**综合小作业 1:迷宫寻路 程序设计**

**学号:3220100733 姓名:倪旌哲 联系方式:Tel:17805829234**

# 算法基本思想

采用回溯算法进行寻路，首先任意选定一个方向前进（代码实现中为右方），若无法继续前进则换方向前进，若四个方向均不可前进则根据上一步的方向回退一步并在上一个方向重复上述过程，若最终某一步走出后上下左右四个单元格中存在&则表示找到路径，若最终回退至$位置则说明无解。

# 项目结构

## 文件map.h map.c

该组文件中定义了以下结构体

typedef struct {

int x;

int y;

}Position\_t; 用于描述位置信息

typedef struct {

int row;

int col;

char\* map\_array;

} Map\_t; 用于描述地图数据

该组文件中包含以下函数

void Init\_Map();

用于地图的初始化和读入

int What\_Is(Position\_t pos);

返回pos处的地图要素类型

Position\_t Get\_Neigbor(Position\_t pos, int direction);

返回在pos的direction方向的Position\_t对象

void Set\_Footprint(Position\_t pos);

在pos处设置足迹

void Mark\_Wrong\_Way(Position\_t pos);

在pos处标记错误道路点

void Print\_Map();

不加处理的打印地图

void Print\_Path\_On\_Map();

打印只包含足迹标记和原地图要素的地图

## 文件pathfinder.h pathfinder.c

该组文件中包含以下函数

void Init\_Finder();

用于初始化Path\_Finder的数据，如起点位置

void Find\_Path();

通过回溯算法寻路的函数

## 文件 main.c

该文件依次调用函数进行地图数据初始化，寻路者初始化后，调用函数开始寻路并将足迹添加到地图数据中。最后调用函数打印地图，无解则输出No Solution

# 测试数据与结果

## Test1

## Test2

## Test3

# 程序代码

## map.h

/\*-------------------------------------------

\* File name: map.h

\* Author: Jingzhe Ni

\* brief: 定义了用于描述地图和位置的两个结构体，

声明了和地图操作和移动操作有关的函数

---------------------------------------------\*/

#ifndef \_\_MAP\_H\_\_

#define \_\_MAP\_H\_\_

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdlib.h>

#include <memory.h>

/\*-----------------------Macro Definition----------------------\*/

#define RELEASE //if you want to run in debug mode, please change RELEASE to DEBUG

//and you can see more detailed solving process.

//定义了各种地图要素，可以应对边缘墙体缺失的情况

#define WALL 0

#define ROAD 1

#define VOID 2

#define FINISH 3

#define START 4

#define FOOTPRINT 5

#define WRONGWAY 6

//23 - 方向 可以得到相反方向。这个特性用于回溯上一步时根据上一步的方向确定回溯方向

//23 - UP == DOWM, 23 - RIGHT == LEFT

#define UP 10

#define RIGHT 11

#define LEFT 12

#define DOWN 13

/\*-----------------------Type definition-----------------------\*/

typedef struct {

int x;

int y;

}Position\_t;

typedef struct {

int row;

int col;

char\* map\_array;

} Map\_t;

//使用单例模式，创建唯一map实例

Map\_t map;

/\*---------------Public Function Declaration------------------\*/

void Init\_Map();

int What\_Is(Position\_t pos);

Position\_t Get\_Neigbor(Position\_t pos, int direction);

void Set\_Footprint(Position\_t pos);

void Mark\_Wrong\_Way(Position\_t pos);

void Print\_Map();

void Print\_Path\_On\_Map();

#endif

## map.c

/\*-------------------------------------------

\* File name: map.c

\* Author: Jingzhe Ni

\* brief: 实现了地图输入输出，获取指定位置的地图要素，位置移动等函数

---------------------------------------------\*/

#include "map.h"

#include <stdio.h>

void Init\_Map() {

int row, col;

scanf("%d %d", &row, &col);

getchar();

map.row = row;

map.col = col;

map.map\_array = (char\*)malloc(map.row \* map.col \* sizeof(char));

memset(map.map\_array, ' ', map.row \* map.col \* sizeof(char));

for (int i = 0; i < map.row; i++) {

for (int j = 0; j < map.col; j++)

scanf("%c", &(map.map\_array[i \* map.col + j]));

getchar();

}

//Print\_Map();

}

int What\_Is(Position\_t pos) {

if (pos.y >= map.row || pos.y < 0 || pos.x >= map.col || pos.x < 0)

return VOID;

switch (map.map\_array[pos.y \* map.col + pos.x]) {

case '$': return START;

case ' ': return ROAD;

case '#': return WALL;

case '&': return FINISH;

case '\*': return FOOTPRINT;

case 'x': return WRONGWAY;

}

}

Position\_t Get\_Neigbor(Position\_t pos, int direction) {

Position\_t temp\_pos = pos;

switch (direction) {

case LEFT: {

temp\_pos.x -= 1; break;

}

case RIGHT: {

temp\_pos.x += 1; break;

}

case UP: {

temp\_pos.y -= 1; break;

}

case DOWN: {

temp\_pos.y += 1; break;

}

}

return temp\_pos;

}

// x = 2 y = 2 in 5x5 array, index is 12 y\*col+x

void Set\_Footprint(Position\_t pos) {

map.map\_array[pos.y \* map.col + pos.x] = '\*';

}

void Mark\_Wrong\_Way(Position\_t pos) {

map.map\_array[pos.y \* map.col + pos.x] = 'x';

}

void Print\_Path\_On\_Map() {

printf("\nSolution:\n");

for (int i = 0; i < map.col \* map.row; i++) {

printf("%c", map.map\_array[i] == 'x' ? ' ' : map.map\_array[i]);

if ((i+1) % map.col == 0)

printf("\n");

}

//printf("%c", map.map\_array[map.col \* map.row - 1]);

return;

}

#ifdef DEBUG

void Print\_Map() {

printf("\n\n");

for (int i = 0; i < map.col \* map.row; i++) {

printf("%c", map.map\_array[i]);

if ((i+1) % map.col == 0)

printf("\n");

}

//printf("%c", map.map\_array[map.col \* map.row - 1]);

return;

}

#endif

## pathfinder.h

/\*-------------------------------------------

\* File name: pathfinder.h

\* Author: Jingzhe Ni

\* brief: 声明了和回溯寻路相关的函数

---------------------------------------------\*/

#ifndef \_\_PATHFINDER\_H\_\_

#define \_\_PATHFINDER\_H\_\_

#include "map.h"

//单例模式，创建唯一当前位置实例

Position\_t current\_position;

void Init\_Finder();

void Find\_Path();

#endif

## pathfinder.c

/\*-------------------------------------------

\* File name: pathfinder.c

\* Author: Jingzhe Ni

\* brief: 实现了初始化Path\_Finder的函数和回溯寻路功能的函数

---------------------------------------------\*/

#include "pathfinder.h"

#include <stdlib.h>

#include <stdio.h>

extern Position\_t current\_position;

extern Map\_t map;

int found\_solution = 0;

int last\_direction = 0;

#ifdef DEBUG

int depth = 1;

#endif

void Init\_Finder() {

current\_position.x = 0;

current\_position.y = 0;

//将当前位置设置为起点

for (int i = 0; i < map.col \* map.row; i++) {

if (map.map\_array[i] == '$') {

current\_position.x = i % map.col;

current\_position.y = i / map.col;

}

}

}

void Find\_Path() {

#ifdef DEBUG

depth++;

printf("\nReached %d layers\n", depth);

#endif

for (int direction = UP; direction <= DOWN; direction++)

if (What\_Is(Get\_Neigbor(current\_position, direction)) == FINISH) {

found\_solution = 1;

return;

}

int last\_direction\_ = last\_direction;

if (What\_Is(Get\_Neigbor(current\_position, RIGHT)) == ROAD && !found\_solution) {

current\_position = Get\_Neigbor(current\_position, RIGHT);

Set\_Footprint(current\_position);

last\_direction = RIGHT;

#ifdef DEBUG

Print\_Map();

#endif

Find\_Path(map);

}

if (What\_Is(Get\_Neigbor(current\_position, DOWN)) == ROAD && !found\_solution) {

current\_position = Get\_Neigbor(current\_position, DOWN);

Set\_Footprint(current\_position);

#ifdef DEBUG

Print\_Map();

#endif

last\_direction = DOWN;

Find\_Path(map);

}

if (What\_Is(Get\_Neigbor(current\_position, UP)) == ROAD && !found\_solution) {

current\_position = Get\_Neigbor(current\_position, UP);

Set\_Footprint(current\_position);

#ifdef DEBUG

Print\_Map();

#endif

last\_direction = UP;

Find\_Path(map);

}

if (What\_Is(Get\_Neigbor(current\_position, LEFT)) == ROAD && !found\_solution) {

current\_position = Get\_Neigbor(current\_position, LEFT);

Set\_Footprint(current\_position);

#ifdef DEBUG

Print\_Map();

#endif

last\_direction = LEFT;

Find\_Path(map);

}

if (found\_solution)

return;

if (What\_Is(current\_position) == START && last\_direction\_ != 0 && !found\_solution) {

#ifdef DEBUG

printf("Step back to Start, no solution!\n");

#endif

return;

}

if (last\_direction\_ != 0) {

Mark\_Wrong\_Way(current\_position);

current\_position = Get\_Neigbor(current\_position, 23-last\_direction\_);

#ifdef DEBUG

printf("Marked one place as wrong way!\n");

Print\_Map();

#endif

return;

}

return;

}

## main.c

#include "pathfinder.h"

#include "map.h"

#include <stdio.h>

extern int found\_solution;

int main() {

Init\_Map();

Init\_Finder();

Find\_Path();

if (found\_solution)

Print\_Path\_On\_Map();

else

printf("No Solution");

return 0;

}