银行家算法代码解析

（C++实现）

【周吉瑞小组】

一、 实验目的

通过编写一个模拟动态资源分配的银行家算法程序，进一步深入理解死锁、产生死锁的必要条件、安全状态等重要概念，并掌握避免死锁的具体实施方法。

二、 实验内容

1）初始化让系统拥有一定的资源。

2）用户用键盘输入的方式请求资源。

3）如果预分配后，系统处于安全状态，则修改系统的资源分配情况。

4）如果预分配后，系统处于不安全状态，则提示不能满足请求。

三、 算法解释

（1） 基本过程

先对用户提出的请求进行合法性检查，即检查请求的资源是否不大于需求的资源，是否不大于可利用的资源。若请求合法，则进行试分配。最后对试分配后的状态调用安全性检查算法进行安全性检查。若安全，则分配，否则不分配，恢复原来状态，拒绝申请。

（2） 数据结构

- 可利用资源向量 *int Available[N]*　　N 为资源种类
- 最大需求矩阵 *int Max[M][N]*　　M 为进程的数量

- 分配矩阵 *int Allocation[M][N]*
- 还需资源 *int Need[i][j] = Max[i][j] - Allocation[i][j]*
- 申请资源数量 *int Request[N]*
- 工作向量 *int Work[N]* *int Finish[M]*

（3） bank()银行家算法

Requesti：进程 Pi 的请求向量。0 <= j <= N-1

1) 若 *Requesti[j]* ≤ *Need[i,j]*，转向(2)，否则出错。

2) 若 *Requesti[j]* ≤ *Available[j]*，转向(3)，否则等待。

3) 系统试探着把资源分配给进程 Pi，修改下面内容：

Available[j] = Available[j] - Requesti[j];

Allocation[i,j] = Allocation[i,j] + Requesti[j];

Need[i,j] = Need[i,j] - Requesti[j];

4) 试分配后，执行安全性算法，检查此次分配后系统是否处于安全状态。若安全，才正式分配；否则，此次试探性分配作废（逆向恢复），进程 Pi 等待。

（4） safe()安全性算法

1) 初始化：设置两个向量 *Work*(1×N)和 *Finish*(1×M)。

Work：系统可提供给进程继续运行所需各类资源数

（初态赋值：*Work[i] = Available[i]*）

Finish：系统是否有足够资源分配给进程

（初值 false）

2) 从进程集合中找寻满足下面条件进程：

Finish[i] = false;　　*Need[i,j] ≤ Work[j];*

若找到，执行(3)，否则，执行(4)。

3) 进程 Pi 获得资源，可顺利执行，完成释放所分配的资源。

Work[j] = Work[j]+Allocation[i,j];

Finish[i] = true; go to (2);

注意：以上式子还可以写为：

Work[k] = Work[k] - Need[i][k];

Work[k] = Work[k] + Max[i][k];

Finish[i] = true; go to (2);

4) 若所有进程 *Finish[i] = true*，表示系统处于安全状态，否则处于不安全状态。

四、　程序流程

```
/* 案例 */
/*************************************************************************************

-----------------------------------------------
||                                           ||
||                银行家算法                 ||
||                                           ||
-----------------------------------------------

系统目前的可用资源【Available】:
                    a  b  c
                    3  3  2

系统当前的资源分配情况如下：

          【Max】      【Allocation】    【Need】
          a  b  c       a  b  c        a  b  c

 【P0】    7  5  3       0  1  0        7  4  3
 【P1】    3  2  2       2  0  0        1  2  2
```

【P2】	9	0	2	3	0	2	6	0	0
【P3】	2	2	2	2	1	1	0	1	1
【P4】	4	3	3	0	0	2	4	3	1

【系统是安全的!】
存在一个安全序列:【P1】 【P3】 【P0】 【P2】 【P4】

```
--------------------------------------------------
||                                              ||
||                请求分配：输入 1              ||
||                退出分配：输入 0              ||
||                                              ||
--------------------------------------------------
```

请选择：1

请输入请求分配资源的进程号【0-4】：1

请输入进程 P1 要申请的资源个数：
a b c
1 0 2

系统目前的可用资源【Available】:

a	b	c
2	3	0

系统当前的资源分配情况如下：

	【Max】			【Allocation】			【Need】		
	a	b	c	a	b	c	a	b	c
【P0】	7	5	3	0	1	0	7	4	3
【P1】	3	2	2	3	0	2	0	2	0
【P2】	9	0	2	3	0	2	6	0	0
【P3】	2	2	2	2	1	1	0	1	1
【P4】	4	3	3	0	0	2	4	3	1

【系统是安全的!】
存在一个安全序列:【P1】 【P3】 【P0】 【P2】 【P4】

```
--------------------------------------------------
||                                              ||
||                请求分配：输入 1              ||
||                退出分配：输入 0              ||
||                                              ||
```

请选择：1

请输入请求分配资源的进程号【0-4】：4

请输入进程 P4 要申请的资源个数：
a b c
3 3 0

进程 P4 申请的资源大于系统现在可利用的资源
【系统尚无足够资源，不予分配!】

	请求分配：输入 1	
	退出分配：输入 0	

请选择：1

请输入请求分配资源的进程号【0-4】：0

请输入进程 P0 要申请的资源个数：
a b c
0 2 0

系统目前的可用资源【Available】:

a	b	c
2	1	0

系统当前的资源分配情况如下：

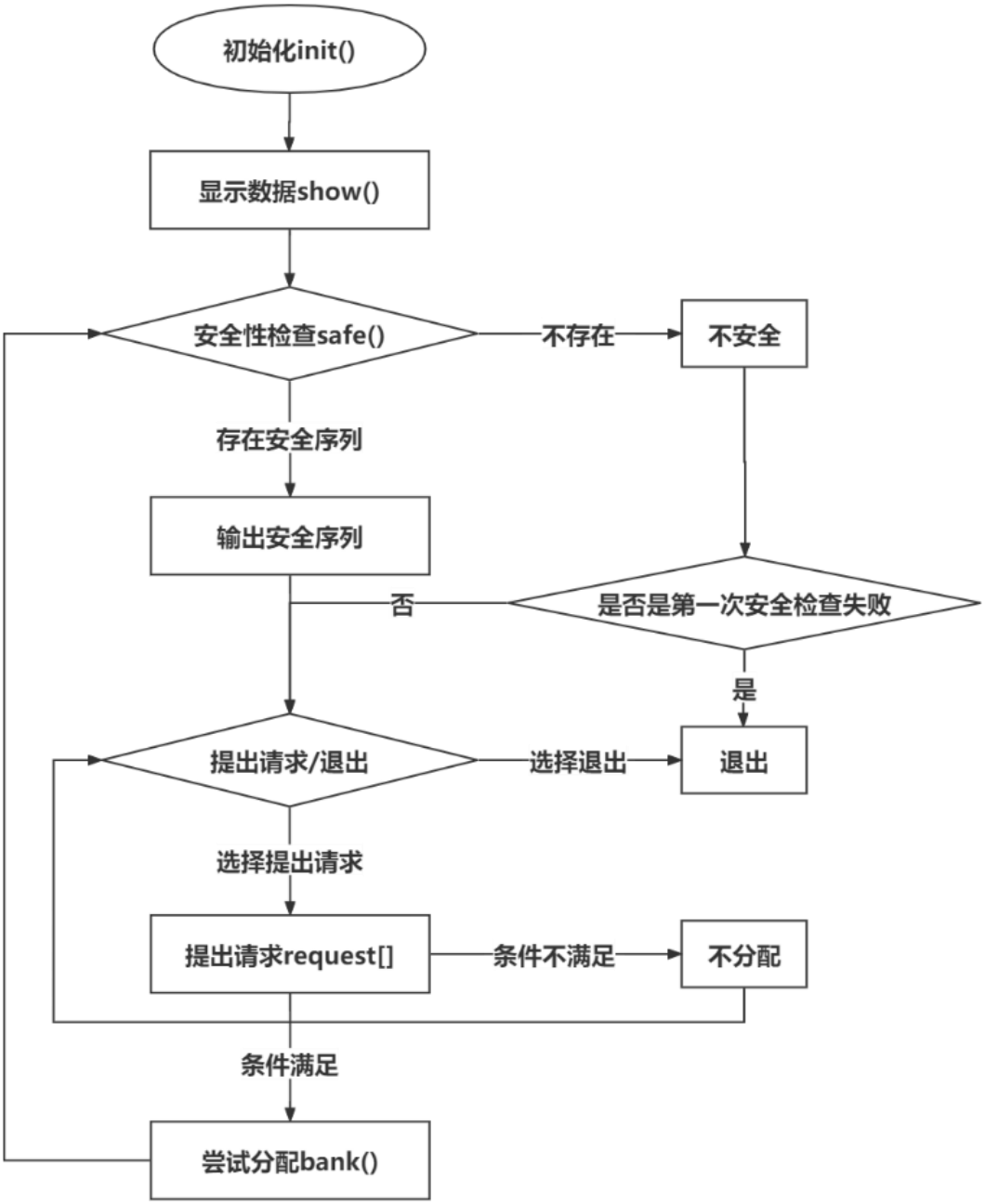
	【Max】			【Allocation】			【Need】		
	a	b	c	a	b	c	a	b	c
【P0】	7	5	3	0	3	0	7	2	3
【P1】	3	2	2	3	0	2	0	2	0
【P2】	9	0	2	3	0	2	6	0	0
【P3】	2	2	2	2	1	1	0	1	1
【P4】	4	3	3	0	0	2	4	3	1

【系统不安全，请重新分配!】

系统目前的可用资源【Available】:

```
                    a  b  c
                    2  3  0

系统当前的资源分配情况如下：

          【Max】      【Allocation】    【Need】
          a  b  c       a  b  c       a  b  c

 【P0】    7  5  3       0  1  0       7  4  3
 【P1】    3  2  2       3  0  2       0  2  0
 【P2】    9  0  2       3  0  2       6  0  0
 【P3】    2  2  2       2  1  1       0  1  1
 【P4】    4  3  3       0  0  2       4  3  1

--------------------------------------------------
||                                              ||
||                请求分配：输入 1              ||
||                退出分配：输入 0              ||
||                                              ||
--------------------------------------------------
请选择：0

--------------------------------
Process exited after 83.06 seconds with return value 0
请按任意键继续. . .

*********************************************************************************/
```

（流程图在下一页）


初始化init()
显示数据show()
安全性检查safe()
不存在
不安全
存在安全序列
输出安全序列
否
是否是第一次安全检查失败
是
提出请求/退出
选择退出
退出
选择提出请求
提出请求request[]
条件不满足
不分配
条件满足
尝试分配bank()