**네트워크 프로그래밍 보고서(5조)**

2017113223 이승우

2017114482 나현준

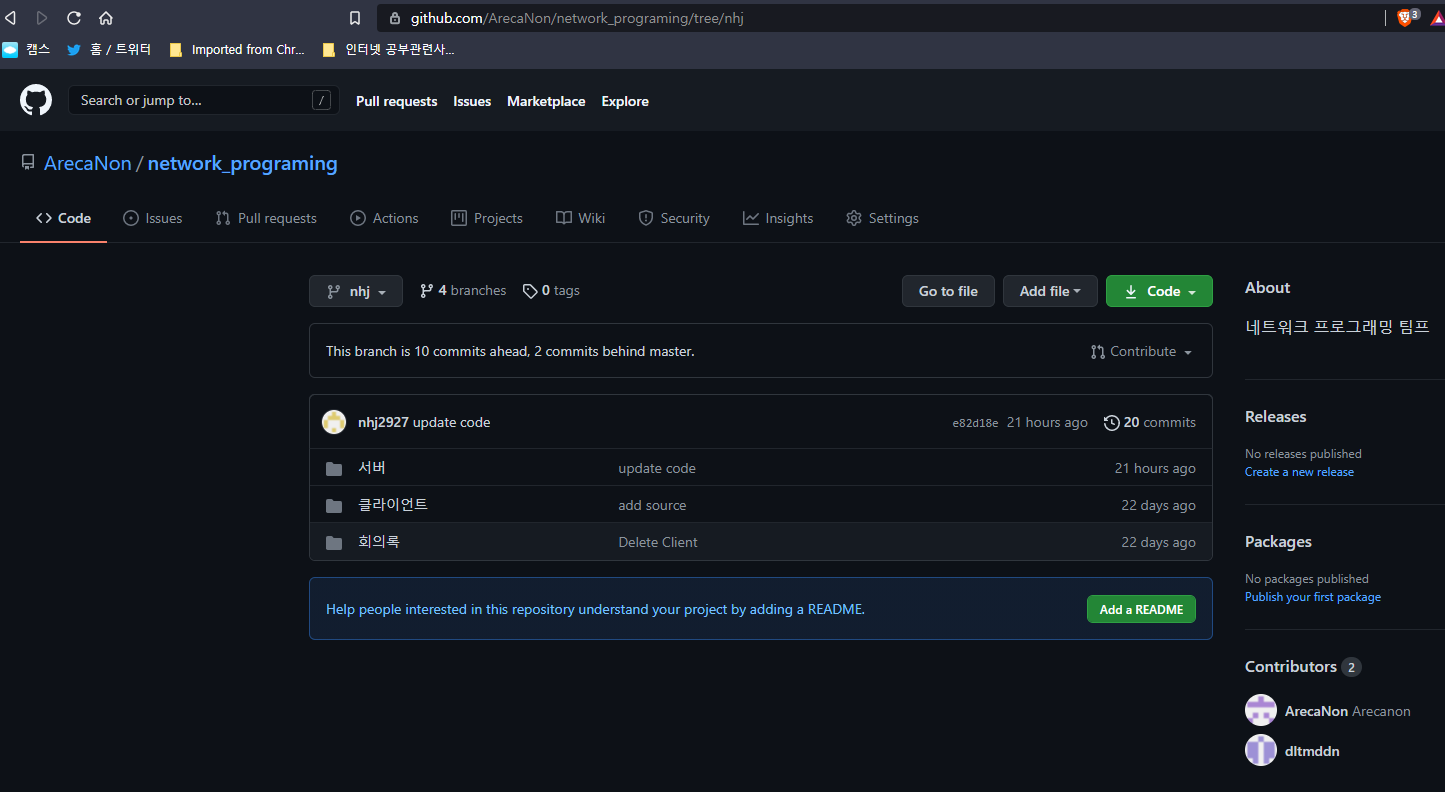
2018112892 황윤호

**1. 주제선정 및 툴 선정방식**

우리 조는 간단한 게임을 제작하자는 의견이 주를 이뤘고 테트리스, 윷놀이, 머드게임 등등 다양한 의견이 나왔으나 가장 네트워크라는 주제와 잘 어울린다고 생각되는 마피아게임을 선택하게 되었다.  
팀원들 대부분이 다른 언어나 툴을 다룰 줄 몰랐기에 강의에서 배운 가장 기본이 되는 c언어를 기반으로 dos창으로 게임을 만들기로 하였다.

**2. 코드 공유 및 작업 공유**

현재 코로나사태로 인해 직접 모여서 작업하기가 힘든 것도 있고, 코드공유가 중요하다고도 생각했기에 깃헙을 이용하여 코드를 공유, 작업을 진행하기로 하였다.

  
깃헙의 화면 일부 - <https://github.com/ArecaNon/network_programing/tree/nhj>

그리고 각자 집에서도 회의를 하기 위하여 디스코드를 이용한 회의를 진행하면서 작업을 해나갔다.

**3. 설계**

**(1) 서버모델 설정**

마피아 게임을 구현하기 위해서는 다수의 클라이언트를 동시에 처리할 수 있는 서버모델을 선택해야 되는데 윈도우 환경에서 사용할 수 있는 후보로 select, multi thread, IOCP 가 있었다. select함수는 간단하지만 함수를 호출할 때마다 인자로 매번 관찰대상에 대한 정보를 전달해야 되고 결과값을 받으면 모든 파일 디스크립터를 대상으로 반복하면서 확인해 주어야 한다. multi thread는 연결하는 클라이언트마다 thread를 생성해 주어야 되기 때문에 자원소모가 크다. 하지만 IOCP는 반복문을 통해 핸들을 찾을 필요도 없고 적절한 thread수를 통해 처리할 수 있기 때문에 좋은 성능을 낼 수 있는 구조라고 생각하였고 따라서 IOCP모델을 선택하여 구현하기로 했다.

**(2) 게임흐름**

1. 클라이언트가 서버에 접속하면 현재 방에 몇 명 들어가있는지 확인할 수 있고 몇 번 방에 들어갈 지 결정할 수 있다. 방들은 서로 독립적으로 게임을 진행한다.

2. 게임 시작 전 플레이어들간 채팅을 통해 소통할 수 있다.

3. 방에 일정 인원이 차면 게임은 자동으로 시작된다.

4. 플레이어들은 역할을 부여 받고 낮이 된다.

5. 낮 동안 플레이어들끼리 채팅을 통해 죽일 플레이어를 정하고 투표를 한다.

6. 투표를 통해 죽은 플레이어를 확인하고 밤이 된다.

7. 밤 동안 마피아들끼리 채팅을 할 수 있으며 죽일 시민을 결정한다.

8. 마피아에 의해 죽은 플레이어를 확인하고 낮이 된다.

9. 마피아 수와 시민수가 같아지거나 마피아가 다 죽을 시 게임은 종료되고 게임 결과를 확인한다.

**(3) 데이터 구조 설정**

서버와 클라이언트간 데이터를 주고 받을 때 사용되는 구조를 설정하였다.

|  |  |  |
| --- | --- | --- |
| 상태 | 데이터 길이 | 데이터 |

모든 구조는 다음과 같은 구조를 기본으로 한다.

1. 시작 메시지

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| A | 데이터 길이 | 플레이어 번호 | 역할 | 회의시간 | 투표시간 |

게임 시작 시 서버에서 클라이언트로 보내는 메시지이다.

2. 채팅 메시지

|  |  |  |  |
| --- | --- | --- | --- |
| B | 데이터 길이 | 플레이어 번호 | 채팅 메시지 |

클라이언트에서 서버로 보내면 해당 방에 있는 플레이어들에게 전달된다.

3. 낮 트리거 메시지

|  |  |  |
| --- | --- | --- |
| C | 데이터 길이 | 죽은 플레이어 번호 |

서버에서 클라이언트에게 낮임을 알려주고 밤에 마피아에 의해 죽은 플레이어의 번호를 전달한다.

4. 밤 트리거 메시지

|  |  |  |
| --- | --- | --- |
| D | 데이터 길이 | 죽은 플레이어 번호 |

서버에서 클라이언트에게 밤임을 알려주고 낮에 투표에 의해 죽은 플레이어의 번호를 전달한다.

5. 투표 메시지

|  |  |  |
| --- | --- | --- |
| E | 데이터 길이 | 투표한 플레이어 번호 |

클라이언트에서 서버에게 투표한 플레이어 번호를 전달한다.

6. 마피아 메시지

|  |  |  |
| --- | --- | --- |
| F | 데이터 길이 | 죽일 플레이어 번호 |

클라이언트에서 서버에게 마피아가 죽인 플레이어 번호를 전달한다.

7. 게임 종료 메시지

|  |  |  |
| --- | --- | --- |
| G | 데이터 길이 | 게임 결과 |

서버에서 클라이언트에게 게임이 종료됨을 알려주고 게임결과를 전달한다.

8. 연결 메시지

|  |  |  |
| --- | --- | --- |
| H | 데이터 길이 | 플레이어 번호 |

서버에서 클라이언트에게 방에 접속한 플레이어 번호를 전달한다.

9. 연결해제 메시지

|  |  |  |
| --- | --- | --- |
| I | 데이터 길이 | 플레이어 번호 |

서버에서 클라이언트에게 방에서 나간 플레이어 번호를 전달한다.

10. 방 인원 메시지

|  |  |  |  |
| --- | --- | --- | --- |
| J | 데이터 길이 | 1번방 현재 인원 | 2번방 현재 인원 |

서버에서 클라이언트에게 현재 방의 인원을 전달한다.

11. 방 선택 메시지

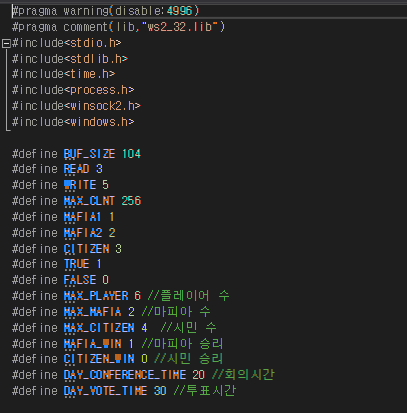
|  |  |  |
| --- | --- | --- |
| K | 데이터 길이 | 방 번호 |

클라이언트에서 서버에게 입장할 방 번호를 전달한다.

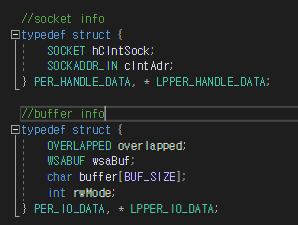
**4. 코드**

**서버**

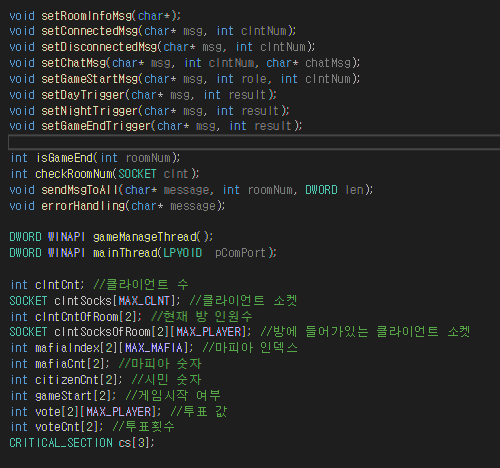
**(1) 헤더 파일**



구현에 필요한 상수들을 정의 하였다.



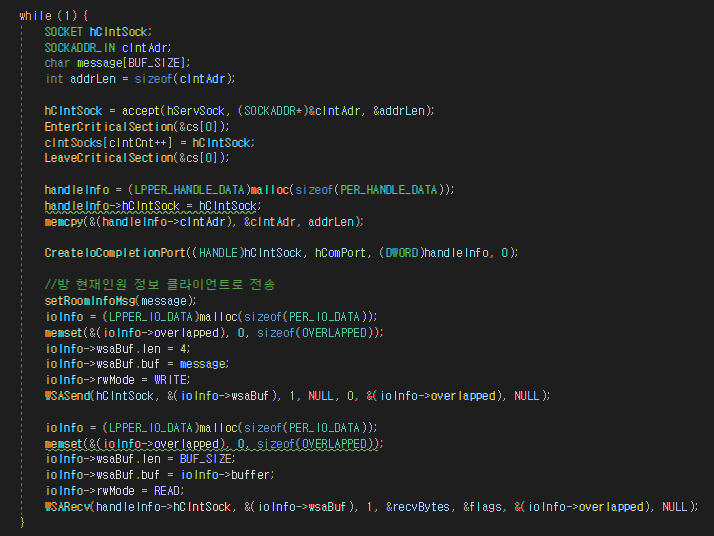
소켓정보와 버퍼정보를 담을 구조체를 선언하였다.



구현에 사용되는 변수와 함수들을 선언하였다.

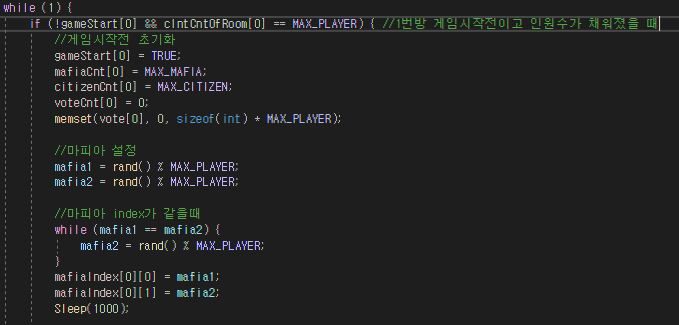
**(2) 소스 파일**

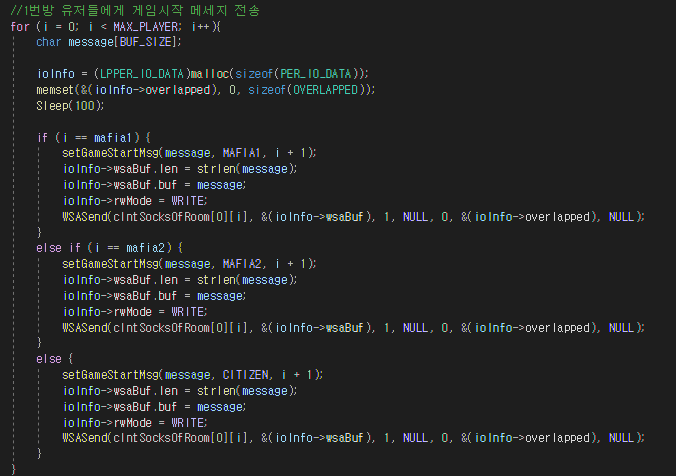
**- main 함수**



서버는 클라이언트 접속이 있을 경우 setRoomInfoMsg 함수를 통해 방의 현재 인원에 대한 정보 메시지를 설정하고 클라이언트에게 전송한다.

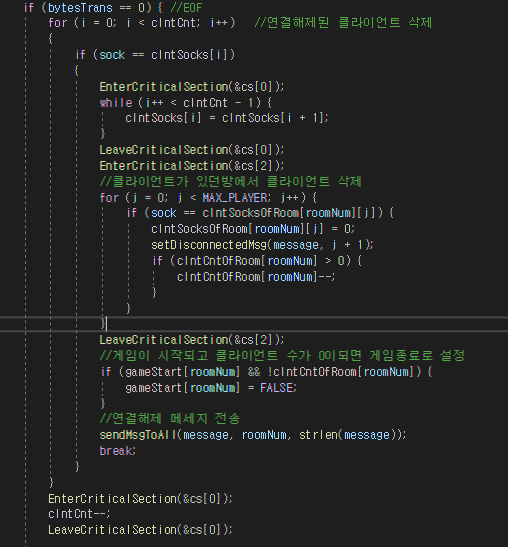
**- gameManageThread**



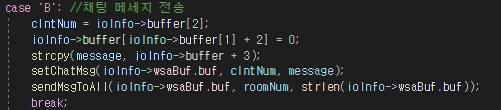


방에 일정 인원이 차면 게임 진행에 필요한 변수들을 초기화 해주며 마피아를 랜덤하게 설정해준다. 그리고 게임시작을 알리는 메시지를 setGameStartMsg를 통해 설정하고 각 플레이이어들에게 전송해준다. 위의 코드는 1번방에 대한 처리이며 2번방도 동일하게 처리된다.

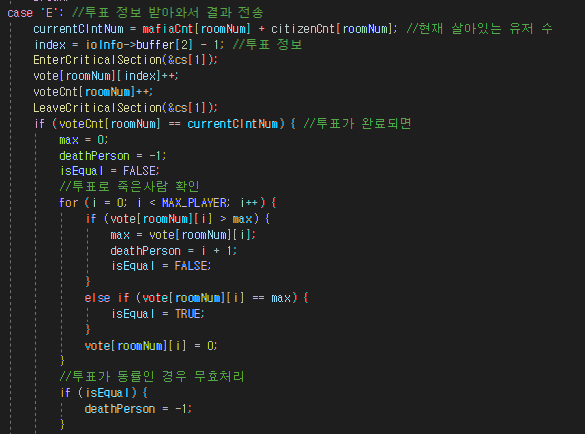
**- mainThread**



클라이언트에서 연결 해제를 하는 부분이다. 해당 클라이언트 소켓을 배열에서 삭제해주며 count도 감소시킨다. 또한 클라이언트가 방에 입장한 상태에서 연결 해제한 경우 해당 방에서 클라이언트 소켓을 삭제해주고 count도 감소시킨다.



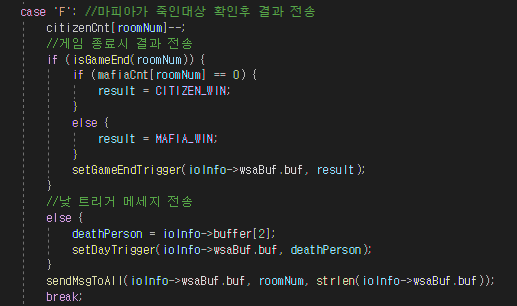
채팅 메시지가 왔을 경우 setChatMsg를 통해 메시지를 설정해주고 그 방에 있는 클라이언트들에게 모두 전송해준다.



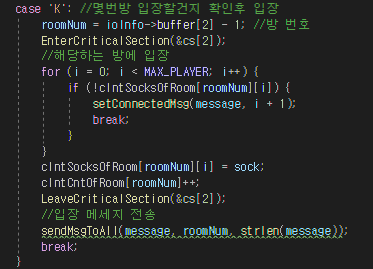


클라이언트에서 투표 정보 메시지를 보낼 경우 가장 많은 투표를 받은 플레이어를 계산한다.

투표한 후 isGameEnd함수를 통해 게임종료 조건을 확인하고 만족하면 setGameEndTrigger을 통해 메시지를 설정하고 만족하지 않으면 setNightTrigger을 통해 메시지를 설정한 후 해당 방의 플레이어들에게 전송해준다.

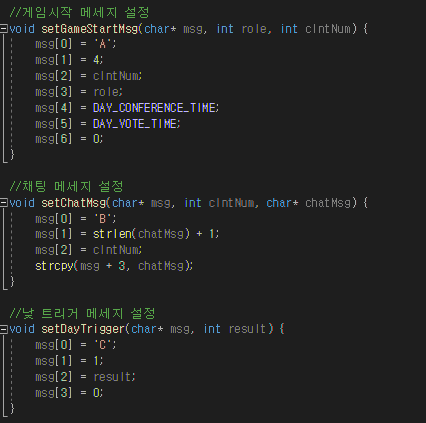


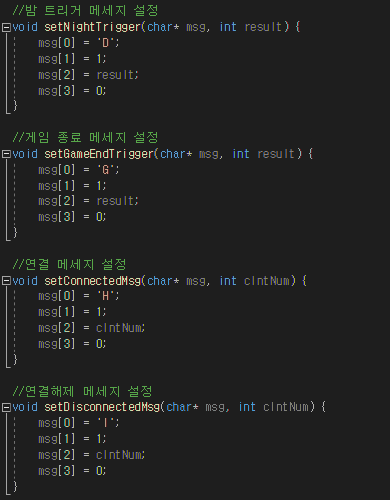
클라이언트에서 마피아가 죽인 플레이어 번호를 보낼 경우 시민수를 줄이고 게임종료 조건을 확인한 후 메시지를 설정해 게임종료 메시지나 낮 트리거 메시지를 해당 방의 플레이어들에게 전송한다.

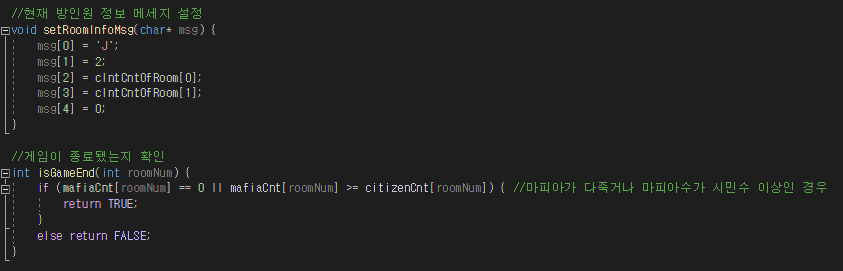


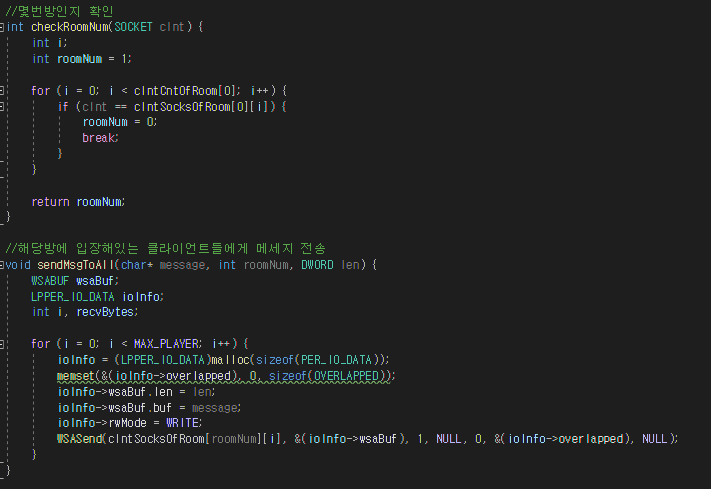
클라이언트가 몇번방에 입장할 건지 확인 후 그 방에 해당하는 배열에 소켓을 추가해주고 count를 증가시킨다.

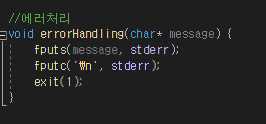
**- 사용되는 함수 정의**







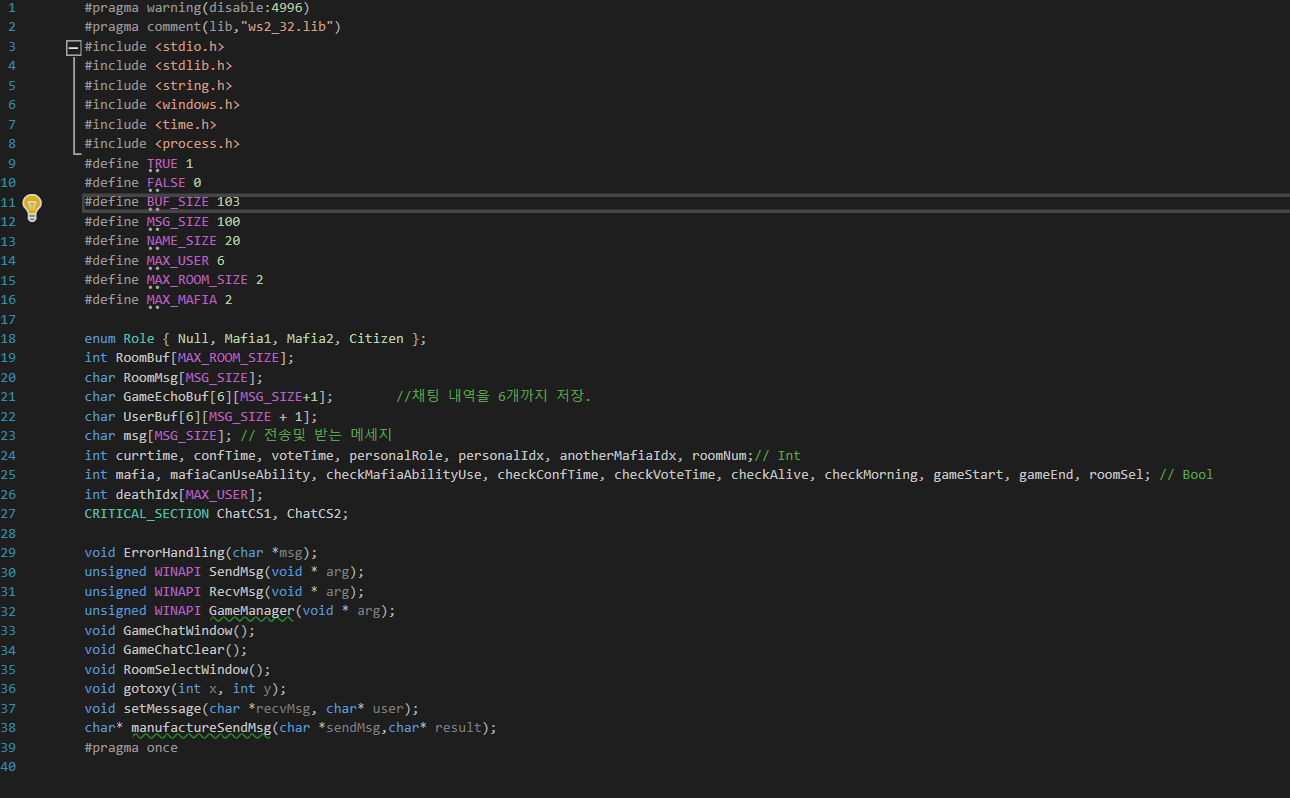




**클라이언트**

c언어의 dos창을 기반으로 제작하였기에 딱히 사용한 api는 없다.  
코드 기반을 수업중에 예제로 사용하였던 클라이언트 코드를 기반으로 만들었으며, 일부 프로그램에서 dos에 gui그리는 방식을 참고해서 사용했다   
이 일부 프로그램은 깃허브에 올라와있으나 참고만 했을뿐 가져다 쓰지는 않았다.

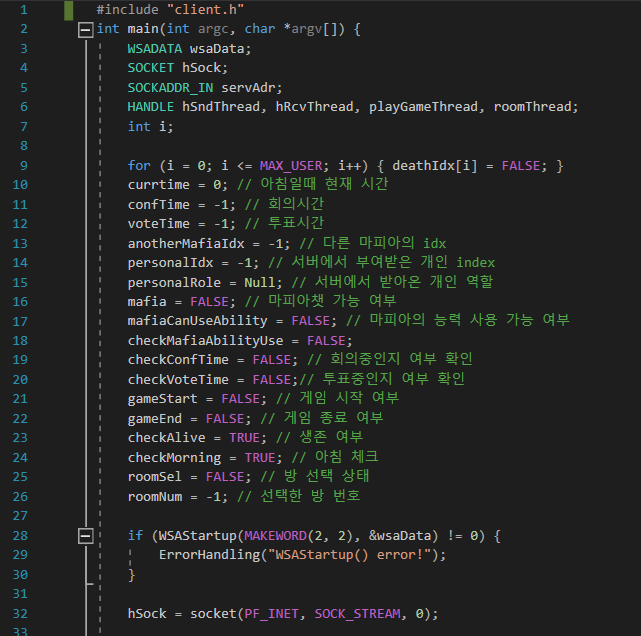
**(1) 헤더파일**



헤더부분으로 함수와 변수를 정의해놨다.   
저희 조에서 정한 규칙에 따라서 변수명과 함수명을 정리해놨다.

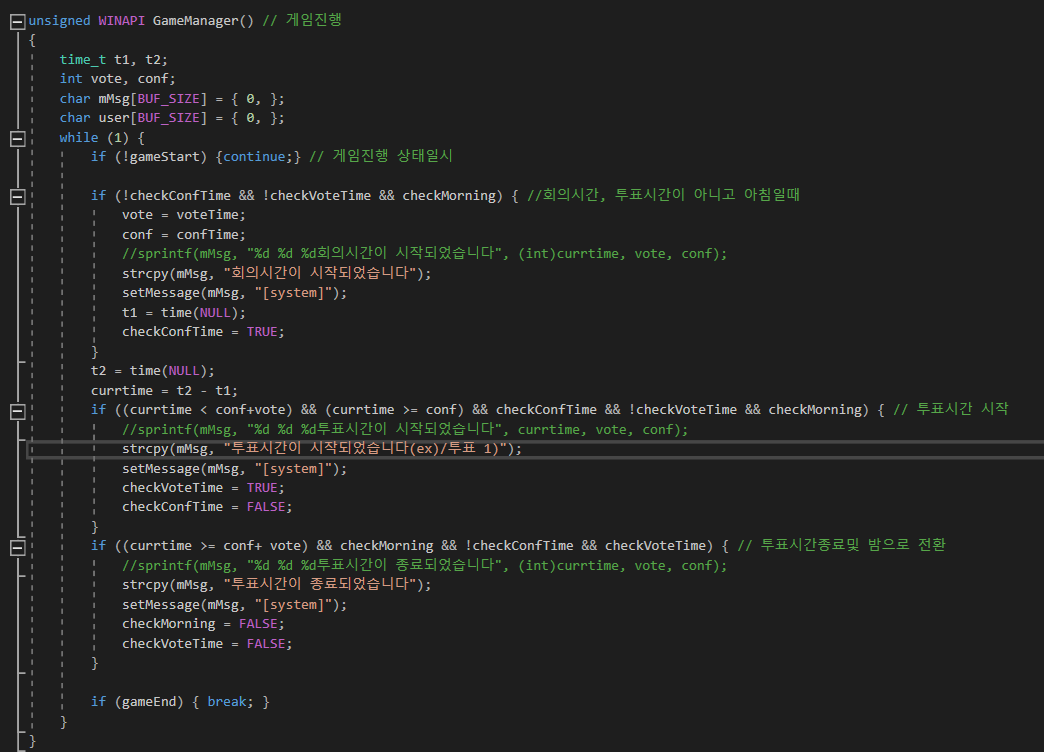
**(2) 소스파일**

**- main함수**

  
변수의 초기화, 소켓생성 및 설정, 임계영역설정, 쓰레드 정의 등이 되어있다..  
쓰레드는 총 hSndThread, hRcvThread, playGameThread로 3가지이다.



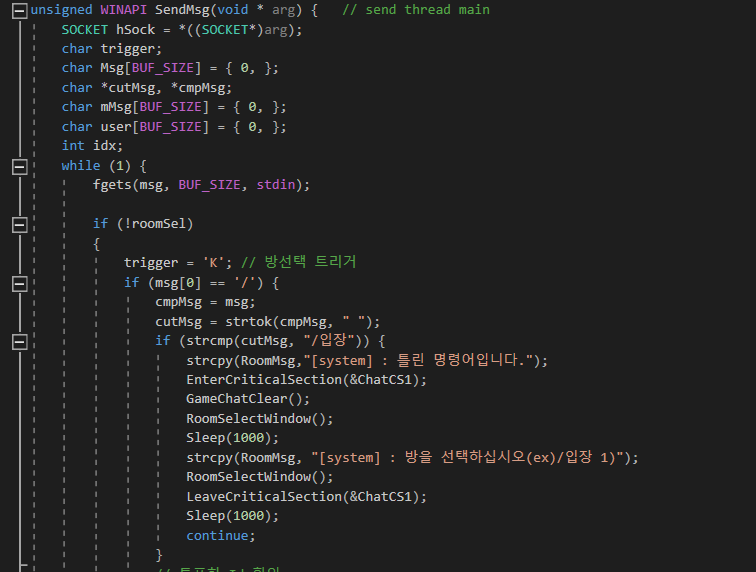
**- GameManager**

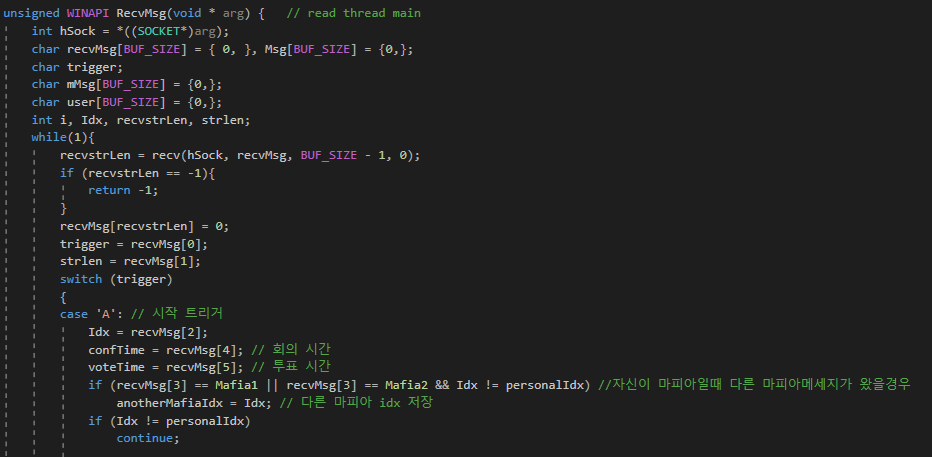
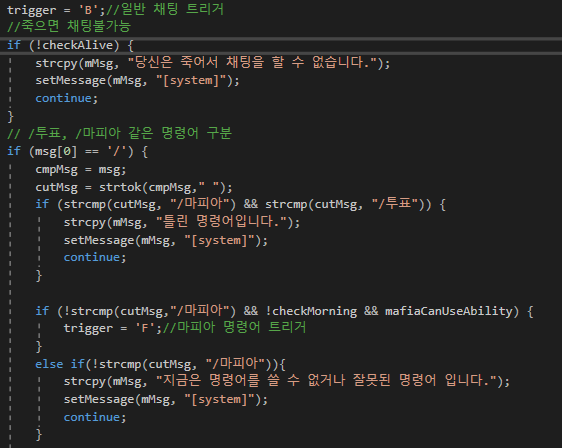


playGameThread에 들어가는 함수인 GameManager로 게임의 시작을 확인하고 게임 시작시 시간을 체크하여서 회의시간, 투표시간, 투표종료를 화면에 출력해주고, 상태변수를 바꿔준다.  
여기서 상태변수는 checkMoring, checkVoteTime과 같이 아침이나 회의시간, 투표시간임을 확인해주는 변수들을 말한다.

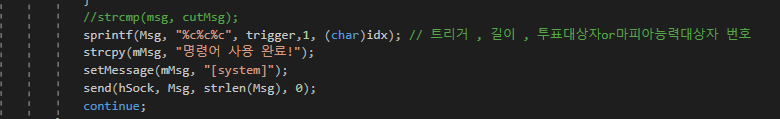


**- SendMsg, RecvMsg**



GameManager에서 수정한 상태변수나 recvMsg에서 수정된 상태변수를 확인한다.(이 경우는 roomSel)  
그것으로 현재 메시지를 작성하고있는 위치가 방선택지역인지, 대기실인지 확인하고 그에 따라서 메시지가 어떤 정보를 담고있는지를 나타내는 trigger변수를 수정한다. 그리고 작성한 메시지가 명령어인지 일반 메세지인지 문자열분리를 통해서 알아내어 최종적으로 해당 trigger에 알맞은 메시지를 서버로 보내주는 역할을 한다.  
또한 상태메세지로는 위치뿐만 아니라 유저의 현재 상태에 맞춰서 해당 명령어를 쓸 수 있는지, 혹은 자신이 메시지를 보낼 수 있느지에 대한 정보를 화면에 보여준다.  


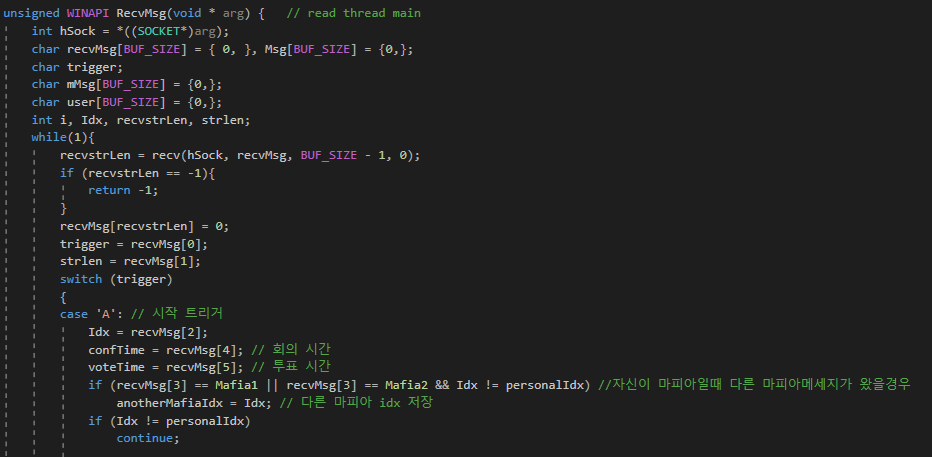
예를 들어서 위와 같은 코드에서는 이미 방에 들어와 있는 것을 roomSel을 통해 알아서 내려왔기에 trigger을 일반 메시지용인 B로 전환, checkAlive로 유저가 살아있는지를 체크해서 죽었으면 채팅이 출력되지 못한다.  
그리고 입력한 메시지가 ‘/’로 시작할 경우 명령어로 보고 문자열분리를 통해서 “마피아”나 “투표”가 아니면 틀린 명령어임을 화면에 표시, 만약 “마피아”이면 trigger을 F로 변경하되 만약 유저가 마피아가 아니거나 밤이 아니라면 명령어를 사용할 수 없음을 알려주고 trigger도 변하지않고, 메시지는, 즉 명령어는 사용되지 않는다.

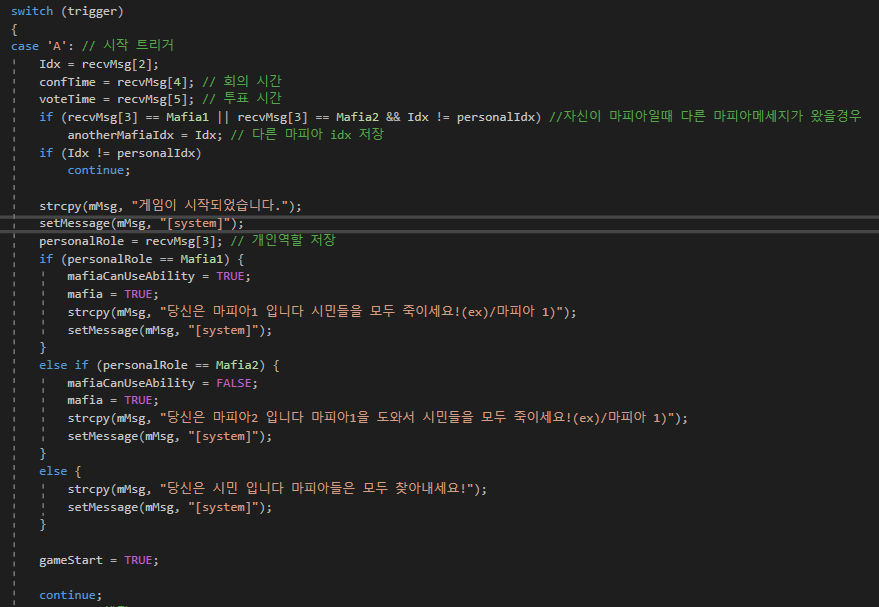


그리고 명령어를 사용한 메시지를 경우 명령어 뒤의 대상 유저를 나타내는 idx만 잘라내어서 sprint를 통해서 문자열을 합치고 이를 서버에 보내준다.

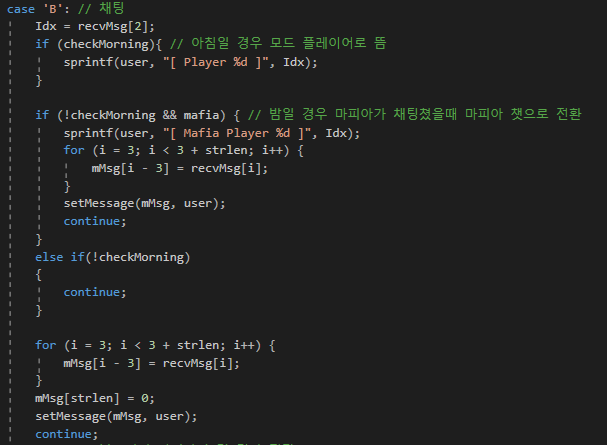


일반 메시지는 그냥 메시지를 트리거와 합쳐주고 서버로 보내주며, 게임이 끝났을 시 종료한다.

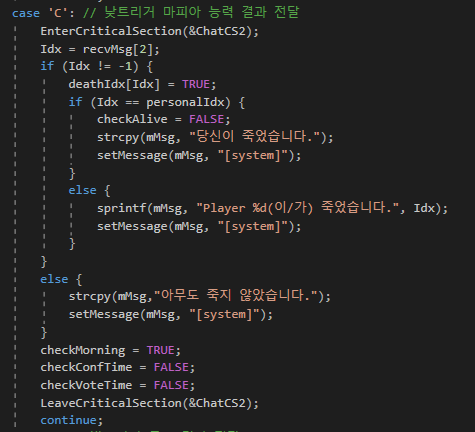


hRcvThread에 들어가는 함수인 RecvMsg로 서버에서 보내는 메시지를 받아서 맨앞의 버퍼에 담긴 trigger에 따라서 각각 알맞은 처리를 한 뒤 화면에 출력해주는 함수이다.  


trigger는 switch문을 통해서 구분 하였으며, 위의 A트리거는 게임인원이 모두 들어왔을 때 유저번호, 유저직업, 회의시간, 투표시간을 받아와서 유저의 변수를 초기화해주고, 각각 직업에 따라서 직업과 해야되는 일을 알려주는 화면을 출력준다.



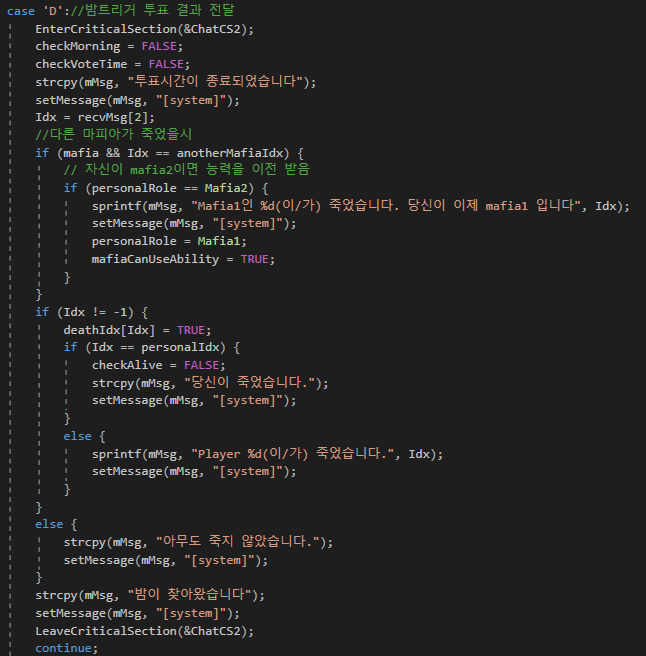
B트리거의 경우 일반 채팅으로서 서버에서 보내준 유저번호와 메시지를 그대로 출력해줍니다.  
그런데 만약 상태변수에서 밤이고, 마피아임이 확인이되면 밤에는 마피아 둘만 서로 채팅을 볼 수 있게 출력해준다.



C트리거는 낮트리거로 상태변수중 checkMoring을 통해서 아침으로 바꿔주고, 마피아의 능력을 통해서 죽은 사람을 판별, 알려줍니다. 그리고 명령어사용시 죽은 사람에게는 못하게 하기위해서 deathIdx를 통해서 죽은 사람을 저장해준다.

또한 자신이 죽었다면 checkAlive를 FALSE로 바꿔서 죽었다는 것을 알려준다.

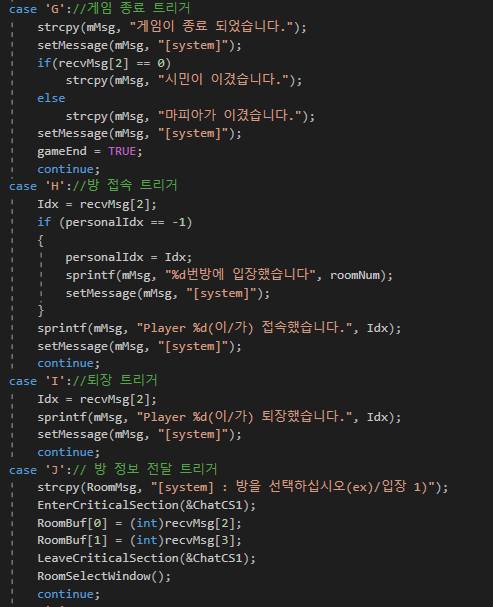
GameManager과 메시지가 겹치지않게 하기위해서 임계영억을 설정해주었다.



D트리거는 밤트리거로 투표에 의해서 죽은 사람을 알려주고, 이때 마피아가 죽었으면, 마피아의 종류(1,2 1이 메인 마피아로 능력사용가능)를 알아내어서 1번 마피아가 죽었으면 mafiaCanUseAbility를 TRUE로 바꿔주고, personalRole도 Mafia1로 바꿔주어서 2번 마피아에게 능력을 넘겨준다.

그리고 C트리거와 마찬가지로 죽은사람을 처리하고, 만약 죽은사람이 없으면 없음을 출력해준다.

GameManager과 메시지가 겹치지않게 하기위해서 임계영억을 설정해주었다.



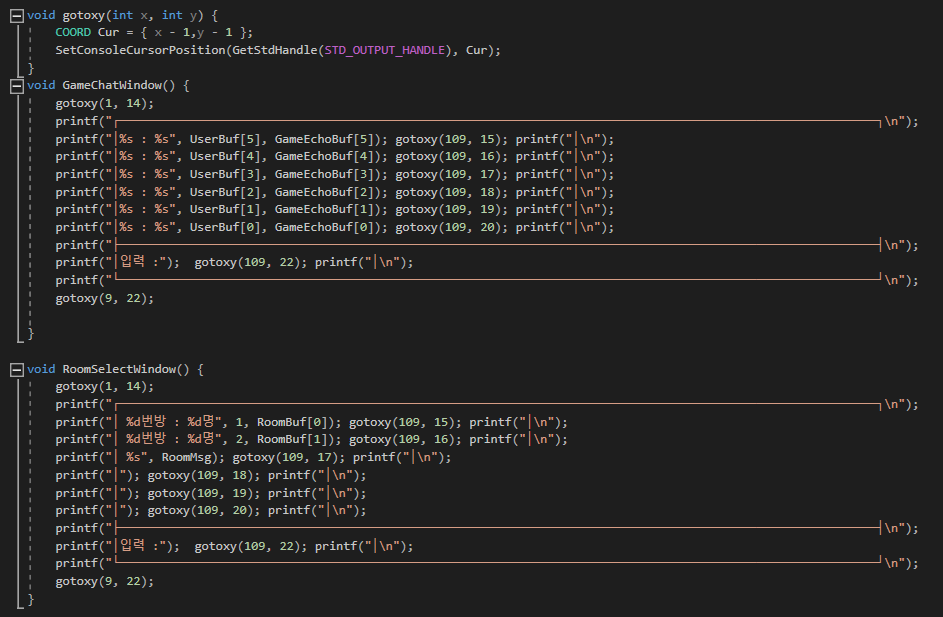
G트리거는 게임종료를 나타내며, 받은 정보를 통해 누가 이겼는지 알려준다.

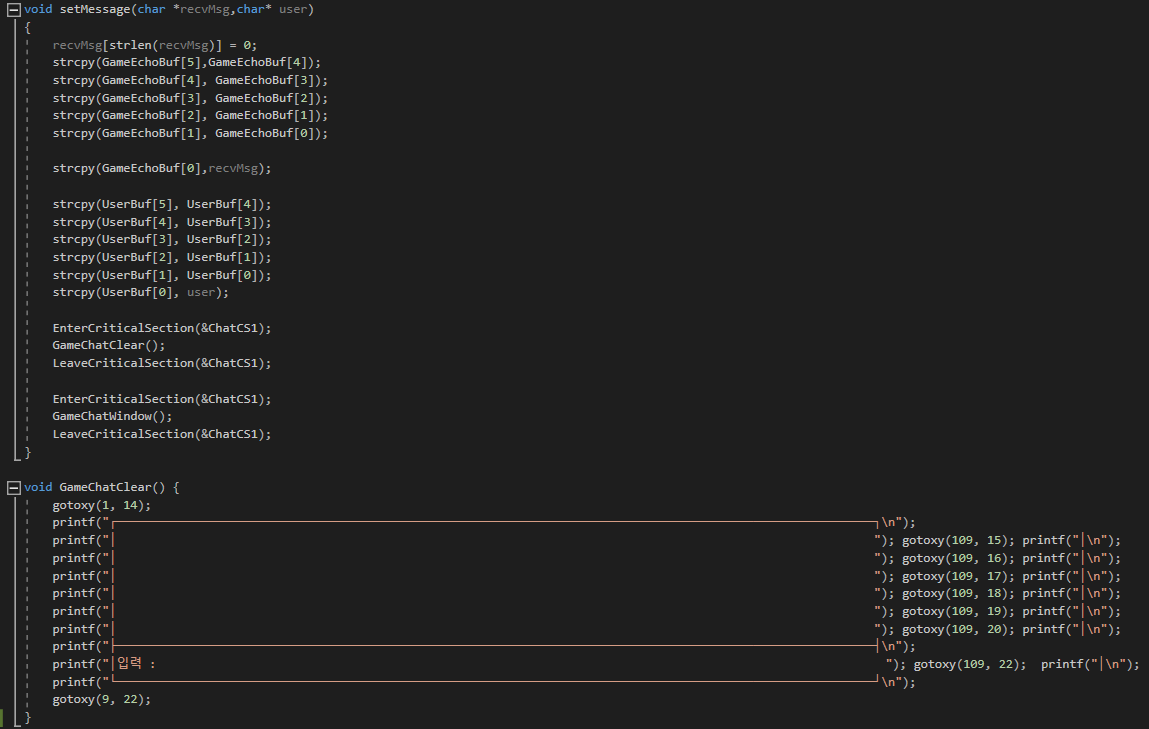
H트리거는 방접속을 알려주며, 누가 방에 들어왔는지 유저번호를 통해 알려준다.

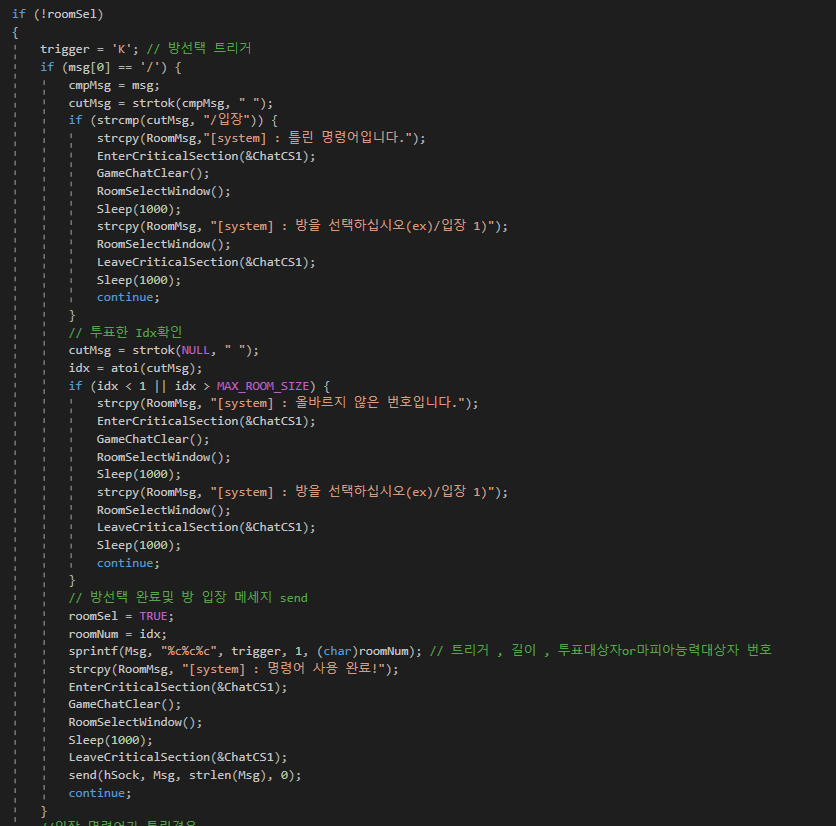
I트리거는 퇴장을 알려주며 누가 나갔는지 알려준다

J트리거는 방정보를 알려주며 현재 1번방과 2번방에 몇 명이 접속해 있는 정보를 알려준다.

**- GUI**



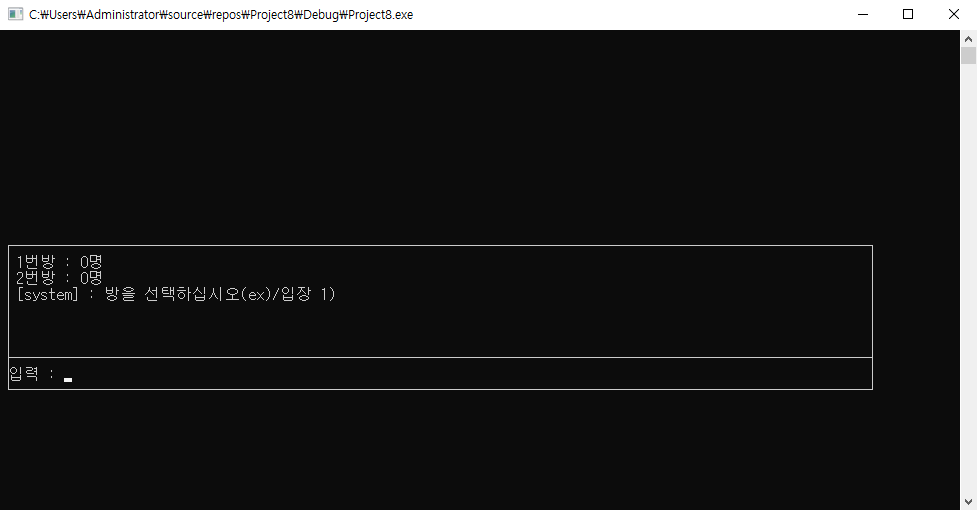
화면을 출력해주는 함수들인 GameChatWindow, RoomSelectWindow와 메시지를 설정해주고 6개의 메시지까지만 저장, GameChatWindow에 출력해주는 setMessage, 화면을 초기화해주는 GameChatClear함수이다.

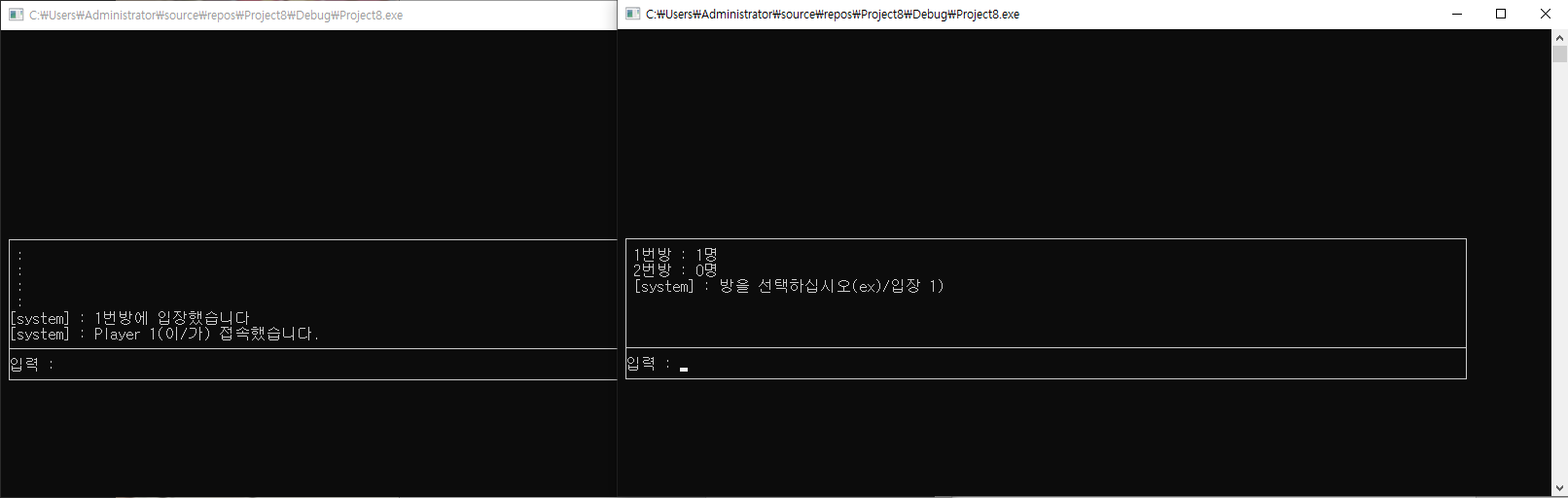
기본적으로 setMessage에 임계영역이 잡혀있어서 다른 함수에서는 임계영역을 잘 잡지 않았으나,  


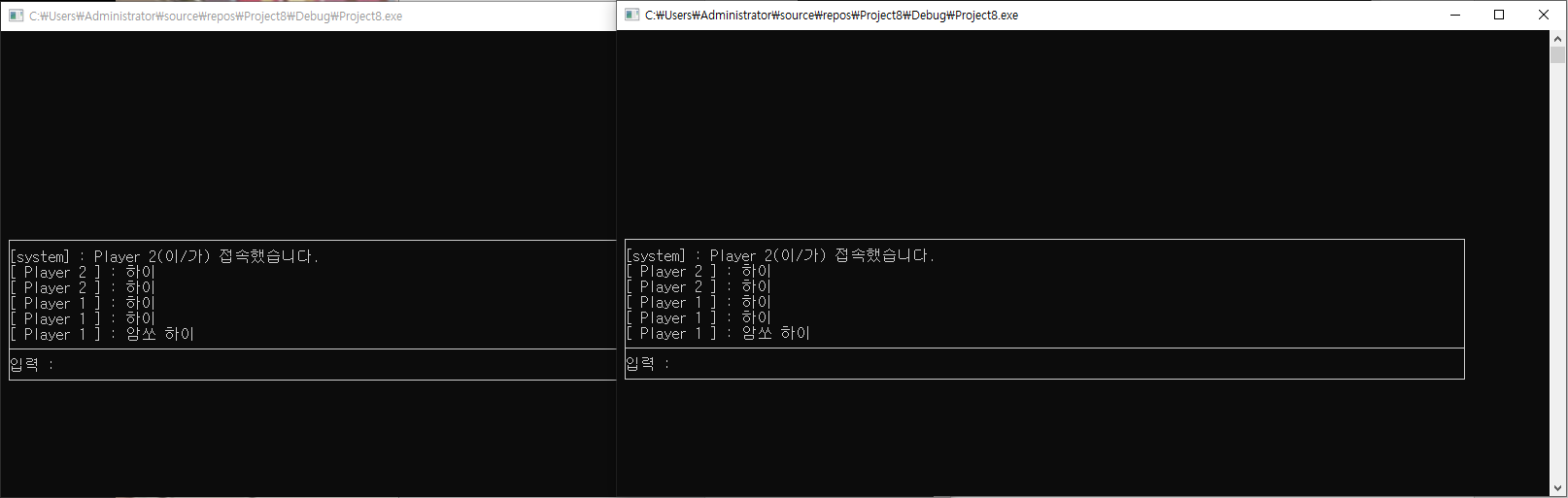
RoomSelectWindow는 위의 SendMsg함수처럼 그때그때마다 임계영역을 잡아주었다.

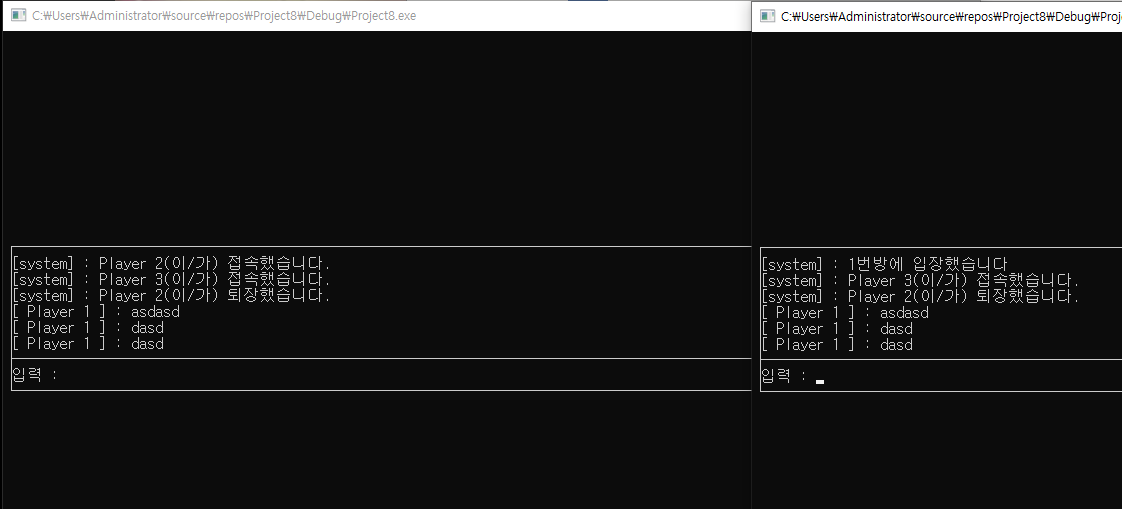
GUI관련 함수들은 네트워크에는 크게 영향이 없다 생각하여서 다른 프로그램에서 참고하여서 사용하였다.

**5. 결과**

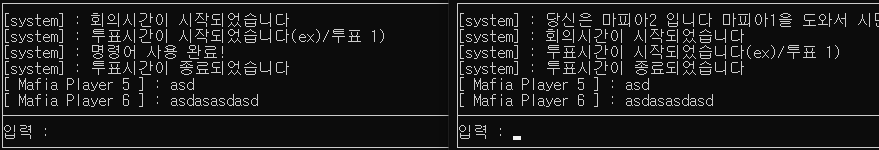
첫 입장 시 화면

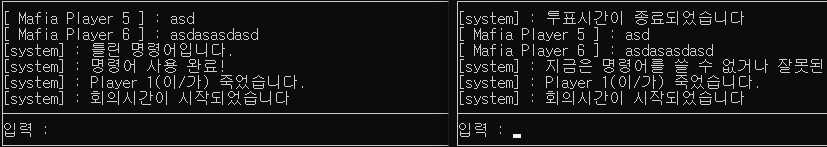
1명이 1번방에 접속해 있고 새로운 클라이언트를 켰을 때 화면

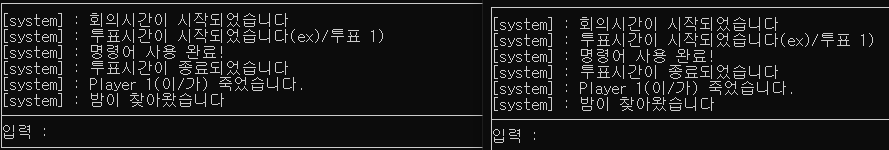
클라이언트 간의 대화 모습

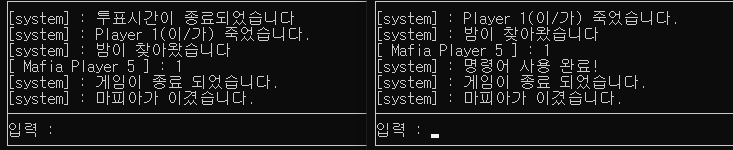
퇴장했을 때 모습

게임 시작된 모습

마피아 챗

마피아 능력 사용

투표 결과

게임 종료