迷宫问题的广度优先搜索求解

（采用Visual Studio 2017 15.6.4 Windows SDK 10.0.16299.0）

这是第四次上机作业的另一部分，主要是对一个给定迷宫的求解。为此沿用了此前为十字路口管理写的动态分配二维数组的代码，作为迷宫的图纸，然后一步步广度优先查找直到找到结尾。为此需要队列的结构存储。队列使用顺序存储结构。然后由于进行广度优先搜索时标准的栈结构会把之前经历过的内容弹出，为此将迷宫的每一个点都对应了一个表示这个点为终点的最佳路径的栈。

0、要完成的目标

实现输入迷宫与求解最佳路径。

1、最终的程序结构

用户输入文件路径或者输入行数，列数和具体的迷宫形态。以0表示可以进入的格子，1表示不可以的，2表示入口，3表示出口。

沿用原先的十字路口的代码，用二维数组（char类型节省空间）表示这个输入的迷宫，输入时把0，2，3对应为数组中的0，1对应为1，并记录开始和末尾（没有开始或末尾或者行列数不一致则报错（输出错误原因和内容））。

然后从入口开始（并处理开始的初始赋值）依次把可能的并且没有到达过走法压入搜索用队列和记录用栈并把前往的位置标记为已搜索过（标记为2）然后把下一个点之前所走过的路径复制到一个指向走法栈结构的指针指向的空间。

在弹出计算用队列时同时把对应点的路径复制到当前路径，然后重复上面的计算，用一个循环来完成这样的操作。最后“遍历”到终点时结束并输出当前走法，同时提示是否继续。若继续，就重新计算（同时删除之前内容，避免内存泄漏）。

2、一些抽象

原则上对迷宫的广度优先搜索的想法有点像菲涅尔原理在光学中的想法，一步步地向外“扩散”。也实际上是对一个树的逐层遍历。但遍历这个树同时要找到整个遍历的路径（由此看其实可能又有一种直接“四叉”树的解法）。

3、一些遇到的问题和解决方法

首先写程序干的事情是写栈结构。为了让栈，像老师说的，不会假溢出，写了一个类，把正确操作封装在了函数里。

在写程序的时候然后遇到的问题就是标准的队列并不能提供到达某个位置的路径。当时想了很久，最后的解决方案是把每个已到达点在标记为已遍历的同时把这个点对应的唯一的走法也对应地记录下来，然后在需要的时候调用。

然后迷宫的二维数组的元素就变成了一个指向对应路径的指针和一个字符。然后就用原来的二维数组的模板写了出来。

一开始程序也是可以运行的，但是开始出现问题。于大一点的迷宫，输入有时会报错，然后有时候二维数组算着算着会突然有数字变成似乎是随机的数。然后加上析构函数以后更是直接在析构时报错，说内存访问错误。每次运行错误都不一样。

调试了很久，最后一点点作运行测试，确定问题出在二维数组上，再进一步确定仅仅是字符的二维数组是没有问题的。为此专门把指针数组放在了一个单独的数组里，然后这个数组接着报错。

后来改了很久代码，怎么改都没有变化，最后干脆不直接开（行\*列）的连续空间，再用指针数组指向行的开头了，而是每一行单独作为一个指针数组，然后再用二级指针指向这些“行”的开头。然后就没问题了。

对于这个的问题我真的没办法用数据结构或者算法的方法解释为什么这样就好了。毕竟原则上是没有区别的，访问方式都是通过两次指针引导访问需要访问的数据。

我觉得很有可能是编译器我的问题，可能VS2017的编译器在处理指针时会在指针后面的某些地方作为存储指针指向空间大小的地方，而导致在我的数组中错误地写入了我在使用的空间。

4、一些可以改进的地方

首先，实际上“当前路径”这个变量不需要真的在内存中单独存在。只要把所以“当前路径”都换成对应二维数组位置的指针指向的路径，就可以大大减少复制运算的需要，提高效率。

其次，实际上最好的这种广度优先的搜索的存储结构并不是标准的栈结构，而是弹栈以后不删除的一种存储结构。这样就可以通过一个下标来标记每个“节点”的“父节点”，从而完全不需要专门开辟空间，进行复制运算来记录路径，也不用专门写一个循环的顺序栈来完成栈的功能。

尽管这样做需要的存储空间看似多很多，但是因为之前用顺序栈时为了安全实际上也开辟的是这么大的顺序栈，所以优势是很明显的。但是因为似乎老师说了要用栈结构，所以没有再修改。

5、总结

以后在使用new和delete作智能指针时尽量避免前面出问题的操作。