**네트워크 게임 프로그래밍**

**레포트**

**게임공학과**

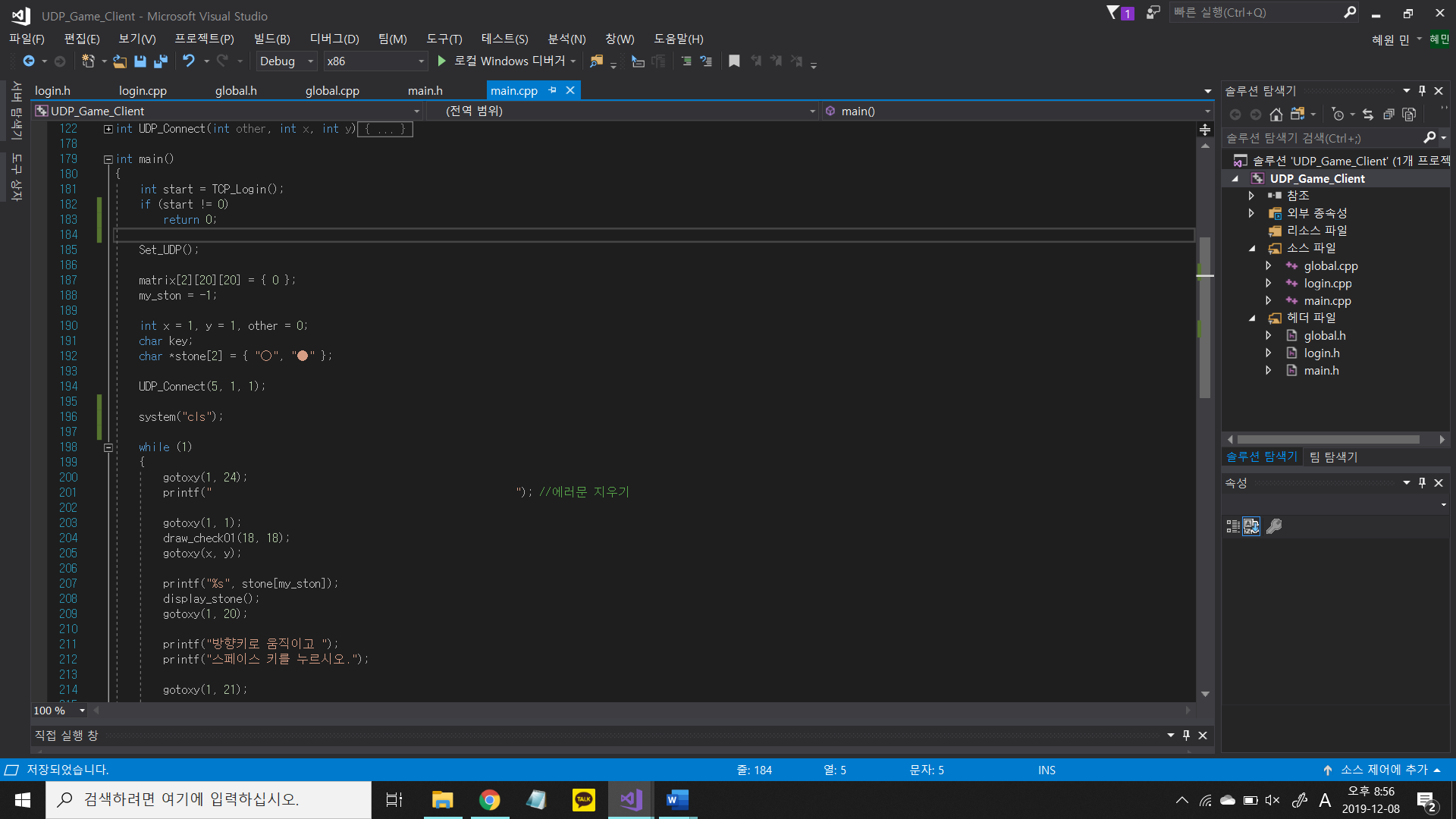
**2017180010**

**민혜원**

1. **프로젝트 명 : 대전 오목 게임**
2. **주요 기능** 
   1. 게임 방 참가
   2. 게임 승패 판정 및 차례 알림
3. **구현 내용**
   1. 멀티 스레드 TCP 서버 – 클라이언트 통신을 사용해 알맞은 비밀번호를 입력하면 게임 방에 참가할 수 있게 한다.
      1. TCP 서버에서 비밀번호를 입력해 설정할 수 있다.
      2. 2명 이상 방에 참가하려 접속하면 방에 참가할 수 없게 한다.
   2. UDP 서버 – 클라이언트 통신을 사용해 돌 정보를 송수신한다.
      1. 클라이언트에서 돌을 두면 이 정보를 서버에 전송한다.
      2. 서버는 이 정보를 보관하고 있다가 상대편 클라이언트에게 요청이 오면 정보를 전송한다.
   3. 멀티 캐스트를 이용해 승패 판정과 차례를 알린다.
      1. 서버는 클라이언트에게서 돌을 두었단 정보를 받으면 멀티 캐스트로 멀티 캐스트 리시버에게 누가 돌을 두었고, 차례가 어떻게 되는지 알린다.
      2. 서버는 승패를 판정하고 승패가 났을 경우 멀티 캐스트로 멀티 캐스트 리시버에게 누가 승리했는지 알린다.
4. **구현 내용 설명**
   1. 방참가
      1. 멀티 스레드 TCP 서버-클라이언트 구조를 이용해 게임 방에 참가하는 부분을 구현했다.
      2. TCP 프로토콜을 사용한 이유 : 비밀번호를 전송하고 결과를 수신하는 역할만 하기 때문에 빠른 속도가 필요 없고, 신뢰성이 높아야 되기 때문에 TCP 프로토콜을 사용했다.
      3. 서버에서 비밀번호를 설정한다. 클라이언트는 비밀번호를 입력하고, 서버에 전송한다. 전송한 비밀번호가 설정한 비밀번호와 일치하면 게임 방으로 입장하고, 일치하지 않으면 다시 입력받는다.
      4. 멀티 스레드를 사용헀기 때문에 여러 클라이언트와 데이터를 송수신할 수 있다. 이때, 오목 대전이기 때문에 클라이언트가 둘만 필요하기 때문에 세번째 스레드부턴 알맞은 비밀번호를 입력했어도, 게임 방으로 넘어가지 못한다.
   2. 게임 실행
      1. UDP 서버-클라이언트 구조를 사용해 게임 실행을 구현했다.
      2. UDP 프로토콜을 사용한 이유 : 대전 형식의 게임이기 때문에 신뢰성보단 전송 속도가 중요하다 생각해 속도가 빠른 UDP 프로토콜을 사용했다. 또한 한 클라이언트에게 받은 데이터를 다른 클라이언트에게 전송해줘야 되기 때문에 다중 스레드에서 공용 데이터에 접근하는 것보다 스레드가 필요 없는 UDP 프로토콜이 더 적합하다고 생각했다.
      3. 게임 구조
         1. 게임화면으로 전환하자마자 클라이언트는 서버에 돌 배정(흑돌인지 백돌인지)을 요청한다. 서버는 클라이언트에게 돌을 배정해 전송한다.
         2. 클라이언트가 돌을 두면 이 정보를 서버에 전송한다. 서버는 흑돌이든, 백돌이든, 전 차례에 둔 돌의 정보를 변수에 저장하고 있는데, 이 정보를 클라이언트에 전송하고, 클라이언트가 전송한 돌의 정보로 변수를 갱신한다.
         3. 클라이언트는 돌을 두기 전 엔터 키를 눌러 서버에 이전 상대 클라이언트가 둔 돌의 정보를 달라 요청한다. 서버는 이 요청을 받고 변수에 저장된 돌의 정보를 클라이언트에게 전송해준다.
   3. 차례 알림 및 승패 판정
      1. 멀티 캐스트를 사용해 차례 알림 및 승패 판정 알림을 구현했다.
      2. 멀티 캐스트를 사용한 이유 : 대전에 참가하는 클라이언트들만 아니라, 참가하지 않는 사람들도 멀티 캐스트 리시버만 실행하면 대전 정보를 받을 수 있게 구현하기 위해 멀티 캐스트를 사용했다.
      3. 구조
         1. 서버는 게임 서버용 UDP와 동일하다. 서버는 클라이언트가 둔 돌의 정보를 받으면 이 정보를 이용해 승패 판정 함수를 돌려 승패를 판정하고, 승패 관련 정보를 멀티 캐스트 리시버에게 전송한다.
         2. 멀티 캐스트 리시버는 승패 판정 정보를 받았을 때, 승패가 나지 않았다는 정보를 받으면 차례를 화면에 출력하고, 승패가 났으면 승리한 돌이 누군지 화면에 출력한다.
5. **소스 코드 설명**

\* 예제를 활용한 부분 및 통신과 관련되지 않은 부분은 이미지 없이 설명만 한다.

* 1. 클라이언트(UDP\_Game\_Client 폴더)
     1. Main 함수
        1. 위치 : main.cpp



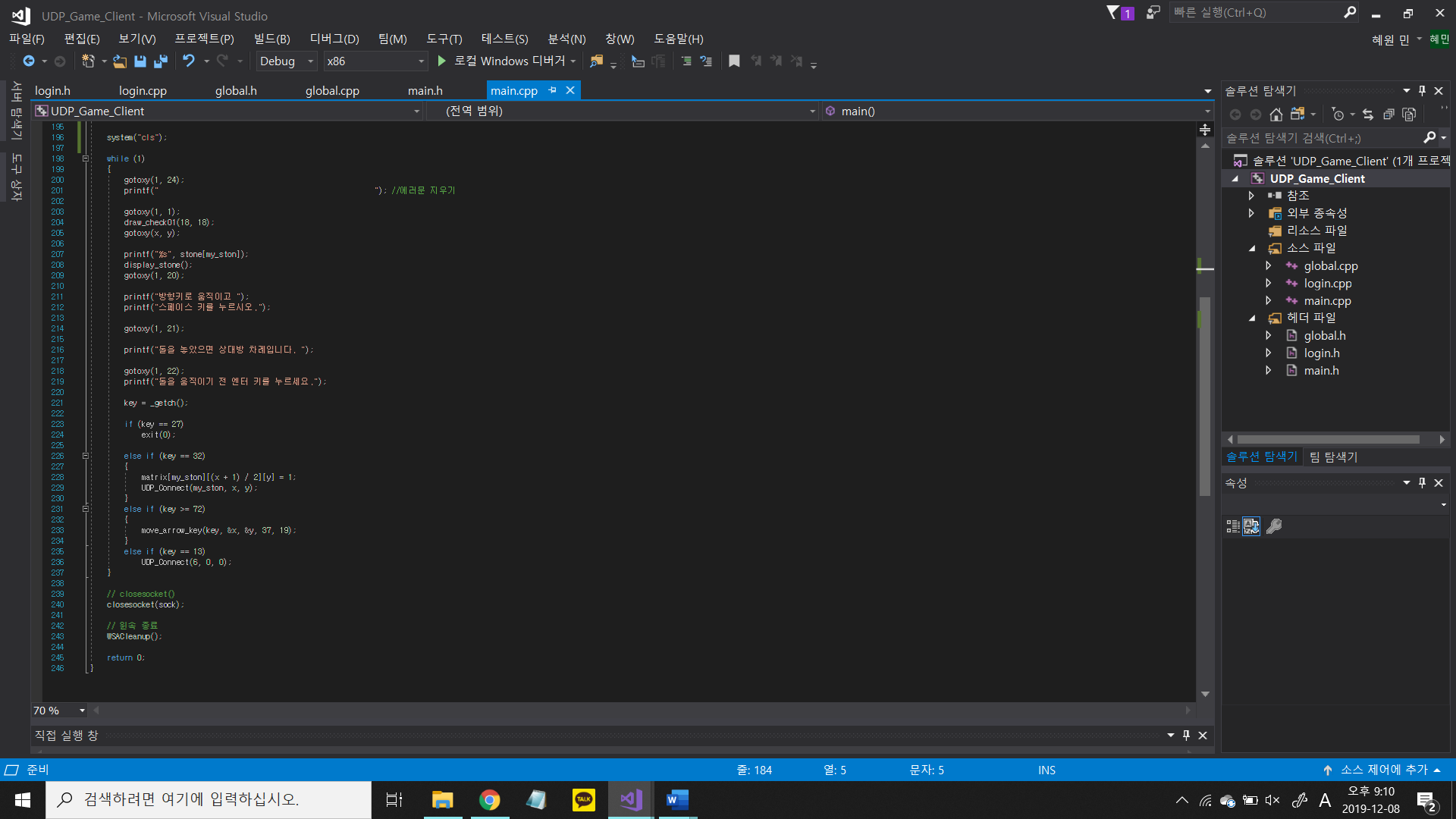
**A**

**B**

**C**

**D**

* + - * 1. TCP 서버에 로그인하기 위한 함수인 TCP\_Login 함수를 호출한다. TCP 서버와의 통신을 담당하는 함수다. 로그인에 성공하면 0을 반환하고 실패하면 프로그램을 종료한다.
        2. UDP 소켓 생성 및 소켓 주소 초기화하는 함수
        3. 오목 관련 변수 초기화한다. Matrix는 보드판, other은 흑돌, 백돌 관련, x, y는 이동 좌표.
        4. UDP 통신하는 함수 UDP\_Connect 함수를 한 번 호출해 서버에 돌을 배정해달라 요청하고, 돌을 배정 받는다.



**J**

**I**

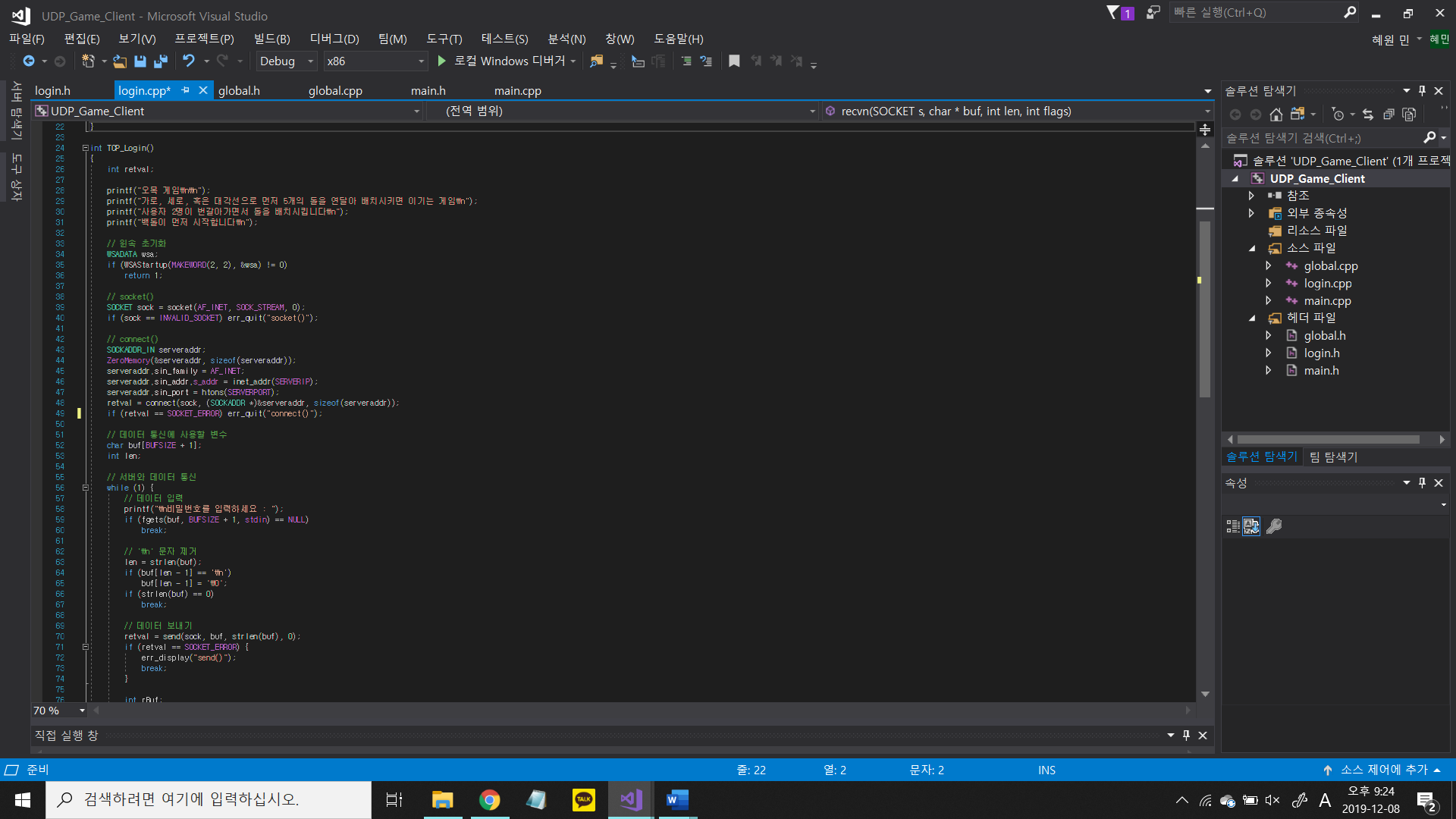
**H**

**G**

**F**

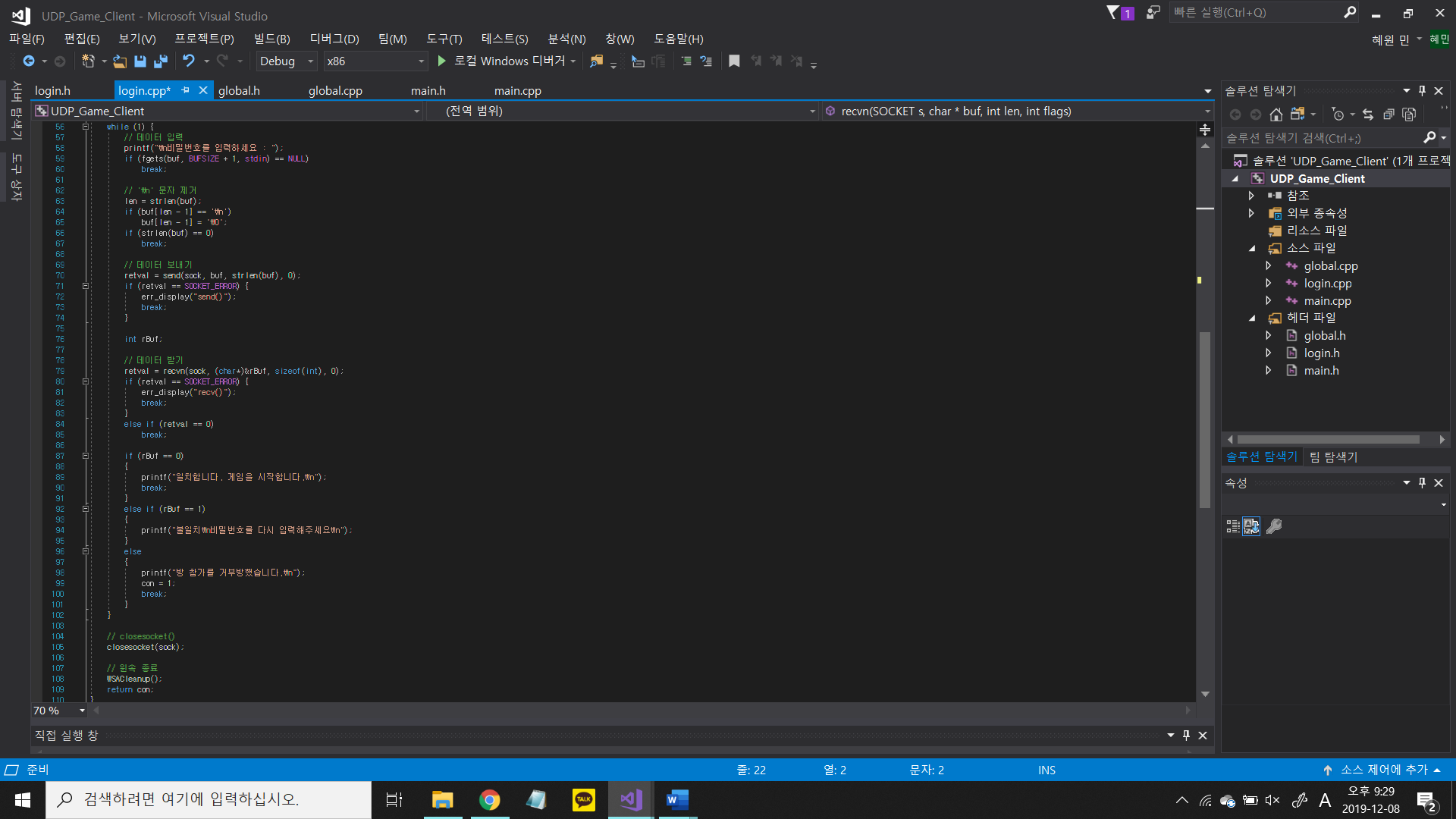
**E**

* + - * 1. 게임 설명을 출력한다.
        2. 키보드 입력을 받는다.
        3. Esc를 입력받으면 종료한다.
        4. 스페이스를 누르면 게임판에 돌 정보를 저장하고, UDP 서버로 이 정보를 전송한다.
        5. 방향키를 누르면 이동한다.
        6. 엔터키를 누르면 UDP 서버로 이전 차례 돌 정보를 달라 요청을 보내 돌 정보를 받아 게임판에 저장한다.
    1. TCP\_Login 함수
       1. 위치 : login.cpp
       2. 역할 : TCP 서버와의 통신을 담당한다.



**A**

* + - * 1. 윈속 초기화, 소켓 생성, TCP 서버 접속을 수행한다. (TCP\_Client 예제를 활용했다.)



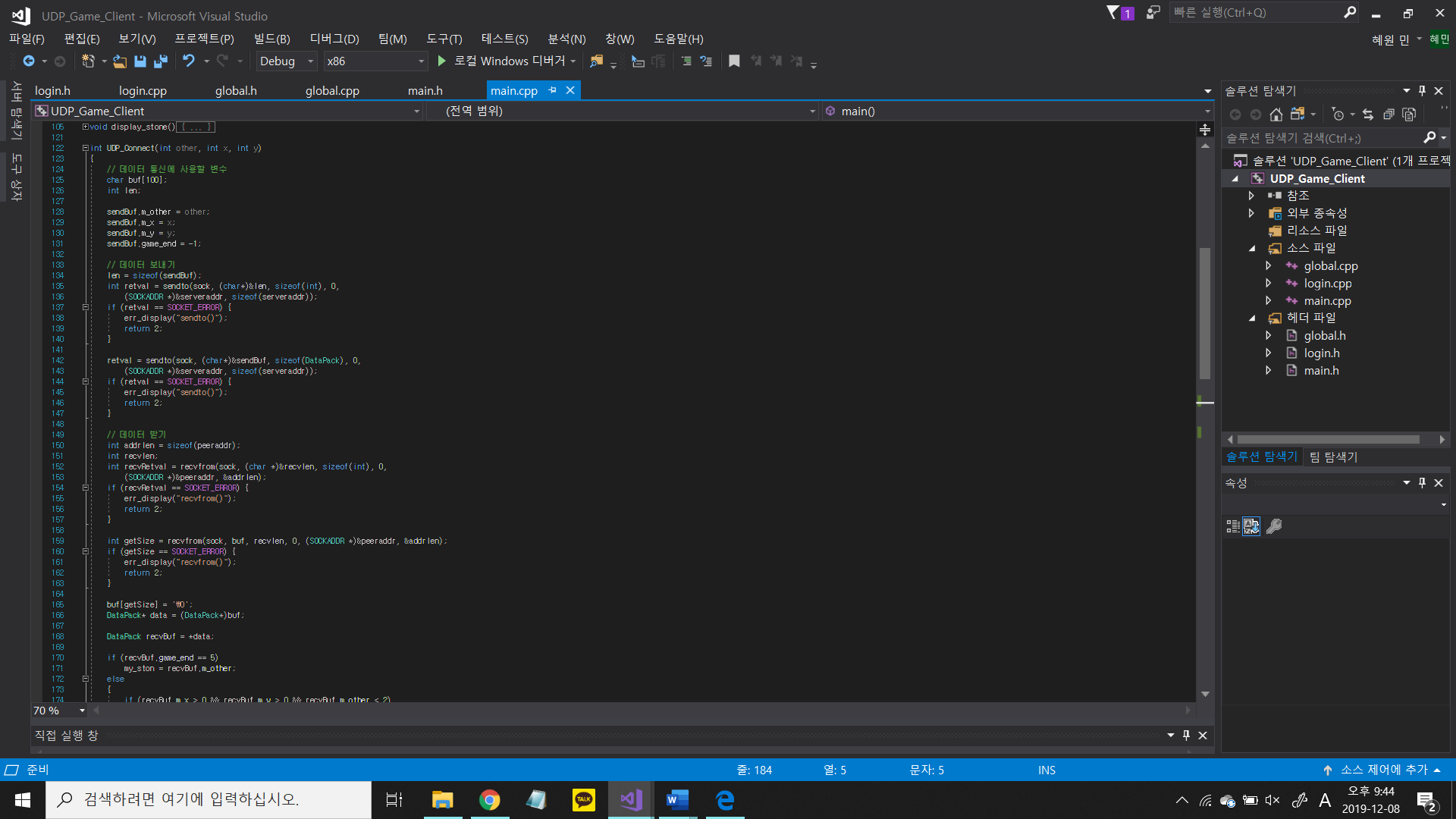
E

D

C

B

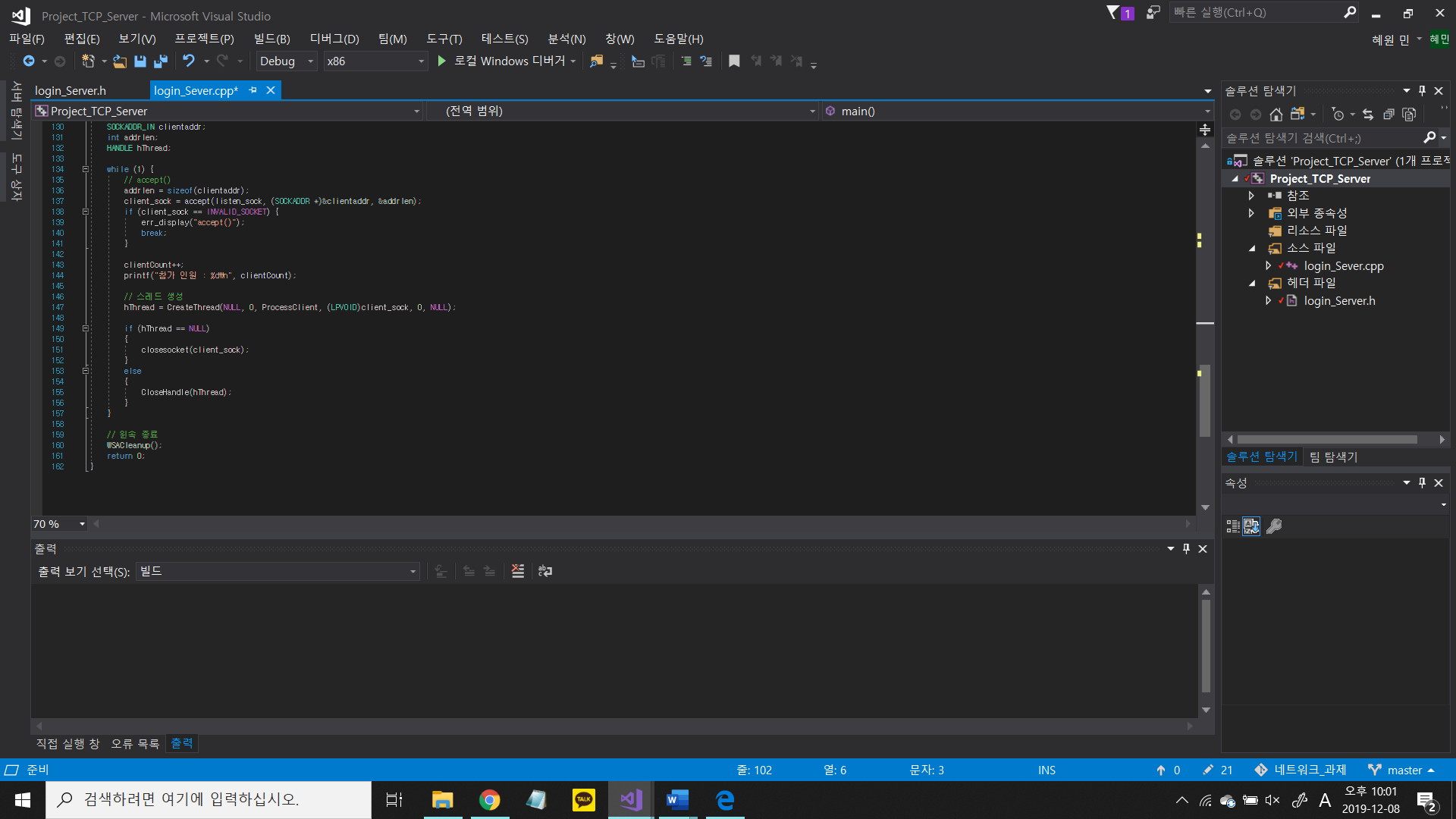
* + - * 1. 비밀번호를 입력받는다. (TCP\_Client 예제를 활용했다.)
        2. send 함수를 사용해 비밀번호를 TCP 서버에 전송한다. (TCP\_Client 예제를 활용했다.)
        3. recv 함수를 사용해 비밀번호가 일치하는지 여부를 TCP 서버로부터 받는다. 이때, 서버가 보내는 정보는 int 형이기 때문에 받는 데이터도 int형으로 받고, 받을 데이터 크기도 int형의 크기(4 바이트)로 설정한다.
        4. 비밀번호가 일치하면 0값을 리턴하면서 종료하고, 틀리면 loop문 처음으로 돌아가며, 방 참가를 거부 당하면 1값을 리턴해 게임 화면을 출력하지 못하게 한다.
    1. Set\_UDP 함수
       1. 위치 : main.cpp
       2. 역할 : UDP 서버와 통신하기 위해 UDP 소켓을 생성하고 소켓 주소를 초기화한다. (UDP\_Server 예제를 활용했다.)
    2. UDP\_Connect 함수
       1. 위치 : main.cpp
       2. 역할 : UDP 서버와의 통신을 담당한다.



B

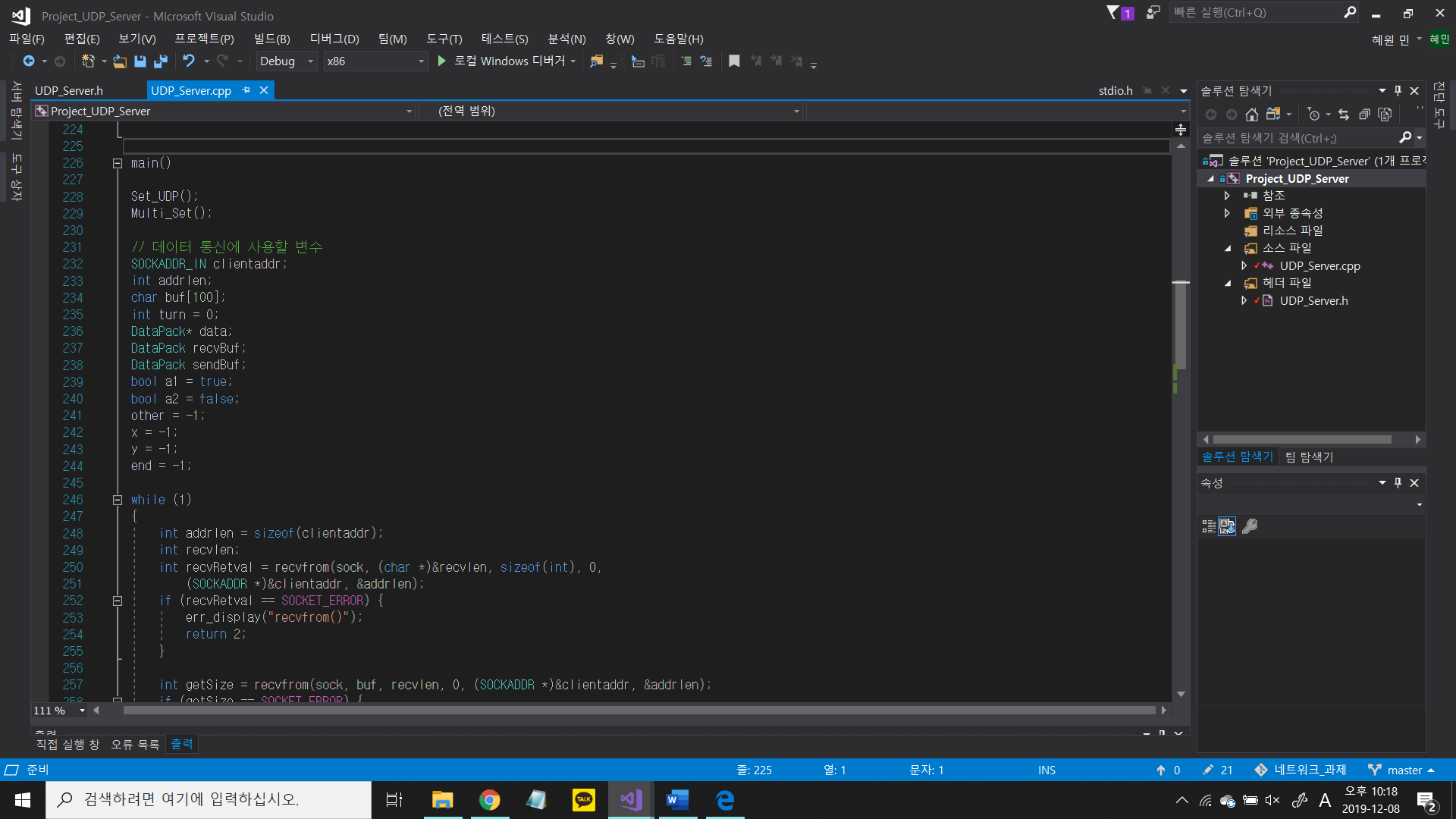
A

* + - * 1. 돌의 정보를 구조체 데이터로 전송한다. 먼저 보낼 구조체 크기를 고정 데이터로 전송한다. Int형으로 전송한다. 그 후 구조체를 가변 데이터로 전송한다.
        2. 서버로부터 돌의 정보를 구조체 데이터로 수신한다. 마찬가지로 받을 구조체 크기를 고정 데이터로 수신하고, 그 후 구조체를 가변 데이터로 수신한다.
        3. 받은 데이터가 상대편이 둔 돌의 데이터가 아니라면(서버로부터 돌의 종류를 받는 경우 등) 게임 판에 넣지 않는다.
  1. TCP 서버 (Project\_TCP\_Server 폴더)
     1. Main 함수
        1. 위치 : login\_Server.cpp



D

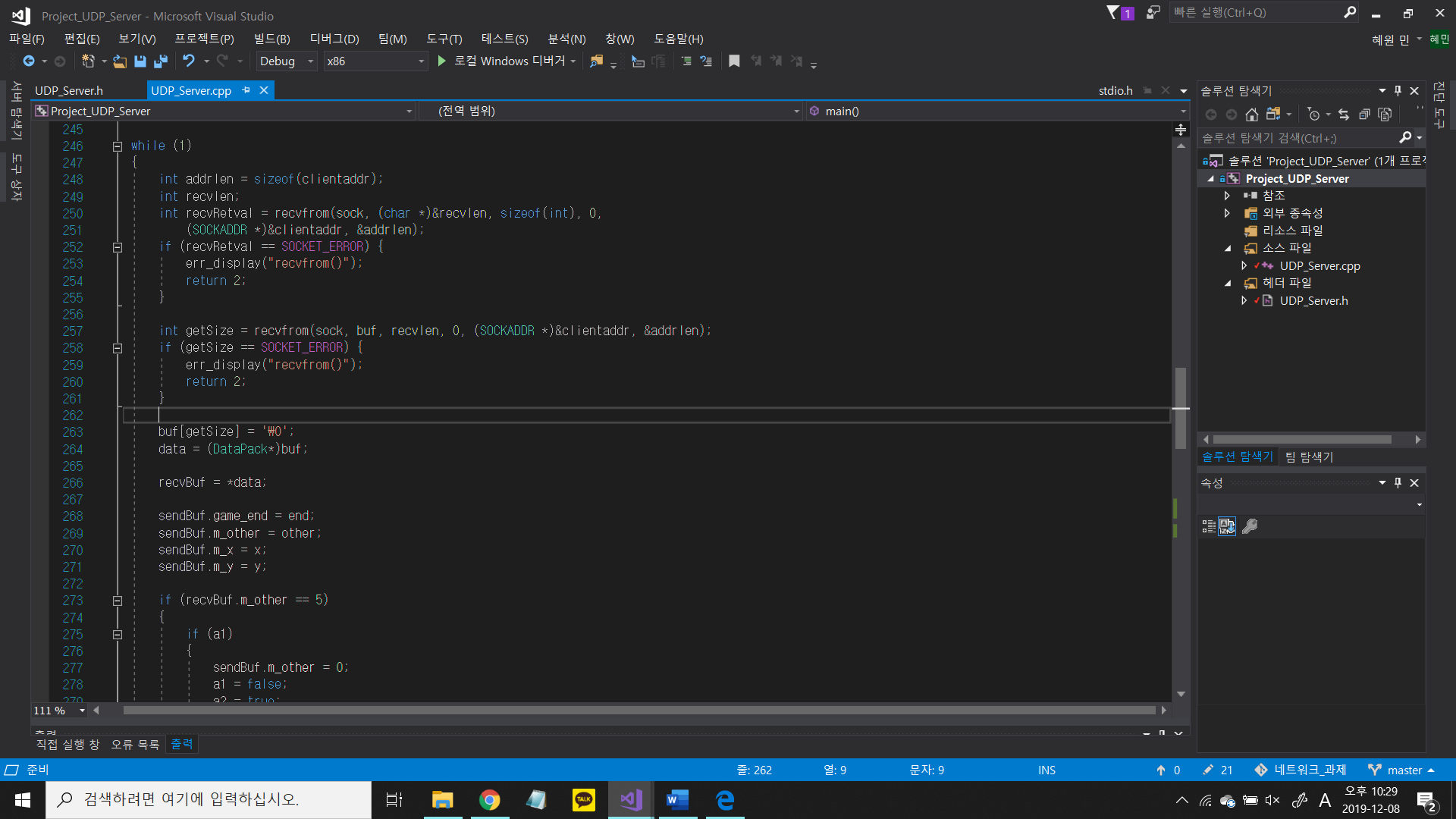
* + - * 1. 비밀번호를 입력 받는다. 비밀번호를 정상적으로 입력할 때까지 loop문을 돌리고, 정상적인 비밀번호를 입력받으면 빠져나온다.
        2. 윈속 초기화, 소켓 생성, 지역 주소 결정, listen 상태로 변경을 수행한다. (TCP\_Server 예제를 활용했다.)
        3. 클라이언트가 연결 요청을 할 때마다 연결 요청을 처리한다.
        4. 연결 요청이 올 때마다 연결할 스레드가 늘어나기 때문에 스레드 카운트를 증가시킨다. 이 정보는 2명 이상 클라이언트가 접속하려하면 방 참가를 거절하기 위해 사용한다.
        5. 연결이 오면 스레드를 생성한다. (스레드에서 돌리는 함수는 클라이언트와 통신을 하는 ProcessClient라는 함수로, TCP\_Server 예제를 활용했다. 받은 데이터를 설정한 비밀번호와 비교해 결과를 전송하는데, 클라이언트 수가 2명 이상이면 방 참가가 안 된다는 결과를 전송한다.)
  1. UDP 서버 (Project\_UDP\_Server 폴더)
     1. Main 함수
        1. 위치 : UDP\_Server.cpp



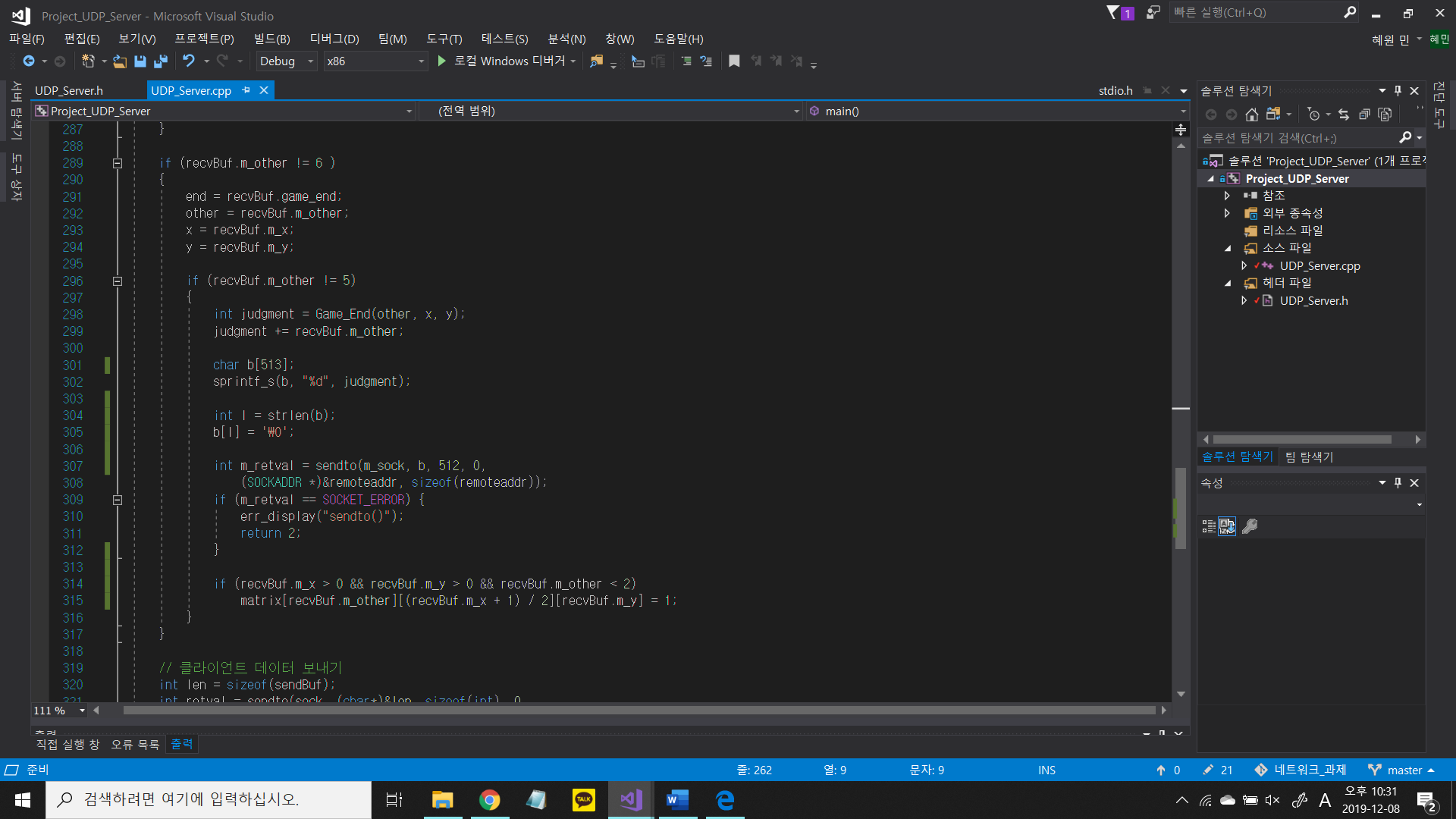
B

A

* + - * 1. UDP 서버와 클라이언트 간의 통신을 위한 윈속 초기화, 소켓 생성, 주소 결정을 수행한다. (UDP\_Server 예제를 활용했다. )
        2. 멀티 캐스트로 멀티 캐스트 리시버에게 데이터를 전송해야 되기 때문에 멀티 캐스트 TTL 설정, 멀티 캐스트 그룹 가입을 수행한다. 멀티 캐스트 통신용 소켓을 새로 생성해주었다. (UDP\_Server 예제 및, Multicast \_Sender 예제를 활용했다.)
        3. 통신에 사용하는 변수들을 선언 및 초기화 해준다.

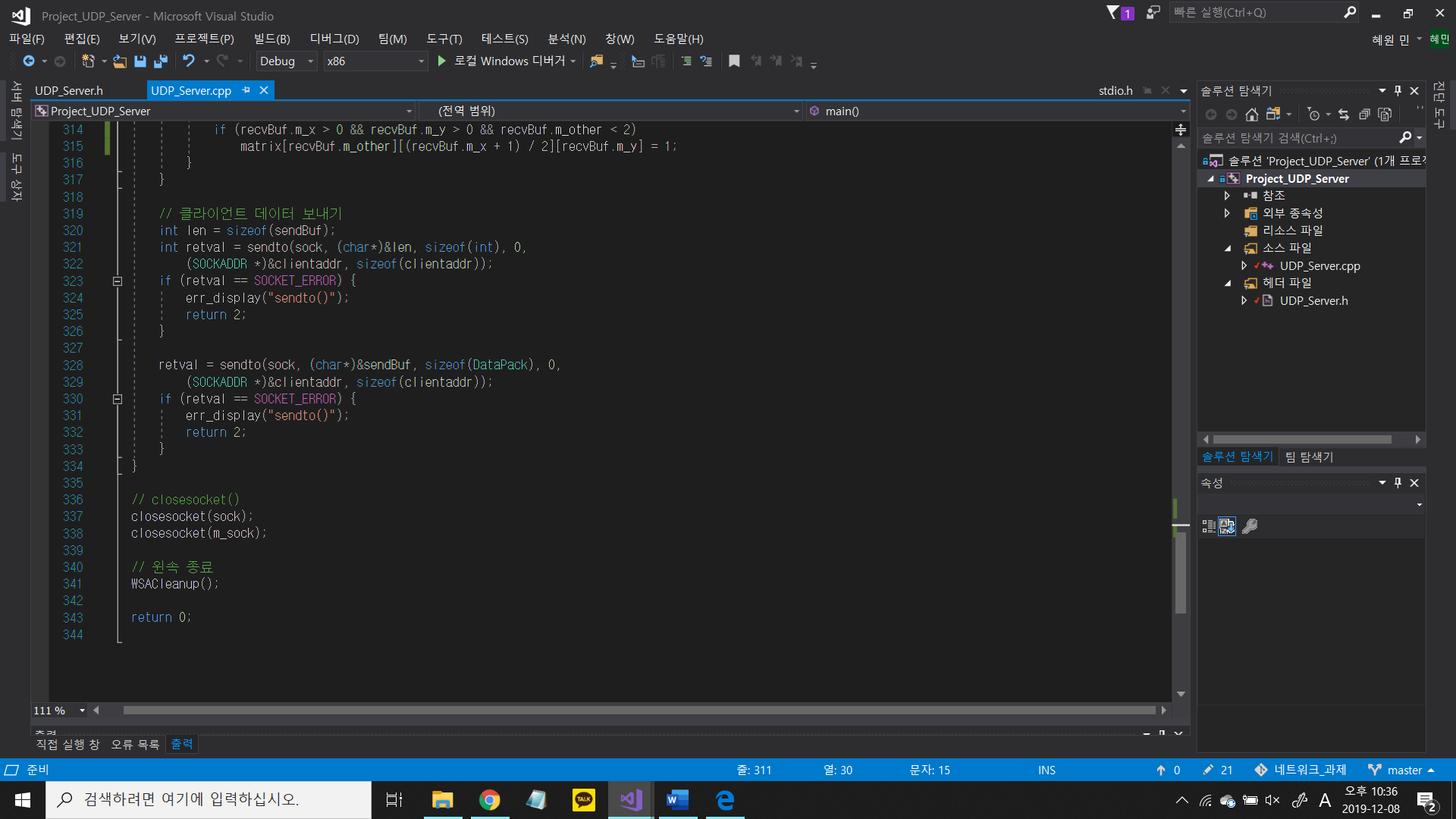


* + - * 1. 클라이언트로부터 데이터를 받는다. 클라이언트와 동일하게 구조체 크리를 먼저 고정 데이터로 받고, 구조체를 가변 데이터로 받는다.



E

* + - * 1. 클라이언트로부터 받은 데이터가 상대편 돌 데이터 요청이나, 돌 배정 요청이 아닌, 자신이 둔 돌 데이터일 경우, Game\_End 함수에서 승패 판정을 해 결과를 받는다. 이 결과와 흑돌인지 백돌인지에 관련된 정보를 합쳐 멀티 캐스트 리시버에게 전송한다.
        2. 클라이언트로부터 받은 데이터가 클라이언트가 둔 돌에 대한 정보일 경우 게임판에 저장한다.



* + - * 1. 클라이언트에게 데이터를 전송한다. 클라이언트와 동일하게 구조체 크기를 고정 데이터로 전송하고, 구조체를 가변 데이터로 전송한다.
        2. 종료 시 소켓을 두 개 사용했기 때문에 (클라이언트와의 통신용, 멀티 캐스트 리시버와의 통신용) 두 소켓을 모두 반환해준다.
  1. 멀티 캐스트 리시버 (UDP\_Multicast\_Client 폴더)
     1. Main 함수
        1. 위치 : 소스.cpp
           1. 윈속 초기화, 소켓 생성, 멀티캐스트 그룹 가입 등을 수행하고 loop문 내에서 recvfrom 함수로 UDP 서버로부터 데이터를 받는다. 받은 데이터를 비교해 차례나 승패를 출력한다. 종료시 멀티 캐스트 그룹을 탈퇴하고 소켓을 반환하고 윈속을 종료한다. (Multicast\_Receiver 예제를 활용했다.)