< MAZE 구현 과제 - 2022113672 황지현 >

<소스코드>

/\*#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#define MAX\_STACK\_SIZE 100

typedef struct { //해당 위치에서 방향을 검색하는 구조체 선언

short int vert;//수직

short int horiz;//수평

}offsets;

offsets move[4] = { {-1,0},{0,1},{1,0},{0,-1} };//북동남서 표시

typedef struct { //입구에서 출구까지 가는 길을 스택에 저장하기 위한 구조체.

short int row;//행

short int col;//열

short int dir;//길을 갔다가 다시 돌아올 경우 dir에 저장한 값부터 방향을 탐색하게 됨.

}element;

element stack[MAX\_STACK\_SIZE] = { 0 }; //출구로 가기위한 길을 표시할 스택.

int\*\* maze = 0; //maze를 동적할당함.

int strow, stcol, endrow, endcol;//입구좌표 행,열 / 출구좌표 행,열

int mark[100][100] = { 0 };//한 번 간길은 mark에 1로 저장함.

int resultrow, resultcol;//findexit에서 maze를 출력해주기 위해 maze row,col값을 저장해놓고 함수 안에서 사용함.

void setmaze(FILE\* fp); //maze와 mark 정보를 넣어주는 함수.

void findexit();//maze 내에서 길을 찾아주고 이를 바탕으로 출력함.

void push(int\* top, element position);//스택에 출구로 가기위한 길을 넣어주는 push함수.

element pop(int\* top);//갈곳이 없을 경우 다시 되돌아가기 위해 pop해줌.

void main() {

FILE\* fp = fopen("maze1.txt", "r");

FILE\* fp1 = fopen("maze2.txt", "r");

FILE\* fp2 = fopen("maze3.txt", "r");

FILE\* fp3 = fopen("maze4.txt", "r");

setmaze(fp);

findexit();

fclose(fp);

setmaze(fp1);

findexit();

fclose(fp1);

setmaze(fp2);

findexit();

fclose(fp2);

setmaze(fp3);

findexit();

fclose(fp3);

//함수 수행 후 maze를 free함.

for (int i = 0;i < resultrow + 2;i++) {

free(maze[i]);

}

free(maze);

}

void setmaze(FILE\* fp) {

int data = 0; //maze의 각 요소

int i, j; //변수

int row, col; //행,열

fscanf(fp, "%d %d", &row, &col);//maze의 행과 열의 데이터 받기.

resultcol = col; resultrow = row;//findexit에서 사용하기 위해 값 넣어줌.

for (int i = 0;i < row;i++) {//mark함수 초기화

for (int j = 0;j < col;j++) mark[i][j] = 0;

}

maze = (int\*\*)calloc((row + 2), sizeof(int\*)); //maze에 1로 테두리를 만들어 줌.

for (int i = 0;i < row + 2;i++) {

maze[i] = (int\*)calloc((col + 2), sizeof(int));

for (int j = 0;j < col + 2;j++) maze[i][j] = 1;//다 1로 선언

}

printf("MAZE\n");

for (int i = 1;i < row + 1;i++) {

for (int j = 1;j < col + 1;j++) {

fscanf(fp, "%d", &data);//row+2, col+2크기로 모두 1로 선언된 maze 이차원 배열에 fscanf를 통해 txt파일에서 data를 읽어 배열에 저장함.

maze[i][j] = data;

printf("%3d", maze[i][j]);

}

printf("\n");

}

fscanf(fp, "%d %d %d %d", &strow, &stcol, &endrow, &endcol);//입구좌표와 출구좌표 받기

maze[strow + 1][stcol + 1] = 10; //나중에 출력시 S로 출력.

maze[endrow + 1][endcol + 1] = 11; //나중에 출력시 F로 출력.

}

void findexit() {

int i, row, col, next\_row, next\_col, dir;//현재좌표:row,col 다음좌표가 next\_row,next\_col

int found = 0;//길이 있는지 없는지 표시

element position;//스택에 넣어줄 좌표

mark[strow][stcol] = 1;//mark에 입구좌표를 1로 선언.

int top = 0;//스택의 맨윗 값

stack[0].row = strow + 1, stack[0].col = stcol + 1, stack[0].dir = 0;//stack[0]에 입구좌표를 넣어줌.

while (top > -1 && !found) {

position = pop(&top);//처음에 element positon에 stack[0]값을 대입해줌. 그 후로는 더이상 갈 길이 없을 때 되돌아가게함.

row = position.row; col = position.col; dir = position.dir;//현재좌표값 대입.

while (dir < 4 && !found) { //4방향 찾기

next\_row = row + move[dir].vert; //현재 좌표기준으로 시계방향으로 돌아가고, 다음 좌표값.

next\_col = col + move[dir].horiz;

if (maze[next\_row][next\_col] == 11) {//출구좌표를 만났을 경우.

position.row = row; position.col = col; position.dir = ++dir;//현재좌표값 position에 넣기.

push(&top, position);//스택에 push

found = 1;//길 찾음 표시(while문 빠져나옴)

}

else if (maze[next\_row][next\_col] == 0 && mark[next\_row][next\_col] == 0) {//간적없는 길이고 maze도 갈수있는길일때.

mark[next\_row][next\_col] = 1;//갔었던 길 표시

position.row = row; position.col = col; position.dir = ++dir;//현재좌표값 position값에 대입. 다시 만났을 때 다음 방향부터 검색하게 하기위해 ++dir.

push(&top, position);//스택에 push

row = next\_row; //미리대입해놨던 다음좌표값 대입.

col = next\_col;

dir = 0;//방향 초기화

}

else ++dir;//검사한 방향이 1일 경우 방향만 +1.

}

}

if (found) {//길이 있을 경우

for (i = 1;i < top + 1;i++) maze[stack[i].row][stack[i].col] = 12;//스택에 저장해놨던 길을 maze배열에서 12로 바꿈. 나중에 출력시 X로 출력.

printf("The path is:\n");

for (i = 1;i < resultrow + 1;i++) { //테두리 빼고 출력해야하므로 1~resultrow+1까지 출력

for (int j = 1;j < resultcol + 1;j++) { //테두리 빼고 출력해야하므로 1~resultcol+1까지 출력

if (maze[i][j] == 10) printf(" S");

else if (maze[i][j] == 11) printf(" F");

else if (maze[i][j] == 12) printf(" X");

else printf("%3d", maze[i][j]);

}

printf("\n");

}

printf("\n");

}

else {//길이 없을 경우

printf("No path!\n");

for (int i = 1;i < resultrow + 1;i++) {//테두리 빼고 출력해야하므로 1~resultrow+1까지 출력

for (int j = 1;j < resultcol + 1;j++) {//테두리 빼고 출력해야하므로 1~resultcol+1까지 출력

if (maze[i][j] == 10) printf(" S");

else if (maze[i][j] == 11) printf(" F");

else printf("%3d", maze[i][j]);

}

printf("\n");

}

printf("\n");

}

}

void push(int\* top, element position) {//position값 push

(\*top)++;

stack[\*top].row = position.row;

stack[\*top].col = position.col;

stack[\*top].dir = position.dir;

}

element pop(int\* top) {//pop해줌.

element result;

if (\*top < 0) printf("stack is empty\n");

else {

result = stack[\*top];

(\*top)--;

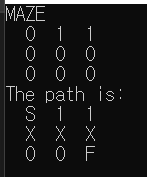
}

return result;

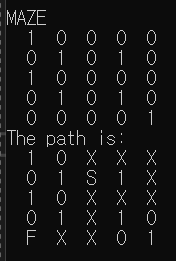
}\*/

<실행결과>

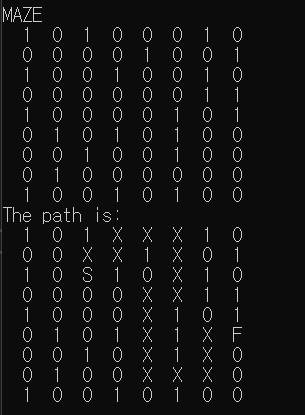
-maze1.txt



-maze2.txt



-maze3.txt



-maze4.txt

