实验 5 银行家算法

目标：参考教材 P139 页银行家算法，采用 C/C++/Java 语言实现银行家算法，并采用算例进行验证。

参数：

- N——系统中进程的个数
- M——系统中资源种类的个数
- Resource[M]——每类资源数量向量
- Claim[N×M]——每个进程对资源最大需求量矩阵
- Allocate[N×M]——当前状态下，每个进程已拥有（或已分配）的资源矩阵
- Need[N×M]——当前状态下，每个进程仍需要的资源数目矩阵
- Available[M]——当前状态下，剩余资源数量向量
- Request(i)[M]——第 i 个进程请求资源的向量

其中 N、M、Resource 和 Claim 在算法运行前是固定的，不会随着资源的分配而发生变化。Allocate、Need、Available 会随着请求向量 Request(i)[M]的接受不断发生变化。

以上参数满足如下约束：

1. 每个进程对每类资源的需求量不超过系统能够提供的最大资源数量：

$$\forall i \in [1..N], j \in [1..M]: Cij \leq Rj$$

2. 已分配的每类资源总量不超过系统能够提供的最大资源数量：

$$\forall j \in [1..M]: \left(\sum_{i=1}^{N} Aij\right) \leq Rj$$

3. 已分配的资源不超过所需求的资源数目：Allocate [i,j]≤Claim[i,j]
4. Need[i,j]=Claim[i,j]-Allocate[i,j]
5. Available[j]= Resource[j]-(Allocate[1,j]+ Allocate[1,j]+…+ Allocate[N,j])

接口说明：

1、 全局常量：N、M、Resource[M]、Claim[N×M]。

2、 全局变量：Allocate[N×M]、Need[N×M]、Available[M]、Request(i)[M]

3、 银行家算法的接口

bool Banker(in Allocate[N×M]， in Request[M], in i)

- Allocate：输入参数
- Request：输入参数
- i: 输入参数，表示第 i 个进程的资源请求
- 返回值：1——接受 Request(i)请求；0——拒绝 Request(i)请求

算例：

1. N=4，

   M=3

   Resource=(9,3,6)

   $\text{claim} = \begin{bmatrix} 3 & 2 & 2 \\ 6 & 1 & 3 \\ 3 & 1 & 4 \\ 4 & 2 & 2 \end{bmatrix}$。

2. 初始时，Allocate[i,j]=0，表示初始时没有进程得到任何资源。假定进程对资源的请求序列为：

   Request(1)[M]=(1,0,0);
   Request(2)[M]=(2,1,0);
   Request(2)[M]=(2,0,1);
   Request(3)[M]=(2,1,1);
   Request(4)[M]=(0,0,2);
   Request(2)[M]=(1,0,1);
   Request(1)[M]=(1,0,1);

   请用 Banker 算法判断每一次资源请求是否接受，如果接受请求，请给出请求接受后的资源分配状态，即 Allocate 矩阵、Need 矩阵和 Available 向量。