# 实验1 进程管理

## 一、实验目的

理解并掌握银行家算法的基本设计思想、关键数据结构和算法流程。

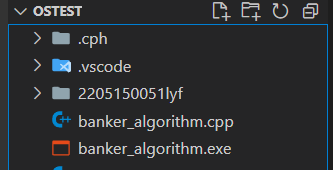
## 二、实验开发环境

系统环境：windows 11

运行环境：vscode

测试环境：vscode

## 三、源程序文件和源码清单



## 四、实验步骤

1.**初始化系统状态：**设置初始的Available、Max、Allocation和Need矩阵。

2.**请求资源：**当进程请求资源时，检查请求是否合法（即请求量不超过进程的最大需求和系统的可用资源）。

3.**试探性分配**：临时分配资源给进程，并更新系统状态。

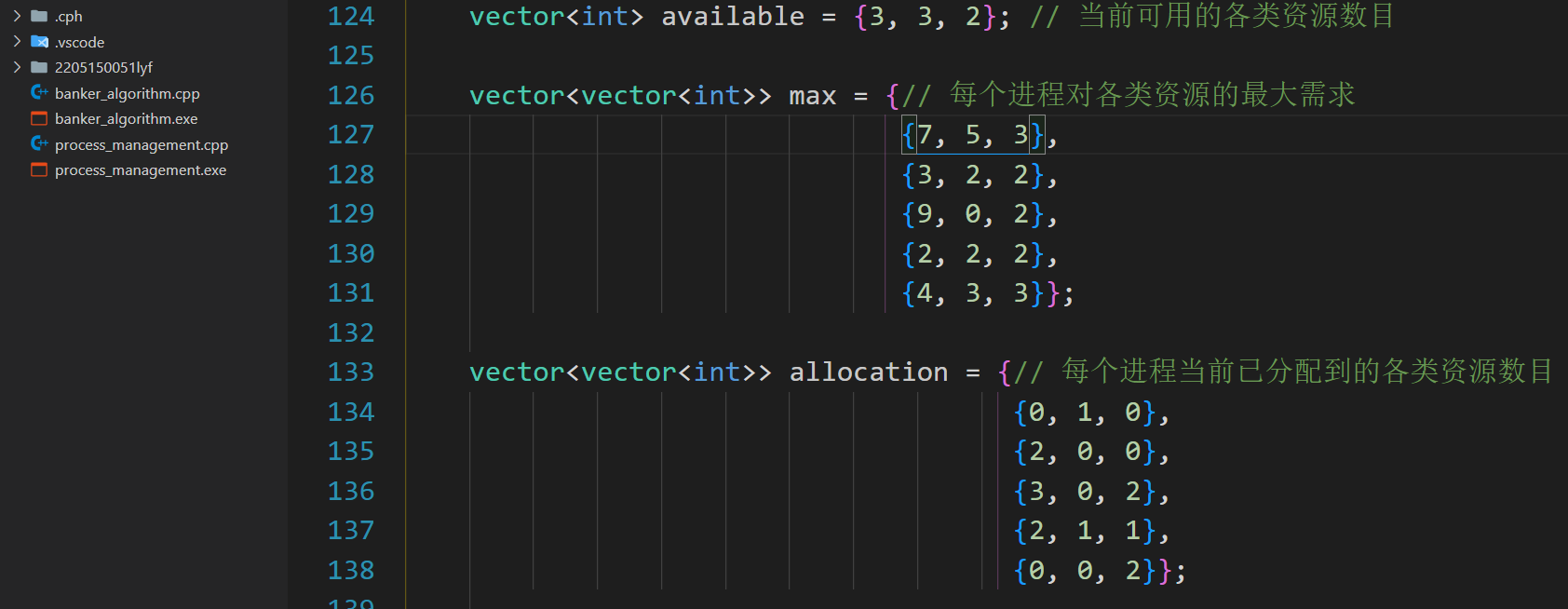
4.**安全性检查：**调用安全性算法检查系统是否处于安全状态。如果安全，正式分配资源；否则，恢复原状态，拒绝请求。

## 五、算法流程

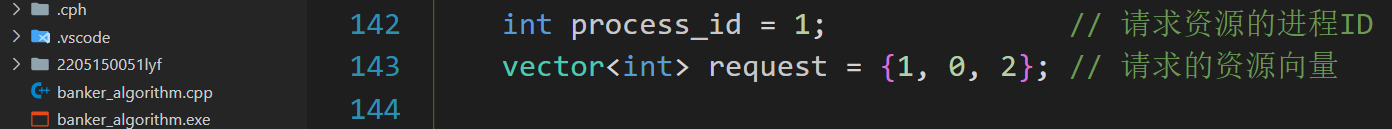
|  |
| --- |
| 银行家算法伪代码 |
| 1. 类 BankersAlgorithm 2. 私有属性： 3. 整数 n // 进程数量 4. 整数 m // 资源类型数量 5. 向量 available // 当前可用的各类资源数目 6. 矩阵 max // 每个进程对各类资源的最大需求 |
| 1. 矩阵 allocation // 每个进程当前已分配到的各类资源数目 2. 矩阵 need // 每个进程尚需的各类资源数目 3. 构造函数 BankersAlgorithm(num\_processes, num\_resources, avail, max\_matrix, alloc\_matrix) 4. 初始化 n, m, available, max, allocation 5. 计算 need 矩阵 7. 方法 calculateNeed() |
| 1. 对于每个进程 i 2. 对于每个资源 j 3. need[i][j] = max[i][j] - allocation[i][j] 4. 方法 isSafeState() 返回 布尔值 5. work = available 6. finish = [false, ..., false] // 初始化为未完成 7. safeSequence = [] 8. 当 safeSequence.size() < n 9. found = false 10. 对于每个进程 i 11. 如果未完成 finish[i] 12. canProceed = true 13. 对于每个资源 j 14. 如果 need[i][j] > work[j] 15. canProceed = false 16. 退出内循环 17. 如果 canProceed 为真 18. 更新 work 19. safeSequence.push\_back(i) 20. finish[i] = true 21. found = true 22. 如果未找到合适进程 23. 返回 false 24. 返回 true 25. 方法 requestResources(process\_id, request) 返回 布尔值 26. 对于每个资源 i 27. 如果 request[i] > need[process\_id][i] 或 request[i] > available[i] 28. 返回 false // 请求不合法 29. 更新资源分配 30. 对于每个资源 i 31. available[i] -= request[i] 32. allocation[process\_id][i] += request[i] 33. need[process\_id][i] -= request[i] 34. 如果 isSafeState() 为真 35. 返回 true // 请求被批准 36. 否则 37. 恢复状态 38. 对于每个资源 i 39. available[i] += request[i] 40. allocation[process\_id][i] -= request[i] 41. need[process\_id][i] += request[i] 42. 返回 false // 请求被拒绝 43. 主程序 44. 初始化 n, m, available, max, allocation 45. 创建 BankersAlgorithm 对象 ba 46. 设置 process\_id 和 request 47. 如果 ba.requestResources(process\_id, request) 48. 输出 "资源请求被批准。系统处于安全状态。" 49. 否则 50. 输出 "资源请求被拒绝。系统将进入不安全状态。" |

## 六、实验结果

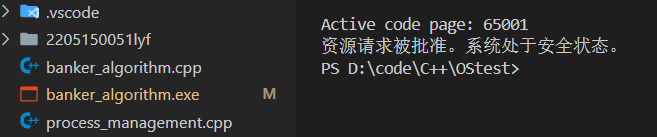
输入数据



测试进程



结果



## 七、总结体会

安全性检测这部分是银行家算法的重中之重了，具体的思路是：如果找到一个进程，使得当前可利用资源（Available）能够满足它的需求（Need），就先对其分配资源，然后收回分配给它之前的资源（Allocation），接着更新Available（Available+=Allocation），然后将其加入安全性序列。继续找一个进程，重复上面操作。需要注意的是，必须要循环多次，因为前面没有符合条件的进程可能因为后面的资源更新又符合条件了。最后，临界条件是如果循环一遍后，还是找不到符合条件的进程，那么说明系统不安全，退出循环，反之如果所有的进程都加入安全性序列，就说明此系统是安全的，退出循环。