1255：迷宫问题

题目描述

定义一个二维数组：

```
int maze[5][5] = {
0,1,0,0,0,
0,1,0,1,0,
0,0,0,0,0,
0,1,1,1,0,
0,0,0,1,0,
};
```

它表示一个迷宫，其中的1表示墙壁，0表示可以走的路，只能横着走或竖着走，不能斜着走，要求编程序找出从左上角到右下角的最短路线。

输入格式

一个5 × 5的二维数组，表示一个迷宫。数据保证有唯一解。

输出格式

左上角到右下角的最短路径，格式如样例所示。

输入样例

```
0 1 0 0 0
0 1 0 1 0
0 0 0 0 0
0 1 1 1 0
0 0 0 1 0
```

输出样例

```
(0, 0)
(1, 0)
(2, 0)
(2, 1)
(2, 2)
(2, 3)
(2, 4)
(3, 4)
(4, 4)
```

解析

最短路径，广搜模板。需要注意的是，在本题中需要记录前一个点的信息，以供后续打印完整路径。

编码

```
#include <bits/stdc++.h>

using namespace std;

int const MaxNum = 26;
int const Limit = 5;

//节点结构体
struct Node {
    int x;    //坐标
    int y;
    int lastIndex; //记录上一个节点的号码，用于后续输出
    int step;  //移动步数

    //当前节点的基本信息
    Node(int nx, int ny, int nIndex, int stepNum) {
        x = nx;
        y = ny;
        lastIndex = nIndex;
        step = stepNum;
    }

    Node() = default;
};

bool Vis[MaxNum][MaxNum];  //判断是否走过
int gapX[4] = {0, 0, -1, 1};  //上下左右偏移的x
int gapY[4] = {-1, 1, 0, 0};  //上下左右偏移的y
Node NodeQueue[MaxNum * MaxNum];
int Maps[MaxNum][MaxNum]; //地图信息;


//检测是否可以通过
bool CanPass(int newX, int newY) {
    //保证没有越界
    if (newX >= 0 && newY >= 0 && newX < Limit && newY < Limit) {
        //没有被访问过
        if (!Vis[newX][newY]) {
            if (Maps[newX][newY] == 0) {
                return true;
            }
        }
```

```
    }
    return false;
}

//打印终点信息
void PrintfEnd(int lastIndex) {
    Node toPrint[25];
    int i = 0;
    //依次倒序寻找前一个节点
    while (lastIndex != -1) {
        Node last = NodeQueue[lastIndex];
        lastIndex = last.lastIndex;
        toPrint[i++] = last;
    }
    //倒序输出最终的节点
    for (int j = i - 1; j >= 0; --j) {
        printf("(%d, %d)\n", toPrint[j].x, toPrint[j].y);
    }
    printf("(%d, %d)\n", Limit - 1, Limit - 1);
}


//广度搜索
void Bfs(Node start, Node end) {
    //初始化
    memset(Vis, false, sizeof(Vis));
    memset(NodeQueue, 0, sizeof(NodeQueue));
    //设置头尾
    int head = 0;
    int tail = 0;
    //将首点加入队列
    NodeQueue[head] = start;
    //设置该节点已经被访问
    Vis[start.x][start.y] = true;
    //标记当前节点已经走过
    while (head <= tail) {
        //取出一个节点
       Node node = NodeQueue[head];
        //向四个方向移动;
        for (int k = 0; k < 4; ++k) {
            int newX = node.x + gapX[k];
            int newY = node.y + gapY[k];
            //找到目标;
            if (newX == end.x && newY == end.y) {
                PrintfEnd(head);
                return;
            }
            //下个节点可以被访问
          if (CanPass(newX, newY)) {
```

```
                    //设置该节点已经被访问
                    Vis[newX][newY] = true;
                    tail++;
                    //将数据加入新的队列
                    NodeQueue[tail] = Node(newX, newY, head, node.step + 1)
                }
            }
            head++;
        }
    }

    //读取输入的数据
    void ReadInfo() {
        Node start = Node(0, 0, -1, 1);
        Node end = Node(Limit - 1, Limit - 1, -1, 1);
        //读入字符串信息;
        for (int i = 0; i < Limit; ++i) {
            for (int j = 0; j < Limit; ++j) {
                cin >> Maps[i][j];
            }
        }
        Bfs(start, end);
    }

    int main() {
        ReadInfo();
        return 0;
    }
```


;