2020년 1학기

프로그래밍과 문제해결

Assignment #3

담당교수:윤은영

학번:20200725

학과:무은재학부

이름:윤승우

POVIS ID:ysw1110

명예서약(Honor code)

"나는 이 프로그래밍 과제를 다른 사람의 부적절한 도움 없이 완수하였습니다."

Problem: 세균전

1. 문제의 개요

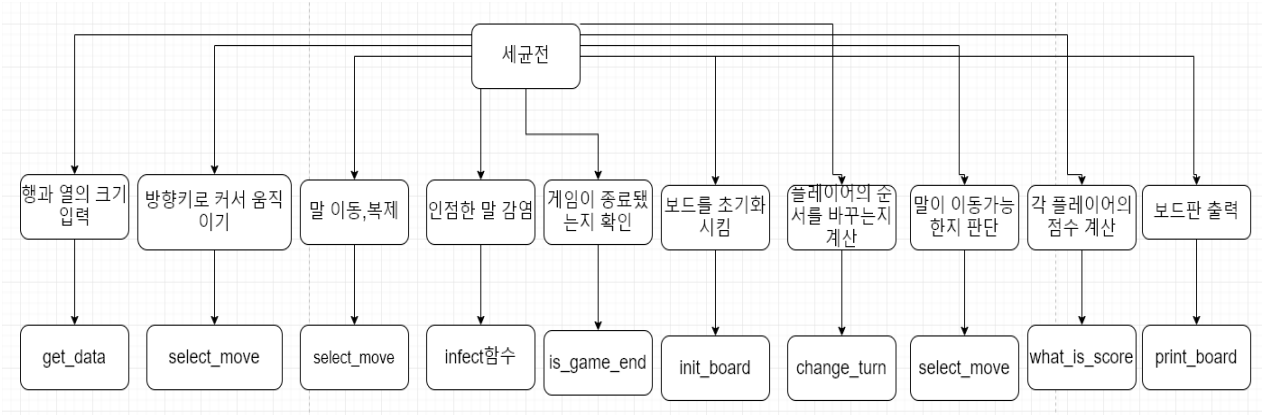
이 프로그램을 간략히 설명하자면 다음과 같다

-프로그램을 실행하면 컴퓨터가 사용자로부터 행과 열의 크기를 입력받는다.

-입력받은 크기에 따라 보드판을 생성하고 세균전 게임을 진행한다.

-경기가 종료되면 재시작하거나 프로그램을 종료할 수 있다.

이 내용을 구조도로 표현하자면 다음의 사진과 같다.



입력부1: 행과 열의 크기를 입력받는다

입력부2: 커서를 움직인다.

처리부1: 말 이동 혹은 복제시킨다.

처리부2: 이동시킨 말 근처의 상대편 말들을 감염시킨다.

처리부3: 게임이 종료됐는지 확인한다.

처리부4: 보드를 초기화시킨다.

처리부5: 플레이어의 순서를 바꾸는지 계산한다.

처리부6: 말이 이동가능 한지 판단한다.

처리부7: 각 플레이어의 점수를 계산한다.

출력부1: 보드판을 출력한다.

2. 알고리즘

본 프로그램 작성을 위한 알고리즘을 pseudo 코드 형태로 나타내면 다음과 같다.

1.헤더 포함

2.사용자 정의 함수의 프로토타입

get_data:보드판의 행과 열의 개수를 입력받는 함수
start_game:게임을 시작하는 함수, 게임이 종료된 후 게임을 계속 진행한다면1,아니면 0을 반환
print_board:점수상황,순서,게임판을 출력하는 함수
init_board:맨 처음 게임판을 만든 후 초기 조건에 따라 게임판 안의 값들을 초기화 시켜주는 함수
select_move:말을 선택하고 이동시킴, 그리고 플레이어의 순서를 반환하는 함수
what_is_score:현재 각 플레이어의 점수를 구하는 함수
infect:이동시킨 말 근처의 다른 플레이어의 말들을 이동시킨 플레이어의 말로 바꾸는 함수
change_turn:플레이어의 순서를 바꾸는지 확인하는 함수, 플레이어의 말 중 움직일 수 있는 말이 없다면 그 플레이어의 순서는 돌아오지 않고 원래 플레이어의 순서 계속 진행함. Plater1의 순서이면 1을 반환, player2의 순서이면 2를 반환
is_game_end:게임이 종료됐는 확인하는 함수, 종료되면 0, 그렇지 않으면 1 반환

메인 함수

1.변수들 선언

2.start_game함수의 반환값이 1인 동안 아래 문장들 반복

1.get data호출

2.동적할당으로 메모리 할당

3.init_board 호출

4.start_game 호출

5.동적할당 했던 메모리 free 통해 메모리 지워줌

get_data함수

1.게임 시작화면 출력

2.scanf통해 행과 열 개수 입력받음(4미만 혹은 20초과일 경우 다시 입력받음)

3.양옆과 위 아래에 2칸만큼의 여유공간을 만들어주기 위해(여유 공간은 막힌 곳, 커서가 이동할 수 없는 곳으로 가정할 것)행과 열의 개수에 4씩 추가해줌

start_game 함수

1.is game end함수를 호출함과 동시에 이 함수의 반환값이 0인동안 아래 반복

1.화면 지움

2.what_is_score호출

3.print_board호출

4.select_move호출

5.화면 지움

6.what_is_score호출

2.print_board호출

3.player1의 점수의 player2의 점수를 비교하여 상황에 맞는 게임결과화면 출력

4.게임 종료 후 y 혹은 n을 입력받아 게임을 계속 진행 혹은 프로그램 종료

is_game_end 함수

1.player1과 player2의 총점이 보드판 크기와 같아지면 게임 종료

2.player1의 말이 1개도 존재하지 않는 경우에 게임 종료

3.player2의 말이 1개도 존재하지 않는 경우에 게임 종료

4. 1,2,3 모두 해당하지 않는 경우 게임 끝난 상태 아님

select_move함수

1.Player1의 순서인 동안에 아래 문장들 수행

1.커서 배치

2.저번 턴에서의 마지막 커서 위치를 통해 커서위치 계산

3.space바를 누르기 이전까지, 혹은 spacebar를 누르더라도 그 자리에 자신의 말이 없는 동안

While문 이용해서 아래 문장들 반복

1.getch

2.getch로 i를 입력받으면 위로,k를 입력받으면 아래로,j를 입력받으면 왼쪽,i를 입력받으면 오른쪽으로 커서를 이동시킴

3.space바(공백)을 입력받으면 아래 문장들 수행

1.아직 말을 선택하지 않은 상태에서 커서 위치에 자신의 말이 있을 때 그 말을 선택된 player1의 말로 바꿔줌, 현재 커서의 위치 저장

2.이미 말을 선택한 경우에서 다른 말을 또 선택하는 경우 이전에 선택했던 말은 선택 취소, 새말을 선택된 말로 바꿈, 현재 커서의 위치 저장

3.말을 선택한 상태에서, 커서가 비어있는 보드판에 있을 때 spacebar를 입력하면 우선 현재 커서의 위치를 저장하고 if else 구문 이용하여 말을 이동시킨 거리가 2이면 말을 이동, 거리가 1이면 말을 복제함

4.change turn함수를 호출하고 change turn함수의 반환값을 return 함

2.player2의 순서일 때에 대하여

1.커서 배치

2.저번 턴에서의 마지막 커서 위치를 통해 커서위치 계산

3.space바를 누르기 이전까지, 혹은 spacebar를 누르더라도 그 자리에 자신의 말이 없는 동안

While문 이용해서 아래 문장들 반복

1.getch

2.getch로 i를 입력받으면 위로,k를 입력받으면 아래로,j를 입력받으면 왼쪽,i를 입력받으면 오른쪽으로 커서를 이동시킴

3.space바(공백)을 입력받으면 아래 문장들 수행

1.아직 말을 선택하지 않은 상태에서 커서 위치에 자신의 말이 있을 때 그 말을 선택된 player2의 말로 바꿔줌, 현재 커서의 위치 저장

2.이미 말을 선택한 경우에서 다른 말을 또 선택하는 경우 이전에 선택했던 말은 선택 취소, 새말을 선택된 말로 바꿈, 현재 커서의 위치 저장

3.말을 선택한 상태에서, 커서가 비어있는 보드판에 있을 때 spacebar를 입력하면 우선 현재 커서의 위치를 저장하고 if else 구문 이용하여 말을 이동시킨 거리가 2이면 말을 이동, 거리가 1이면 말을 복제함

4.change turn함수를 호출하고 change turn함수의 반환값을 return 함

change_turn함수

1.player1의 순서인 경우 아래문장들 수행

1.모든 player2의 말들에 대해 각각의 말들로부터 1칸,2칸 떨어진 공간 중 빈 공간이 없다면 그 말은 움직일 수 없는 말이다. 그러므로 순서가 돌아가지 않고 player1의 순서가 계속된다. 그러므로 turn변수에 1을 대입한다.

2.player2의 순서인 경우 아래 문장들 수행

1.모든 player1의 말들에 대해 각각의 말들로부터 1칸,2칸 떨어진 공간 중 빈 공간이 없다면 그 말은 움직일 수 없는 말이다. 그러므로 순서가 돌아가지 않고 player1의 순서가 계속된다. 그러므로 turn변수에 1을 대입한다.

3.turn을 return한다.

infect함수

1.player1의 순서이면 커서로부터 1칸 떨어져 있는 모든 player2의 말들을 전부 player1의 말들로 바꿔준다.

2.player2의 순서이면 커서로부터 1칸 떨어져 있는 player1의 말들을 전부 player2의 말들로 바꿔준다.

Print_board함수

1.현재 점수상황과 현재 누구의 순서인지 출력

2.for문을 이용하여 +--과 ₩n를 보드의 크기에 맞게 출력한다.

3.if else를 이용하고 SetConsoleTextAttribute함수를 이용해서 색을 조절하며 커서가 있는 곳에는 [ ]를 출력하고, 빈공간에는 |를, 말이 있는 곳에는 @를 출력한다

Init_board함수

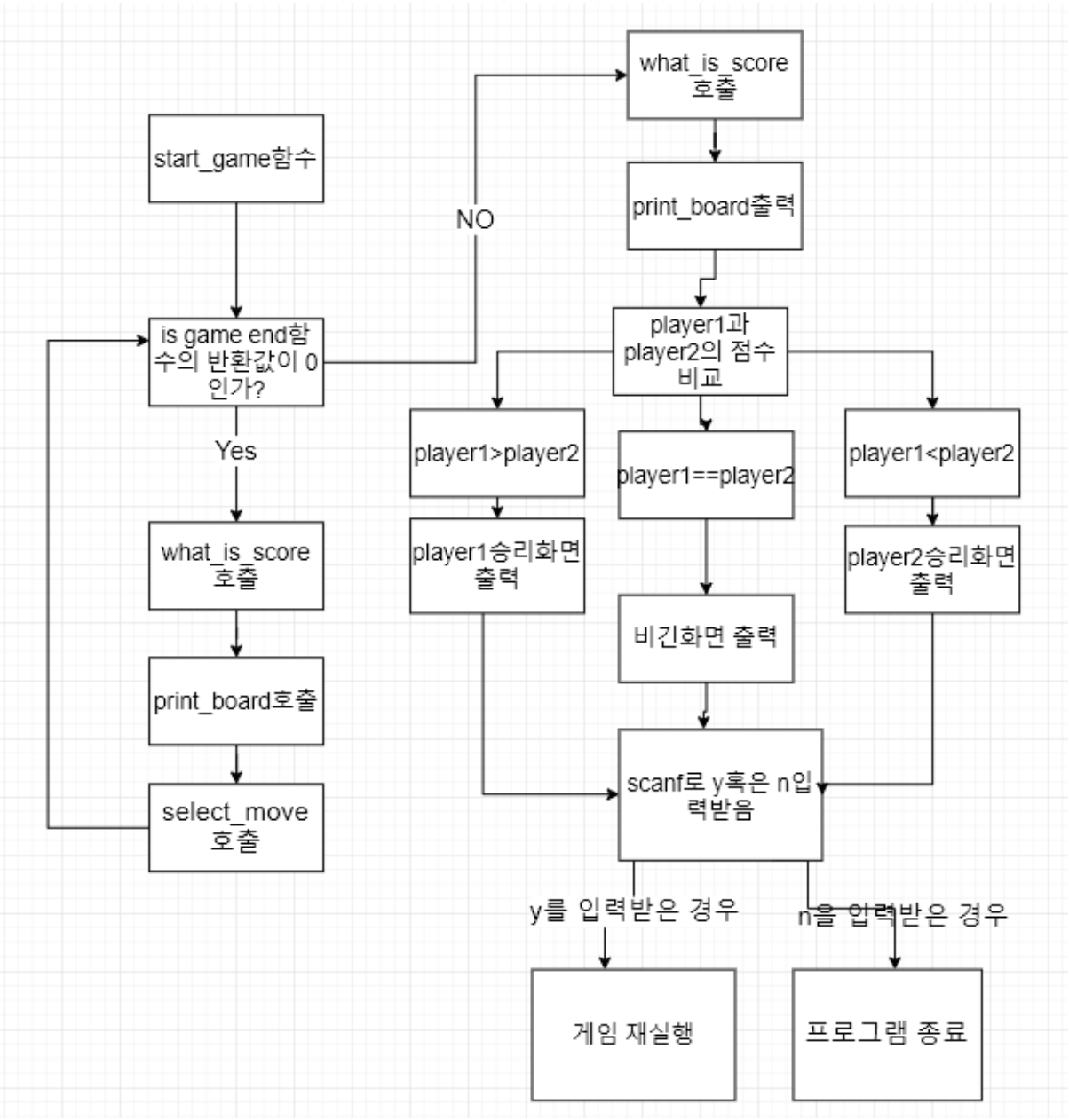
1.입력받은 행과 열의 개수에 맞춰서 실제로 사용할 공간이 아닌 양 옆, 위 아래를 막아둔다(보드판에 어떤 말이 있는지를 나타내는 board_horse배열에 9를 대입한다. 9가 들어있으면 막혀있는 보드판이라고 가정할 것이다.)

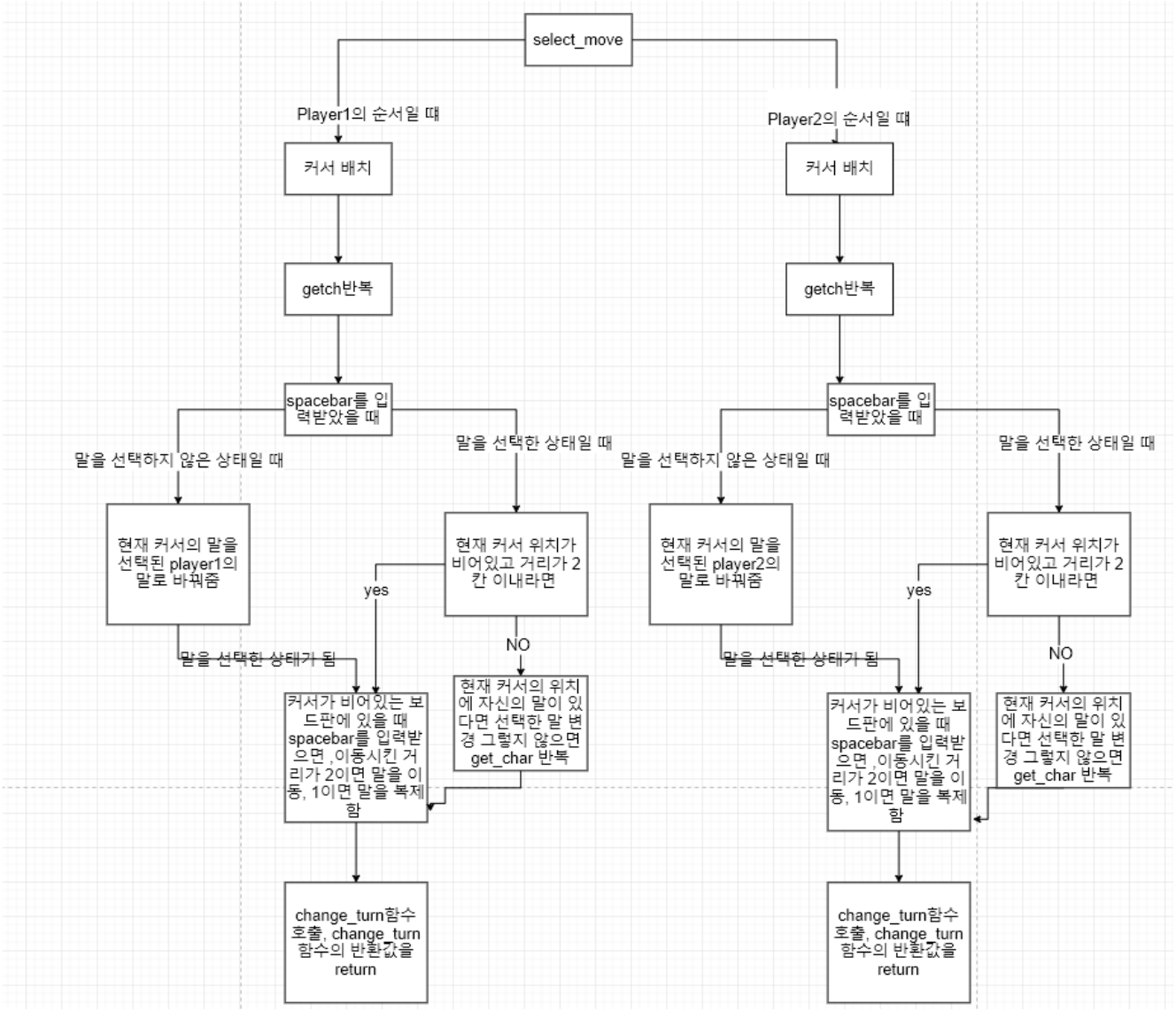
2.실제로 사용할 배열의(0,0)지점과 (0,col-1)지점에 player1의 말을 놓는다

3.실제로 사용할 배열의(row-1,0),(row-1,col-1)지점에 player2의 말을 놓는다

4.맨 처음 커서의 위치는 실제로 사용할 배열의 (0,0)지점으로 해놓는다.

위의 알고리즘을 Flowchart를 통해 표현하면 다음과 같다.





3.프로그램 구조 및 설명

a) 프로그램의 시작

- get_data함수를 호출하여 컴퓨터가 행과 열의 개수를 입력 받는다.

b) 보드판 생성

-동적할당을 이용하여 입력받은 행과 열의 개수에 따라 메모리를 할당한다. 이 때 배열에 위 아래로 2칸 씩 여유공간을 만들어 주기 위해 행과 열의 크기를 4개씩 더 크게 만든다.

- init_board이용하여 맨 처음 커서의 위치를 설정해놓는다. 또한 실제로 사용될 배열이 아닌 나머지 칸들은 9를 넣어서 막아둔다.(배열에 9가 들어가있으면 막힌 곳으로 가정)

c)게임 시작

- start_game함수를 호출한다.

-게임이 종료됐는지 확인하는 is_game_end함수를 호출하고 게임이 종료되지 않았다면 게임을 계속 실행한다.

-what_is_score함수를 호출하여 각 플레이어의 점수를 계산하고 print_board함수를 호출하여 보드판을 출력한다.

-select_move함수를 호출한다, select_move는 커서 움직임, 말 선택, 말 이동, infect함수 호출을 통한 상대편 말 감염시키기를 담당한다.

-화면을 지우고 what_is_score을 호출하여 각 플레이어의 점수를 다시 한 번 계산해 준 후 print_board함수를 호출하여 보드판을 출력한다.

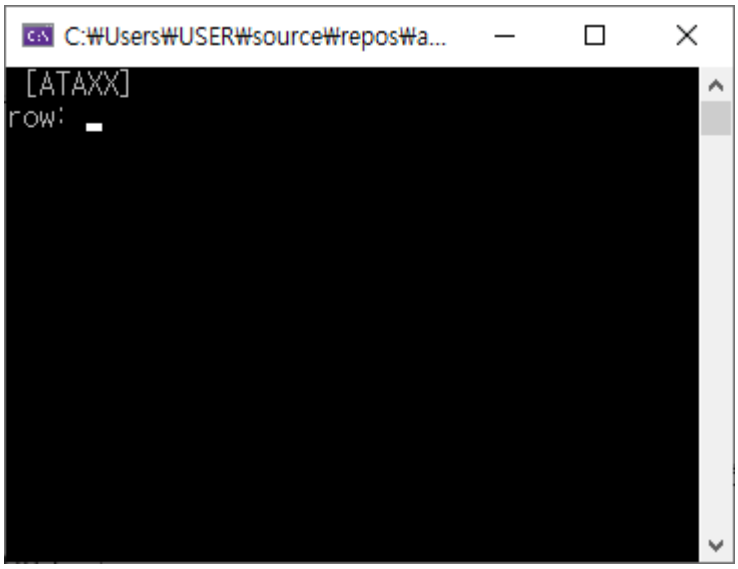
-plyaer1의 점수와 player2의 점수를 비교하여 상황에 맞는 게임결과화면 출력한다.

-scanf를 통해 게임을 계속 진행할 지 말지 입력받는다. y를 입력받으면 main함수 내에서 다시 동적할당, get_data호출, init_board호출, start_game을 호출한다. n을 입력받으면 프로그램이 종료된다.

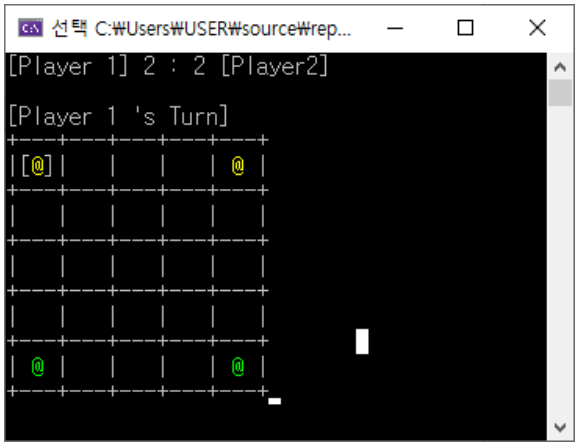
d)메모리 할당 취소

-free를 통해 할당했던 메모리를 취소시킨다.

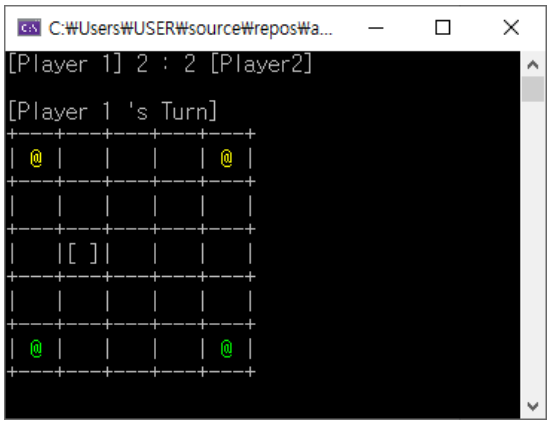
4.프로그램 실행 방법 및 예제



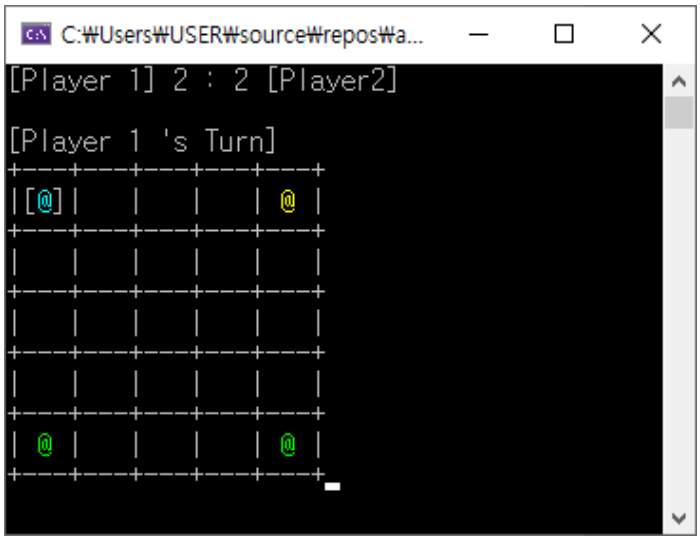
프로그램을 실행하면 우선 행과 열의 크기를 입력받는다.



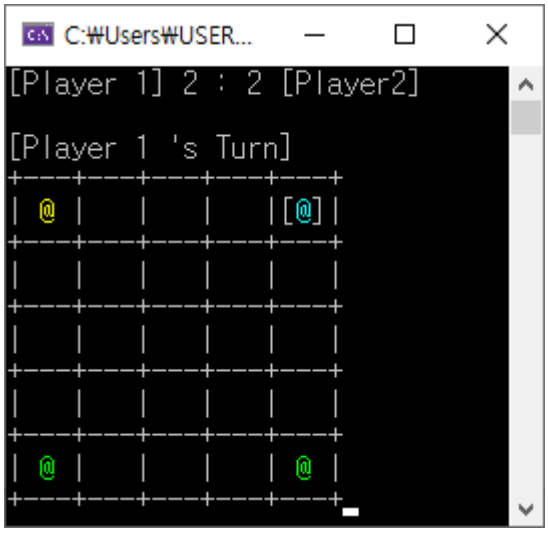
항상 player1이 먼저 시작하며 커서의 초기 위치는 배열의 (0,0)지점이다.



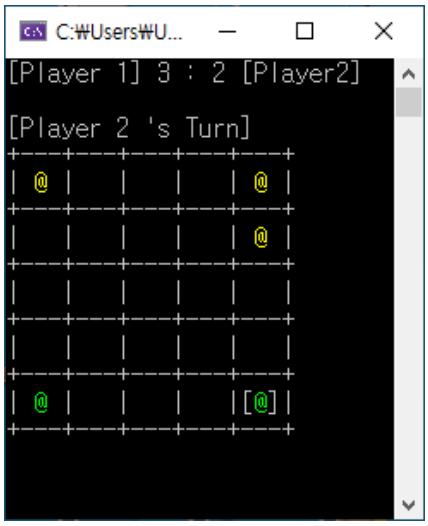
I,j,k,l 을 입력함으로써 커서의 위치를 이동시킬 수 있다.



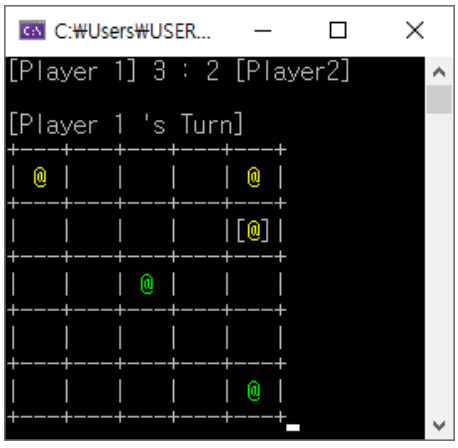
말을 선택하면 하늘색으로 바뀐다.



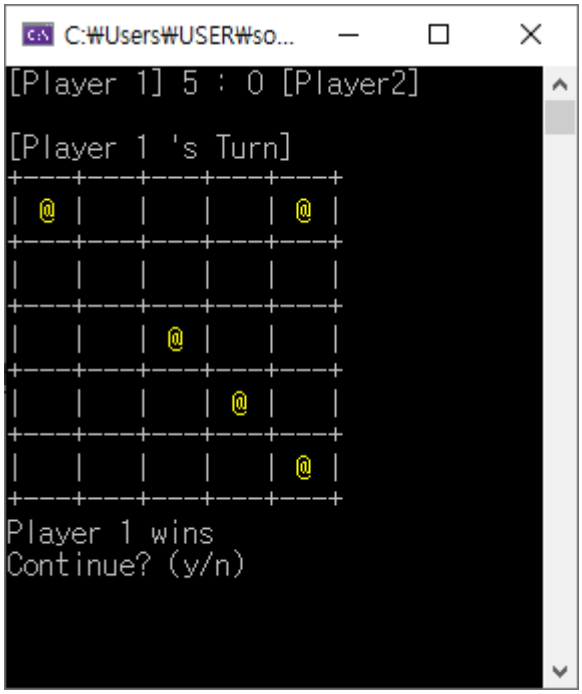
다른 말을 선택하면 이전에 선택했던 말은 취소되고 새 말이 선택된다.



말을 1칸 이동시켰기에 말이 복제됐다. Player2의 순서가 되었다. Player2의 초기 커서 위치는(row-1,col-1) 이다.

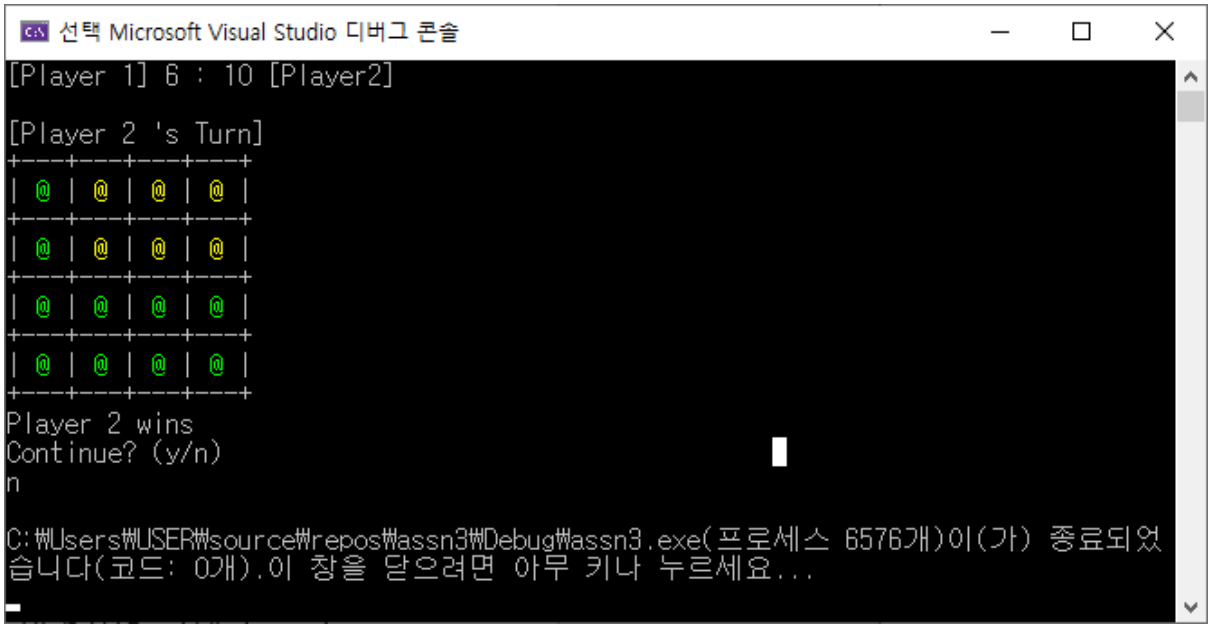


2칸 이동시켰기에 이동됐다. Player1의 순서가 되었으며 커서의 위치는 이전 위치에서 시작된다.

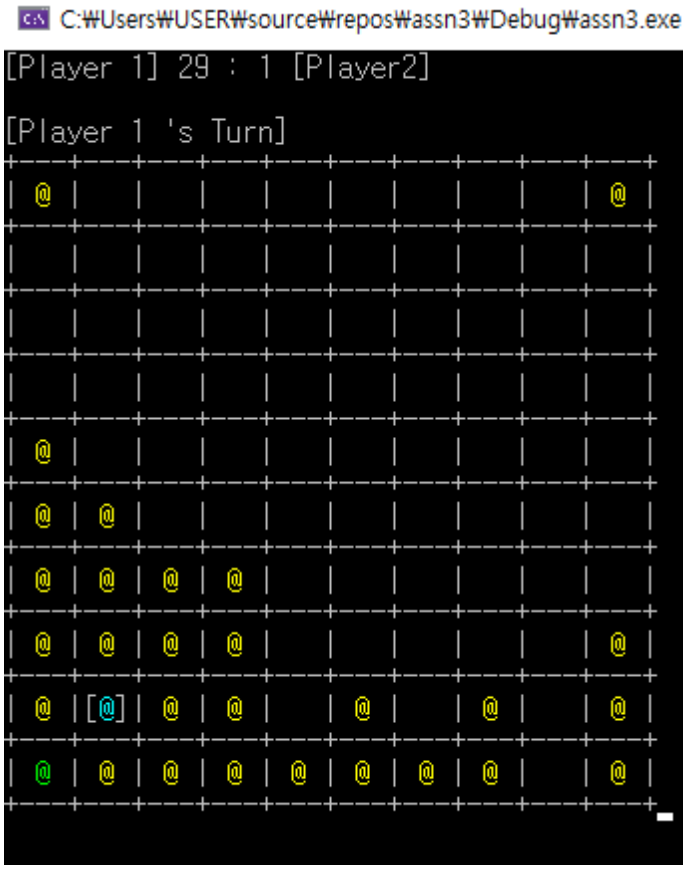


모든 말이 player1의 말이 됐기에 player1이 승리하게 된다.

y를 입력하면 게임을 재시작한다.



양쪽 플레이어가 말을 이동시킬 수 없는 상태이기에 경기가 종료됐다. n을 입력하면 그대로 프로그램이 종료된다.



말이 가둬져서 말을 이동시킬 수 없는 상태가 되면, 플레이어의 순서가 바뀌지 않고 이전 플레이어가 남은 게임판을 가득 채울 때 까지 계속 순서가 진행된다.

5.토론

-change_turn함수에서 특정 플레이어의 말이 가둬지면 플레이어의 순서를 바꾸지 않고 이전 플레이어가 판을 가득 채울 때 까지 계속 진행하게 되는데, 이 때 특정 플레이어의 말이 가둬지는 것을 확인하는 방법이 고민됐다. 결국 입력받은 행과 열 보다 위,아래,왼쪽,오른쪽 각각 입력받은 행과 열의 개수보다 2칸 씩 넉넉하게 더미 보드판을 만들었다., 더미 보드판은 상대편의 말에 의해 막혀있을 때와 똑같은 효과를 주기 위해 9라는 숫자를 넣어놓는 방법으로 특정 플레이어의 말의 주변이 막혀있는지 그렇지 않은지 확인했다.

-플레이어가 마지막으로 커서를 뒀던 위치를 기억해야한다. 그래서 따로 변수를 선언해서, 플레이어가 말을 이동시키는 순간의 커서의 위치를 저장해놨다.

6.결론

-assn3을 만들어 보는 과정에서 동적할당에 대해 배울 수 있었다. 또한 동적할당을 한 후에는 free함수를 이용해서 할당했던 메모리를 비워줘야 하는 것을 다음부터 까먹지 않게 될 것 같다. 조건문과 반복문도 많이 이용됐기 때문에 조건문과 반복문을 이용하는 알고리즘을 짜는 실력도 많이 는 것 같다.

7.개선 방향

-이중포인터를 이용한 후 전부 *연산자를 이용하여 역참조하였는데, []연산자를 이용했으면 좀 더 보기 쉽고 빨리 코드를 작성했을 수 있을 것 같다.

-사용자 정의 함수들에서 if else를 이용해 맨 처음부터 player1인 상황과 player2인 상황으로 나누어 코드를 만들어서 코드의 양이 player1 혼자 일 때 보타 2배가 되었다. 그렇게 하지 말고 공통적인 코드를 만든 다음, player1과 player2에 따라 나뉘는 상황에서만 if else를 이용하였으면 코드 길이를 훨씬 줄일 수 있었다.