System SW 실습 3 term project 1 – 2015313057 류승범

1. 필수 구현 테이블

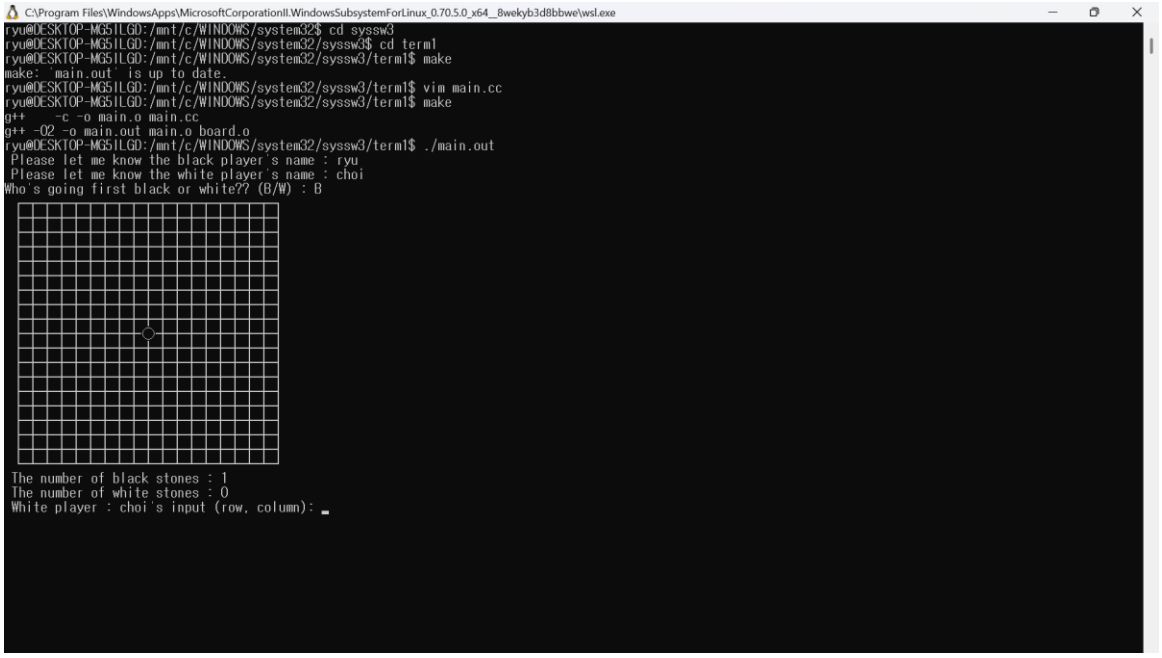
[인터페이스] 바둑판 화면 및 검은 돌, 흰 돌 착수 화면 구현	구현 완료
[인터페이스] 플레이어 이름 입력/표시, 선공 입력, 돌 수 표시	구현 완료
[게임룰] 선공 첫 수는 무조건 정 중앙 위치에 (10,10) 자동착수	구현 완료
[게임룰] 한 턴씩 번갈아 가면서 착수 위치 입력	구현 완료
[게임룰] 쌍삼 착수 금지 알림	구현 완료
[판정] 6목불인정(게임 계속 진행)	구현 완료
[판정] 더 이상 놓을 곳이 없거나"20000" 입력 시 무승부	구현 완료
[판정] 승리 또는 무승부 시 화면에 결과를 알리고 재시작/종료입력	구현 완료

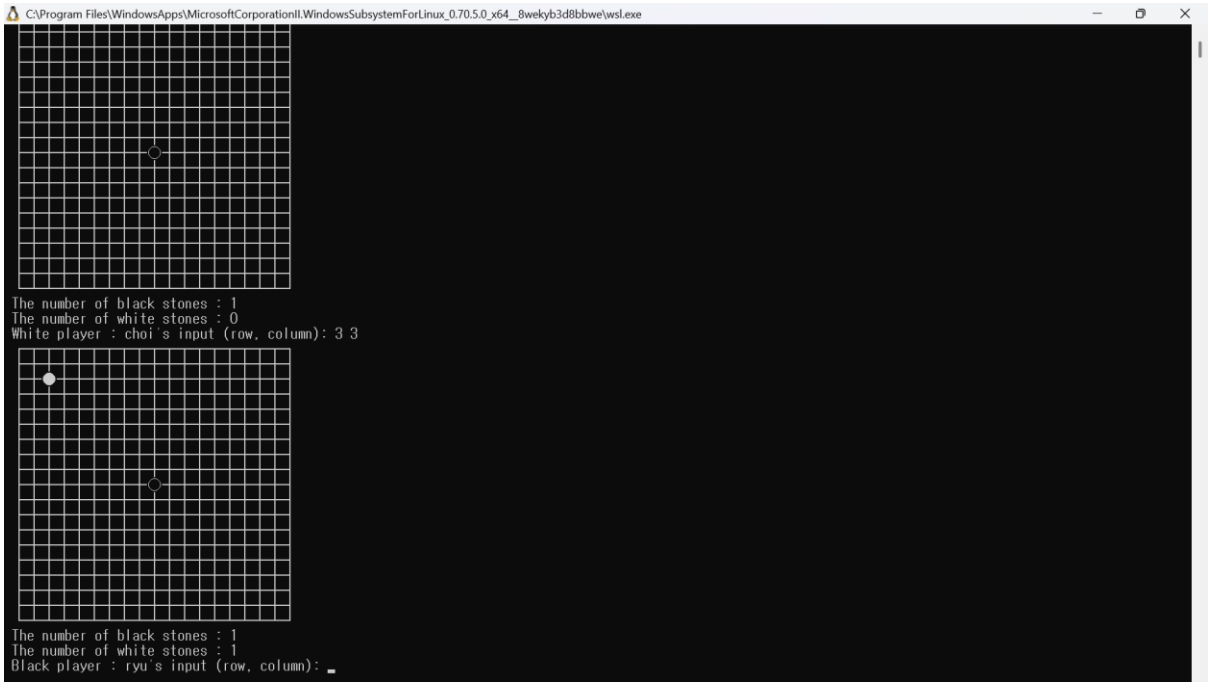
2. 실행 방법 요약

Makefile 을 통해 컴파일을 하면 됩니다.

즉 make 명령어를 사용하면 해당 폴더에 main.out 이 생성되는 데 이 파일을 실행합니다.

먼저 흑/백 플레이어 이름을 각각 입력해주고, 선공을 B/W 로 입력해주면 자동으로 첫 수가 놓인 채로 출력됩니다.





3. Abstract Data Table

먼저 BOARD 는 생성자입니다. 클래스 안의 정보들을 초기화 하고, 선수 위치를 (10, 10)으로 고정해줍니다.

Void print(); 함수는 오목판을 board_arr[][] 대로 출력해줍니다. Board_arr 배열의 0 값은 비어있음을, 1 값은 흑색 돌이 있음을, 2 값은 백색 돌이 있음을 의미합니다.

Void get_name(); 함수는 이름을 std::cin 을 통해 입력 받습니다.

Void get_first(); 함수는 선공의 대상을 B/W 캐릭터를 통해 입력 받습니다.

Bool get_stone(); 함수는 각각의 위치를 입력 받아 객체의 x, y 값에 저장합니다. 게임이 끝나게 될 경우엔 false 를 리턴하고, 그렇지 않은 경우엔 true 를 리턴합니다.

Void put_stone(); 함수는 get_stone 함수에서 호출되며 해당하는 x, y 값을 토대로 Board_arr 배열에 1(흑색) 또는 2(백색)을 저장합니다.

Bool check_pos(); 함수는 돌을 놓으려고 하는 위치에 다른 돌이 있어 중복되거나 영역 밖을 나간 경우에 false 를 리턴합니다. 이 함수 또한 get_stone 함수에서 호출됩니다. 정상적으로 놓을 수 있는 경우 true 를 리턴합니다.

Bool check_ssangsam(); 함수는 돌을 놓으려고 하는 위치가 쌍삼 규칙에 위배되는 지 판단합니다. 위배될 경우 false 를 리턴하고, 아닐 경우 true 를 리턴합니다. 이 함수 또한 get_stone 함수에서 호출됩니다.

Bool judge(); 함수는 돌을 놓을 자리가 없어서 무승부를 해야하거나, 게임 중 누군가가 승리하여 종료해야 할 경우 false 를 리턴합니다. 진행해야 하는 경우는 true 를 리턴합니다. 이 함수 또한 get_stone 함수에서 호출됩니다.

Bool end_phase(); 게임이 끝나게 되어 결과창을 출력해주는 함수입니다. 한번 더 하고 싶은 경우 (Y/N) 중 하나를 캐릭터로 입력 받습니다. Main.cc 에서 직접 호출됩니다.

4. 객체 지향 설계

먼저 클래스의 변수에 대해 설명하겠습니다.

Board_arr[19][19] 라는 인티저 배열이 0, 1, 2 각 값을 가지며 전체 오목판을 나타낼 수 있도록 했습니다.

N_black, n_white는 각 색깔 별 총 돌 개수를

End_flag는 게임 결과를

String p1, p2는 플레이어 이름을

Int x, y는 현재 위치를

Int p는 현재 플레이어를 나타냅니다.

설계 관점에서 클래스 안의 멤버 함수 들에게 각 필수 구현 기능별 역할을 하나씩 주는 데에 중점을 뒀습니다.

이를 통해 한 가지 기능에서 버그가 발생하면 해당 함수에 대해서만 디버깅을 하면 된다는 점이 편리하게 다가왔습니다.