

数据结构课程设计报告



**学 号：\_\_\_\_\_\_\_\_\_1806300153\_\_\_\_\_\_\_\_\_\_\_\_\_**

**姓 名：\_\_\_\_\_\_\_\_\_曾祥毅\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**班 级：\_\_\_\_\_\_\_\_\_\_软件185\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**指导老师：\_\_\_\_\_\_\_\_綦 科、张 为\_\_\_\_\_\_\_\_\_\_**

**自评成绩：\_\_\_\_\_\_\_\_\_\_\_良\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**成 绩: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

2019-12-20

广州大学计算机科学与网络工程学院

**广州大学学生实验报告**

**开课学院及实验室**： 电子楼517室 2019 **年**12**月**21**日**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **学院** | 计算机学院 | | **年级、专业、班** | 软件185 | **姓名** | 曾祥毅 | **学号** | 1806300153 |
| **实验课程名称** | | 数据结构 | | | | | **成绩** |  |
| **实验项目名称** | | 课程设计 | | | | | **指导老师** |  |

1. **题目内容**

**迷宫求解问题**

[问题描述]

设计一个8X8的迷宫, 迷宫入口为(1,1) ，出口为（8,8）， 设计算法， （1）求出从入口到出口的所有可能路径。（2）求出从入口到出口的最短路径。

[基本要求]

（1） 用矩阵表示一个迷宫，并输出该迷宫；。

（2） 显示所有路径的移动轨迹。

1. **程序说明**

程序自动生成随机迷宫，且该迷宫只有一种正确路径。程序包括迷宫生成算法和迷宫求解算法两部分。

**迷宫生成算法：**使用prime迷宫生成算法，基本算法思想为：

1.初始地图所有位置均设为墙

2.任意插入一个墙体进墙队列

3.判断此时墙体是否可以设置为路（判断依据在于上下左右四个位置是否只有一个位置是路）

4.若设置为路，则将该位置周围（上下左右）的所有墙插入队列，接着执行第五步；若无法设置为路，直接执行第五步

5.从墙队列中删去当前位置所在节点

6.若墙队列不为空，则从队列中随机选取一面墙重新执行从第三布开始执行，直到墙队列为空

**迷宫求解算法：**采用深度优先搜索算法，基本思想如下：

1. 从起点开始，设置起点为已经访问过（走过），以一定顺序检查四周是否有路
2. 若一个方向有路且未被走过，则访问该点，设置该点为已经访问，若迷宫尚未解决，则标记该点为路径，并继续寻找四周是否有路（从第一步开始执行）
3. 若走到某一点后四个方向都没有路或都走过且迷宫未解决，则清除该点的路径并返回上一个点（返回第二步）
4. 若走到终点，设置迷宫已解决并返回
5. 最终递归出口为遍历完所有路
6. **程序源代码**

#include <iostream>

#include <ctime>

#include<vector>

using namespace std;

class LabyM

{

private:

int laby[12][12];

int visited[12][12];

bool solved;

int exitX;

int exitY;

public:

LabyM();

void showLaby();

void generate();

void solve(int, int);

};

//初始化空迷宫

LabyM::LabyM()

{

for (int j = 0; j < 11; j++) for (int k = 0; k < 12; k++)laby[j][k] = 1;

for (int j = 0; j < 11; j++) for (int k = 0; k < 12; k++)visited[j][k] = 0;

solved = false;

}

void LabyM::showLaby()

{

for (int j = 0; j < 11; j++)

{

for (int k = 0; k < 11; k++)

{

if (laby[j][k] == 1)cout << "■";

else if (laby[j][k] == 5)cout << "☆";

else cout << " ";

}

cout << endl;

}

}

void LabyM::generate()

{

srand((unsigned)time(0));

//最外围设置为路，可以有效的保护里面一层墙体，并防止挖出界

for (int i = 0; i < 12; i++)

{

laby[i][0] = 0;

laby[0][i] = 0;

laby[11][i] = 0;

laby[i][11] = 0;

}

//墙队列

vector<int> X;

vector<int> Y;

//初始位置

X.push\_back(2);

Y.push\_back(2);

//当墙队列为空时结束循环

while (X.size())

{

//在墙队列中随机取一点

int r = rand() % X.size();

int x = X[r];

int y = Y[r];

//判读上下左右四个方向是否为路

int count = 0;

for (int i = x - 1; i < x + 2; i++)

{

for (int j = y - 1; j < y + 2; j++)

{

if (abs(x - i) + abs(y - j) == 1 && laby[i][j] == 0) {

++count;

}

}

}

//若四周无路或只有一个方向有路

if (count <= 1)

{

//挖去这一块

laby[x][y] = 0;

//在墙队列中插入新的墙

for (int i = x - 1; i < x + 2; i++)

{

for (int j = y - 1; j < y + 2; j++)

{

if (abs(x - i) + abs(y - j) == 1 && laby[i][j] == 1)

{

X.push\_back(i);

Y.push\_back(j);

}

}

}

}

//删除当前墙

X.erase(X.begin() + r);

Y.erase(Y.begin() + r);

}

//设置迷宫进出口

laby[2][1] = 0;

visited[2][2] = 1;

for (int i = 9; i >= 0; i--)

{

if (laby[i][9] == 0)

{

laby[i][10] = 0;

exitX = i;

exitY = 10;

break;

}

}

//输出初始迷宫

showLaby();

}

void LabyM::solve(int x, int y)

{

laby[2][1] = 5;

//如果当前点是出口

if (x == exitX && y == exitY)

{

visited[x][y] = 1;

laby[x][y] = 5;

solved = true;

return;

}

//如果当前点非出口

else

{

visited[x][y] = 1;

laby[x][y] = 5;

//寻找周围是否有路

for (int i = x - 1; i < x + 2; i++)

{

for (int j = y - 1; j < y + 2; j++)

{

//如果上下左右有路且没走过，且迷宫未解决，访问该路

if (abs(x - i) + abs(y - j) == 1 && laby[i][j] == 0 && visited[i][j] == 0 && !solved)

{

solve(i, j);

}

}

}

//访问完周边所有方块后，若迷宫未解决,移除当前的的路标并返回

if (!solved)

{

laby[x][y] = 0;

return;

}

return;

}

}

int main()

{

LabyM lm;

cout << "生成唯一解迷宫（8×8）：" << endl;

lm.generate();

lm.solve(2, 2);

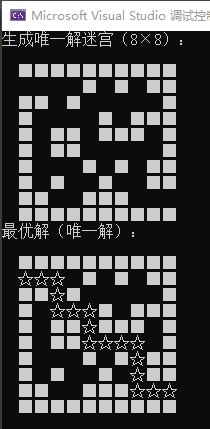
cout << "最优解（唯一解）：" << endl;

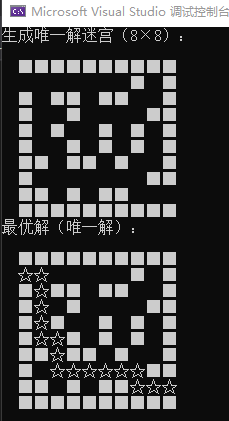
lm.showLaby();

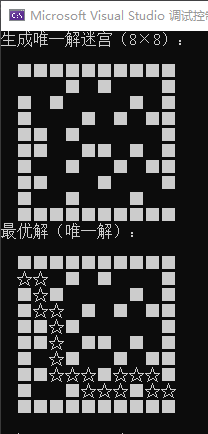
return 0;

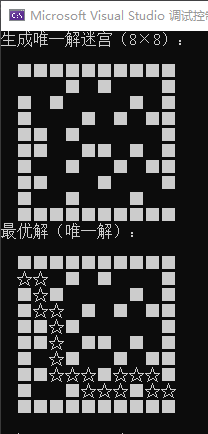
}

1. **程序测试结果**









**五、结果分析、收获和体会**

程序运行可以生成唯一解的迷宫，且能正确显示迷宫的求解路径。

一开始生成迷宫的时候，只用随机排布的方格总是会遇到路径被封死无法从入口走到出口的情况。索性就去研究了一下迷宫的生成算法，然后了解到了prime迷宫生成算法（和prime最小生成树算法无关）。接着就是迷宫的求解算法，思想基本是对迷宫进行深度优先搜寻，找到出口即停止修改，写的途中遇到的问题就是递归出口和路径显示的问题，和深度优先遍历略有不同的就在于回溯时要清除记录。

**六、改进意见和建议**

关于迷宫求解算法，由于递归工作栈的原因，当前写的算法无论迷宫复杂程度如何，都要遍历完整个迷宫，若用自己写的栈的话或许会更好。