**数据结构上机实习报告**

题目：2.9 迷宫问题

班级：1403013 学号：14030130061 姓名：刘豪 完成日期：2015.12.04

**一、需求分析**

1.**二、概要设计**

为了实现上述程序功能，我设计了一个栈类和迷宫类。

class Stack{

基本操作：

Stack()

初始化一个栈;

~Stack()

清除一个栈;

void push(int \_val)

压栈;

void pop()

弹栈;

int top()

返回栈顶元素;

int size()

返回栈中元素个数;

void del()

删除一个元素;

}

class Maze{

基本操作：

Maze()

随机生成一个迷宫;

Maze(int n,int m)

生成一个n行m列的迷宫;

bool solve()

通过dfs的栈展开来寻找路径;

void print()

打印结果;

}

**三、详细设计**

#include<iostream>

#include<cstdio>

#include<cstdlib>

#include<ctime>

using namespace std;

class Stack{

struct Node{

int val;

Node \*nex;

};

private:

Node \*head;

int \_sz;

public:

Stack();

~Stack();

void del();

void push(int \_val);

int top();

void pop();

int size();

};

void Stack::del(){

if(head==NULL)

return ;

\_sz--;

Node \*tmp=head;

head=head->nex;

delete tmp;

}

Stack::Stack(){

head=NULL;

\_sz=0;

}

Stack::~Stack(){

Node \*tmp;

while(head!=NULL){

(\*this).del();

}

}

int Stack::top(){

return head->val;

}

void Stack::pop(){

(\*this).del();

}

void Stack::push(int \_val){

\_sz++;

Node \*tmp=new Node;

tmp->val=\_val;

tmp->nex=head;

head=tmp;

}

int Stack::size(){

return \_sz;

}

class Maze{

private:

static const int MAXN=100;

static const int MAXM=100;

int maze[MAXN][MAXM];

bool vis[MAXN][MAXM];

bool tag[MAXN][MAXM][4];

int \_n,\_m;

int \_s,\_e;

Stack Sta;

public:

Maze();

Maze(int n,int m);

bool solve();

void print();

};

Maze::Maze(){

srand((unsigned)time(0));

\_n=rand()%20+5;

\_m=rand()%20+5;

\_s=rand()%20+20;

do{

\_e=rand()%20+(\_n\*\_m-21);

}while(\_e==\_s);

vis[\_s/\_m][\_s%\_m]=true;

maze[\_s/\_m][\_s%\_m]=2;

maze[\_e/\_m][\_e%\_m]=3;

for(int i=0;i<\_n;i++){

for(int j=0;j<\_m;j++){

if(maze[i][j]==2||maze[i][j]==3)

continue;

vis[i][j]=false;

maze[i][j]=rand()%10;

maze[i][j]=maze[i][j]<8?1:0;

for(int k=0;k<4;k++)

tag[i][j][k]=false;

}

}

}

Maze::Maze(int n,int m){

srand((unsigned)time(0));

\_n=n;

\_m=m;

\_s=rand()%20+20;

do{

\_e=rand()%20+(\_n\*\_m-21);

}while(\_e==\_s);

vis[\_s/\_m][\_s%\_m]=true;

maze[\_s/\_m][\_s%\_m]=2;

maze[\_e/\_m][\_e%\_m]=3;

for(int i=0;i<\_n;i++){

for(int j=0;j<\_m;j++){

if(maze[i][j]==2||maze[i][j]==3)

continue;

vis[i][j]=false;

maze[i][j]=rand()%10;

maze[i][j]=maze[i][j]<8?1:0;

for(int k=0;k<4;k++)

tag[i][j][k]=false;

}

}

}

bool Maze::solve(){

int dx[4]={1,-1,0,0};

int dy[4]={0,0,1,-1};

vis[\_s/\_m][\_s%\_m]=1;

Sta.push(\_s);

while(Sta.size()){

if(Sta.top()==\_e)

return true;

int x=Sta.top()%\_m;

int y=Sta.top()/\_m;

bool flag=false;

for(int i=0;i<4;i++){

int tx=x+dx[i];

int ty=y+dy[i];

if(tag[y][x][i]||vis[ty][tx]||!maze[ty][tx]||(tx<0||tx>=\_m||ty<0||ty>=\_n))

continue;

Sta.push(ty\*\_m+tx);

vis[ty][tx]=true;

tag[y][x][i]=true;

flag=true;

break;

}

if(!flag){

vis[y][x]=false;

for(int i=0;i<4;i++)

tag[y][x][i]=false;

Sta.pop();

}

}

return false;

}

void Maze::print(){

if(Sta.size())

Sta.pop();

while(Sta.size()>1){

int x=Sta.top()%\_m;

int y=Sta.top()/\_m;

maze[y][x]=-1;

Sta.pop();

}

for(int i=0;i<\_n;i++){

for(int j=0;j<\_m;j++){

switch(maze[i][j]){

case -1:

printf("\*");break;

case 0:

printf("#");break;

case 1:

printf(".");break;

case 2:

printf("@");break;

case 3:

printf("^");break;

default:

break;

}

}

printf("\n");

}

printf("\n");

}

int main(){

int n,m;

printf("请输入迷宫的行数和列数: ");

scanf("%d%d",&n,&m);

Maze a(n,m);

printf("\n符号代表的意义:\n'.'=空地 '#'=墙 '\*'=答案 '@'=起点 '^'=终点\n");

a.print();

if(a.solve())

a.print();

else

printf("There is no answer\n");

}

**四、调试分析**

1.通过头插法和头结点删除法建立了一个栈。

2.由于是通过dfs的栈展开来做的，所以找到的解并不是最优解，可以考虑改进通过bfs来求解。

**五、用户手册**

本程序的运行环境为WIN10，执行文件为:棋盘问题.exe

**六、测试结果**

