

“程序设计基础”

课程设计报告

**（一） 需求和规格说明**

编写一个推箱子游戏游戏，使用图形界面。推箱子游戏的目的是用最少的步数把所有的箱子推到目标位置。

游戏规则为：

1. 游戏开始时，在游戏区域内随机的出现箱子、障碍物、小人以及箱子的目标位置。

2. 箱子只可以被推动，不可以被拉动。

3. 用户通过键盘的来控制小人上下左右的移动来推动箱子。记录用户移动 小人的次数。

4. 如果箱子被推到死角，则游戏结束。

5. 所有箱子被推到目标位置，游戏胜利。

**（二） 设计**

首先设计一个二维矩阵用用不同数字来标志不同位置的信息：

0: 空地

1: 目的地

2: 墙壁

4: 箱子

8: 玩家

16:箱子已在目的地

32:人在目的地上。

每次循环用户进行操作后，将表示地图信息的二维矩阵数字更新后输出，以此实现玩家移动的功能。二维矩阵的更新，首先通过判断玩家执行移动后要到达的地方进行分类，分类后再进行一系列的条件判断执行不同的操作，以此形成一颗条件树，算法如下：

首先通过二重循环找到玩家的位置。

1. 要到达的地方为空地：
2. 人已在目的地上：

要到达的地方变为玩家，玩家原来的位置变为目的地。

1. 人不在目的地上：

要到达的地方变为玩家，玩家原来的位置变为空地。

1. 要到达的地方为目的地：

1）人已在目的地上：

要到达的地方变为人在目的地上，玩家原来的位置变为目的地。

2）人不在目的地上：

要到达的地方变为人在目的地上，玩家原来的位置变为空地。

1. 要到达的地方为箱子：
2. 箱子被推后到达空地：
3. 人已在目的地上：

箱子到达的地方变为箱子，玩家要到达的地方变为玩家，玩家原来的位置变为目的地。

1. 人不在目的地上：

箱子到达的地方变为箱子，玩家要到达的地方变为玩家，玩家原来的位置变为空地。

1. 箱子被推后到达目的地：
2. 人已在目的地上：

箱子到达的地方变为箱子已在目的地，玩家要到达的地方变为玩家，玩家原来的位置变为目的地。

1. 人不在目的地上：

箱子到达的地方变为箱子已在目的地，玩家要到达的地方变为玩家，玩家原来的位置变为空地。

1. 要到达的地方为箱子已在的目的地
2. 人已在目的地上：
3. 箱子可推并且箱子被推后到达目的地：

箱子到达的地方变为箱子已在目的地，玩家要到达的地方变为人在目的地上，玩家原来的位置变为目的地。

1. 箱子可推并且箱子被推后到达空地：

箱子到达的地方变为箱子，玩家要到达的地方变为人在目的地上，玩家原来的位置变为目的地。

1. 人不在目的地上：
2. 箱子可推并且箱子被推后到达目的地：

箱子到达的地方变为箱子已在目的地，玩家要到达的地方变为人在目的地上，玩家原来的位置变为空地。

1. 箱子可推并且箱子被推后到达空地：

箱子到达的地方变为箱子，玩家要到达的地方变为人在目的地上，玩家原来的位置变为空地。

以上游戏的主要功能基本实现,除主要功能外本游戏还能实现悔步，重置本关和多关卡功能。

悔步：将每次的移动所要改变的值先压入栈再改变，注意最后把要到达的地方压栈，执行悔步操作时要达到的地方此时已在栈顶，通过判断栈顶的值来决定要弹栈几次。弹栈后将地图修改为原来的数据实现悔步。

重置本关：保持难度不变，将地图变为原值即实现了重置本关。

多关卡：人为设计多个地图，将其加载为当前地图实现多关卡功能。

为实现代码整洁，保护数据，本游戏设计了Map类，来实现各种功能，并保护当前游戏的数据。

**表 1 属性和方法定义**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **类名** | **成员类别** | **类型** | **成员名** | **概述** |
| Map | 属性 | int | Flag | 进程标志。  0：正常游戏  1：通过  -1：箱子推入死角 |
| stack<int> | back | 用于实现悔步 |
| int | moveCnt | 移动步数 |
| int | difficulty | 游戏难度 |
| int | VALIDM | 当前游戏地图有效的行数 |
| int | VALIDN | 当前游戏地图有效的列数 |
| int(\*)[M] | map | 当前游戏地图数据矩阵 |
| 方法 | void | loadingImg | 加载若干图片 |
| void | mapPrint | 输出地图 |
| void | mapUpdate | 更新地图数据 |
| void | mapBackUpdate | 更新悔步后的地图数据 |
| void | mapCopy | 将设计好的地图数据赋予当前游戏地图 |
| void | mapSwitch | 选择不同的地图 |
| void | setDifc | 设定游戏难度 |
| bool | isOver | 判断游戏结束 |
| int | getDifc | 获取当前游戏难度 |
| int | getVALIDM | 获取当前游戏地图有效行数 |
| int | getVALIDN | 获取当前游戏地图有效列数 |

1. **用户手册**

程序运行时，根据游戏内提示进行操作。

1. **调试及测试**

经测试，游戏当前各功能暂时正常。

1. **运行实例**



1. **进一步改进**

目前当前的游戏地图是固定最大为30\*30，而有效的的部分该地图小，会造成空占内存，可用动态二维数组来实现当前游戏地图的可变化。

1. **心得体会**

通过本次课程设计，我收获良多。程序可视化方面，通过网上自学，学会了通过EasyX将程序可视化。程序设计方面，在面对悔步功能时，我想到了调用堆栈来实现悔步，并掌握了c++中stl的部分应用。整个程序设计，加强了我对c++的熟练程度，以及代码整洁，规范性。

1. **对课程设计的建议**

可以向做课程设计的学生提供一些有益于做课程设计的软件，平台等。

**（九） 附录——源程序**

**//PushBox.cpp文件（主程序）**

#include <conio.h>

#include <Windows.h>

#include "Map.h"

#pragma comment(lib,"Winmm.lib")

extern IMAGE img[6];

extern IMAGE bgp, help;

int main(){

int difficulty;

Map gMap(0);

initgraph(360, 300); //输出开始界面

putimage(0, 0, &help);

setbkmode(TRANSPARENT);

settextcolor(BLUE);

settextstyle(50, 0, \_T("Consolas"));

outtextxy(35, 90, \_T("Move Box Game"));

settextstyle(16, 0, \_T("Consolas"));

outtextxy(100, 200, \_T("按任意键开始游戏！"));

outtextxy(85, 250, \_T("Students Id: 2019216174"));

mciSendString(\_T("open 1.mp3 alias mymusic"), NULL, 0, NULL);

mciSendString(\_T("play mymusic repeat"), NULL, 0, NULL);

system("pause");

cleardevice(); //输出选择难度界面

putimage(0, 0, &help);

settextstyle(50, 0, \_T("Consolas"));

outtextxy(20, 90, \_T("请选择游戏难度"));

settextstyle(16, 0, \_T("Consolas"));

outtextxy(50, 150, \_T("从0-5输入一个数字，难度依次递增"));

difficulty = \_getch();

while (difficulty < '0' || difficulty > '5'){

difficulty = \_getch();

}

gMap.setDifc(difficulty);

outtextxy(135, 184, \_T("游戏开始！"));

outtextxy(100, 200, \_T("按任意键开始游戏！"));

system("pause");

gMap.mapSwitch();

initgraph(20 \* gMap.getVALIDN() + 200, 30 \* gMap.getVALIDM());

gMap.mapPrint();

do{

if (gMap.Flag == 1){ //通关后初始化

while (!gMap.back.empty()){ //将悔步的栈清空

gMap.back.pop();

}

if (\_getch() != '0'){

gMap.Flag = 0;

cleardevice();

difficulty++;

gMap.setDifc(difficulty);

gMap.mapSwitch();

initgraph(20 \* gMap.getVALIDN() + 200, 30 \* gMap.getVALIDM());

gMap.mapPrint();

}

else {

break;

}

}

else if (gMap.Flag == -1){ //箱子推入死角

while (!gMap.back.empty()) {

gMap.back.pop();

}

if (\_getch() == 'r'){

gMap.Flag = 0;

cleardevice();

gMap.mapSwitch();

initgraph(20 \* gMap.getVALIDN() + 200, 30 \* gMap.getVALIDM());

gMap.mapPrint();

}

else{

break;

}

}

switch (\_getch()){ //正常游戏

case 'w': //上移

gMap.mapUpdate(-1, 0);

break;

case 's': //下移

gMap.mapUpdate(1, 0);

break;

case 'a': //左移

gMap.mapUpdate(0, -1);

break;

case 'd': //右移

gMap.mapUpdate(0, 1);

break;

case 'b': //悔步

gMap.mapBackUpdate();

break;

case 'r': //重置本关

while (!gMap.back.empty()) {

gMap.back.pop();

}

gMap.mapSwitch();

initgraph(20 \* gMap.getVALIDN() + 200, 30 \* gMap.getVALIDM());

default:

break;

}

cleardevice();

gMap.mapPrint();

} while (gMap.isOver());

cleardevice(); //输出游戏结束界面

initgraph(360, 300);

putimage(0, 0, &help);

setbkmode(TRANSPARENT);

settextcolor(BLUE);

settextstyle(50, 0, \_T("Consolas"));

outtextxy(80, 90, \_T("Game Over"));

mciSendString(\_T("stop mymusic"), NULL, 0, NULL);

mciSendString(\_T("close mymusic"), NULL, 0, NULL);

system("pause");

return 0;

}

**//Map.h文件**

#ifndef \_Map\_

#define \_Map\_

#include <graphics.h>

#include <stdlib.h>

using namespace std;

#include <stack>

#define M 30 // 定义最大的地图30\*30

using namespace std;

class Map{

public:

Map(int iDifc);

void loadingImg(); //加载若干图片

bool isOver(); //判断游戏是否结束

void mapPrint(); //输出地图;

void mapUpdate(int row, int col); //更新地图数据

void mapBackUpdate(); //更新悔步后的地图数据

void mapCopy(int\* map, int\* mapX); //将设计好的地图数据赋予当前游戏地图

void mapSwitch(); //选择不同的地图，目前总共6关，可继续添加

void setDifc(int iDifc) { difficulty = iDifc; };//设定游戏难度

int getDifc() { return difficulty; }; //获取当前游戏难度

int getVALIDM() { return VALIDM; }; //获取当前游戏地图有效的行数

int getVALIDN() { return VALIDN; }; //获取当前游戏地图有效的列数

int Flag = 0; //进程标志。0：正常游戏 1：通关 -1：箱子推入死角

stack<int> back; //用于实现悔步

private:

int moveCnt = 0; //移动步数

int difficulty = 0; //游戏难度

int VALIDM; //当前游戏地图有效的行数

int VALIDN; //当前游戏地图有效的列数

int map[M][M]; //当前游戏地图数据矩阵

};

#endif // \_Map\_

**//Map.cpp文件**

#include "Map.h"

IMAGE img[6];

IMAGE bgp, help;

Map::Map(int iDifc){

difficulty = iDifc;

loadingImg();

}

void Map::loadingImg(){ //加载若干图片

loadimage(img + 0, \_T("Image\\pic0.png"), 20, 30);

loadimage(img + 1, \_T("Image\\pic1.png"), 20, 30);

loadimage(img + 2, \_T("Image\\pic2.png"), 20, 30);

loadimage(img + 3, \_T("Image\\pic3.png"), 20, 30);

loadimage(img + 4, \_T("Image\\pic4.png"), 20, 30);

loadimage(img + 5, \_T("Image\\pic5.png"), 20, 30);

loadimage(&bgp, \_T("Image\\background.png"), 600, 900);

loadimage(&help, \_T("Image\\help.png"), 360, 420);

}

bool Map::isOver(){ //判断游戏是否结束

int row;

int col;

int num = 0;

for (row = 0; row < VALIDM; row++){

for (col = 0; col < VALIDN; col++){

if (map[row][col] == 4){

if ((map[row][col + 1] == 2 && (map[row + 1][col] == 2 || map[row - 1][col] == 2))

|| (map[row][col - 1] == 2 && (map[row + 1][col] == 2 || map[row - 1][col] == 2)))

{

outtextxy(40, 32, \_T("箱子推入死角"));

outtextxy(30, 48, \_T("是否重新开始?"));

outtextxy(40, 64, \_T("按r重新开始！"));

outtextxy(30, 80, \_T("非r键退出游戏！"));

Flag = -1;

return 1;

}

num++;

}

}

}

if (num == 0){

if (difficulty == '5'){

outtextxy(5, 32, \_T("恭喜你通过全部关卡"));

system("pause");

return 0;

}

outtextxy(40, 32, \_T("恭喜你通关"));

outtextxy(20, 48, \_T("是否进入下一关?"));

outtextxy(40, 64, \_T("按0退出游戏"));

outtextxy(20, 80, \_T("非0键进入下一关"));

Flag = 1;

}

return 1;

}

void Map::mapPrint(){ //输出地图;

int row;

int col;

putimage(0, 0, &help);

putimage(150, 0, &bgp);

setbkmode(TRANSPARENT);

settextcolor(BLUE);

settextstyle(16, 0, \_T("Consolas"));

outtextxy(40, 0, \_T("当前步数："));

char num[20];

sprintf\_s(num, "%d", moveCnt);

outtextxy(120, 0, num);

settextstyle(16, 0, \_T("Consolas"));

outtextxy(70, 98, \_T("提示"));

outtextxy(45, 114, \_T("w:向上移动"));

outtextxy(45, 130, \_T("s:向下移动"));

outtextxy(45, 146, \_T("d:向右移动"));

outtextxy(45, 162, \_T("a:向左移动"));

outtextxy(45, 178, \_T("b:悔步"));

outtextxy(45, 194, \_T("r:重新开始"));

for (row = 0; row < VALIDM; row++){

for (col = 0; col < VALIDN; col++){

switch (map[row][col]){

case 0:

break;

case 1:

putimage(col \* 20 + 150, row \* 30, &img[0]);

break;

case 2:

putimage(col \* 20 + 150, row \* 30, &img[1]);

break;

case 4:

putimage(col \* 20 + 150, row \* 30, &img[2]);

break;

case 8:

putimage(col \* 20 + 150, row \* 30, &img[3]);

break;

case 16:

putimage(col \* 20 + 150, row \* 30, &img[4]);

break;

case 32:

putimage(col \* 20 + 150, row \* 30, &img[5]);

break;

default:

break;

}

}

}

}

void Map::mapUpdate(int row, int col){ //更新地图数据

int preRow = 0, preCol = 0;

int i;

int j;

for (i = 0; i < VALIDM; i++){ //找到player当前的位置

for (j = 0; j < VALIDN; j++){

if (map[i][j] == 8 || map[i][j] == 32){

preRow = i;

preCol = j;

}

}

}

if (map[preRow + row][preCol + col] == 0){ //要到的地方为空地

if (map[preRow][preCol] == 32){

back.push(row); //将移动前的数据压入悔步栈

back.push(col);

back.push(map[preRow][preCol]);

back.push(map[preRow + row][preCol + col]);

map[preRow + row][preCol + col] = 8;

map[preRow][preCol] = 1;

moveCnt++;

}

else{

back.push(row);

back.push(col);

back.push(map[preRow][preCol]);

back.push(map[preRow + row][preCol + col]);

map[preRow + row][preCol + col] = 8;

map[preRow][preCol] = 0;

moveCnt++;

}

return;

}

if (map[preRow + row][preCol + col] == 1){ //要到的地方为目的地

if (map[preRow][preCol] == 32){

back.push(row);

back.push(col);

back.push(map[preRow][preCol]);

back.push(map[preRow + row][preCol + col]);

map[preRow + row][preCol + col] = 32;

map[preRow][preCol] = 1;

moveCnt++;

}

else{

back.push(row);

back.push(col);

back.push(map[preRow][preCol]);

back.push(map[preRow + row][preCol + col]);

map[preRow + row][preCol + col] = 32;

map[preRow][preCol] = 0;

moveCnt++;

}

return;

}

if (map[preRow + row][preCol + col] == 4){ //要到的地方为箱子

if (map[preRow + row + row][preCol + col + col] == 0){

if (map[preRow][preCol] == 32){

back.push(row);

back.push(col);

back.push(map[preRow + row + row][preCol + col + col]);

back.push(map[preRow][preCol]);

back.push(map[preRow + row][preCol + col]);

map[preRow + row + row][preCol + col + col] = 4;

map[preRow + row][preCol + col] = 8;

map[preRow][preCol] = 1;

}

else{

back.push(row);

back.push(col);

back.push(map[preRow + row + row][preCol + col + col]);

back.push(map[preRow][preCol]);

back.push(map[preRow + row][preCol + col]);

map[preRow + row + row][preCol + col + col] = 4;

map[preRow + row][preCol + col] = 8;

map[preRow][preCol] = 0;

}

moveCnt++;

}

if (map[preRow + row + row][preCol + col + col] == 1){

if (map[preRow][preCol] == 32){

back.push(row);

back.push(col);

back.push(map[preRow + row + row][preCol + col + col]);

back.push(map[preRow][preCol]);

back.push(map[preRow + row][preCol + col]);

map[preRow + row + row][preCol + col + col] = 16;

map[preRow + row][preCol + col] = 8;

map[preRow][preCol] = 1;

}

else{

back.push(row);

back.push(col);

back.push(map[preRow + row + row][preCol + col + col]);

back.push(map[preRow][preCol]);

back.push(map[preRow + row][preCol + col]);

map[preRow + row + row][preCol + col + col] = 16;

map[preRow + row][preCol + col] = 8;

map[preRow][preCol] = 0;

}

moveCnt++;

}

return;

}

if (map[preRow + row][preCol + col] == 16){ //要到的地方为箱子已在的目的地

if (map[preRow][preCol] == 32){

if (map[preRow + row + row][preCol + col + col] != 2

&& map[preRow + row + row][preCol + col + col] != 4

&& map[preRow + row + row][preCol + col + col] != 16)

{

if (map[preRow + row + row][preCol + col + col] == 1){

back.push(row);

back.push(col);

back.push(map[preRow + row + row][preCol + col + col]);

back.push(map[preRow][preCol]);

back.push(map[preRow + row][preCol + col]);

map[preRow + row + row][preCol + col + col] = 16;

map[preRow + row][preCol + col] = 32;

map[preRow][preCol] = 1;

}

if (map[preRow + row + row][preCol + col + col] == 0){

back.push(row);

back.push(col);

back.push(map[preRow + row + row][preCol + col + col]);

back.push(map[preRow][preCol]);

back.push(map[preRow + row][preCol + col]);

map[preRow + row + row][preCol + col + col] = 4;

map[preRow + row][preCol + col] = 32;

map[preRow][preCol] = 1;

}

moveCnt++;

}

}

else{

if (map[preRow + row + row][preCol + col + col] != 2

&& map[preRow + row + row][preCol + col + col] != 4

&& map[preRow + row + row][preCol + col + col] != 16)

{

if (map[preRow + row + row][preCol + col + col] == 1){

back.push(row);

back.push(col);

back.push(map[preRow + row + row][preCol + col + col]);

back.push(map[preRow][preCol]);

back.push(map[preRow + row][preCol + col]);

map[preRow + row + row][preCol + col + col] = 16;

map[preRow + row][preCol + col] = 32;

map[preRow][preCol] = 0;

}

if (map[preRow + row + row][preCol + col + col] == 0){

back.push(row);

back.push(col);

back.push(map[preRow + row + row][preCol + col + col]);

back.push(map[preRow][preCol]);

back.push(map[preRow + row][preCol + col]);

map[preRow + row + row][preCol + col + col] = 4;

map[preRow + row][preCol + col] = 32;

map[preRow][preCol] = 0;

}

moveCnt++;

}

}

return;

}

}

void Map::mapBackUpdate(){ //更新悔步后的地图数据

int i, j;

int preRow = 0, preCol = 0;

int row, col;

int prePlayer, prePos, preFarPos;

if (!back.empty())

{

for (i = 0; i < VALIDM; i++) { //找到player当前的位置

for (j = 0; j < VALIDN; j++) {

if (map[i][j] == 8 || map[i][j] == 32) {

preRow = i;

preCol = j;

}

}

}

switch (back.top()) { //通过弹栈来更新悔步后的地图

case 0:

case 1:

prePos = back.top();

back.pop();

prePlayer = back.top();

back.pop();

col = back.top() ? -back.top() : 0;

back.pop();

row = back.top() ? -back.top() : 0;

back.pop();

map[preRow][preCol] = prePos;

map[preRow + row][preCol + col] = prePlayer;

moveCnt--;

break;

case 4:

case 16:

prePos = back.top();

back.pop();

prePlayer = back.top();

back.pop();

preFarPos = back.top();

back.pop();

col = back.top() ? -back.top() : 0;

back.pop();

row = back.top() ? -back.top() : 0;

back.pop();

map[preRow][preCol] = prePos;

map[preRow + row][preCol + col] = prePlayer;

map[preRow - row][preCol - col] = preFarPos;

moveCnt--;

break;

default:

break;

}

}

return;

}

void Map::mapCopy(int\* map, int\* mapX){ //将设计好的地图数据赋予当前游戏地图

int i, j;

for (i = 0; i < VALIDM; i++){

for (j = 0; j < VALIDN; j++){

\*(map + i \* M + j) = \*(mapX + i \* VALIDN + j);

}

}

return;

}

void Map::mapSwitch(){ //选择不同的地图，目前总共6关，可继续添加

// 0: 空地

// 1: 目的地

// 2: 墙壁

// 4: 箱子

// 8: 玩家

// 16:箱子已在目的地

// 32:人在目的地上

if (difficulty == '0'){

VALIDM = 10;

VALIDN = 10;

int map0[10][10] = {

{ 2, 2, 2, 2, 2, 2, 2, 2, 2, 2 },

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

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

{ 2, 0, 0, 0, 0, 4, 0, 0, 0, 2 },

{ 2, 0, 0, 0, 0, 8, 4, 1, 0, 2 },

{ 2, 0, 0, 0, 0, 0, 4, 0, 0, 2 },

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

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

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

{ 2, 2, 2, 2, 2, 2, 2, 2, 2, 2 }

};

mapCopy((int\*)map, (int\*)map0);

}

if (difficulty == '1'){

VALIDM = 13;

VALIDN = 14;

int map1[13][14] = {

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

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

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

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

{ 0,2,0,4,0,0,0,4,0,0,0,4,0,2 },

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

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

{ 2,0,0,0,0,0,2,0,0,0,8,0,0,2 },

{ 2,0,4,0,0,0,2,0,2,2,2,2,2,2 },

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

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

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

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

};

mapCopy((int\*)map, (int\*)map1);

}

if (difficulty == '2'){

VALIDM = 7;

VALIDN = 8;

int map2[7][8] = {

{2, 2, 2, 2, 2, 2, 2, 2},

{2, 1, 0, 0, 0, 4, 1, 2},

{2, 0, 2, 4, 0, 2, 0, 2},

{2, 0, 4, 0, 8, 0, 0, 2},

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

{2, 1, 0, 0, 4, 0, 1, 2},

{2, 2, 2, 2, 2, 2, 2, 2}

};

mapCopy((int\*)map, (int\*)map2);

}

if (difficulty == '3'){

VALIDM = 7;

VALIDN = 9;

int map3[7][9] = {

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

{ 2, 2, 2, 0, 8, 0, 2, 2, 2 },

{ 2, 0, 4, 4, 4, 4, 4, 0, 2 },

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

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

{ 2, 2, 2, 2, 2, 2, 2, 2, 2 },

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

};

mapCopy((int\*)map, (int\*)map3);

}

if (difficulty == '4'){

VALIDM = 10;

VALIDN = 12;

int map4[10][12] = {

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

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

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

{2,0,4,1,1,1,1,1,2,0,2,2},

{2,0,0,2,2,1,1,1,4,0,0,2},

{2,0,0,0,0,4,2,2,4,2,0,2},

{2,0,4,2,4,0,0,0,4,0,0,2},

{2,2,0,8,0,2,2,2,4,2,0,2},

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

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

};

mapCopy((int\*)map, (int\*)map4);

}

if (difficulty == '5'){

VALIDM = 14;

VALIDN = 17;

int map5[14][17] = {

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

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

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

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

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

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

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

{0,0,0,0,0,2,0,4,4,4,0,2,2,0,0,0,0},

{0,2,2,2,2,2,0,0,4,0,4,0,2,2,2,2,2},

{2,2,0,0,0,2,4,0,4,0,0,0,2,0,0,0,2},

{2,8,0,4,0,0,4,0,0,0,0,4,0,0,4,0,2},

{2,2,2,2,2,2,0,4,4,0,4,0,2,2,2,2,2},

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

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

};

mapCopy((int\*)map, (int\*)map5);

}

moveCnt = 0;

return;

}