天津理工大学中环信息学院

课程教案

2018 ～ 2019 学年 第 一 学期

教 学 单 位 计算机工程系

教 研 室 计算机科学与技术教研室

课 程 名 称 数据结构

主 讲 教 师 史英杰

职 称 / 职 务 讲师

使 用 教 材 数据结构教程—李春葆

课程教案29

|  |  |  |
| --- | --- | --- |
| 授课题目（教学章、节或主题）：  **栈和队列的应用-迷宫问题** | 课时安排 | 2学时 |
| 教学的和要求（分掌握、熟悉、了解三个层次）：  1．掌握：灵活运用两个数据结构，知道用栈和队列求解迷宫的差异  2．熟悉：栈和队列的特性  3．了解： 栈和队列的概念 | | |
| 教学内容（包括基本内容、重点、难点）：  1．基本内容：栈和队列的概念、栈和队列的特性、灵活运用两个数据结构，知道用栈和队列求解迷宫的差异  2．重点：灵活运用两个数据结构，知道用栈和队列求解迷宫的差异  3．难点：灵活运用两个数据结构，知道用栈和队列求解迷宫的差异 | | |
| 讲课进程和时间分配  １、问题描述：以一个m×n的长方阵表示迷宫，0和1分别表示迷宫中的通路和障碍。设计一个程序，对任意设定的迷宫，求出一条从入口到出口的通路，或得到没有通路的结论。  ２、基本要求：首先实现一个以链表作存储结构的栈类型，然后编写一个求解迷宫的非递归程序。求得的通路以三元组(i,j,d)的形式输出，其中：(i,j)指示迷宫中的一个坐标，d表示走到下一坐标的方向。如：对于下列数据的迷宫，输出的一条通路为：(1,1,1),(1,2,2),(2,2,2),(3,2,3),(3,1,2)…  ３、测试数据：迷宫的测试数据如下：左下角(1,1)为入口，右下角(8,9)为出口。   |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |   ４、实现提示：计算机解迷宫通常用的是“穷举求解”方法，即从入口出发，顺着某一个方向进行探索，若能走通，则继续前进；否则沿着原路退回，换一个方向继续探索，直至出口位置，求得一条通路。假如所有可能的通路都探索到而未能到达出口，则所设定的迷宫没有通路。可以二维数组存储迷宫数据，通常设定入口点的下标为(1,1)，出口点的下标为(n,n)。为处理方便起见，可在迷宫的四周加一圈障碍。对于迷宫中任一位置，均可约定有东、南、西、北四个方向可通。  ５、选作内容： 算术表达式求值。 | | |
| 讨论、思考题、作业：  练习：  作业：  思考题：  讨论： | | |
| 授课类型（请打√）：理论课√ 讨论课□ 实践课□ 练习课□ 其他□ | | |
| 教学方式（请打√）：传统讲授√ 讨论□ 翻转□ 示教□ 指导□ 其他□ | | |
| 教学资源（请打√）：多媒体√ 模型□ 实物□ 挂图□ 音像□ 其他□ | | |

#include<stdio.h>

#include<stdlib.h>

#define Maxsize 121

bool flag=false;//游戏状态

int Map[11][10] = {

{0,1,2,3,4,5,6,7,8,9},

{1,0,0,1,0,0,0,1,0,1},

{2,0,0,1,0,0,0,1,0,2},

{3,0,0,0,0,1,1,0,1,3},

{4,0,1,1,1,0,0,1,0,4},

{5,0,0,0,1,0,0,0,0,5},

{6,0,1,0,0,0,1,0,1,6},

{7,0,1,1,1,1,0,0,1,7},

{8,1,1,0,0,0,1,0,1,8},

{9,1,1,0,0,0,0,0,0,9},

{0,1,2,3,4,5,6,7,8,0}

};//地图

struct Box

{

int x;//横坐标

int y;//纵坐标

int d; //下步方向

};

struct LinkStack

{

Box Data;

LinkStack \*next;

};//

LinkStack \*CreatLinkStack()//初始化栈

{

LinkStack \*Top;

Top=(LinkStack\*)malloc(sizeof (LinkStack));

Top->next=NULL;

return Top;

}

LinkStack \*PushStack(LinkStack \*Top,Box b1)//进栈

{

LinkStack \*Point;

Point=(LinkStack\*)malloc(sizeof (LinkStack));

Point->next=Top->next;

Point->Data=b1;

Top->next=Point;

return Top;

}

LinkStack \*PopStack(LinkStack \*Top)//出栈

{

LinkStack \*Point;

Point=Top->next;

Top->next=Point->next;

free(Point);

return Top;

}

Box InitGame(LinkStack \*Top)//初始化游戏

{

int fx,fy,sx,sy;

Box sb,fb;

LinkStack \*Point;

printf("请输入起点的横坐标：");

scanf("%d",&sx);

printf("请输入起点的纵坐标：");

scanf("%d",&sy);

printf("请输入终点的横坐标：");

scanf("%d",&fx);

printf("请输入终点的纵坐标：");

scanf("%d",&fy);

sb.x=sx;sb.y=sy;

fb.x=fx;fb.y=fy;

PushStack(Top,sb);

return fb;

}

bool IsEmpty(LinkStack \*Top)

{

if(Top->next==NULL) return true;

return false;

}

LinkStack \*FindAnswer(LinkStack \*Top)

{

Box FinalPoint;

LinkStack \*Point;

int xi,yi;

FinalPoint=InitGame(Top);

while(!IsEmpty(Top)&&!flag)

{

int di=-1;

Point=Top->next;

while(di<4)

{

xi=Point->Data.x;

yi=Point->Data.y;

di++;

switch(di)

{

case 0:yi=yi-1;break;

case 1:xi=xi+1;break;

case 2:yi=yi+1;break;

case 3:xi=xi-1;break;

}

if(yi<0||xi<0)continue;//当坐标为负数，重置坐标结束当前循环进入下一循环。

if(Map[yi][xi]==0) break;

}

if(xi==FinalPoint.x&&yi==FinalPoint.y)//如果当前坐标为迷宫的终点，结束循环

flag=true;//找到答案后将游戏标志置为True

if(Map[yi][xi]!=0)//如果当前坐标的值不是0，重置。结束当前循环，进入下一循环

{

// if(Map[yi][xi]!=-1)

Map[yi][xi]=2;

PopStack(Top);

continue;

}

if(Map[yi][xi]==0)//如果当前坐标的值是0，将当前点进栈

{

Box bi;

bi.x=xi;bi.y=yi;bi.d=di;

Map[yi][xi]=-1;

PushStack(Top,bi);

}

}

return Top;

}

void DispStack(LinkStack \*Top)//输出栈

{

int i=0;

int count=0;

LinkStack \*Point;

Box Path[Maxsize];

//Point=Top->next;

if(IsEmpty(Top))

{

printf("该迷宫无解！");

return;

}

while(!IsEmpty(Top))

{

Path[i++]=Top->next->Data;

PopStack(Top);

}

while(i>=1)

{

i--;

printf("(%d,%d,%d)",Path[i].x,Path[i].y,Path[i-1].d);

count++;

if(i>=1) printf("->");

if(count%5==0) printf("\n");

}

}

int main()

{

LinkStack \*Top;

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

{

for(int j=0;j<10;j++)

printf("%2d ",Map[i][j]);

printf("\n");

}

Top=CreatLinkStack();

Top=FindAnswer(Top);

DispStack(Top);//输出谜底

printf("\n");

getchar();

}