

CSED101. Programming & Problem solving

Spring, 2017

Programming Assignment #3 (70 points)

이건희(zogondragon@postech.ac.kr)

- **Due:** 2017.05.04 23:59
- **Development Environment:** GNU C Compiler (GCC) and Vim Editor (Editor is optional)
- **제출물**
 - **C Code files (*.c and *.h)**
 - 프로그램의 소스 코드를 이해하기 쉽도록 반드시 주석을 붙일 것.
 - **보고서 파일** (.doc(x) or .hwp) 예) assn3.doc(x) 또는 assn3.hwp
 - AssnReadMe.pdf 를 참조하여 작성할 것.
 - 리눅스 서버에 접속하는 것부터 시작해서 프로그램 컴파일 및 실행하는 과정까지를 화면 캡처하여 보고서에 포함시키고 간단히 설명 할 것!!
 - 명예서약(Honor code): 표지에 다음의 내용을 포함한다. “나는 이 프로그래밍 과제를 다른 사람의 부적절한 도움 없이 완수하였습니다.” 보고서 표지에 명예서약이 없는 경우는 과제를 제출하지 않은 것으로 처리한다.
 - 소스코드와 보고서 파일을 LMS를 이용하여 제출한다.
- **주의사항**
 - 과제에 대한 질문은 수강생들과 공유하기 위하여 LMS의 질의응답 게시판을 이용한다. 이외 사항은 이메일 등의 개별 연락보다는 오피스아워를 이용한다.
 - 각 문제에 해당하는 요구사항을 반드시 지킬 것.
 - 컴파일 & 실행이 안되면 0점 처리된다. 반드시 리눅스 서버에서 본인이 직접 테스트해보고 제출한다.
 - 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.)
 - 과제 작성시 전역변수는 사용할 수 없으며, 사용자 정의 함수를 적절히 작성하도록 한다.
 - 이번 과제에서는 추가 기능 구현에 대한 추가 점수는 없습니다.

■ Problem 1: 탐험 게임 지도 만들기(35점)

(목적)

- 2차원 배열의 선언과 사용을 익힌다.
- 함수에서 2차원 배열을 매개변수로 사용하는 방법을 익힌다.

(주의사항)

- 보고서는 "**assn3.doc**" or "**assn3.hwp**"로 저장 할 것 (보고서는 통합하여 작성)
- 전역변수, goto 문, 구조체는 사용하지 않는다.
- 프로그램 구현 시, main() 함수를 호출하여 사용하지 않는다. 즉, 소스 코드 내에 main(); 이라고 호출하지 않는다.
- 적절히 사용자 정의 함수를 작성하여야 한다. (main() 함수 내에 모든 기능을 구현 시 감점)
- 과제에서 정의하는 함수와 파일은 반드시 작성한다. (미 작성시 감점). 추가로 더 만들 함수가 있으면 만들어서 사용해도 된다.
- **map.h**(문서에서 내용 설명)와 **map.c(main() 함수를 포함한 모든 사용자 정의 함수의 선언과 정의를 포함)**를 작성한다. 이 문제에서는 다른 소스파일을 추가로 사용하지 않는다.
- 출력은 아래의 "실행예제"와 동일하거나 비슷하게 작성 할 것.
- 초기 위치를 설정할 때는 rand() 함수를 이용해서 맵의 임의의 위치에 들어가게 해야 하는데, 이 때 srand() 함수를 이용한 seed 설정을 하여 매번 실행할 때마다 다른 값을 가지게 해야 한다.
- 과제 문서 사이사이에 있는 지시사항과 맨 마지막 부분에 있는 팁 모음을 반드시 끝까지 읽고 구현을 시작한다. 부분점수가 걸려 있는 지시사항들을 놓치지 않도록 주의한다.
- 구현에 반드시 필요한 함수의 예시로는 CreateMap(), PrintMap(), DrawSquare(), DrawCircle() 등이 있다. 나머지 추가 사용자 정의 함수의 경우 꼭 이대로 이름을 짓지 않아도 되지만 소스코드 안에서 본인의 함수와 변수의 작명법이 통일성을 가질 수 있도록 한다. (참조: <https://google.github.io/styleguide/cppguide.html> 또는 [https://msdn.microsoft.com/en-us/library/ms229002\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/ms229002(v=vs.110).aspx) 등을 활용. 인터넷에 naming convention 혹은 naming guideline 으로 검색하여 좋은 작명을 하는 습관을 들이자.)

(설명)

간단한 RPG 탐험 게임의 바탕이 되는 지도와 화면 입출력 기능을 리눅스 콘솔 입출력으로 구현한다.

- 2차원 배열(array)로 게임 맵을 구현한다.
- 게임 맵의 한 칸은 char 데이터 타입으로 구성되어 있다.
- 사용자는 이 지도를 탐험하는 화면을 상하좌우로 움직일 수 있다.
- 탐험에 사용되는 명령어는 h, j, k, l, H, J, K, L, c, q 이다. 자세한 내용은 아래에서 설명한다.

I. 맵 설정

(1) map.h 파일

map.h 파일에는 MAP_ROW, MAP_COL, VIEW_ROW, VIEW_COL, MOVE_STEP, RADIUS의 상수를 define 하여 사용한다. map.h 파일의 내용은 아래와 같다.

```
#ifndef MAP_H
#define MAP_H

#define MAP_ROW 64
#define MAP_COL 144

#define VIEW_ROW 16
#define VIEW_COL 36

#define MOVE_STEP 10
#define RADIUS 5

#endif
```

- **MAP_ROW:** 전체 지도의 세로 크기를 결정하는 상수
- **MAP_COL:** 전체 지도의 가로 크기를 결정하는 상수
- **VIEW_ROW:** 화면에 보여지는 부분의 세로 크기를 결정하는 상수
- **VIEW_COL:** 화면에 보여지는 부분의 가로 크기를 결정하는 상수
- **MOVE_STEP:** 사용자 입력으로, 대문자 H, J, K, L 을 이용하여 view 를 옮길 때 한 번에 몇 칸씩 옮길지 결정하는 상수
- **RADIUS:** 맵 생성시, 동그라미의 반지름을 결정하는 상수

학생들의 구현 편의를 위하여 RADIUS 를 제외한 모든 상수는 짝수라고 가정한다. 또한

조교가 채점 시, 위 상수를 바꾸어가면서 채점할 것이므로 학생들도 제출하기 전에 스스로 이 값을 바꿔가면서 테스트 해 보도록 하자. 각 상수들의 허용 범위는 $40 \leq \text{MAP_ROW} \leq 400$, $40 \leq \text{MAP_COL} \leq 400$, $10 \leq \text{VIEW_ROW} \leq 50$, $10 \leq \text{VIEW_COL} \leq 50$, $1 \leq \text{MOVE_STEP} \leq 20$, $1 \leq \text{RADIUS} \leq 20$ 이라고 생각하고 과제를 구현한다.

위의 사용자 정의 헤더 파일 map.h 를 map.c 에서 include 하여 사용한다. 또한 **보고서에 왜 "#ifndef"와 "#define", 그리고 "#endif" 와 같은 부분이 들어가는지** 간략하게 조사하여 그 이유를 서술한다(부분점수 있음).

(2) 맵 생성

탐험할 맵의 전체크기는 MAP_ROW X MAP_COL 의 크기를 가진 2 차원 배열로 구성한다.

main()함수 안에 탐험할 맵을 아래와 같이 선언 후, 사용하도록 한다. (다른 방법으로 구현시 감점)

```
char main_map[MAP_ROW][MAP_COL];
```

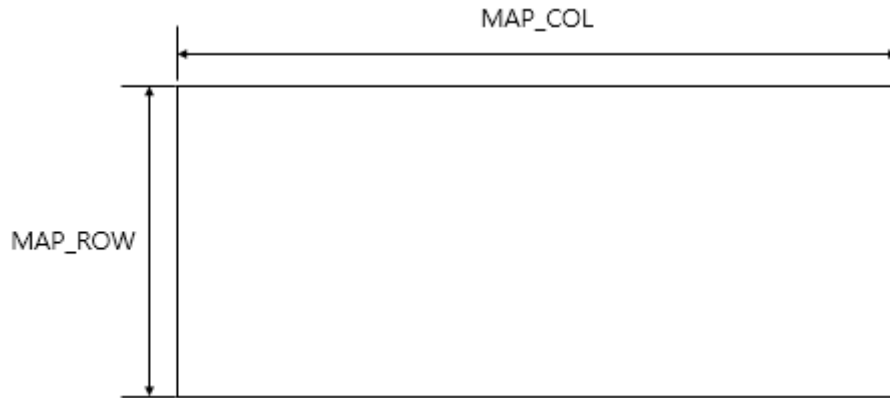


그림 1. 탐색할 맵의 전체 크기

- ***void CreateMap(char main_map[][MAP_COL], ...)***

탐색할 지도를 생성하는 함수로, 전달 받은 2차원 배열의 배열의 내용을 전부 ' '(space 1개)로 초기화 한 후, 아래에서 정의하는 DrawCircle()함수와 DrawSquare()함수를 호출하여 지도를 완성하는 함수이다.

(배열 이외에 필요한 매개변수가 있으면 추가하여 구현한다.)

(3) 탐색할 맵 안에 무인도 배치하기

맵의 특정한 위치에 동그라미 모양의 무인도 1개와 네모 모양의 섬 2개를 배치한다. 동그라미 무인도의 모양은 'c'로 표현하고 네모 모양의 섬은 's'로 표현한다.

맵의 한가운데(중심점) 좌표는 (MAP_ROW/2, MAP_COL/2)로 정의한다. 이 것은 MAP_ROW와 MAP_COL 이 짝수이기 때문에 중심점을 어떻게 지정할지에 대한 모호성이 발생하는데 그 모호성을 없애고 통일하기 위한 것이므로 **다르게 구현할 경우 감점한다.**

가. 동그라미 그리기

아래의 예시처럼, 동그라미는 'c'를 이용하여 나타내며, 반드시 아래의 함수를 구현하여 이용하도록 한다.

- ***void DrawCircle(int center_row, int center_col, int radius, char main_map[][MAP_COL])***

: 동그라미의 중심의 위치와 반지름 radius (r)를 전달받아 동그라미를 지도에 그리는 함수

$(x-a)^2+(y-b)^2 \leq r^2$ 식을 이용하여 속이 꽉 찬 동그라미를 그린다. (원의 중심의 좌표 (a, b)에서 일정한 거리(반지름 r)에 있는 점의 좌표 (x, y)의 집합을 원이라고 한다.) 실행예제와 동일한 결과를 내기 위해서는 \leq 인 것에 유의한다. 또한 **math.h 라이브러리를 사용하지 않고 ***

(곱하기) 연산자를 이용하여 계산한다.

위와 같이 구현한 함수를 CreateMap() 함수 안에서 **DrawCircle(MAP_ROW/2, MAP_COL/2,**

RADIUS, main_map); 로 호출되며, 전체를 출력하면 아래와 같이 맵이 채워진다. 아래는

MAP_ROW: 20, MAP_COL: 30, RADIUS: 5일 때의 예시로, 동그라미가 아래와 같이 구성된다. **(출력 예시와 동일하게 출력되도록 함수를 정의한다. 다를 시 감점)**

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
0																														
1																														
2																														
3																														
4																														
5																c														
6													c	c	c	c	c	c	c											
7												c	c	c	c	c	c	c	c	c										
8												c	c	c	c	c	c	c	c	c										
9												c	c	c	c	c	c	c	c	c										
10											c	c	c	c	c	c	c	c	c	c	c									
11												c	c	c	c	c	c	c	c	c										
12												c	c	c	c	c	c	c	c	c										
13												c	c	c	c	c	c	c	c	c										
14													c	c	c	c	c	c	c											
15																c														
16																														
17																														
18																														
19																														

그림 2. 맵 생성 예시 - 동그라미

동그라미의 화면 출력 예시는 아래의 그림 6-3을 참고한다.

나. 사각형 그리기

void DrawSquare(int top_left_row, int top_left_col, int bottom_right_row, int bottom_right_col, char main_map[][MAP_COL]) 함수는 top_left 와 bottom_right 의 2개의 점의 좌표를 입력 받으면 해당 두 점을 포함하는 사각 영역에 해당하는 맵 칸을 's' 로 칠하는 함수이다.

CreateMap() 함수 안에서 **DrawSquare(10, 10, 20, 40);** 와 **DrawSquare(25, 15, 30, 30);** 함수를 호출하여 **사각형 2개**를 그린다. 이 때 혹시 여러 도형이 한 칸에 겹치게 될 경우에 대한 처리는 따로 하지 않아도 된다. 더 늦게 그린 도형의 맵 칸('c' 혹은 's')으로 자연스럽게 겹쳐 보이게 되기 때문이다.

사각형을 그릴 때 인수들의 범위는 다음과 같이 가정한다. $0 \leq \text{top_left_row} \leq \text{MAP_ROW}-1$, $0 \leq \text{top_left_col} \leq \text{MAP_COL}-1$, $\text{top_left_row} \leq \text{bottom_right_row} \leq \text{MAP_ROW}-1$, $\text{top_left_col} \leq \text{bottom_right_col} \leq \text{MAP_COL}-1$. 그리고 만약 top_left_row 와 bottom_right_row, 그리고 top_left_col과 bottom_right_col 이 겹친다면 해당 겹치는 칸에 대해서 's'로 표시하게 구현한다.

(4) 화면출력

MAP은 전체를 출력하는 게 아니라, VIEW 크기에 해당하는 부분이 출력된다.

MAP과 VIEW 사이의 관계는 다음 그림과 같다.

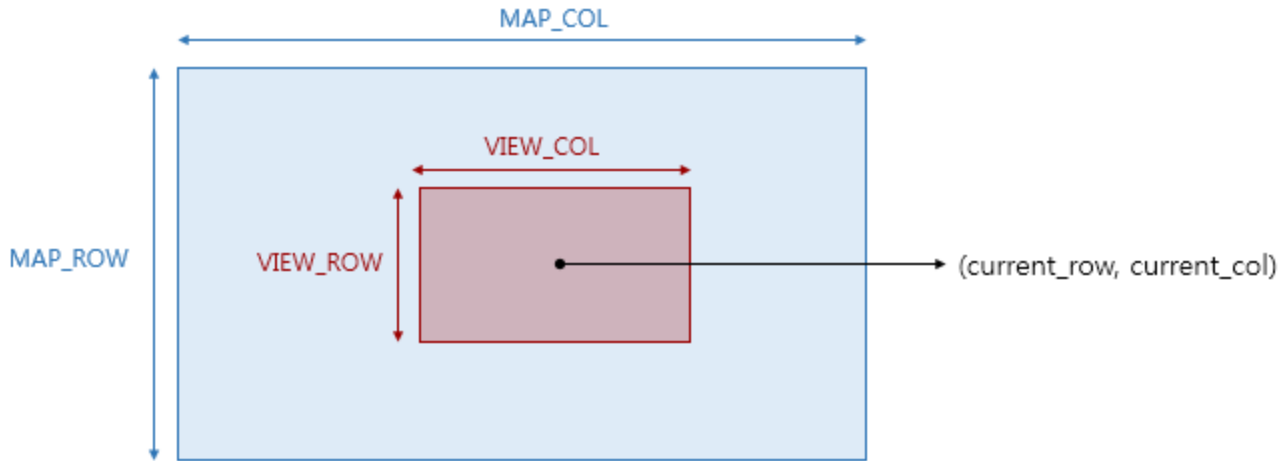


그림 3-1. MAP과 VIEW 사이의 관계

큰 MAP 안에 작은 VIEW 가 있는 형태이다. VIEW가 MAP의 가장자리로 옮겨졌을 때는 다음 그림과 같다.

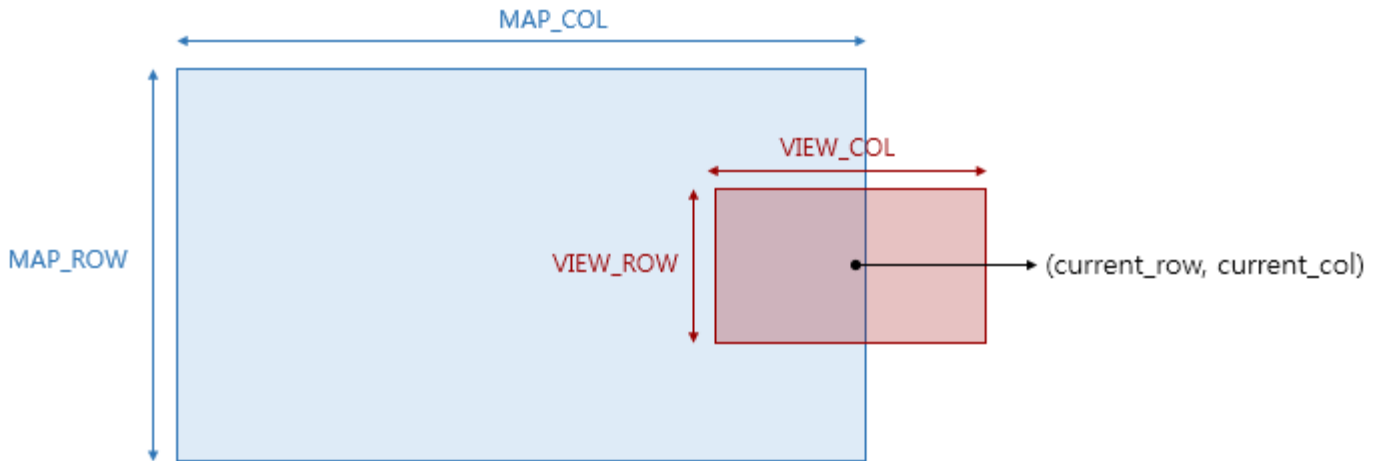


그림 3-2. VIEW를 MAP의 가장자리로 이동시

VIEW는 상하좌우로 움직일 수 있고 그 중심점은 `current_row`, `current_col` 로 표기한다. VIEW의 영역이 MAP 밖을 보여주게 될 경우 X자로 표시한다. (실행예시 그림 7참조)

VIEW의 중심점은 **VIEW_ROW** 와 **VIEW_COL** 이 짝수이기 때문에 **$(VIEW_ROW/2, VIEW_COL/2)$** 로 구현한다. 아래 그림 4는 VIEW_ROW:6, VIEW_COL:10의 경우에 중심점의 위치가 (3, 5)라는 것을 보여준다. 다르게 구현할 시 감점한다.

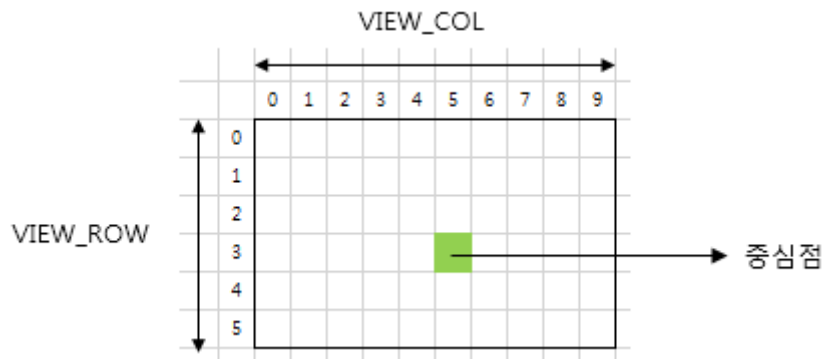


그림 4. VIEW의 중심점 예시

II. 탐험하기

프로그램을 실행하면, 아래와 같은 화면이 출력되고, 사용자 입력을 기다린다.

최초 시작 위치인 (current_row, current_col) 은 rand() 함수를 이용해 맵 범위내에서 무작위로 설정되게 구현한다. 즉 여러 번 프로그램을 실행했을 때 각각 다른 위치에서 시작되어야 한다.

가. 화면구성

사용자의 현재 위치 (초기 위치는 rand() 함수를 이용하여 설정)를 기준으로 가로 세로 일정 범위 안에 있는 맵 칸들을 화면에 출력한다.

그 아래에 사용자의 현재 위치를 row와 col의 형태로 지도 하단에 출력한다.

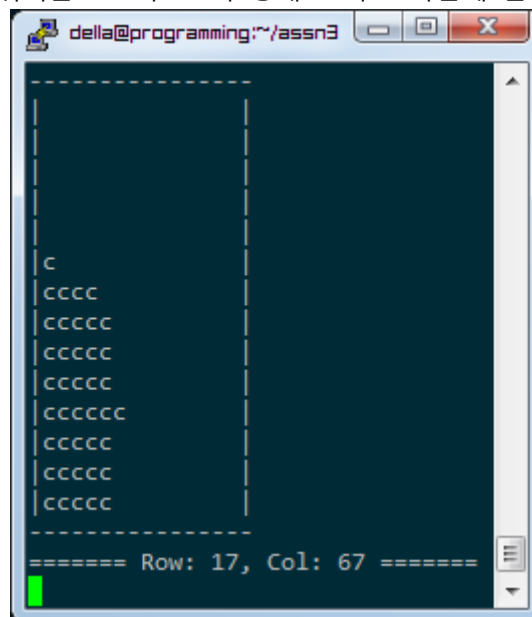


그림 5. 화면구성

- **void PrintMap(char main_map[][MAP_COL], int current_row, int current_col, ...)**
위의 실행예제에서 보이는 사각형 테두리와 VIEW 내용(VIEW_ROW x VIEW_COL)을 출력

나. 명령어 입력

사용자로부터 입력 받는 명령어는 다음과 같다. 아래의 10 개의 알파벳 이외의 문자 입력에 대해서는 무시한다.

- ① **h** : 왼쪽으로 1칸 이동, **H** : 왼쪽으로 MOVE_STEP칸 만큼 이동.
- ② **j** : 아래로 1칸 이동, **J** : 아래로 MOVE_STEP칸 만큼 이동.
- ③ **k** : 위로 1칸 이동, **K** : 위로 MOVE_STEP칸 만큼 이동.
- ④ **l** : 오른쪽으로 1칸 이동, **L** : 오른쪽으로 MOVE_STEP칸 만큼 이동.
- ⑤ **c** : 맵 정중앙에 화면을 위치시킨다. MAP_ROW와 MAP_COL 이 짝수이기 때문에 (MAP_ROW/2, MAP_COL/2) 로 구현한다.
- ⑥ **q** : 프로그램을 종료한다.

명령어를 입력 받으면, 현재 출력된 화면을 지우고 명령어를 따라 이동한 위치의 화면이 새롭게 출력되도록 한다. 아래 "화면 지우기" 기능을 참고하여 구현하자.

• 화면 지우기

화면을 지우기 위해서는 stdlib.h를 포함한 system 함수를 사용한다. 아래 코드를 그대로 실행해 본 뒤, system("clear")를 주석 처리하여 실행해봄으로써 차이를 파악한다.

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    printf("HELLO WORLD\n");
    system("clear"); // 윈도우의 경우 system("cls");

    return 0;
}
```

(실행예시)

MAP_ROW: 40, MAP_COL: 120, VIEW_ROW: 14, VIEW_COL: 14, MOVE_STEP: 4, RADIUS: 5로 define 되어 있는 경우의 예시이다.

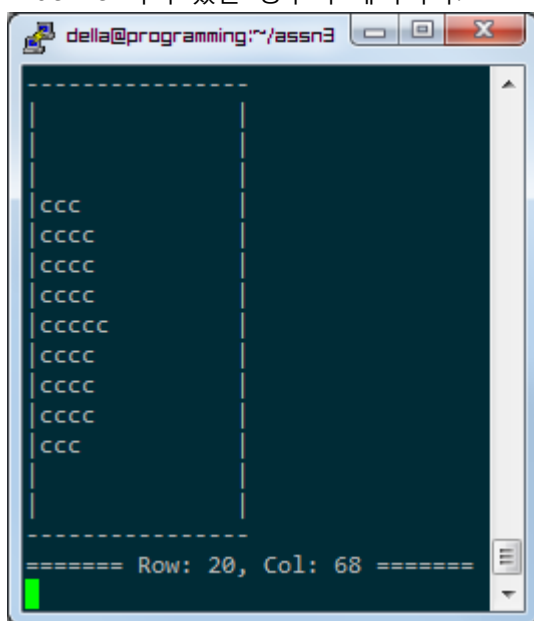


그림 6-1. 실행초기화면 예시

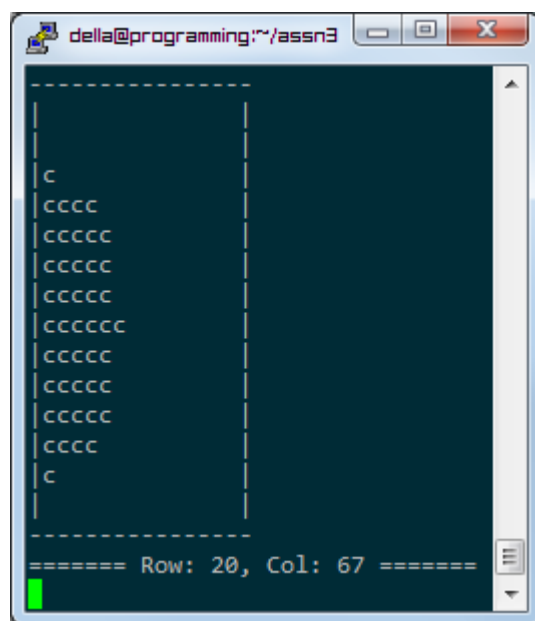


그림 6-2. h키(왼쪽 1칸 이동) 누른 경우

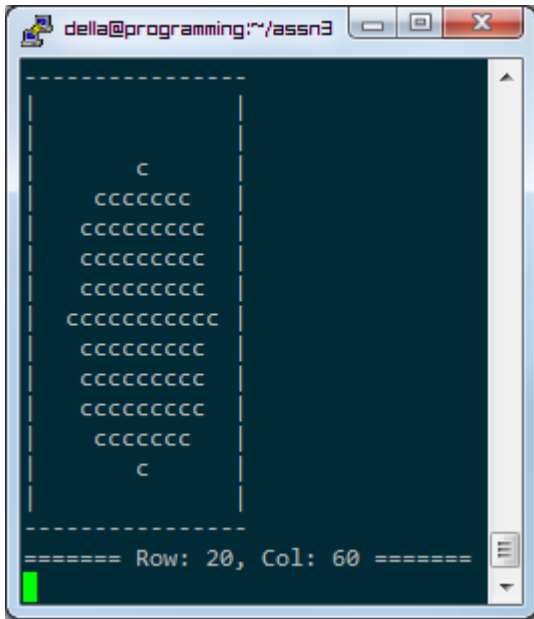


그림 6-3. c키(맵 중앙으로 이동)를 누른 경우

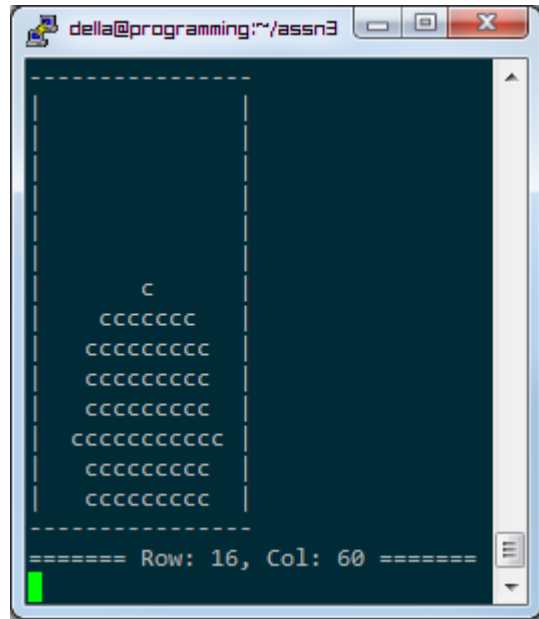


그림 6-4. K키를 입력한 경우

그림 6-1은 초기화면 예시로 랜덤하게 current_row가 20, current_col 이 68로 시작됐다. 이 상태에서 h키를 눌러 왼쪽으로 1칸 이동한 화면이 그림 6-2이며 current_col 이 67로 변경됐음을 알 수 있다.

그림 6-3은 c키를 눌러 맵의 중앙으로 이동했음을 보여주며, 이 상태에서 K키를 눌러 MOVE_STEP(4) 만큼 맵의 위쪽 방향으로 이동한 화면이 그림 6-4 이며, current_row가 20에서 16으로 변경됐음을 알 수 있다.

VIEW 영역이 지도의 범위를 넘어가는 칸은 아래의 예시처럼 X 표시한다.

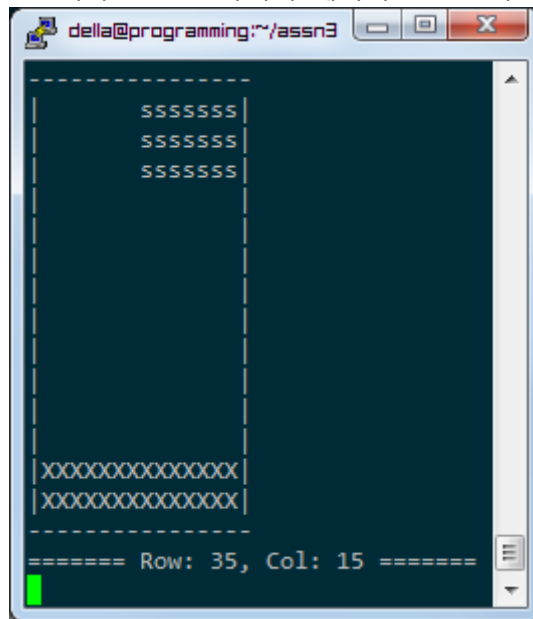


그림 7. VIEW 영역이 지도의 범위를 넘어가는 경우

■ Problem 2: 지뢰찾기 게임 지도 만들기(35점)

(목적)

- 2차원 배열의 선언과 사용을 익힌다.
- 함수에서 2차원 배열을 매개변수로 사용하는 방법을 익힌다.
- 텍스트 파일 출력을 익힌다.

(설계예시)

이 문제는 기능이 단순하므로 **mine.c** 파일 하나로 구현한다. 이 문제에서는 Problem 1에서 생성한 소스코드나 헤더파일을 참조하지 않는다.

(문제)

이번 문제는 Problem 1에서 터득한 배열의 조작법을 바탕으로 지뢰가 n개 설치된 지도를 생성한 뒤 파일 형태로 저장하는 문제이다.

지뢰찾기 게임은 윈도우 기본 게임으로 M x N 행렬에 위치에 있는 지뢰를 찾는 게임이다. 이번 과제에서는 지뢰찾기 게임을 위한 지도를 생성해본다.

(주의사항)

- 전역변수를 사용할 수 없다.
- 소스코드는 **mine.c** 파일 하나로 구현한다.
- 과제에서 정의하는 함수는 반드시 구현하고, 이외 필요한 함수는 적절히 정의해서 사용하여야 한다.
- map을 구성하기 위해, **MAP_ROW**와 **MAP_COL**을 **20으로 define 한 후 사용**하도록 한다. 이 배열의 (define 하지 않고 소스코드에 숫자가 그대로 들어갈 시 감점)
- **2차원 배열 사용**
 - 배열의 크기를 선언할 때 변수를 사용하면 안 된다. (char map[i][j]).
 - 충분한 크기로 배열을 선언 한 뒤 필요한 부분만 사용하도록 한다.
 - 2차원 배열을 함수의 매개변수로 넘겨주기 위해서는 map[size]와 같은 표현을 사용한다.

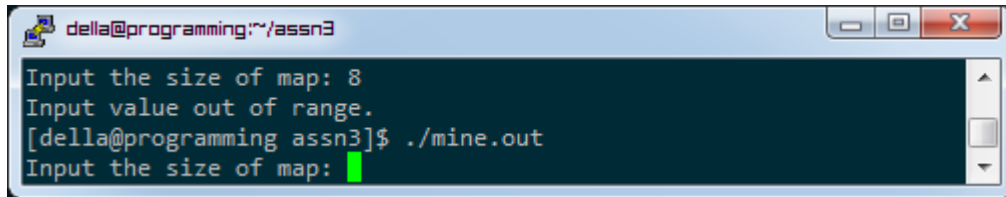
```
int arrayTestFunc(int arr[][colSize], int ...);
int main()
{
    int arr[3][3] = {0, 1, 2, 3, 4, 5, 6, 7, 8};
    int arrSize = 9;

    arrayTestFunc(arr, arrSize);

    return 0;
}
```

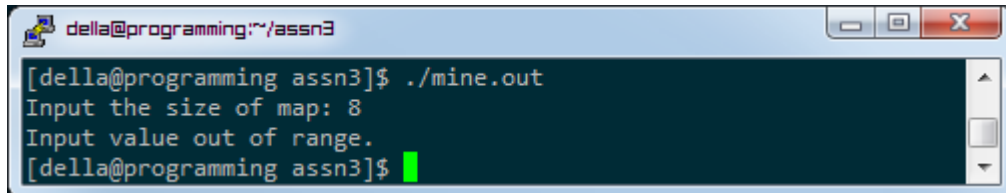
(설명)

프로그램을 실행시키면 아래와 같이 생성할 지뢰찾기 게임의 map 크기(size)를 입력을 기다린다. 지뢰찾기 map은 정사각 size * size 행렬로, 크기는 9부터 20까지로 제한한다. ($9 \leq \text{size} \leq 20$)



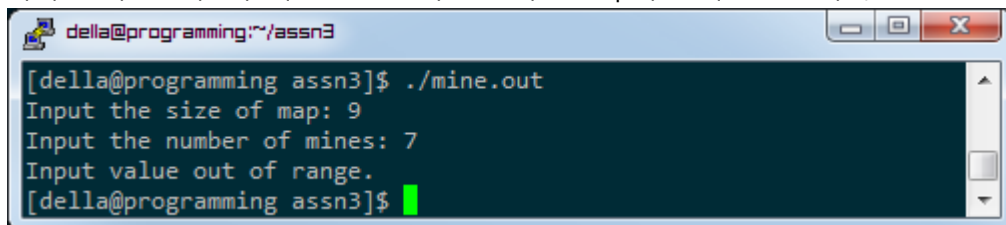
```
della@programming:~/assn3
Input the size of map: 8
Input value out of range.
[della@programming assn3]$ ./mine.out
Input the size of map: █
```

제시한 범위를 벗어나면, 아래의 예시처럼 에러를 출력하고 프로그램을 종료한다.



```
[della@programming assn3]$ ./mine.out
Input the size of map: 8
Input value out of range.
[della@programming assn3]$ █
```

그 다음으로 설치할 지뢰의 수(n)를 입력 받는다. 지뢰의 수는 10개부터 $(\text{size}-1) * (\text{size}-1)$ 개까지로 제한한다. 이 때 size는 입력받은 게임 map의 크기를 말한다. ($10 \leq n \leq (\text{size} - 1)^2$)



```
[della@programming assn3]$ ./mine.out
Input the size of map: 9
Input the number of mines: 7
Input value out of range.
[della@programming assn3]$ █
```

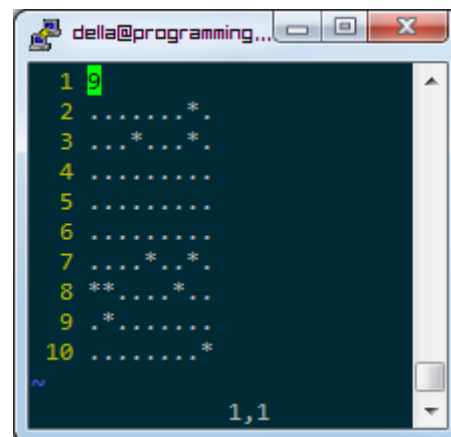
제시한 범위를 벗어나면, 위의 예시처럼 에러를 출력하고 프로그램을 종료한다.

사용자로부터 지정 범위내의 map의 크기와 지뢰의 수를 입력받았으면, size * size에 해당하는 map을 생성하고, 그 map안에 n개의 지뢰가 설치되도록 구성한다. 10개의 지뢰는 같은 칸에 설치되는 경우가 없도록 구현해야한다.

아래 예시는 map의 크기는 9 x 9, 지뢰의 개수는 10개인 예시이다. 이때, 빈칸은 '.' (마침표), 지뢰는 '*' (별) 로 표시했다.

	0	1	2	3	4	5	6	7	8
0	*	.
1	.	.	.	*	.	.	.	*	.
2
3
4
5	*	.	.	*	.
6	*	*	*	.	.
7	.	*
8	*

그림 1. 지뢰찾기 map의 생성 예시



```
della@programming...
1 9
2 .....*
3 ...*...*
4 .....*
5 .....*
6 .....*
7 ...*...*
8 **...*
9 ...*...
10 .....*

1,1
```

그림 2. minemap.txt의 파일 출력 예시

그림 1과 같이 구성된 map을 minemap.txt 파일에 출력한다. 그림 1의 예시를 파일로 출력하면 그림 2와 같다. 파일의 첫번째 줄은 size, 그 다음줄부터 생성한 map을 행 단위로 파일에 출력한다. 이때, 빈칸은 '.' (마침표), 지뢰는 '*' (별) 로 표기하여 출력한다.

다음으로, 생성된 맵을 이용하여 지뢰의 위치(*)를 제외한 나머지 칸에 인접한 칸에 몇 개의 지뢰가 있는지를 찾아, mapsol.txt 파일로 출력한다. (각 칸은 최대 8개의 인접한 칸이 있을 수 있다.)

예를 들면, 그림 3에서 칸(5, 6)에 인접한 8개의 칸에는 2개의 지뢰가 있으므로, 칸(5, 6)의 값은 2가 된다. 칸(7, 0)의 경우에는 5개의 칸이 인접해 있고, 이 칸 중에 3개가 지뢰이므로, 칸(7, 0)의 값은 3이 된다.

	0	1	2	3	4	5	6	7	8
0	0	0	1	1	1	0	2	*	2
1	0	0	1	*	1	0	2	*	2
2	0	0	1	1	1	0	1	1	1
3	0	0	0	0	0	0	0	0	0
4	0	0	0	1	1	1	1	1	1
5	2	2	1	1	*	2	2	*	1
6	*	*	2	1	1	2	*	2	1
7	3	*	2	0	0	1	1	2	1
8	1	1	1	0	0	0	0	1	*

그림 3. 생성된 map의 빈칸을 지뢰수로 채우기

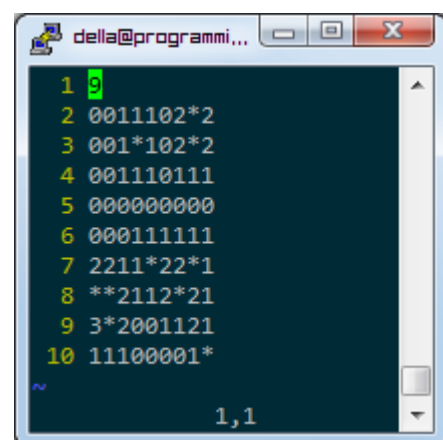


그림 4. mapsol.txt의 파일 출력 예시

생성된 mapsol.txt는 그림 4와 같다. 파일의 첫번째 줄은 size, 그 다음줄부터는 지뢰수로 채운 map을 행 단위로 파일에 출력한다.

아래의 그림 5는 에러 없는 값을 입력했을 때의 실행예시이다. 이를 통해 minemap.txt와 mapsol.txt가 생성된다.

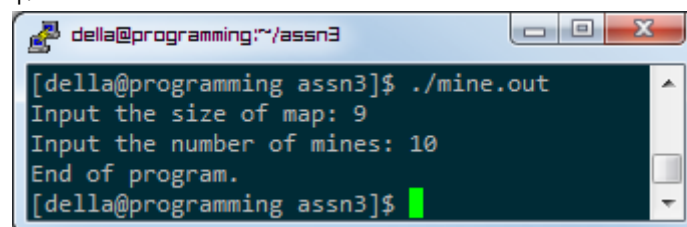


그림 5. 실행 예시

(코드 작성)

- 포인터 전까지 다루었던 문법 (조건문, 반복문, 함수, 배열 등)을 이용하여 작성하며 포인터, goto문, 구조체 등을 사용하지 않는다. 또한, 외부 라이브러리(string.h 등)의 사용은 불가하다.
- **반드시 작성해야 하는 함수**는 다음과 같다. 이 함수들 말고도 자유롭게 추가로 함수를 정의해서 사용하도록 한다.
 - **int main() 함수**
map의 크기와 설치한 지뢰의 수를 입력받아 지뢰찾기 지도를 생성하고, 파일 출력을 하는 함수를 호출하는 부분. 에러처리도 이 함수에서 수행한다. 에러처리가 똑바로 되지 않을시 감점.
char mineMap[MAP_ROW][MAP_COL] 은 main() 함수 안에서 선언하고 함수 매개변수를 통해 2차원 배열을 넘기게 구현한다 (**다른 방법으로 구현할시 감점**).
 - **int GenerateMineMap(int n, char mineMap[][MAP_COL], ...)**
전달받은 2차원 배열 mineMap에 n개의 지뢰를 설치하는 함수로, 지뢰 설치 위치는 random 하게 정하고, n개 모두 다른 위치에 설치되도록 구성한다. (팁: 버그를 방지하기 위해서 함수가 끝나는 시점에 **제대로 n개의 지뢰가 설치되었는지** 확인하여 제대로 설치되었으면 return 0을 하고 제대로 설치되지 않았으면 return 1을 하여 디버그를 쉽게 할 수 있다.)
 - **int SaveMineMap(char mineMap[][MAP_COL], ...)**
mineMap의 내용을 파일에 출력한다.

■ 어싸인 팁 모음

1. **Putty** 의 기본 색깔과 글꼴 조합은 눈이 매우 피로하고 글자를 잘 읽을 수 없다. 색깔을 바꾸어서 눈의 피로를 경감하자. 색깔과 글꼴을 바꾸는 방법은 설정 안에 있으니 잘 찾아보거나 인터넷 검색을 활용한다. 색깔은 Solarized Dark 나 Solarized Light 를 추천한다 (<https://www.nextofwindows.com/how-to-easily-change-default-putty-color-scheme>). 글꼴은 네이버에서 무료로 배포하는 나눔고딕코딩을 16pt 이상 크기로 사용하는 것을 추천한다.

(색깔과 글꼴 예시) - 내용은 무시하고 색깔과 글꼴만 볼 것

```
1 #ifndef MAP_H
2 #define MAP_H
3
4 // For the grading, TA will change this values to other test values.
5 // So students need to code their program accordingly.
6 #define MAP_ROW 256
7 #define MAP_COL 256
8
9 #define VIEW_ROW 24
10 #define VIEW_COL 48
11
12 // Normally we don't use global variable, but in this assignment
13 // students do not know how to use pointer, so we have to use it.
14 char main_map[MAP_ROW][MAP_COL];
15
16 void CreateMap();
17 int PrintMap(int current_row, int current_col);
18
19 #endif /* MAP_H */
~
~
```

2. **VIM 의 기능** 중에는 긴 변수명을 치다가 중간에 Ctrl+N 이나 Ctrl+P 를 누르면 **자동 완성을 시켜주는 기능**이 있다. 매우 편리하니 익숙해지도록 하자. 또 :vs (vertical split) 기능을 이용하면 한 번에 2개의 파일을 열어서 작업할 수 있는데, 헤더 파일과 소스 파일을 같이 열어놓고 작업하면 매우 편리하다. 창 사이를 움직이는 키는 Ctrl+W, W 이고 자세한 사항은 인터넷 검색을 통해 스스로 공부해보자. VIM은 쓰면 쓸 수록 세계 최고의 에디터라는 것을 느끼게 해 주는 좋은 프로그램이다.

3. VIM 의 편집 모드 명령 중 i, o, s, cw, dd, J(대문자), u, dw, Ctrl+F, Ctrl+B, . (마침표), :ls, :b<숫자>, :e <파일이름> 등은 매우 자주 사용되니 익숙해지도록 연습해보자. 또 화면을 움직이는 것도 화살표보다 h, j, k, l 등으로 움직이는 것이 편하다. 이 키들은 UNIX / LINUX 상의 많은 프로그램들이 공통으로 사용하는 이동키이므로 익혀두면 미래에 편리하다.

4. programming.postech.ac.kr 서버에 접속해서 vim ~/.vimrc 를 입력한 뒤 다음 내용을 복사 붙여넣기하면 편리하다. Putty에서 붙여넣기 하려면 VIM에서 i 키를 눌러 입력 모드로 바꾼 뒤

마우스 우클릭을 누르면 된다.

```
set tabstop=4
set softtabstop=4
set shiftwidth=4
set bs=2
set number
set smartindent
set expandtab
set nohls
set scrolloff=5
set syntax=on
```

5. MAP_ROW, MAP_COL, VIEW_ROW, VIEW_COL 등을 제출할 때는 문서에 적혀 있는 초기값으로 제출해야 하지만 동작 과정이 올바른지 확인하기 위해서 학생들은 제출하기 전에 스스로 이 값을 바꿔가면서 테스트해보도록 하자. 이런 연습을 하는 이유는 실제로 나중에 어떤 프로그램을 개발하던지 간에 개발 중간에 요구사항이나 지시사항이 바뀌는 일이 매일 밥 먹듯이 일어나기 때문이다. 미리 대비해서 유동적으로 코딩하는 습관을 길러놓지 않으면 나중에 불편한 상황에 직면할 수 있다.

6. UNIX/LINUX는 서버용 운영체제이기 때문에 putty 창을 여러개 띄워놓고 동시에 작업해도 된다. 창 하나에는 VIM 을 띄워놓고 다른 창에서는 gcc 를 이용하면 매번 VIM을 켜다 켜다 하지 않아도 된다.

7. 프로그램 작성 중 무한루프에 의하여 프로그램이 끝나지 않을 경우 Ctrl+C 를 누르면 된다.

Ctrl+Z 를 눌렀을 경우 프로그램을 끝내는 게 아니라 잠시 멈추게 되는데, 이 경우 ps 와 kill 명령어를 이용하여 수동으로 프로그램을 꺼 주어야 하므로 복잡해진다. 그냥 Ctrl+C 를 이용하도록 한다.

8. 주석은 짧고 간단한 영어라도 괜찮으니 영어로 다는 연습을 해 보자. 현업에서는 여러 플랫폼 사이에 파일을 전달할 때 한글 인코딩이 달라서 내용이 깨지는 경우가 매우 빈번히 발생한다. 이것을 방지하기 위하여 영어로 주석과 코드를 작성하는 습관을 들인다.

9. 혹시 윈도우에서 개발하고 리눅스로 파일만 옮기는 학생들의 경우 반드시 linux 에서 제대로 컴파일 되고 동작되는지, 또 소스 파일은 제대로 열리는지 확인하도록 한다. linux 환경에서 제대로 실행되지 않으면 감점된다.