**数据结构课程设计报告**

**课程设计题目：迷宫问题**

**一、问题描述**

迷宫求解是实验心理学中的一个经典问题，心理学家把一只老鼠从一个无顶盖的大盒子的入口处赶进迷宫，迷宫中设置很多隔壁，对前进方向形成了多处障碍，假设前进的方向有四个，分别是上、下、左、右，心理学家在迷宫的唯一出口处放置了一块奶酪，吸引老鼠在迷宫中寻找通路以到达出口。例如，图1所示为一个迷宫示意图。

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 入口 |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  | 出口 |

图1 迷宫示意图

基本要求：⑴ 设计数据结构存储迷宫；

⑵ 设计存储结构保存从入口到出口的通路；

⑶ 设计算法完成迷宫问题的求解；

⑷ 分析算法的时间复杂度。

**二、数据结构的设计**

通过分析题目得到信息应当使用二维数组这种数据结构

//创建迷宫地图

int map[8][10] = {0};

并将找到的“通路”标注在此地图中，因此也采用二维数组存储从入口到出口的“通路”

**三、算法设计及性能分析**

主函数的调用流程：

1. 首先创建一个二维数组用作地图
2. 初始化当前地图，调用initMap函数
3. 查看当前地图，调用watchMap函数
4. 寻找迷宫的通路，调用findWay函数

并展示搜索的过程（思考题1）

1. 查看寻找完成通路后的地图，调用watchMap函数

函数：共有4个函数

1. void initMap(int map[][10])

传入保存迷宫地图的二维数组，完成地图的初始化

1. void watchMap(int map[][10])

传入保存迷宫地图的二维数组，输出当前地图情况

1. int move(int i, int j, int m)

传入当前位置坐标以及移动的方向，完成位置的移动

1. bool findWay(int map[][10], int i, int j, int fi, int fj)

输入：保存迷宫地图的二维数组，起始位置坐标和终点坐标

输出：true表示找到了通路 false表示未找到通路

算法性能分析：O（n\*m）

伪代码：

/\* 规定：

0 无障碍且未走过

1 为墙

2 为通路

3 为无通路

（i,j）是起点

（fi,fj）是终点 \*/

1. 判断终点位置是否为2

如果找到通路，就返回true, 否则返回false

1. 否则如果map[i][j] == 0 则表示当前坐标没有走过

假定该点是可以走通

2.1如果按照策略 上->右->下->左 “走”

（思考题2）则在后面对应添加左上->左下->右下->右上 即可

能走通返回true 否则false

* 1. 均无法走通则说明该点是走不通，是死路

返回false

1. 否则如果map[i][j] != 0 , 可能是 1、 2、 3

返回false

**四、测试数据**

输入数据：

按照所给图片创建地图如下：

1 1 1 1 1 1 1 1 1 1

1 0 0 1 1 0 1 1 1 1

1 1 0 0 0 1 1 1 1 1

1 0 1 0 0 0 0 0 1 1

1 0 1 1 1 0 1 1 1 1

1 1 0 0 1 0 0 0 0 1

1 0 1 1 0 0 1 1 0 1

1 1 1 1 1 1 1 1 1 1

输出数据：

以下为试探通路的过程：

当前位于坐标(1,1)处

当前位于坐标(1,2)处

当前位于坐标(2,2)处

当前位于坐标(2,3)处

当前位于坐标(2,4)处

当前位于坐标(3,4)处

当前位于坐标(3,5)处

当前位于坐标(3,6)处

当前位于坐标(3,7)处

(2,8)是死路，四面八方都走不通！唯有原路返回了！！

当前位于坐标(4,5)处

当前位于坐标(5,5)处

当前位于坐标(5,6)处

当前位于坐标(5,7)处

当前位于坐标(5,8)处

当前位于坐标(6,8)处

通路找到啦！

以下为搜寻后的迷宫地图：（其中‘2’即为所寻找到的通路）

1 1 1 1 1 1 1 1 1 1

1 2 2 1 1 0 1 1 1 1

1 1 2 2 2 1 1 1 1 1

1 0 1 0 2 2 3 3 1 1

1 0 1 1 1 2 1 1 1 1

1 1 0 0 1 2 2 2 2 1

1 0 1 1 0 0 1 1 2 1

1 1 1 1 1 1 1 1 1 1

**五、心得体会**

问题：

在本题中用到递归算法，由于对底层逻辑还没有搞懂，导致陷入死循环

心得体会：

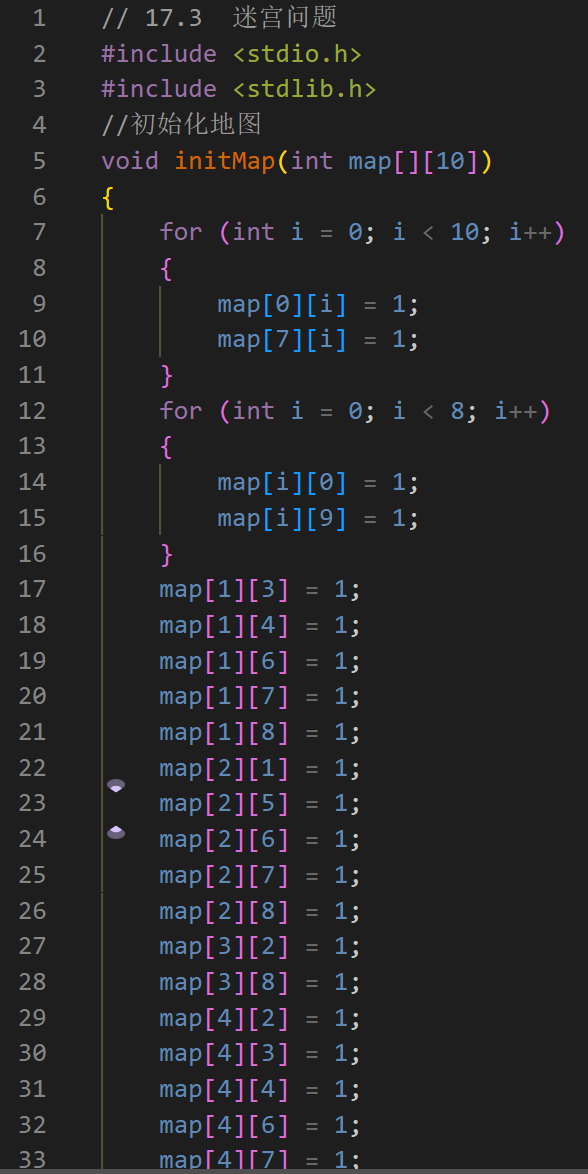
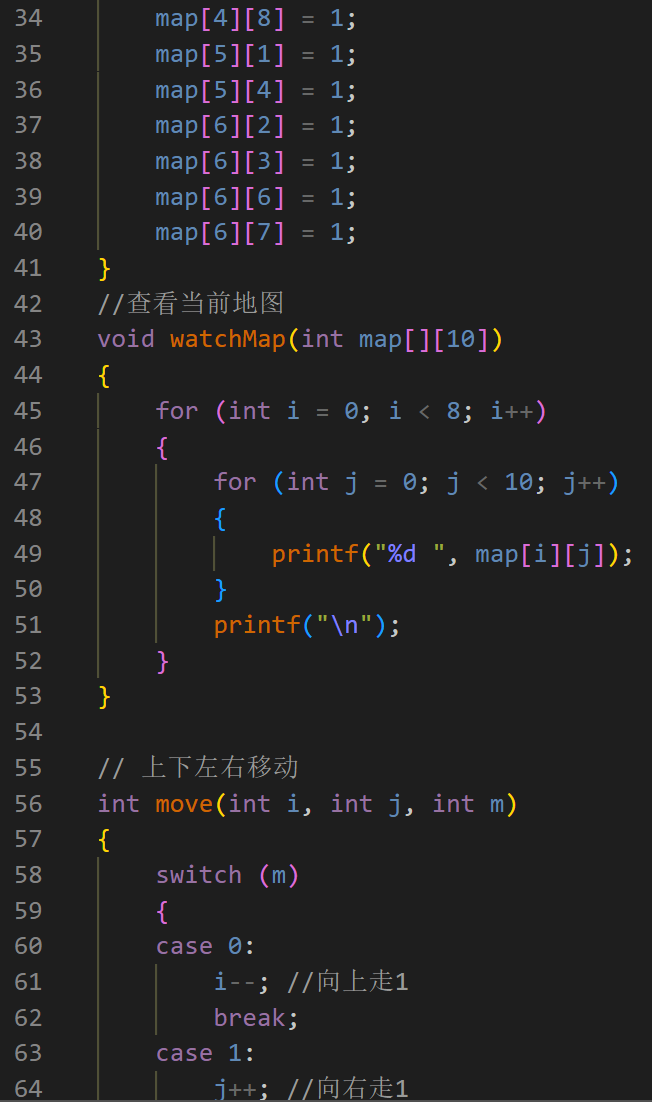
加强了对于递归算法的学习

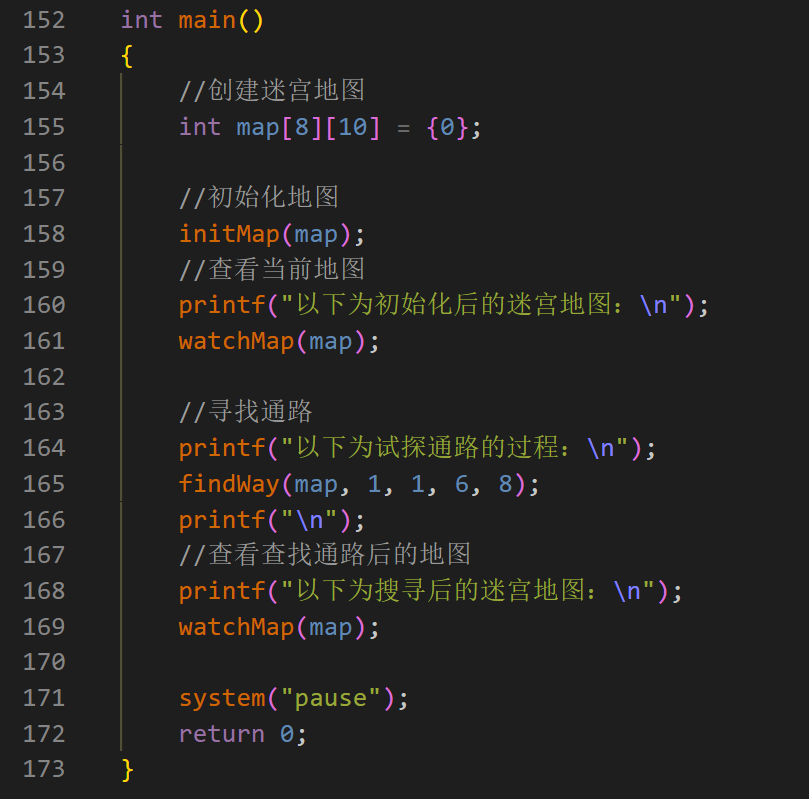
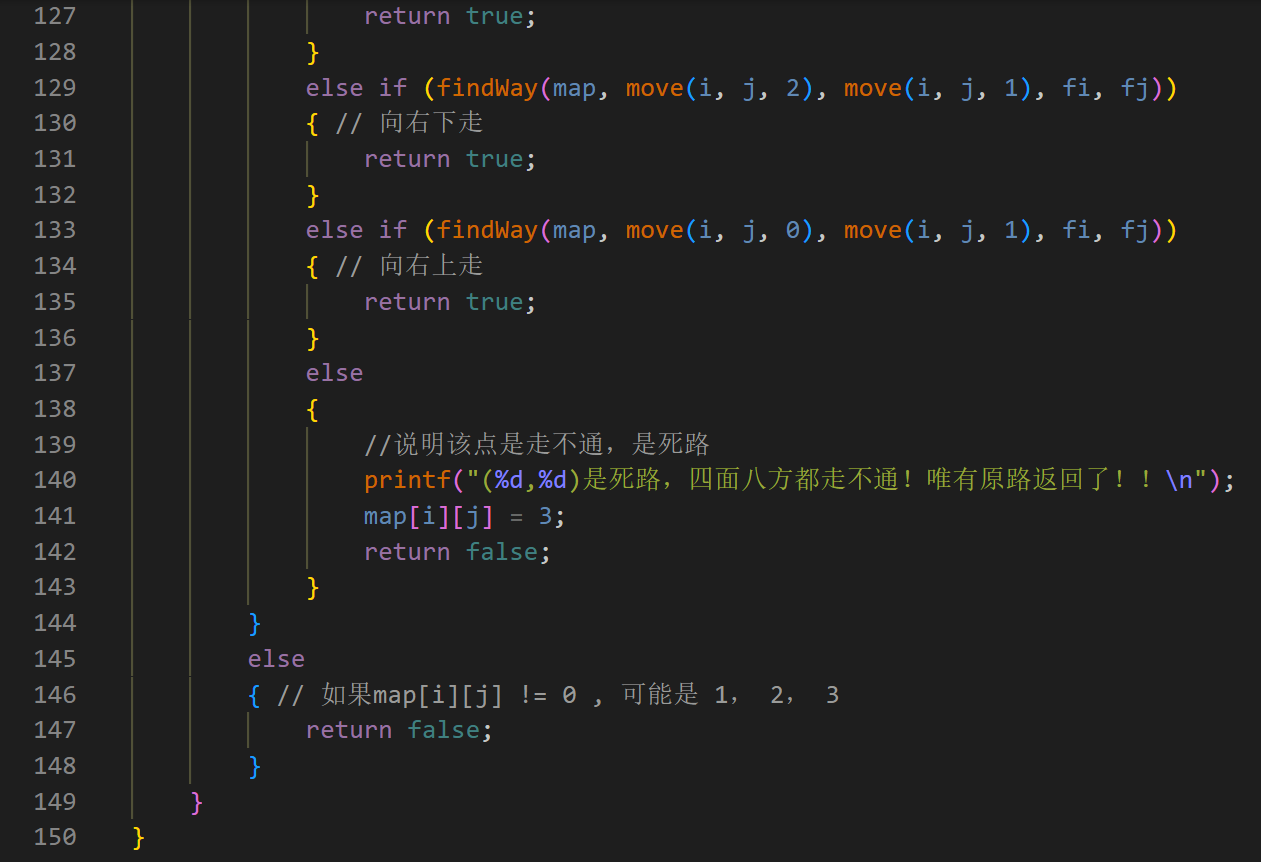
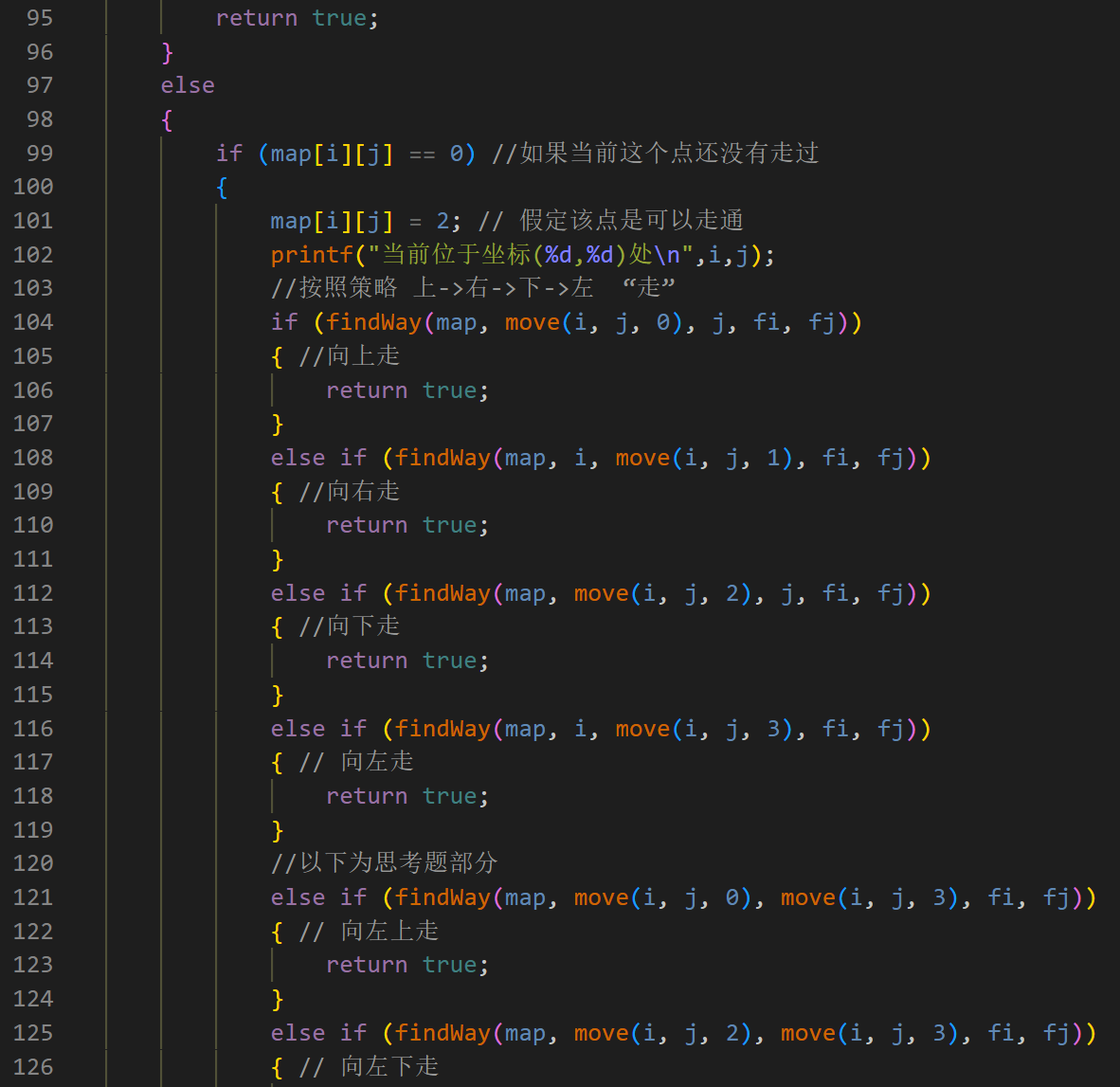
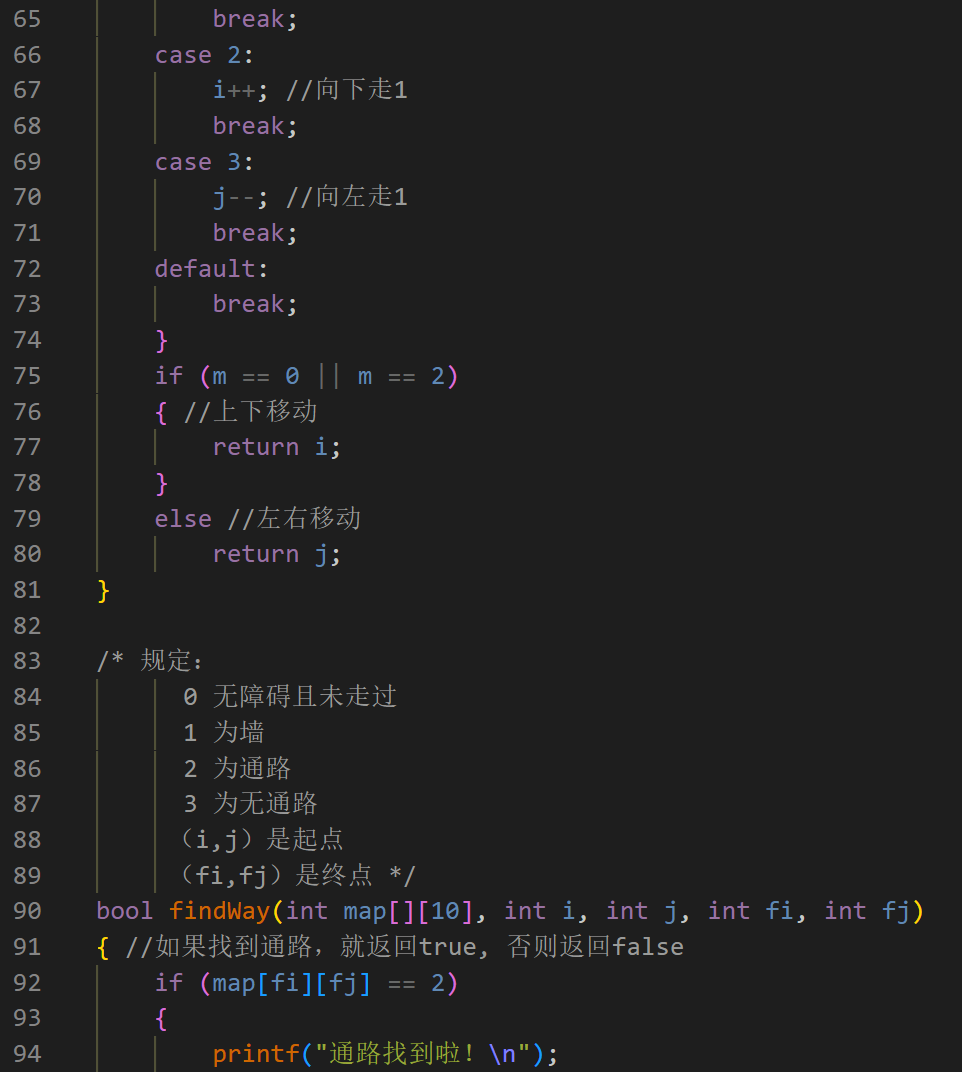
了解并熟悉了回溯法的思想，虽然仍不太熟练，但收获也是很大

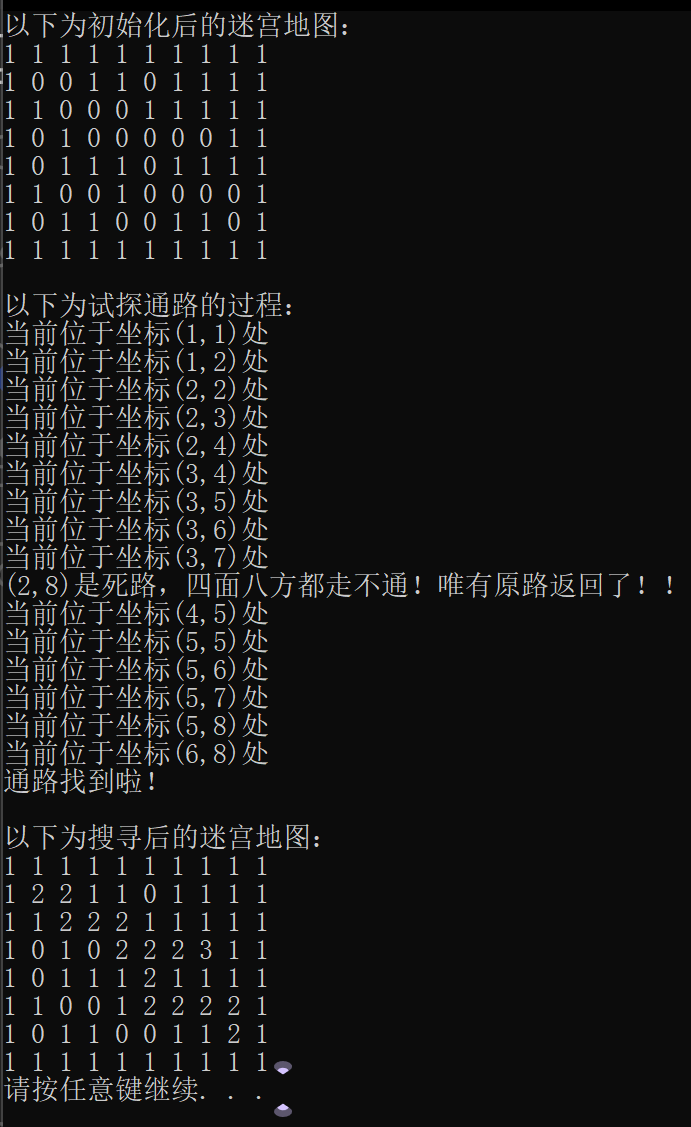
通过本题，对于函数的调用顺序理解也加强了

Debug查找错误的能力得到了加强

**六、程序源码及运行结果**

**** ****



****