数据结构课程设计

项目说明文档

**勇闯迷宫游戏**



同济大学

Tongji University

姓名： 林觉凯

学号： 2253744

指导老师： 张颖

学院专业： 软件学院 软件工程

**目录**

**1.项目分析-------------------------------------------------------------------------3**

**1.1 项目背景分析-------------------------------------------------------------------------3**

**1.2项目功能分析--------------------------------------------------------------------------3**

**1.2.1项目功能要求-------------------------------------------------------------------------------3**

**1.2.2项目输入要求-------------------------------------------------------------------------------3**

**1.2.3项目输出要求-------------------------------------------------------------------------------3**

**1.2.4项目实例-------------------------------------------------------------------------------------4**

**2.项目设计-------------------------------------------------------------------------4**

**2.1 数据结构设计-------------------------------------------------------------------------4**

**2.2算法设计--------------------------------------------------------------------------------4**

**2.3系统流程设计--------------------------------------------------------------------------5**

**3.项目实现-------------------------------------------------------------------------6**

**3.1 ascend\_dimension函数的实现-----------------------------------------------------6**

**3.2 reduce\_dimension函数的实现-----------------------------------------------------6**

**3.3 Search函数的实现-------------------------------------------------------------------6**

**3.4 main函数的实现---------------------------------------------------------------------8**

**4.项目心得与体会----------------------------------------------------------------9**

**1.项目分析**

**1.1 项目背景分析**

迷宫只有两个门，一个门叫入口，另一个门叫出口。一个骑士骑马从入口进入迷宫，迷宫设置很多障碍，骑士需要在迷宫中寻找通路以到达出口。

**1.2项目功能分析**

**1.2.1项目功能要求**

迷宫问题的求解过程可以采用回溯法即在一定的约束条件下试探地搜索前进，若前进中受阻，则及时回头纠正错误另择通路继续搜索的方法。从入口出发，按某一方向向前探索，若能走通，即某处可达，则到达新点，否则探索下一个方向；若所有的方向均没有通路，则沿原路返回前一点，换下一个方向再继续试探，直到所有可能的道路都探索到，或找到一条通路，或无路可走又返回入口点。在求解过程中，为了保证在达到某一个点后不能向前继续行走时，能正确返回前一个以便从下一个方向向前试探，则需要在试探过程中保存所能够达到的每个点的下标以及该点前进的方向，当找到出口时试探过程就结束了。

**1.2.2项目输入要求**

由于题目给出的迷宫固定，所以我们在程序中定义了一个全局变量：迷宫的地图，故该项目不需要输入。

**1.2.3项目输出要求**

1.输出走出迷宫的路径，每个坐标点用<row,column>表示，中间用--->连接。

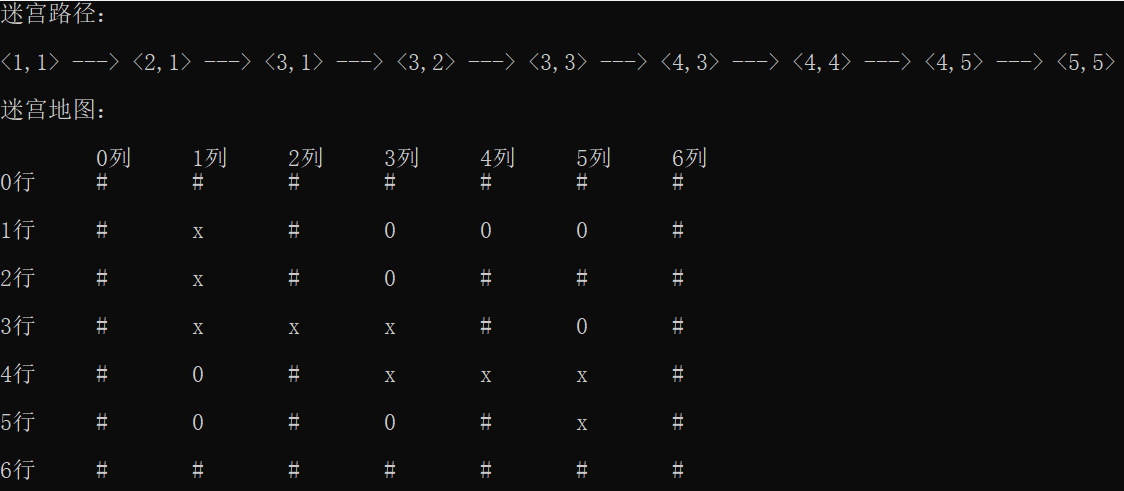
2.输出走过的迷宫的地图。

‘#’——表示迷宫中的墙

‘0’——表示迷宫的通路

‘x’——表示骑士从迷宫入口走过迷宫到迷宫出口的路径。

同时输出每一行和每一列。

**1.2.4项目示例**

**2.项目设计**

**2.1 数据结构设计**

本题的数据存储利用到了以下两个数组完成实现：

1.mazemap[7][7]：迷宫地图二维数组，用来储存迷宫墙、通路和路径信息。

2.StackArray[50]：这个数组用来可行路径的二维坐标转化为一维坐标的结果，利用到了类似“栈”的思维存储从出口到入口的坐标信息。

**2.2算法设计**

勇闯迷宫游戏使用递归回溯搜索法即深度搜索实现。

深度优先搜索算法（Depth First Search , DFS）的思想即回溯法（Backtrack），以下为回溯法思想的如下：在一定的约束条件下沿着某一个方向搜索前进，如果在这一个搜索前进的方向失败，便回头寻找其他搜索的方法继续搜索前进。

在勇闯迷宫游戏求解过程中，我们首先从第一行第一列(入口处)开始计算，每次计算首先判断计算的坐标点的行数和列数是否已经达到了最大值，如果达到了最大值，说明已经遍历至出口，我们用StackArray数组来记录这个位置的坐标信息。然后我们就要判断遍历的这个坐标点是墙还是通路，如果是墙的话，说明这个点根本就走不通，直接return false；如果是通路的话，我们就要对这个点的上下左右四个方向再进行考察(考察的算法就是Search函数)，这个上下左右这四个点有一个是可以走得通的，就说明这个点有效，我们就也用StackArray数组记录下这个坐标点的基本信息。这个算法进行不断地迭代递归，直到从入口坐标一直递归到出口坐标为止(递归开始条件：还未递归到出口的行数和列数)。

**2.3系统流程设计**

**程序开始**

**根据地图开始遍历**

Yes

**退出遍历**

**Start\_row==End\_row**

**Start\_column==End\_column**

**打印**

**迷宫路径**

**迷宫地图**

No

**搜索上下左右四个方向是否通路**

**肯定有一个方向**Yes

**标记位置**

**储存入数组内**

**3.项目实现**

**3.1 ascend\_dimension函数的实现**

//ascend\_dimension函数，用来将一维储存的的长度转化为二维迷宫的坐标位置

void ascend\_dimension(int& number, int& row, int& column)

{

row = number / 7;

column = number % 7 - 1;

}

ascend\_dimension函数是升维行数，它的作用是将用一位储存的长度转化为二维迷宫的具体坐标位置(第几行第几列)。它与下面的reduce\_dimension函数配套使用，由于我们用来储存坐标位置信息的StackArray数组是一维的，所以当我们找到合适的路径的坐标点时，我们需要将二维的坐标点信息转化为一维的信息储存，当在打印迷宫地图的时候，我们又要用ascend\_dimension函数将StackArray函数中的每一个坐标信息再转化为具体的每一个二维坐标点。

**3.2 reduce\_dimension函数的实现**

//reduce\_dimension函数，用来将二维的迷宫坐标位置转化为一维储存的的长度储存起来

int reduce\_dimension(int& row, int& column)

{

int length = row \* 7 + column + 1;

return length;

}

reduce\_dimension函数是降维函数，它的作用是将二维迷宫的具体坐标位置(第几行第几列)转化为用一位储存的坐标信息。它与上面的ascend\_dimension函数配套使用，由于我们用来储存坐标位置信息的StackArray数组是一维的，所以当我们找到合适的路径的坐标点时，我们需要将二维的坐标点信息转化为一维的信息储存，所以这个时候我们就要用到reduce\_dimension函数将这一个二维点坐标转化为其相应的一维的坐标信息来储存到StackArray数组中去。

**3.3 Search函数的实现**

//bool Search函数，用于计算递归回溯，StackArray数组用来储存储存可行路径的二维坐标转化为一维坐标的结果(递归的是从入口向出口递归的，第三返回是从出口到入口返回的，所以StackArray里下标大的储存的是靠近入口的位置坐标)

bool Search(char mazemap[7][7], int\* StackArray, int Start\_row, int End\_row, int Start\_column, int End\_column, int& top)

{

if (Start\_row == End\_row && Start\_column == End\_column)

//如果遍历至尾行尾列，遍历完成，找到出口，记录出口坐标，返回true

{

StackArray[top++] = reduce\_dimension(Start\_row, Start\_column);

return true;

}

if (mazemap[Start\_row][Start\_column] == '#') //如果这一个位置是墙，返回false

{

return false;

}

if (mazemap[Start\_row][Start\_column] == '0') //如果这一个位置是通路

{

mazemap[Start\_row][Start\_column] = '#';

//先将这个位置变为墙，以免对下面的遍历产生影响

if (Search(mazemap, StackArray, Start\_row + 1, End\_row, Start\_column, End\_column, top)) //向下走可行

{

StackArray[top++] = reduce\_dimension(Start\_row, Start\_column); //记录此点坐标信息

return true;

}

else if (Search(mazemap, StackArray, Start\_row - 1, End\_row, Start\_column, End\_column, top)) //向上走可行

{

StackArray[top++] = reduce\_dimension(Start\_row, Start\_column); //记录此点坐标信息

return true;

}

else if (Search(mazemap, StackArray, Start\_row, End\_row, Start\_column + 1, End\_column, top)) //向右走可行

{

StackArray[top++] = reduce\_dimension(Start\_row, Start\_column); //记录此点坐标信息

return true;

}

else if (Search(mazemap, StackArray, Start\_row, End\_row, Start\_column - 1, End\_column, top)) //向左走可行

{

StackArray[top++] = reduce\_dimension(Start\_row, Start\_column); //记录此点坐标信息

return true;

}

mazemap[Start\_row][Start\_column] = '0'; //恢复当前位置现场为'0'

}

return false;

//如果以上四种情况都不能达成，则此点不在路径上，不记录此点坐标信息，返回false

}

Search函数，用于计算递归回溯。在递归的开始，我们首先要判断当前行数和列数是否达到了最大，如果是的话就说明已经到达出口，return true；在递归的过程中，可能会遇到某一个坐标点是‘#’或者‘0’的两种情况，如果是‘#’的话，说明该点是墙，走不通，肯定不在路径上，return false；如果是‘0’的话，说明该点是通路，有可能在路径上，此时还需要对该点坐标的上下左右四个方向进行递归，看看这个点往下走是否能走到下一个能通的点，如果可以的话，则用StackArray数组来储存该点坐标的基本信息，return true。StackArray数组用来储存储存可行路径的二维坐标转化为一维坐标的结果。(递归的是从入口向出口递归的，第三返回是从出口到入口返回的，所以StackArray里下标大的储存的是靠近入口的位置坐标)

**3.4 main函数的实现**

//主函数，由于初始化数据和最后打印迷宫路径和迷宫地图

int main()

{

int StackArray[50] = { 0 };

//运用"栈"的思维，StackArray用来储存可行路径的二维坐标转化为一维坐标的结果

int top = 0;

Search(mazemap,StackArray, 1, 5, 1, 5, top); //递归从第一行第一列开始至第五行第五列

int row = 0, column = 0;

ascend\_dimension(StackArray[--top], row, column); //类似"出栈"的思想打印迷宫路径

mazemap[row][column] = 'x';

cout << "迷宫路径：" << endl << endl;

cout << "<" << row << "," << column << "> ";

while (top != 0)

{

ascend\_dimension(StackArray[--top], row, column);

mazemap[row][column] = 'x';

cout << "---> <" << row << "," << column << "> ";

}

cout << endl << endl;

cout << "迷宫地图：" << endl << endl; //遍历mazemap数组，打印迷宫地图

for (int i = 0; i < 7; i++)

{

cout << "\t" << i << "列";

}

cout << endl;

for (int i = 0; i < 7; i++)

{

cout << i << "行";

for (int j = 0; j < 7; j++)

{

cout << "\t" << mazemap[i][j];

}

cout << endl << endl;

}

return 0;

}

主函数，由于初始化数据和最后打印迷宫路径和迷宫地图.在主函数中，我们首先定义了StackArray数组和top(运用类似栈的思维通过top的++和--完成StackArray数组的位置填充和StackArray数组下标的移动)；之后调用Search函数，设置输入参数递归从第一行第一列开始至第五行第五列，在这之后，再利用类似"出栈"的思想打印迷宫路径，同时在迷宫路径上填充‘x’；最后再通过遍历mazemap数组，打印出迷宫地图。

**4.项目心得与体会**

勇闯迷宫游戏和第四道题N皇后所用的方法都是递归回溯法，个人认为这道题目比N皇后难一些。在这次项目中，我主要收获了以下两个心得：第一是利用一维数组来存储二维的信息单元，通过行数与列数的计算来得到一个对应的位置信息然后存储到一维数组之中去；第二是我又加深了对递归回溯法的印象。我记得上次在写N皇后代码的时候忘记了在每次成功找到方法之后还原现场导致了当时一直没有跑出正确结果，这次我在写勇闯迷宫游戏的时候就着重注意到了这一个问题，所以写得就比较顺。但是本道题目中，在如何存储合适的坐标点信息和在四个方向上的再次判断应该要用怎样形式的语句表达这个问题我想了一段时间。总之，递归回溯法我还需要通过继续练习来巩固。