一、 实验目的

使用 C++模拟银行家算法，避免死锁。

二、 数据结构及说明

定义三个数组：一个一维数组，两个二维数组，分别存放可利用资源、已分配资源以及还需要资源数目

```
int available[num] = {3,3,2};
int allocation[process][num] = {{0,1,0},{2,0,0},{3,0,2},{2,1,1},{0,0,2} };
int need[process][num] = {{7,4,3},{1,2,2},{6,0,0},{0,1,1},{4,3,1} };
```

再定义三个同样的数组，用来备份

```
int available_1[num];
int allocation_1[process][num];
int need_1[process][num];
```

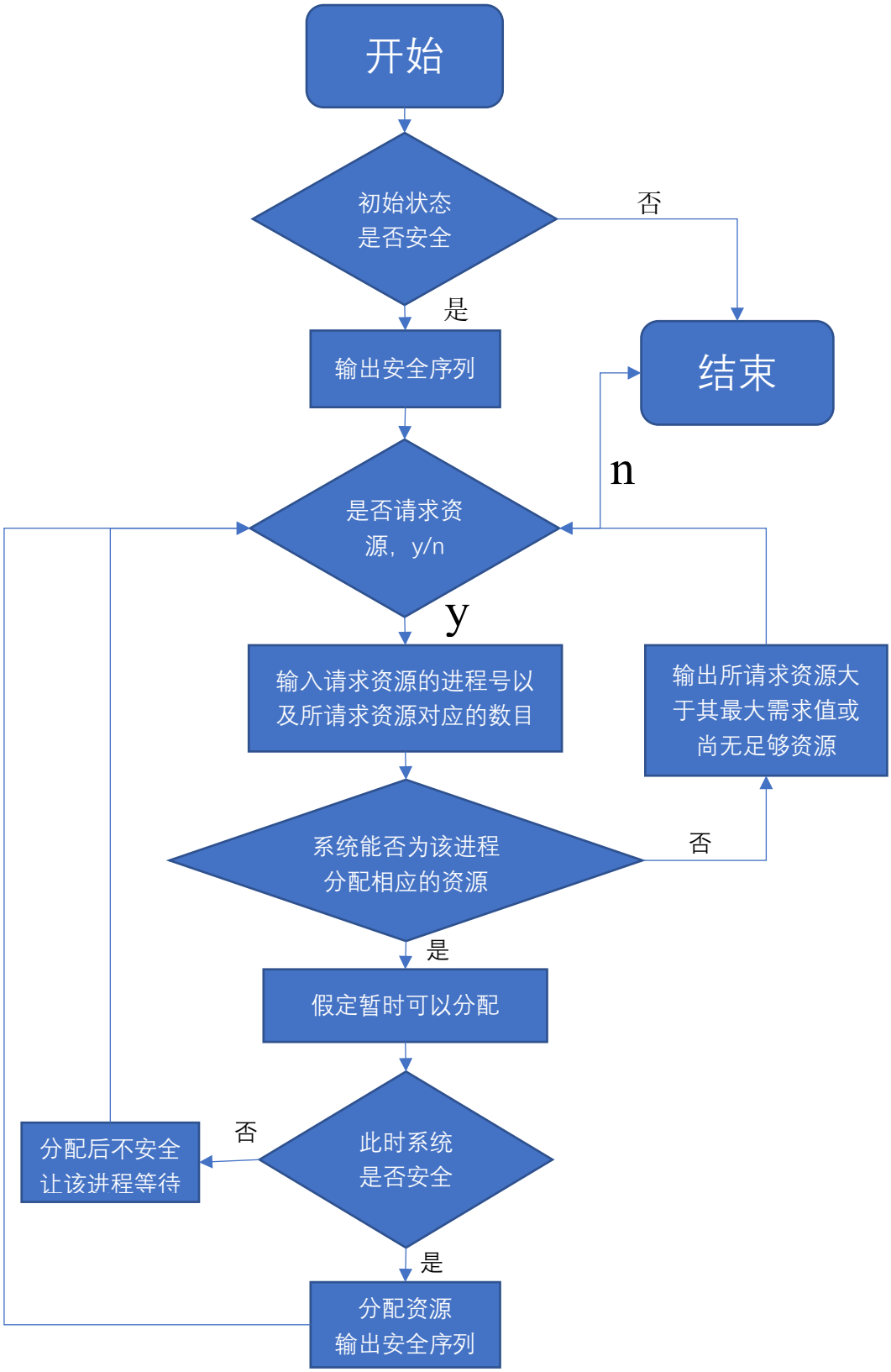
process、num 分别表示进程数目，资源数目

定义一个全局变量 que_pos 队列，用于按序存放安全序列

安全性检测算法中：

定义一个 work[num]数组，存放系统可利用的资源数目，布尔类型数组 finish[process]表示系统是否有资源分配给进程，使之运行完成

三、 流程图


开始
初始状态
是否安全
否
是
结束
输出安全序列
n
是否请求资
源，y/n
y
输入请求资源的进程号以
及所请求资源对应的数目
输出所请求资源大
于其最大需求值或
尚无足够资源
系统能否为该进程
分配相应的资源
否
是
假定暂时可以分配
分配后不安全
让该进程等待
否
此时系统
是否安全
是
分配资源
输出安全序列

四、 实验结果

初始数据：

```
int available[num] = { 3,3,2 };//可利用资源向量
int allocation[process][num] = { {0,1,0},{2,0,0},{3,0,2},{2,1,1},{0,0,2} };//分配矩阵
int need[process][num] = { {7,4,3},{1,2,2},{6,0,0},{0,1,1},{4,3,1} };//需求矩阵
```

结果：

初始状态安全
安全序列为：P1 P3 P4 P0 P2

是否需要请求资源，y/n:y

请输入所请求资源的进程号:1
请依次输入所请求资源对应的数量：1 0 2

P1 进程请求资源:1 0 2
P1 进程已分配资源:2 0 0
P1 进程需要资源:1 2 2
尚可利用资源:3 3 2

可暂时假定可以为 P1 分配资源

分配后状态安全,可以为 P1 分配资源!
安全序列为：P1 P3 P4 P0 P2

是否需要请求资源，y/n:y

请输入所请求资源的进程号:4
请依次输入所请求资源对应的数量：3 3 2

P4 进程请求资源:3 3 2
P4 进程已分配资源:0 0 2
P4 进程需要资源:4 3 1
尚可利用资源:2 3 0

所请求资源大于其最大需求值!

故暂时不能给 P4 分配资源!

是否需要请求资源，y/n:y

请输入所请求资源的进程号:4
请依次输入所请求资源对应的数量：3 3 0

P4 进程请求资源:3 3 0
P4 进程已分配资源:0 0 2
P4 进程需要资源:4 3 1

尚可利用资源:2 3 0

尚无足够资源!

故暂时不能给 P4 分配资源!

是否需要请求资源，y/n:y

请输入所请求资源的进程号:0
请依次输入所请求资源对应的数量：0 2 0

P0 进程请求资源:0 2 0
P0 进程已分配资源:0 1 0
P0 进程需要资源:7 4 3
尚可利用资源:2 3 0

可暂时假定可以为 P0 分配资源

分配后状态不安全, 让 P0 等待!

是否需要请求资源，y/n:y

请输入所请求资源的进程号:0
请依次输入所请求资源对应的数量：0 1 0

P0 进程请求资源:0 1 0
P0 进程已分配资源:0 1 0
P0 进程需要资源:7 4 3
尚可利用资源:2 3 0

可暂时假定可以为 P0 分配资源

分配后状态安全,可以为 P0 分配资源!
安全序列为：P1 P3 P4 P0 P2

五、 结果分析

初始状态安全，可以请求资源

（1） P1 发出请求向量 request（1,0,2），按照银行家算法进行检测
request（1,0,2）≤need（1,2,2）
request（1,0,2）≤available（3,3,2）
则假定可为 P1 分配资源，修改相关向量，然后进行安全性检测，发现存在安全序列{ P1，P3，P4，P0，P2}，所以可以为 P1 分配资源

（2） P4 发出请求向量 request（3,3,2），按照银行家算法进行检测

request（3,3,2）＞need（4,3,1）
即 P4 所请求资源大于其最大需求值，让 P4 等待

（3） P4 发出请求向量 request（3,3,0），按照银行家算法进行检测
request（3,3,0）≤need（4,3,1）
request（3,3,0）＞available（2,3,0）
即尚无足够资源，让 P4 等待

（4） P0 发出请求向量 request（0,2,0），按照银行家算法进行检测
request（0,2,0）≤need（7,4,3）
request（0,2,0）≤available（2,3,0）
则假定可为 P0 分配资源，修改相关向量，然后进行安全性检测，此时可利用资源向量为 available（2,1,0），已经无法满足任何进程的需要，即找不到一个安全序列，系统进入不安全状态，所以此时不应该为 P0 分配其请求资源，让 P0 等待

（5） P0 发出请求向量 request（0,1,0），按照银行家算法进行检测
request（0,1,0）≤need（7,4,3）
request（0,1,0）≤available（2,3,0）
则假定可为 P0 分配资源，修改相关向量，然后进行安全性检测，发现存在安全序列{ P1，P3，P4，P0，P2}，所以可以为 P0 分配资源