CSED101. Programming & Problem solving

Fall, 2016

Programming Assignment #4

(70 points)

방지수 (jisus19@postech.ac.kr)

- ***Due: 2016.12.01 23:59***
- **Development Environment:** Windows Visual Studio 2015
- **제출물**
  - **C Code files (assn4.c)**
    - 프로그램의 소스 코드를 이해하기 쉽도록 반드시 주석을 붙인다.
  - **보고서 파일 (.doc(x) or .hwp) 예) assn4.doc(x) 또는 assn4.hwp**
    - AssnReadMe.pdf 를 참조하여 작성한다.
    - 각각의 기능에 대한 프로그램 실행 화면을 캡쳐하여 보고서에 포함시키고 간단히 설명한다.
  - 소스코드와 보고서 파일을 LMS를 이용하여 제출한다.

- **주의사항**
  - 문제에 해당하는 요구사항을 반드시 지킬 것.
  - 컴파일 & 실행이 안되면 무조건 0점 처리된다.
  - **하루 late시 20%가 감점되며, 3일 이상 지나면 받지 않는다. (0점 처리)**
  - 부정행위에 관한 규정은 POSTECH 전자컴퓨터공학부 학부위원회의 'POSTECH 전자컴퓨터공학부 부정행위 정의'를 따른다. (LMS의 과목 공지사항의 제목 [document about cheating]의 첨부파일인 disciplinary.pdf를 참조할 것.)
  - 추가 기능 구현에 대한 추가 점수는 없습니다.
  - Linked list로 구현하지 않는다.
  - 과제를 작성하는데 있어서 전역변수를 선언하여 사용할 수 없다.

■ Problem: 양과 늑대 게임

(목적)

이번 과제를 통하여 파일 입출력 및 포인터 사용법을 익힌다.

(문제)

주어진 맵에서 양이 움직이는 늑대를 피해 맵에 존재하는 모든 꽃을 먹고 목적지까지 도착하는 프로그램을 작성한다.

(문제 정의)

- 파일로부터 입력 받은 map에서 명령어를 사용해 이동하여 map에 있는 모든 꽃을 먹고 목적지까지 도달한다.
- map은 N x M 행렬로 구성된다. 양의 시작점의 좌표는 (0, 0)이며 목적지의 좌표는 (N-1, M-1)으로 고정된다.
- 양은 동서남북으로 움직일 수 있는 명령어를 입력 받아 움직인다.
- 각 명령어 step 이후에 늑대가 움직이며, 잘못된 명령어 입력도 한 step으로 처리하여 늑대가 움직인다.
- 생명은 5개부터 시작한다.

(설명)

게임을 위해서 input 파일과 output 파일의 입력이 필요하다.
우선 프로그램을 실행시키면 아래와 같이 output 파일 이름을 입력 받는다. 파일이름에는 공백이 없으며 10자를 넘지 않는다고 가정한다.

```
Input the out file name:
```

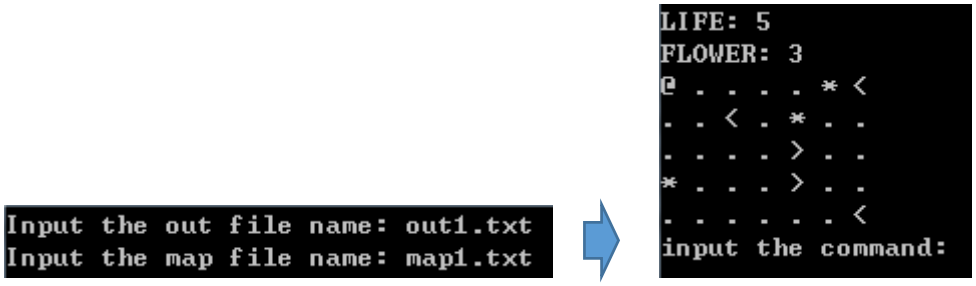
그 후 아래와 같이 맵을 생성하기 위해 읽어야 할 파일명을 요구한다. 파일이름에는 공백이 없으며 10자를 넘지 않는다고 가정한다.

```
Input the map file name:
```

존재하지 않는 파일명 wrong.txt 를 입력한 경우에는 아래와 같은 에러 메시지를 출력 후, 프로그램을 종료한다.

```
Could not open wrong.txt file
```

아래와 같이 파일이름으로 map1.txt 를 입력한 후 엔터키를 누르면, 입력 파일을 읽어서 게임을 위한 맵의 초기상태를 출력한다. 화면이 지워진 후, 맵의 초기 상태를 출력하도록 한다. (화면 지우기 기능은 뒤의 힌트를 참조)



- **map 출력 양식**

다음의 N x M 행렬은 주어진 map을 나타낸다. 양의 시작점의 좌표는 (0, 0)으로 고정되고, 목적지의 좌표는 (N-1, M-1)로 고정된다.

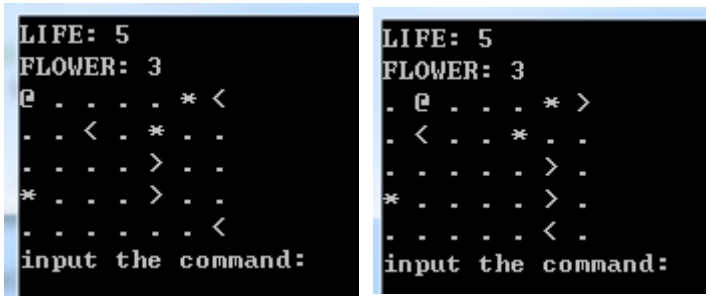
@ : 양의 현재 위치

> : 오른쪽으로 움직이는 늑대의 위치 (dir == 'R')

< : 왼쪽으로 움직이는 늑대의 위치 (dir == 'L')

* : 꽃의 위치 (변하지 않음)

. : 그 외의 공간



map 출력은 매번 화면을 지운 후 출력한다.

- **출력 파일의 양식**

미리 입력 받은 이름의 파일에 map을 출력한다.

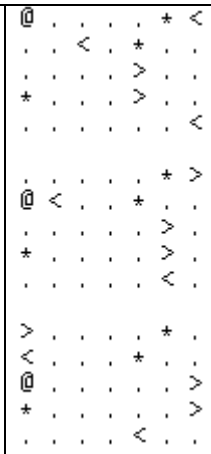
매 step의 map은 enter로 구분한다.

(예) out1.txt

```
@ . . . . * <
. . < . * . .
. . . . > . .
* . . . > . .
. . . . . . <

. . . . . * >
@ < . . * . .
. . . . . > .
* . . . . > .
. . . . . < .

> . . . . * .
< . . . * . .
@ . . . . . >
* . . . . . >
. . . . < . .
```



출력 함수 참조 (map 출력시, 아래의 출력 함수의 printf() 및 fprintf() 형식을 사용할 것!)

```
void print_map(FILE* outfile, char** map, int map_row, int map_col) {
        int i, j;

        for(i = 0; i < map_row; i++){
                for(j = 0; j < map_col; j++){
                        printf("%c ", map[i][j]);
                        fprintf(outfile, "%c ", map[i][j]);
                }
                printf("\n");
                fprintf(outfile, "\n");
        }
        fprintf(outfile, "\n");
}
```

- **입력 파일의 양식**

map 및 늑대와 꽃의 위치는 파일에 정해진 양식을 따른다. 아래는 위의 5 x 7 행렬에 대한 파일의 예이다.

※ 뒤에 있는 문제 힌트 참조 (POSITION 구조체 정의)

map1.txt
#MAP 5 7 #WOLF 5 0 6 L 1 2 L 2 4 R 3 4 R 4 6 L #FLOWER 3 0 5 1 4 3 0

#MAP의 다음 행에는 map size가 (N x M) 형태로 주어진다.
#WOLF의 다음 행에는 늑대의 수가 주어지고, 그 다음 각 행마다 각 늑대의 좌표 및 방향이 주어진다. 방향 L은 왼쪽, R은 오른쪽을 나타내며 게임시작 시 늑대의 이동방향을 나타낸다.
#FLOWER의 다음 행에는 꽃의 수가 주어지고, 그 다음 각 행마다 각 꽃의 좌표가 주어진다.

- **가정 1**: 입력 파일이 위의 정해진 양식을 벗어나는 경우는 없다.
- **가정 2**: #MAP, #WOLF, #FLOWER 구문의 순서는 정해져 있지 않다.
- **가정 3**: 각 행에 늑대는 한 마리 이상 존재하지 않는다.
- **가정 4**: 각 좌표에는 하나의 늑대나 꽃만 존재한다.
- **가정 5**: map의 시작 지점에는 늑대와 꽃이 존재하지 않는다.

파일을 읽어 맵을 생성하면 게임을 위해서 "input the command: "라는 구문이 출력되며 사용자 입력을 위해 기다린다.

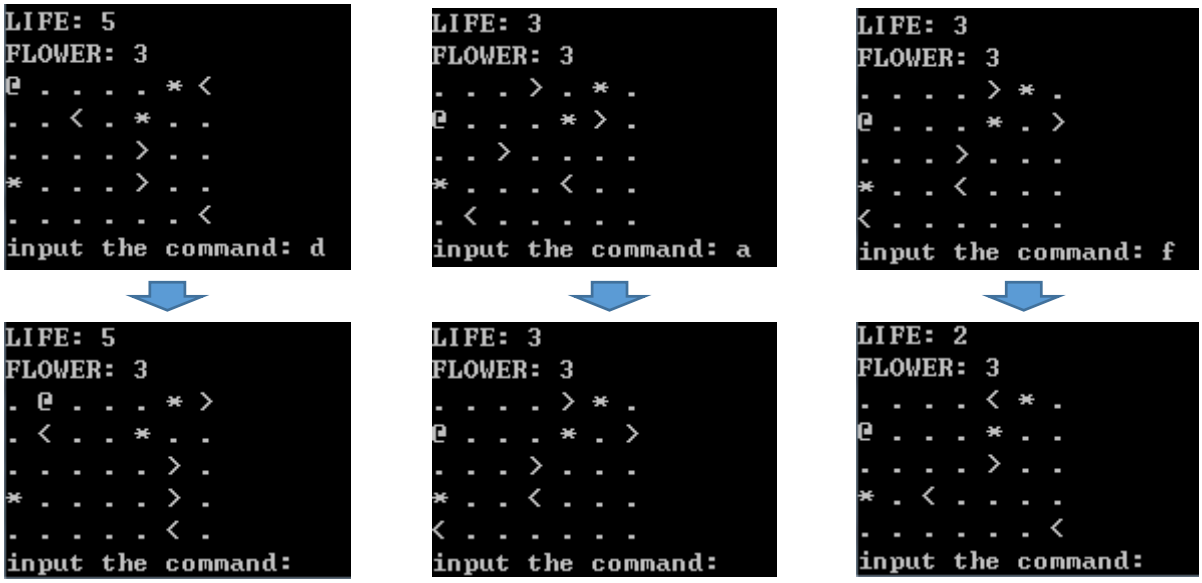
※ 문자열 비교는 **string.h**에 선언되어 있는 strcmp() 함수를 사용할 수 있다.

- **매 Step마다 해야 하는 일: 명령어 처리, 늑대 이동**

프로그램은 현재 생명과 map을 출력하고 매 step마다 사용자에게 명령어를 입력 받는다. 매 step 이후 모든 늑대는 자신의 이동방향에 따라 이동한다. 사용자는 ***w, a, s, d*** 의 명령어(소문자)를 사용할 수 있다. 사용자 명령 입력은 **scanf** 함수를 통해 받도록 한다.

- **양의 위치 이동 명령어 (w, a, s, d)**

w(위), a(왼쪽), s(아래), d(오른쪽) 명령어가 들어왔을 때 현재 양의 위치를 해당 방향으로 이동시킨다. 아래 예제 (1)은 d(오른쪽) 명령어를 입력한 경우로, 양의 위치가 (0, 0)에서 (0, 1)으로 이동한 것을 볼 수 있다.
양은 map 끝에서는 더 이상 이동할 수 없다. 아래 예제 (2)에서처럼 현재 양의 위치가 왼쪽으로 더 이상 이동할 수 없을 때, 왼쪽 이동 명령어 a를 입력하게 되면 양의 위치는 변하지 않는다. 그렇지만 늑대들은 계속 움직인다.



(1) 양의 위치 이동 예제1 **(2) 양의 위치 이동 예제2** **(3) 없는 명령어 입력**

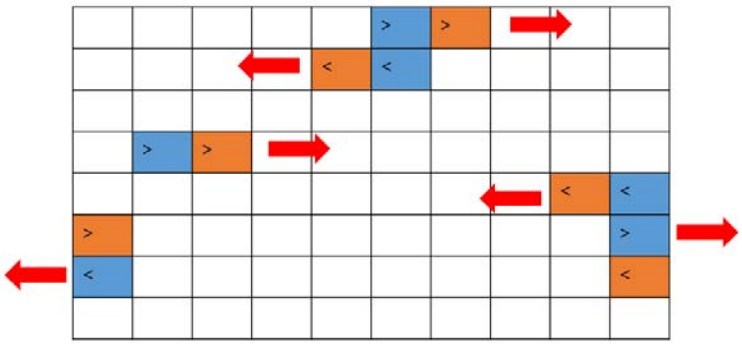
위의 예제 (3)에서처럼, 잘못 입력된 명령어는 예외 처리 하지 않는다. 즉, 양의 위치 변경은 없지만 step은 그대로 진행되어 늑대들은 이동한다.

아래의 예제처럼, 양이 꽃을 만나면 해당 위치의 꽃은 사라지고 양이 먹어야 할 꽃의 개수가 하나 감소한다.

- **늑대들의 이동**

모든 늑대들은 명령어 입력 이후에 움직인다. 각 행에 늑대는 하나 이상 존재하지 않기 때문에 늑대끼리 만나는 경우는 고려하지 않는다.

1. 늑대들은 각 이동방향에 따라 'L'은 왼쪽으로 한 칸, 'R'은 오른쪽으로 한 칸 움직인다. 각 늑대는 map 가장자리에서 이동할 때 그 row의 첫/끝에서 시작한다.



파란색 셀: step(t)에서의 늑대, 주황색 셀: step(t+1)에서의 늑대

2. 늑대들이 꽃을 만났을 때 그 자리에서 방향을 바꾼다.

*	<	
>	*	
<		*

step(t)

*	>	
<	*	
>		*

step(t+1)

- **양이 늑대를 만났을 때 처리**

명령어를 입력 한 후, 양이 이동한 위치와 늑대의 이동 위치가 같을 때(양과 늑대가 만난 경우), 생명이 하나 줄어들고, 만나게 된 늑대는 사라진다.

```
LIFE: 5
FLOWER: 3
. . . > . * .
. . . . * > .
. . > . . . .
* . @ . < . .
. < . . . . .
input the command: d
```

```
LIFE: 4
FLOWER: 3
. . . . > * .
. . . . * . >
. . . > . . .
* . . @ . . .
< . . . . . .
input the command: d
```

```
LIFE: 4
FLOWER: 3
. . . . < * .
> . . . * . .
. . . . > . .
* . . . @ . .
. . . . . . <
input the command:
```

1. 스쳐 지나가기

양의 위치가 (2, 2)에서 (2, 1)로 옮기고, 그 후 늑대가 (2, 1)에서 (2, 2)으로 이동했을 때 생명은 줄지 않는다. (늑대 이동 후에 현재 양의 위치에 늑대가 있는 지를 확인)

```
LIFE: 4
FLOWER: 3
. . . . > * .
. . . . * . >
. > @ . . . .
* . . . . . .
. . < . . . .
input the command: a
```

```
LIFE: 4
FLOWER: 3
. . . . < * .
> . . . * . .
. @ > . . . .
* . . . . . .
. < . . . . .
input the command:
```

2. 양과 늑대가 꽃을 사이에 두고 만났을 때

아래 그림과 같이 양(1,0)과 늑대(1,2)가 꽃(1,1)을 사이에 두고 있고, 양이 d 명령어를 입력 받아 (1,1)로 이동하면 꽃을 먹은 것으로 간주하고, 그 이후에 늑대가 (1,1)로 이동하기 때문에 만난 늑대는 사라지고 생명이 하나 줄어든다.

@ →	*	<

step(t)
LIFE: 5
FLOWER: 1

	@	

step(t+1)
LIFE: 4
FLOWER: 0

```
LIFE: 4
FLOWER: 3
. < . . . * .
. . . > * . .
. . . . @ > .
* . . . . . .
. . . . . < .
input the command: w
```

```
LIFE: 3
FLOWER: 2
< . . . . * .
. . . . @ . .
. . . . . . >
* . . . . . .
. . . . < . .
input the command:
```

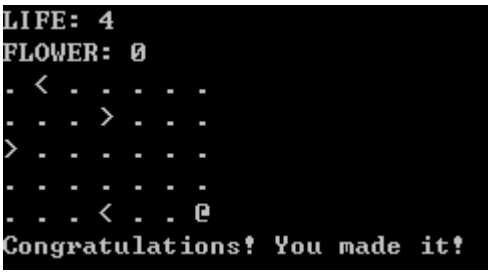
- **프로그램 종료**

양이 모든 꽃을 다 먹고 목적지에 도착했을 때, 혹은 생명이 0 이하일 경우에 프로그램을 종료한다. 프로그램을 종료할 때, 동적 할당된 모든 메모리를 반드시 해제하고 프로그램을 종료한다.

(1) 목적지에 도착 – 맵에 꽃이 남아 있는 경우

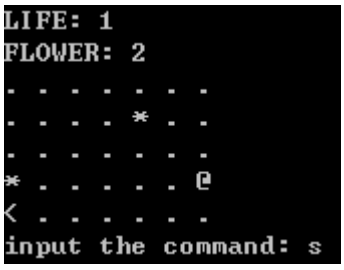
양이 목적지에 도착은 했지만, 맵에 남아 있는 꽃을 다 먹지 못했으므로 게임이 계속 진행됨을 볼 수 있다.

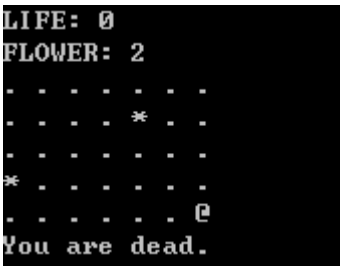
(2) 목적지에 도착 – 맵에 있는 모든 꽃을 다 먹은 경우



양이 맵에 있는 꽃을 다 먹고 목적지에 도착한 경우의 예제로 위의 예시처럼 메시지를 출력하고 프로그램을 종료한다.

(3) 게임 도중에 생명이 0 이하가 된 경우





양이 게임 도중에 주어진 생명이 0 이하가 된 경우로 위의 예시처럼 메시지를 출력하고 프로그램을 종료한다.

(주의사항)

- 출력 파일의 양식은 반드시 출력 예와 같아야 한다.
- 프로그램은 최소한 첨부된 sample data (map1.txt, map2.txt, map3.txt)에 대해서 오류 없이 실행되어야 한다.
  (각 map*.txt에 input*.txt의 명령어가 주어졌을 때 out*.txt의 출력물과 같도록 한다.)
- 동적할당 받은 메모리는 프로그램 종료시 free 시키도록 한다.

(문제 힌트)

■ **위치 및 방향을 표현하는 구조체**

- 위치 및 방향 정보는 구조체의 변수를 이용하여 표현한다.
- 방향 정보는 늑대에게만 유효하다.

```
typedef struct{
	int row;   // x 좌표
	int col;   // y 좌표
	char dir;  // 이동 방향 정보
}POSITION;
```

■ **늑대 위치 지정을 위한 구조체와 동적 할당 방법**

- 늑대의 위치는 구조체 변수를 이용하여 표현한다.
- 늑대의 수는 정해져 있지 않고 입력을 통해 정해지기 때문에 동적 할당을 사용한다.
- 늑대 위치 저장 배열의 각 원소들은 POSITION 구조체이다. 즉, 늑대 n마리에 대해서 구조체 POSITION* 형으로 관리한다.

```
// 늑대 위치 allocation
POSITION *wolves;
wolves = (POSITION*) calloc(n, sizeof(POSITION));
```

■ **map을 생성하기 위한 2차원 배열 동적 할당 방법**

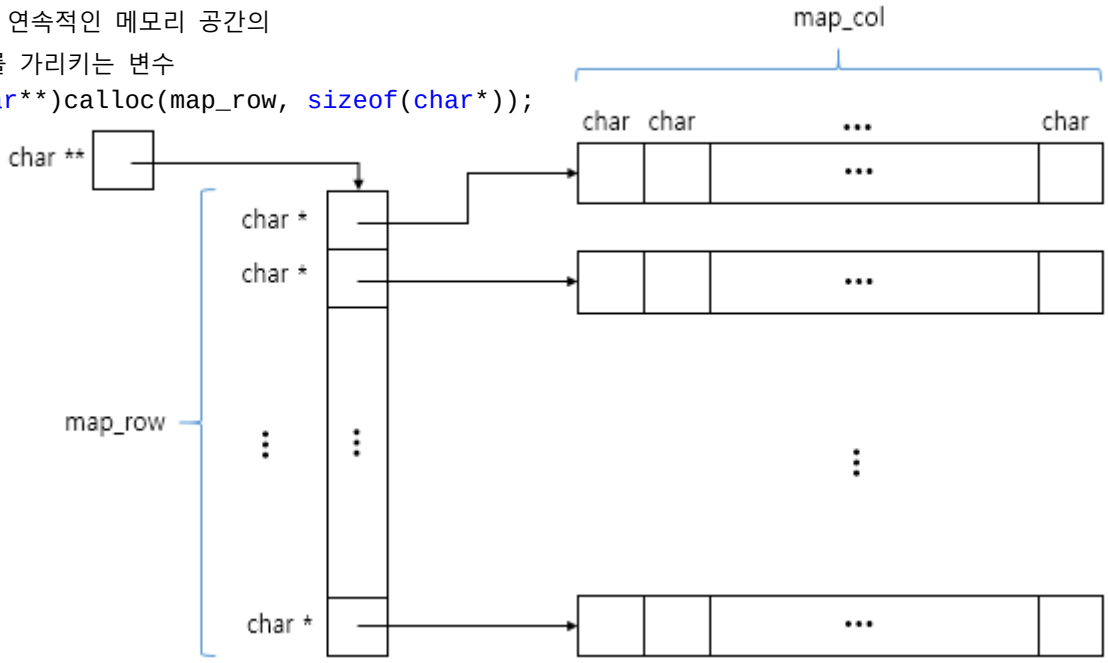
- input 파일은 N(map_row)과 M(map_col)이 주어지고, 양의 위치는 '@'로, 늑대의 위치는 '<' 또는 '>'로, 꽃은 '*'로, 나머지는 '.'로 표시하게 되어 있다. map은 char를 원소로 가지는 2차원 동적 배열로 구현하도록 한다.
- map size는 변경될 수 있기 때문에 동적할당을 이용한다. char **map은 2차원 배열을 동적할당 받기 위한 포인터 변수이다.

```
// map allocation
char **map;
map = (char**) calloc (map_row, sizeof(char*));
for(i = 0; i < map_row; i++)
	*(map + i) = (char*)calloc(map_col, sizeof(char)) ;
```

- **2차원 배열 동적 할당에 대한 모식도**

1. char* 들의 연속적인 메모리 공간의
첫번째 주소를 가리키는 변수

```
map = (char**)calloc(map_row, sizeof(char*));
```



3. char들의 연속적인 메모리 공간

2. 각각의 char*는 char 들의 연속적인
메모리 공간의 첫번째 주소를 가리키는 변수

```
for(i=0; i < map_row; i++)
   *(map+i) = (char *)calloc(map_col, sizeof(char));
```

■ **화면 지우기**

<stdlib.h>를 포함한 뒤, system 함수를 사용하여 현재 콘솔 창에 출력된 내용을 지운다.

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
        printf("HELLO WORLD\n");
        system("cls");
        return 0;
}
```

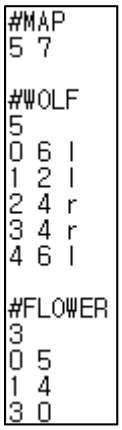
■ **제공된 파일 (data.zip) 설명**

문제와 같이 제공되는 파일 data.zip 의 압축을 풀면 input*.txt, map*.txt, out*.txt 을 확인할 수 있다. 제공되는 파일 (input*.txt, map*.txt, out*.txt)은 프로그램이 잘 동작하는 지 확인하기 위해 사용할 수 있다.

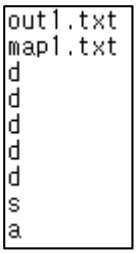
map*.txt: map 정보가 저장된 파일

input*.txt: 프로그램을 실행하기 위하여 필요한 사용자 입력이 순서대로 저장되어 있는 파일
아래의 input1.txt 는 output 파일 이름으로 out1.txt 를 입력 받고, input 파일 이름으로 map1.txt 를 입력 받은 후, d(오른쪽), d(오른쪽), … 사용자 입력이 순서대로 저장되어 있다.

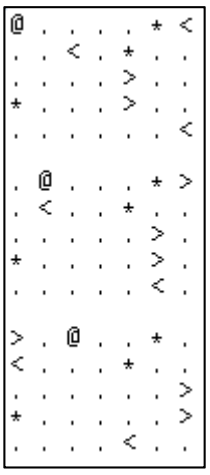
out*.txt: map*.txt 파일을 input*.txt 명령어로 실행한 후 출력되는 파일



map1.txt



input1.txt (후략)



out1.txt (후략)

프로그램을 실행 시킨 후, input1.txt 파일에 있는 순서대로 입력을 하는 경우, map1.txt 를 읽어서 생성된 맵에서 게임이 진행되며, step 마다 맵 상태 출력은 out1.txt 와 같다.

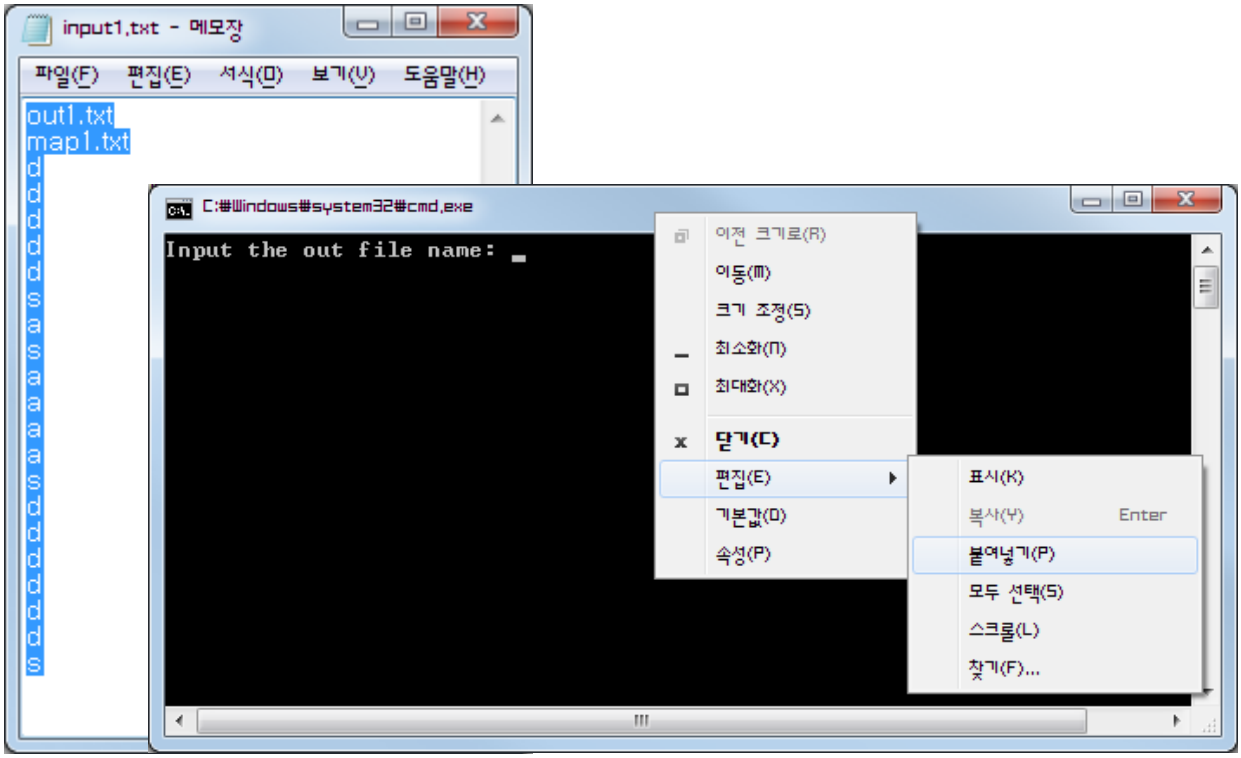
프로그램을 실행 시킨 후, input2.txt 파일에 있는 순서대로 입력을 하는 경우, map2.txt 를 읽어서 생성된 맵에서 게임이 진행되며, step 마다 맵 상태 출력은 out2.txt 와 같다.

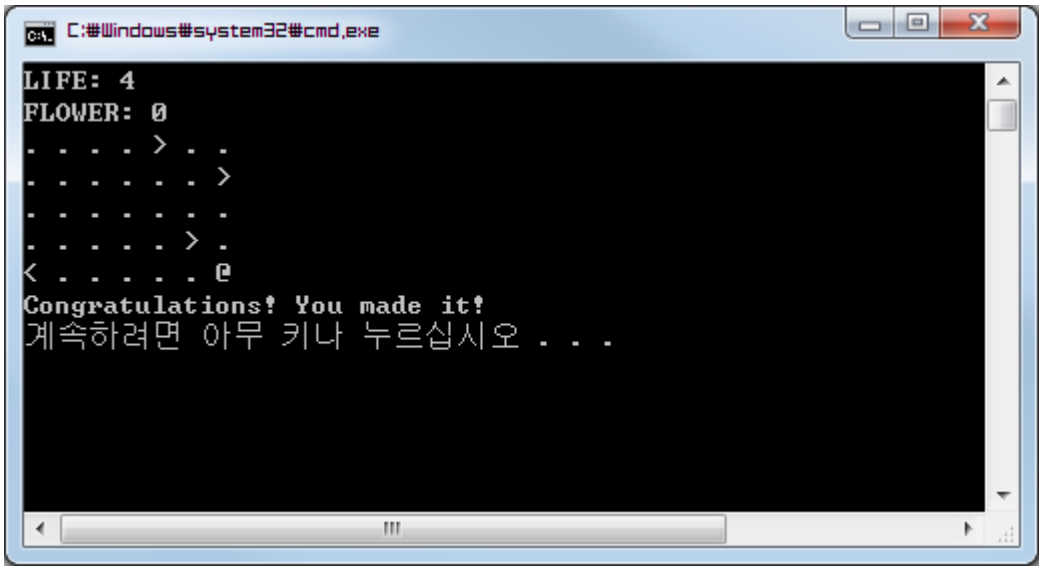
프로그램을 실행 시킨 후, input3.txt 파일에 있는 순서대로 입력을 하는 경우, map3.txt 를 읽어서 생성된 맵에서 게임이 진행되며, step 마다 맵 상태 출력은 out3.txt 와 같다.

※ input*.txt 파일 입력 팁

Visual studio 환경에서 입력할 문자열을 복사한 후, 프로그램 실행 창의 제목표시줄에서 마우스 오른쪽 버튼을 클릭하고 [편집]-[붙여넣기]를 선택하면 사용자 입력이 한번에 입력되어 순서대로 실행된다.

출력된 결과물 파일을 제공되는 파일 out1.txt 와 비교해 볼 수 있다.


input1.txt - 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
out1.txt
map1.txt
C:₩Windows₩system32₩cmd.exe
Input the out file name:
이전 크기로(R)
이동(M)
크기 조정(S)
최소화(N)
최대화(X)
닫기(C)
편집(E)
기본값(D)
속성(P)
표시(K)
복사(Y) Enter
붙여넣기(P)
모두 선택(S)
스크롤(L)
찾기(F)...


C:₩Windows₩system32₩cmd.exe
LIFE: 4
FLOWER: 0
. . . . > . .
. . . . . . >
. . . .
. . . . . > .
< . . . . . @
Congratulations! You made it!
계속하려면 아무 키나 누르십시오 . . .