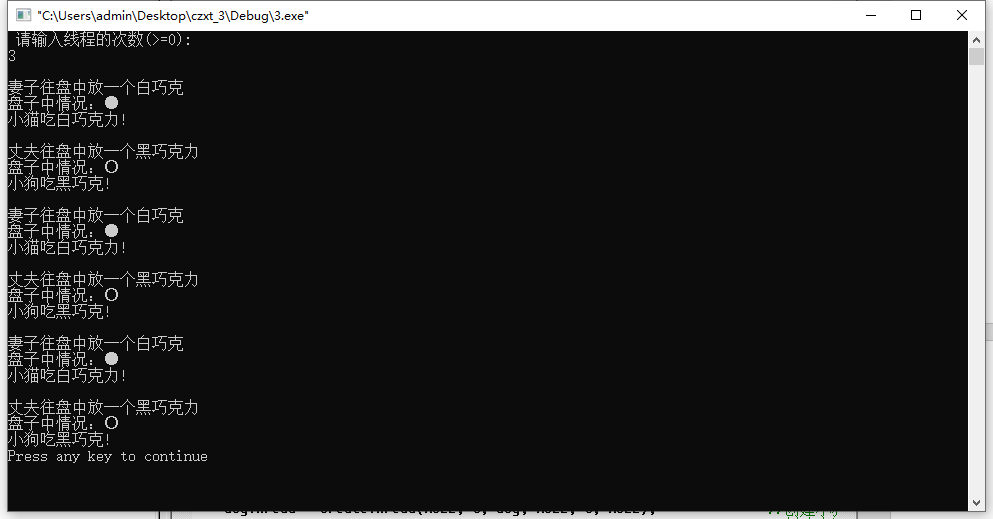


图 5-2-08

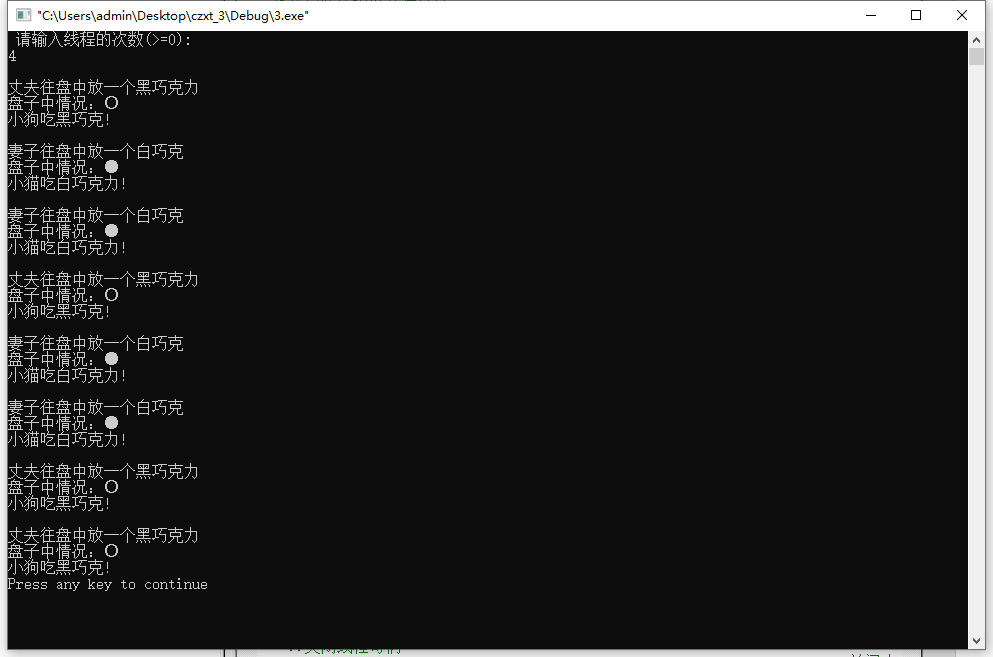


图 5-2-09

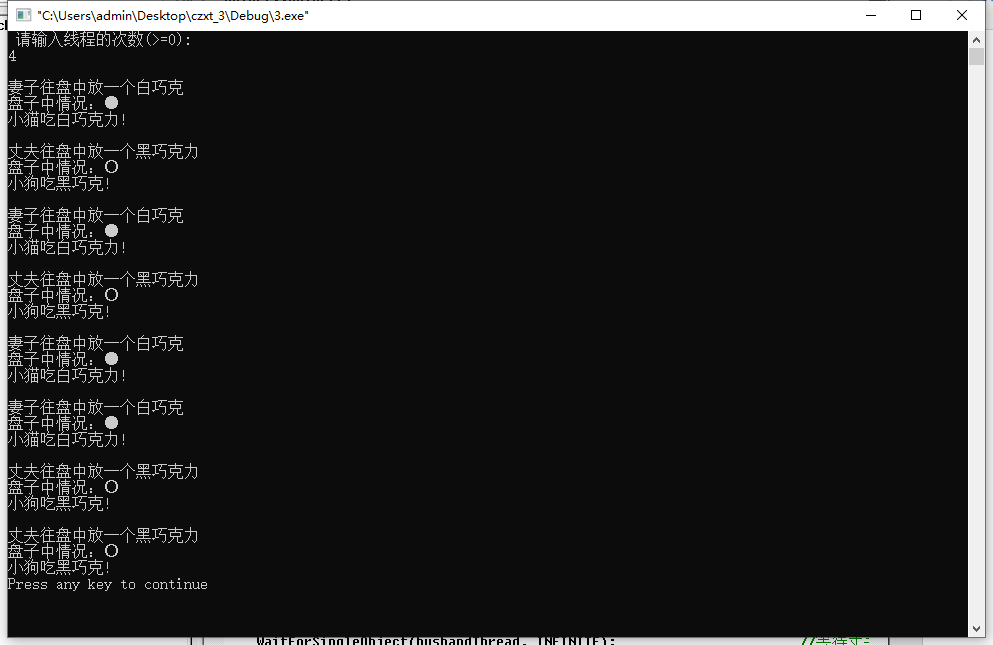


图 5-2-10

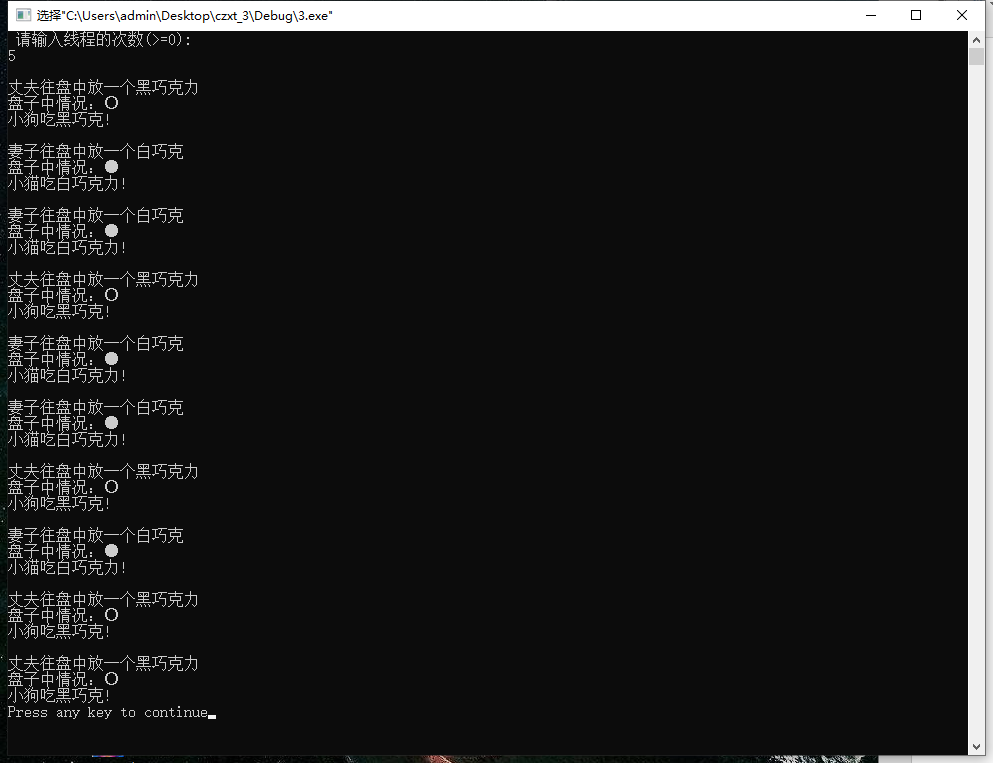


图 5-2-11

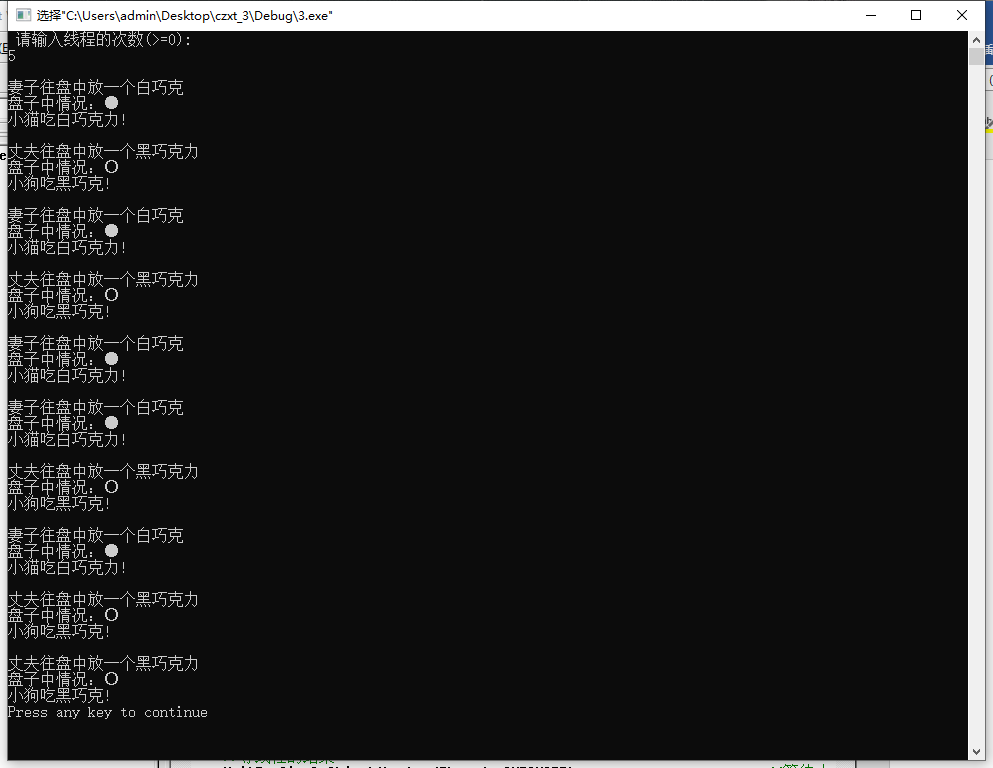


图 5-2-12

# 六、总结

第三个课题还是有点难度的，因为关系到线程的操作，这块知识一直是我们比较薄弱的一块，对多线程的理解更多是停留在理论知识的层面上。

刚拿到课题，我们第一反应是整个题目类似于附带限制条件的生产者消费者，父亲与母亲是两个生产者同时生产产品，但是由于只有一个店面，所以两个生产者只有一个可以进行生产并且出售产品，而小猫、小狗也类似于两个消费者但是目标商品不一样，他们会争夺对应的资源。这就产生了一对特殊的生产者消费者。要实现这个程序我们就要通过上课学习过的一个PV原语操作，通俗的解释就是一个线程的锁定等待和线程的解锁，在父亲摆放巧克力的时候，他会先锁住盘子不允许母亲去放东西，同理母亲在摆放巧克力的时候，父亲同样无法操作，小猫小狗吃巧克力的过程也是同理。

总的来说实验还是比较成功的，整个程序大体上是我们独立完成的，在代码的编写过程中，我们也学习了很多有关多线程的知识。特别感谢老师给出的这个课题，让我们对多线程有了深入的学习与了解，也巩固了课堂所学，对于之后的学习和工作有很大裨益，在平时要多去动手实践，去提高自己分析问题、发现问题、解决问题的能力。