上机报告

一、任务

1.创建一个栈类，一个队列类，一个迷宫类

2.写一个可以随机生成迷宫的函数

3.用递归和栈分别写一个可以解决迷宫的函数

4.用队列找到解决迷宫的最短路径

5.用队列找到所有的路径

二、实现

我先写了一个link类，link是一个链表类，栈、队列均使用链表实现。link的成员有：

info：存储该结点的信息。

isDel：这个bool变量是为了方便析构而创建的。

next：指向下一个节点地址的指针。

prev：指向上一个节点地址的指针。

loop\_num：记录这个链表的长度。

为了便于在其他类中使用，link是一个模板类，而且是一个循环链表。这样我们就能在剩下的类中直接使用它。

之后我写了一个Node类。Node是一个链表类，和link不同的是Node不是循环链表，它是单向的一个链表，它的成员有：

info：存储这个节点的信息。

next：指向下一个节点。

和link类似，Node也是一个模板类。

之后我写了一个Stack。Stack是一个栈类，它是用node实现的，它的成员有：

top：指向当前的栈尾。

栈的功能和递归可以互相替代，在实际使用中栈被实例化为迷宫格点grid的指针。

queue类是一个队列类，这个类是用link实现的。为了节约空间，queue使用了循环链表，这样可以避免一些无谓的空间浪费。它的成员有：

f：指向队首的指针。

r：指向队尾的指针。

Max：这个队列的长度。

n:当前的元素个数。

在实例化过程中，queue被实例化为grid指针的队列。我删除了Queue的默认构造函数，以int为参数的构造函数里queue的默认长度为100。实际使用过程中如果发现空间不够用，每次会自动把queue 的长度扩充100个。直到内存不足为止。

grid是迷宫格点的类，它的成员有：

x：格点的横坐标。

y：格点的纵坐标。

i：格点的信息，0代表通路，1代表墙体，2代表已经走过。

在实际的实现过程中，我并没有专门写一个迷宫类，而是始终使用grid的数组。同时我需要传入迷宫的一些长宽参量。

栈和递归思路基本一致，都是通过深度搜索来找到迷宫的出口。下面以栈为例写一个迷宫搜索伪代码：

while 还没找到出口

if 这个点可以向右走且右边没有被走过

右边的点入栈

else if 这个点可以向下走且左边没有被走过

下边的点入栈

else if 这个点可以向左走且右边没有被走过

左边的点入栈

else if 这个点可以向上走且右边没有被走过

上边的点入栈

else

出栈

而队列的思路相对简单，是一种典型的广度搜索。伪代码如下：

while 队尾不是出口

出队

if 右边的点可以走且没走过

右边的点入队

if 左边的点可以走且没走过

左边的点入队

if 上边的点可以走且没走过

上边的点入队

if 下边的点可以走且没走过

下边的点入队

二者相比，队列的搜索耗时明显多于栈和递归，内存占用也更高，但是搜索出来的路径最短，最符合实际走迷宫的想法。

接下来定义一个Lab类，Lab里存有一个迷宫。它的数据成员有:

Grid: Grid是一个grid数组，可以存一个迷宫的所有格点。

n: n是一个int，可以存这个迷宫的大小信息。

X： 当前位置的横坐标。

Y： 当前位置的纵坐标。

Lab的赋值构造和赋值操作均定义为深层复制，这样对赋值构造的一个Lab进行操作的时候不会影响原来的Lab。

Lab类内有四个类函数，分别是east，west，north，south，这四个类函数返回一个新的Lab，新的Lab相比较原来的Lab分别向东西北南运动。

函数all\_route负责输出所有的迷宫路径。伪代码如下：

迷宫起点入队。

创建一个vector<Lab>记录可行路径

While(队列不空)

Tmp<-队列当前元素；

队列队尾出队。

如果（Tmp是终点）

把Tmp压入可行路径vector

Continue

如果（可以向右走）

Tmp.east入队

如果（可以向左走）

Tmp.west入队

如果（可以向上走）

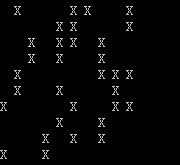
Tmp.north入队

如果（可以向下走）

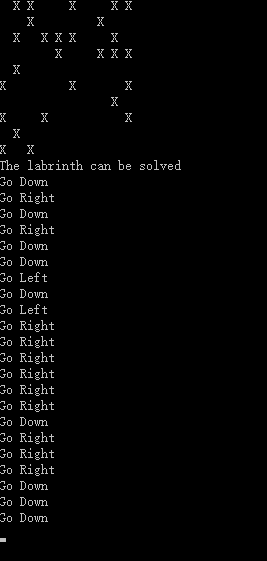
Tmp.south入队

三、输入结果

随机迷宫：（10 \* 10）

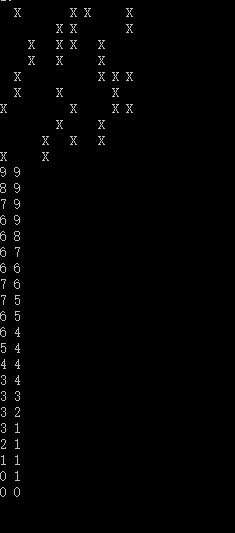


递归:



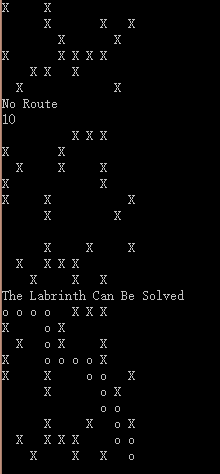
这个是逆向输出路径的。

栈：



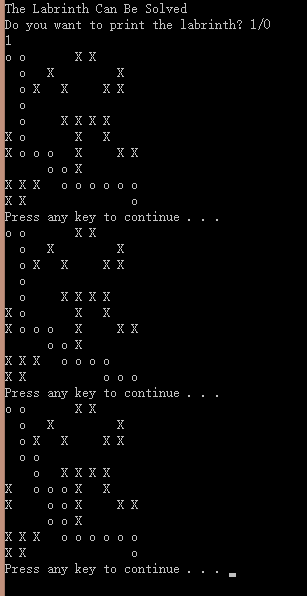
逆向输出所走路径。

队列解决最短路径问题：



第一次无解，第二次的路径已用o标出。

队列输出全部路径：



这是一部分输出结果。也可以直接输出路径数目，输入0即可。

四、总结

这次的上机代码锻炼了我写模板类的能力。在写类的过程中我严格遵照了三五原则，认认真真把类写好再去实现迷宫算法，虽然累，但是在后期过程中少了很多莫名其妙的bug。这告诫我平时在写类的时候不能偷懒，要写全。

栈和队列是两种特别有用的算法，分别可以用于深度和广度搜索。迷宫算法虽然简单，但是却让我更深的了解了这两种算法思路，收获颇丰！