BankerFinally

```
/*可读取文件的银行家算法 */
#include <iostream>
#include <fstream>
#include <vector>

using namespace std;

const int MAX_PROCESS = 10;  // 最大进程数
const int MAX_RESOURCE = 10; // 最大资源数

int n, m;                                    // 进程数和资源数
int available[MAX_RESOURCE];                 // 可用资源数
int max_need[MAX_PROCESS][MAX_RESOURCE];     // 最大需求量
int allocation[MAX_PROCESS][MAX_RESOURCE]; // 已分配资源量
int need[MAX_PROCESS][MAX_RESOURCE];         // 需求资源量
bool finish[MAX_PROCESS];                    // 进程是否完成

vector<int> safe_seq; // 安全序列

// 从文件中读取数据并初始化银行家算法相关数据结构
void init(const char *filename)
{
    ifstream infile(filename);

    if (!infile)
    {
        cout << "无法打开文件 " << filename << "，请检查文件路径是否正确！" << endl;
        exit(1);
```

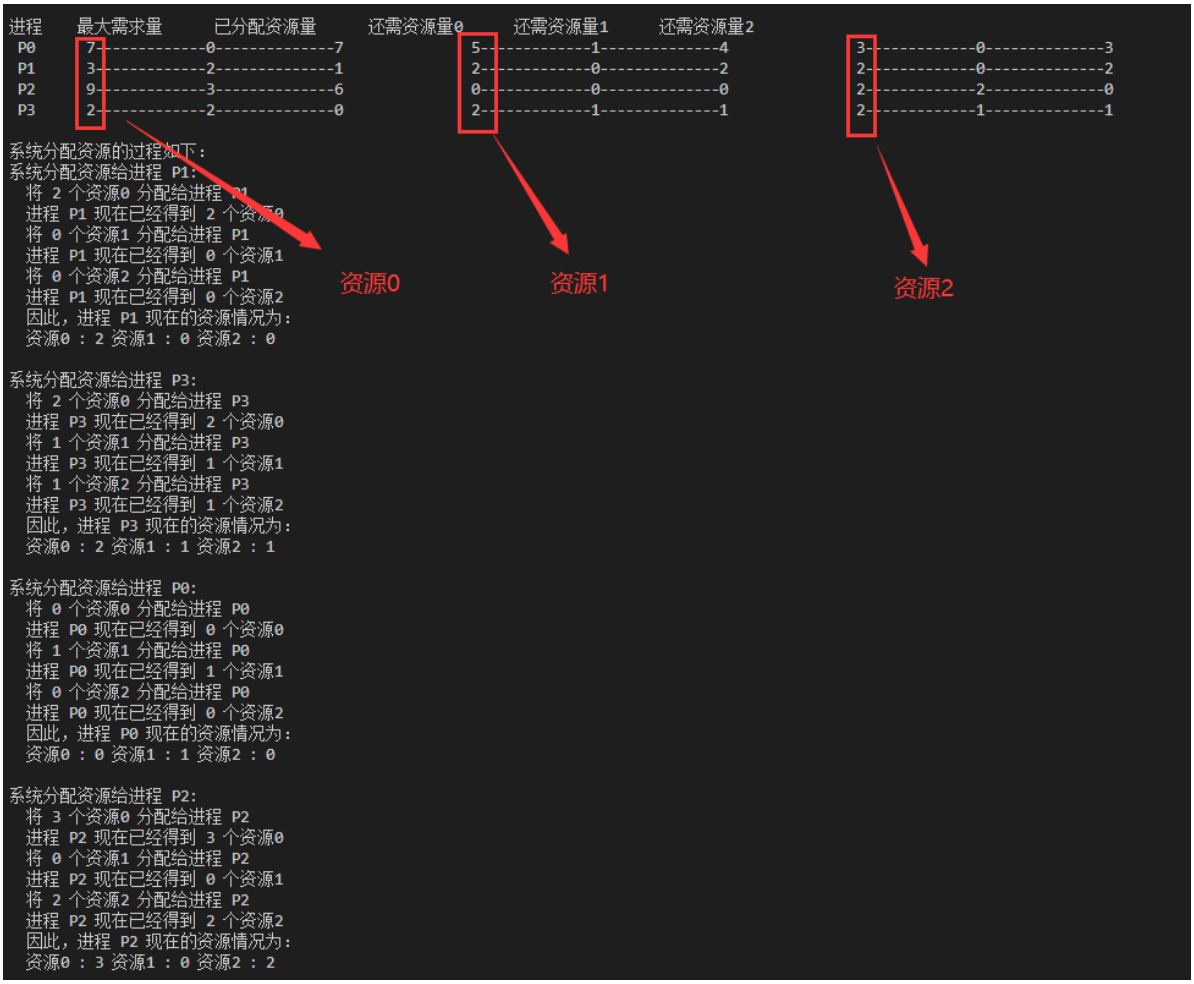
```
    }

    cout << "正在读取文件 " << filename << "..." << endl;

    infile >> n >> m;
    cout << "读取到 " << n << " 个进程和 " << m << " 种资源" << endl;

    for (int i = 0; i < m; i++)
    {
        infile >> available[i];
        cout << "资源" << i << " 的可用数量为 " << available[i] << endl;
    }

    for (int i = 0; i < n; i++)
    {
        cout << "读取进程" << i << "的资源需求量: " << endl;
        for (int j = 0; j < m; j++)
        {
            infile >> max_need[i][j];
            cout << "进程" << i << " 需要 " << max_need[i][j] << " 个资源" << j << endl;
        }
    }

    for (int i = 0; i < n; i++)
    {
        cout << "读取进程" << i << "的已分配资源量: " << endl;
        for (int j = 0; j < m; j++)
        {
            infile >> allocation[i][j];
            need[i][j] = max_need[i][j] - allocation[i][j];
            if (need[i][j] < 0)
            {
```

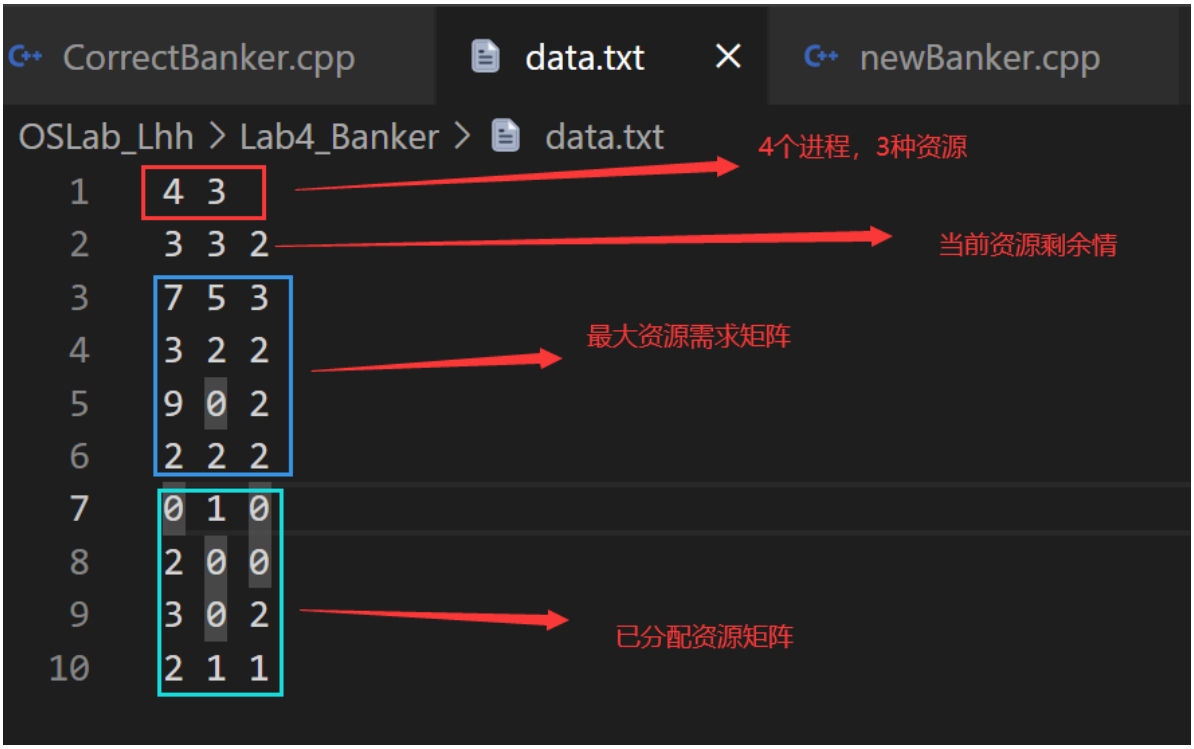
```
                cout << "错误：已分配的资源量大于最大需求量！" << endl;
                exit(1);
            }
            cout << "进程" << i << " 已分配 " << allocation[i][j] << " 个资源" << j << "，还需要 " << need[i][j] << " 个" << endl;
        }
        finish[i] = false;
    }

    infile.close();

    cout << "文件读取成功！" << endl;
}

// 判断是否满足需求
bool check_need(int process_id)
{
    for (int i = 0; i < m; i++)
    {
        if (need[process_id][i] > available[i])
        {
            return false;
        }
    }
    return true;
}

// 模拟分配资源
void simulate()
{
    int count = 0; // 已完成进程数
    while (count < n)
    {
        bool found = false;
        for (int i = 0; i < n; i++)
        {
            if (!finish[i] && check_need(i))
            { // 满足需求
```

```
                for (int j = 0; j < m; j++)
                {
                    available[j] += allocation[i][j]; // 释放资源
                }
                finish[i] = true;
                count++;
                safe_seq.push_back(i); // 将进程加入安全序列
                found = true;
            }
        }
        if (!found)
        { // 没有进程满足需求，说明不安全
            cout << "错误：系统不安全！无法找到安全序列" << endl;
            exit(1);
        }
    }
}

void print_result()
{
    cout << "找到安全序列，进程可以成功执行！" << endl;
    cout << "安全序列为：";
    for (int i = 0; i < n; i++)
    {
        cout << "P" << safe_seq[i];
        if (i != n - 1)
        {
            cout << " -> ";
        }
    }
    cout << endl;

    cout << "进程    最大需求量      已分配资源量      ";
    for (int i = 0; i < m; i++)
    {
        cout << "还需资源量" << i << "      ";
    }
```

```
135     cout << endl;
136 
137     for (int i = 0; i < n; i++)
138     {
139         cout << " P" << i << "      ";
140         for (int j = 0; j < m; j++)
141         {
142             cout << max_need[i][j] << "-------------" << allocation[i][j] << "--------------" << need[i][j] << "                ";
143         }
144         cout << endl;
145     }
146     cout << endl;
147 
148     cout << "系统分配资源的过程如下: " << endl;
149 
150     for (int i = 0; i < n; i++)
151     {
152         cout << "系统分配资源给进程 P" << safe_seq[i] << ":" << endl;
153         for (int j = 0; j < m; j++)
154         {
155             cout << "  将 " << allocation[safe_seq[i]][j] << " 个资源" << j << " 分配给进程 P" << safe_seq[i] << endl;
156             cout << "  进程 P" << safe_seq[i] << " 现在已经得到 " << allocation[safe_seq[i]][j] << " 个资源" << j << endl;
157         }
158         cout << "  因此，进程 P" << safe_seq[i] << " 现在的资源情况为: " << endl;
159         cout << "  ";
160         for (int j = 0; j < m; j++)
161         {
162             cout << "资源" << j << " : " << allocation[safe_seq[i]][j] << " ";
163         }
164         cout << endl
165              << endl;
```

```
    }
}

int main()
{
    const char *filename = "data.txt";
    init(filename);
    simulate();
    print_result();
    return 0;
}
```




CorrectBanker.cpp
data.txt
newBanker.cpp
OSLab_Lhh > Lab4_Banker > data.txt
4 3
3 3 2
7 5 3
3 2 2
9 0 2
2 2 2
0 1 0
2 0 0
3 0 2
2 1 1
4个进程，3种资源
当前资源剩余情
最大资源需求矩阵
已分配资源矩阵