数据结构迷宫大作业

maze_gen(maze)函数可以生成一个迷宫，0为可以走，1为不可以走。

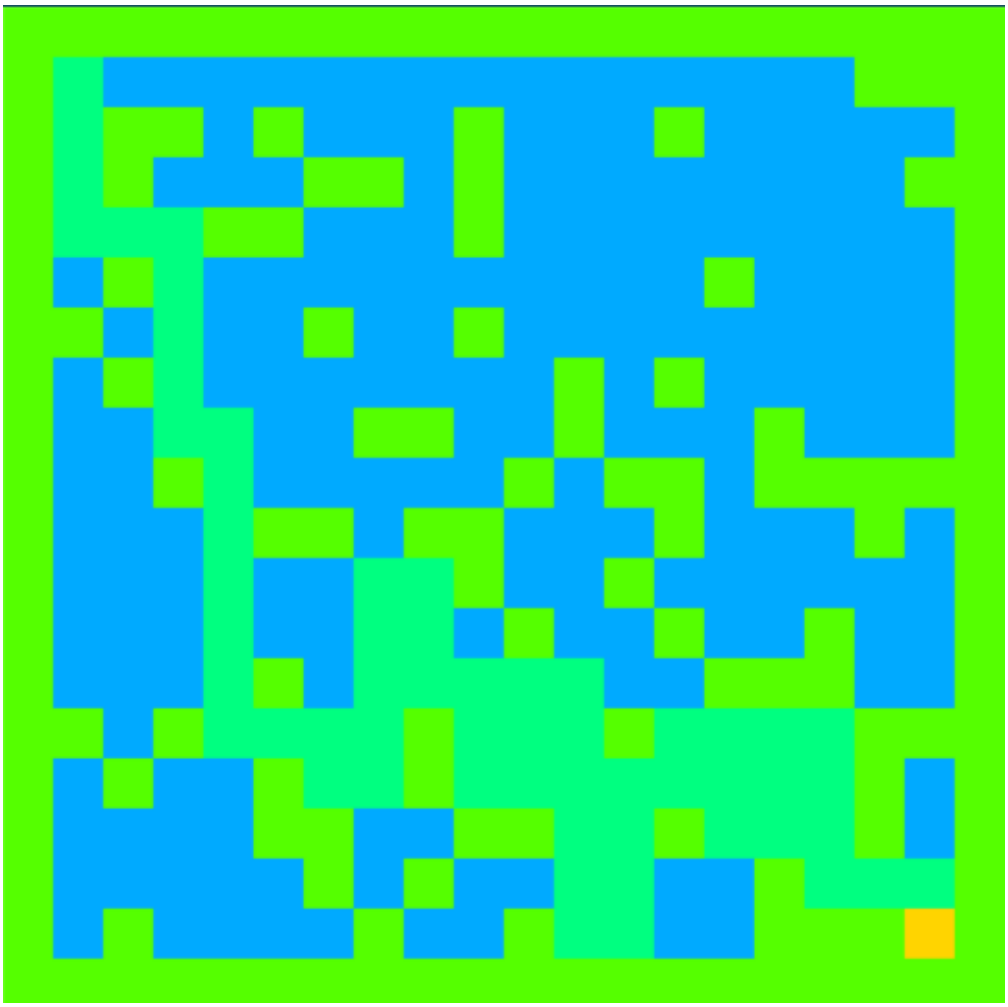
迷宫的长和宽可以通过define中的LENGTH和WIDTH，其中length和width包括迷宫的边界，在打印所有路径時，不推荐使用超过7以上的长宽，会打印出超多路径。

为了表示清楚，使用plot函数可以调用graphic包画出迷宫，不可以走为绿色，可以走为黄色。

在之后走过的路径会被标上2，在图上会被表示成青色。

为了提前剪枝，当一个位置没有通往目的地的路，会被标上3，画图会显示为蓝色。

下图是一个找到路径的示意图，出发位置为左上角，到达位置为右下角。



1.查找是否有路径

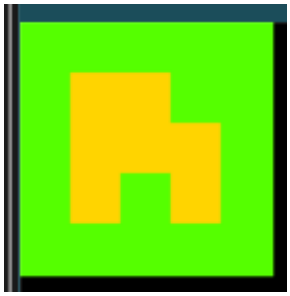
严老师曾说过开二维数组比较浪费，因此我开了一个一位数组通过坐标的运算来表示迷宫maze。

当一个位置的上下左右都走不通会被标为3，在此之后就不会走这个位置。

```
bool find_path(int *maze, int row, int col) {
    if (col == (WIDTH - 2) && row == (lENGTH - 2)) {  return true;}
    if (maze[row * WIDTH + col] == 1|| maze[row * WIDTH + col] == 2|| maze[row *
WIDTH + col] == 3)return false;
    maze[row * WIDTH + col] = 2;
    bool up=find_path(maze, row + 1, col);
```

```
    bool down=find_path(maze, row - 1, col);
    bool left = find_path(maze, row, col + 1);
    bool right = find_path(maze, row, col - 1);
    if (up || down || left || right) {
        return true;
    }
    else {
        maze[row * WIDTH + col] = 3;
        return false;
    }

}
```

运行实例



生成的迷宫见第一张图，走的路径见第二张图。以下是输出结果。

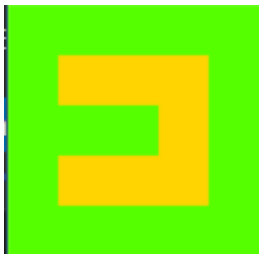
```
1
C:\Users\admin\source\repos\ConsoleApplication1\Debug\ConsoleApplication1.exe (进程 13216)已退出，代码为 0。
要在调试停止时自动关闭控制台，请启用“工具”->“选项”->“调试”->“调试停止时自动关闭控制台”。
按任意键关闭此窗口. . .
```

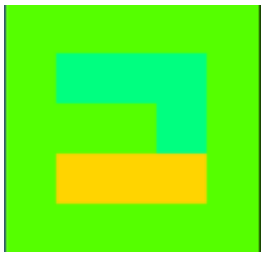
2.打印路径

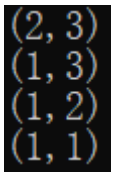
```
bool find_print_path(int* maze, int row, int col) {
    if (col == (WIDTH - 2) && row == (lENGTH - 2)) { return true; }
    if (maze[row * WIDTH + col] == 1 || maze[row * WIDTH + col] == 2 || maze[row
* WIDTH + col] == 3)return false;
    maze[row * WIDTH + col] = 2;
    bool up = find_print_path(maze, row + 1, col);
    bool down = find_print_path(maze, row - 1, col);
    bool left = find_print_path(maze, row, col + 1);
    bool right = find_print_path(maze, row, col - 1);
    if (up || down || left || right) {
        cout << "(" << row << "," << col << ")" << endl;
        return true;
    }
    else {
        maze[row * WIDTH + col] = 3;
        return false;
    }
```

```
}
```

运行实例





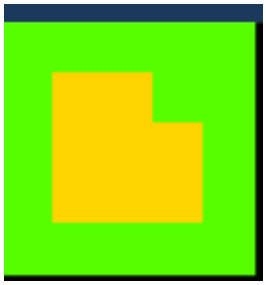


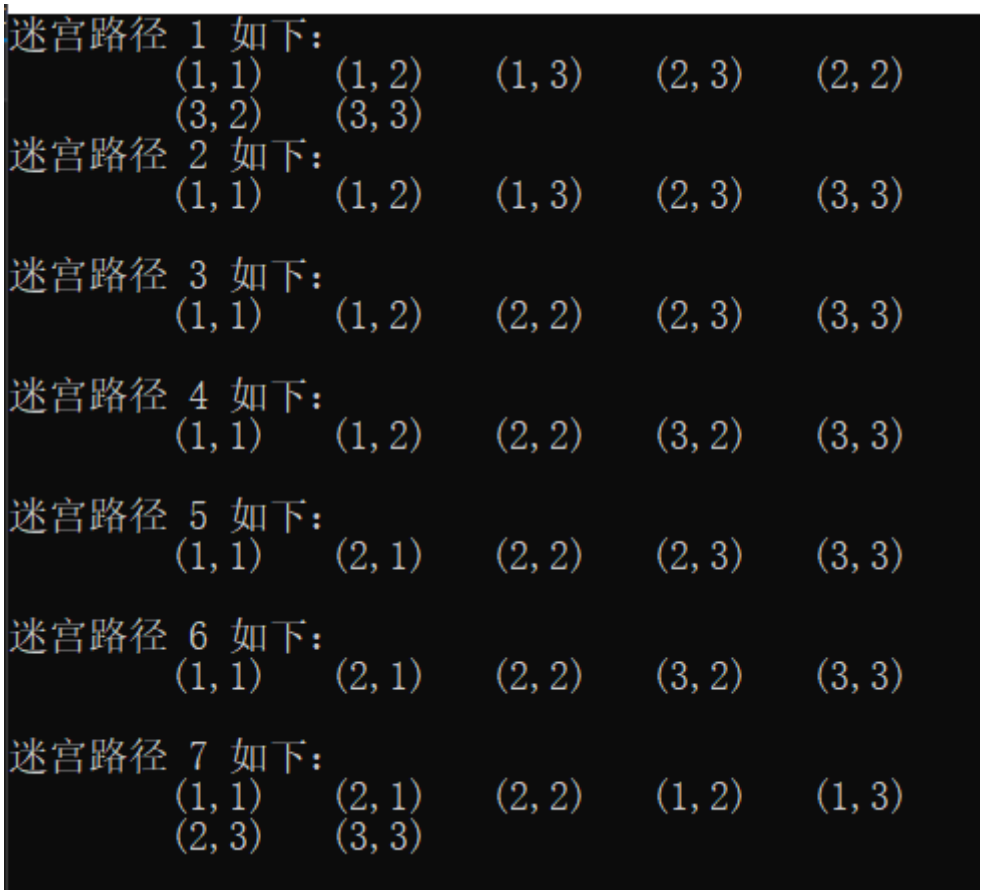
3.打印所有路径

```
typedef struct
{
    int i;
    int j;
}Box;
typedef struct
{
    Box data[1000];
    int length;     //路径长度
}PathType;
int num = 0;
void print_all_path( int row, int col, PathType path) {
    if (col == (WIDTH - 2) && row == (lENGTH - 2)) {
        path.data[path.length].i = col;
        path.data[path.length].j = row;
        path.length++;
        printf("迷宫路径 %d 如下：\n", ++num);
        for (int k = 0;k < path.length;k++) {
            printf("\t(%d,%d)", path.data[k].i, path.data[k].j);
            if ((k + 1) % 5 == 0)       //每输出5个方块后换行
                printf("\n");
        }
        printf("\n");
    }

    else {
        int i;
        if (maze[row * WIDTH + col] != 1 && maze[row * WIDTH + col] != 2) {
            for (i = 0;i < 4;i++) {
```

```
                maze[row * WIDTH + col] = 2;
                path.data[path.length].i = col;
                path.data[path.length].j = row;
                path.length++;
                switch (i)
                {
                case 0: print_all_path(row + 1, col, path);break;
                case 1: print_all_path(row - 1, col, path); break;
                case 2: print_all_path(row, col + 1, path);break;
                case 3:  print_all_path(row, col - 1, path);break;
                default:
                    break;
                }
                path.length--;//这步忘记加了
                maze[row * WIDTH + col] = 0;


            }
        }

    }
}
```

运行实例





调用说明

对main中的三个函数选择性注释，运行对应任务

```
//第一个任务：查找是否有路径
    //bool ispath = find_path(maze,1,1);
    //printf("%d", ispath);
    //plot(maze);

    //第二个任务：打印路径
    find_print_path(maze, 1, 1);
    plot(maze);

    //第三个任务：打印所有路径
    //PathType path;
    //path.length = 0;
    //print_all_path( 1, 1, path);
```