**<실행 환경 설명>**

AWS EC2 인스턴스를 활용하여 두 코드중 server.cpp는 AWS 인스턴스에서, client.cpp는 cmd를 통해 ubuntu로 만든 서버에 접속하여 실행하려 하였으나, cpp 파일을 인스턴스나 ubuntu 서버에 옮기는 과정이 의도대로 되지 않아 cmd창을 두 개 열어 놓고, 제 PC의 IP주소로 접속하여 같은 PORT를 입력하여 통신이 실행되는 환경을 만들었습니다.

**<실행도>**

**Window close (num)**

**Window open (num)**

**클라이언트 접속 요청**

**통신 종료**

**Window open (num + 1)**

**Window close (num)**

**클라이언트 접속 허용**

**client.cpp**

**server.cpp**

**클라이언트 접속 요청 대기**

**IP / PORT 정보 결합**

**socket 생성**

**r.cpp**

**socket 생성**

**<server.cpp>**



<stdio.h> - printf()

<stdlib.h> - exit()

<string.h>

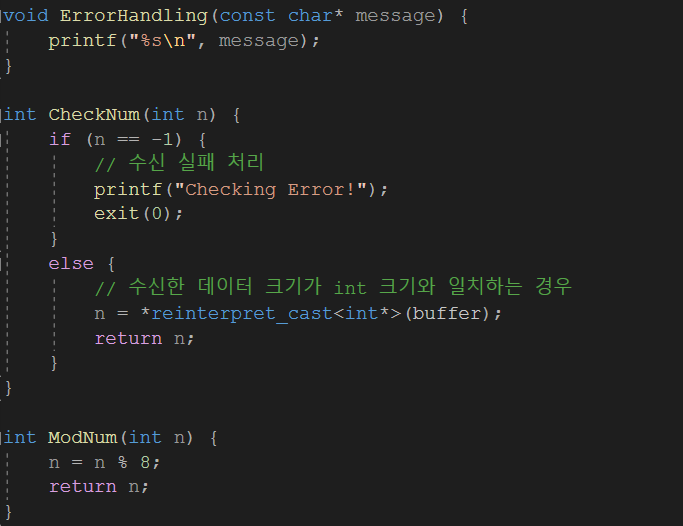
- memset() : 배열이나 구조체 등을 초기화

- strlen() : 문자열 길이 반환

- strcmp() : 문자가 같은지 비교

<WinSock2.h>

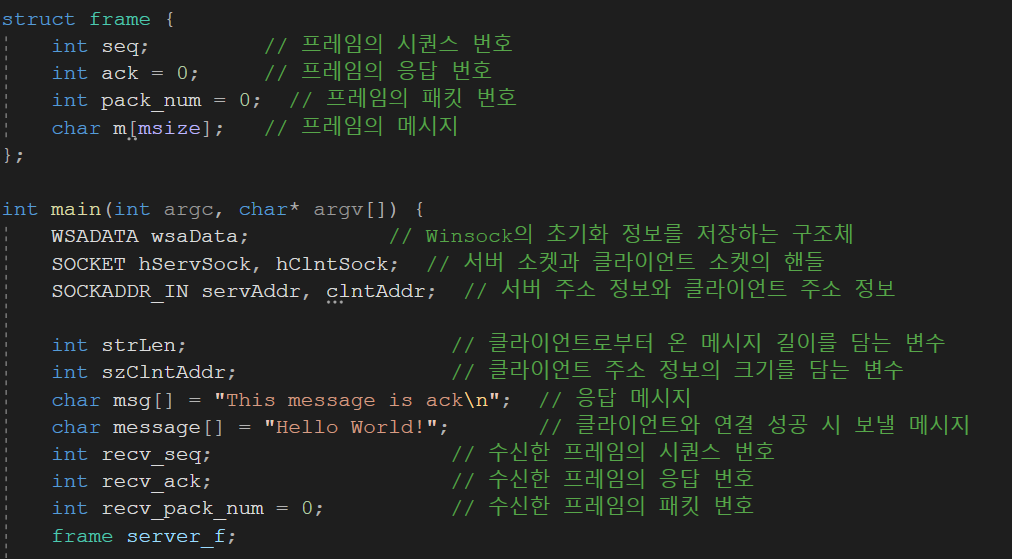
* WSAStartup() : 소켓 라이브러리 초기화
* socket() : 소켓 생성
* bind() : 소켓에 IP 주소와 PORT를 바인딩
* listen() : 소켓을 대기 모드로 설정하고 클라이언트의 연결을 대기
* accept() : 클라이언트의 연결을 수락
* send() : 소켓을 통해 데이터를 전송
* recv() : 소켓을 통해 데이터를 수신
* closesocket() : 소켓을 닫음
* WSACleanup() : 소켓 라이브러리를 정리하고 사용한 자원을 해제
* WSADATA : Windwos 소켓 초기화에 사용되는 구조체
* SOCKET : 소켓을 나타내는 데이터 타입
* SOCKADDR\_IN : 소켓의 주소 정보를 저장하는 구조체

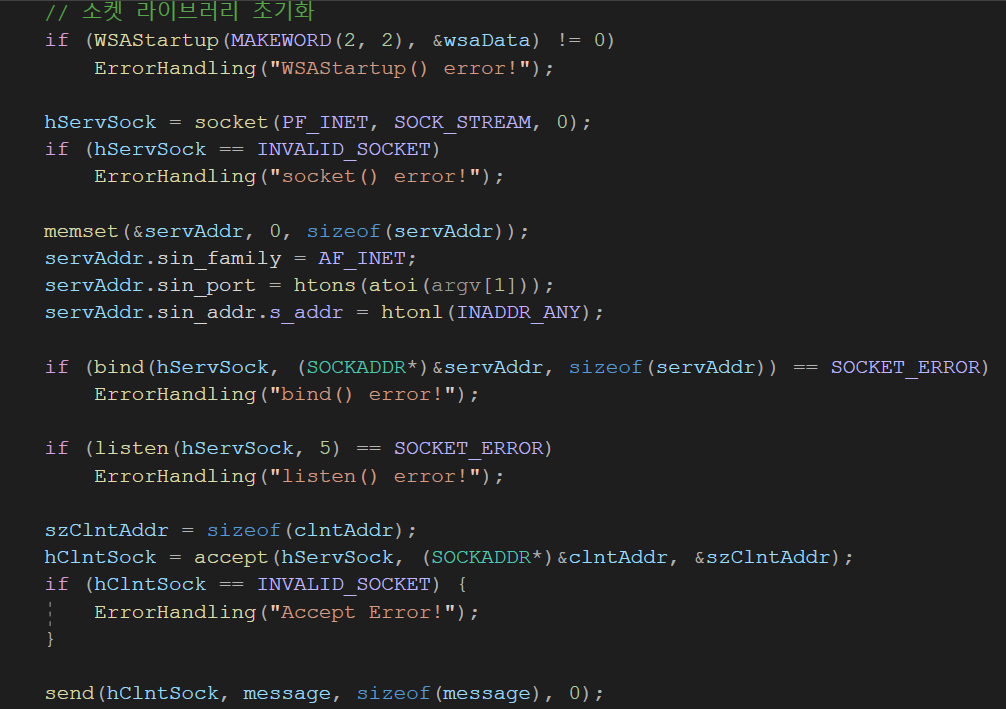


ErrorHandling() : 특정 함수가 정상적으로 시행되지 않았을 때 에러가 발생했음을 알리는 출력 함수

CheckNum() : 상대 측에서 데이터를 char 형태에서 int 형으로 바꾸며, 수신 처리 실패시 프로그램을 종료하는 함수

ModNum : 총 window size = 8 로 가정하여 8 이상일 때 나머지 연산자를 사용하여 반환하는 함수





1. 소켓 라이브러리 초기화

2. hServSock 이름의 소켓 생성

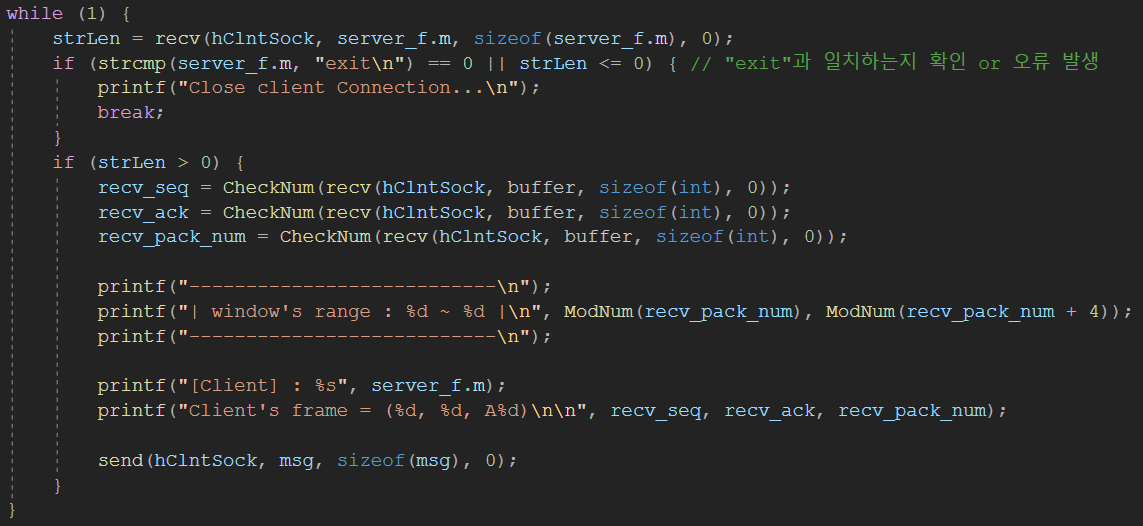
3. IP 주소와 PORT 번호를 받을 servAddr 설정

4. hServSock 소켓에 servAddr 데이터를 적용

5. 클라이언트 측으로부터 연결을 대기

6. szClntAddr 변수에 클라이언트 주소 정보를 담고 연결을 수락

7. 클라이언트에 연결이 성공했다는 메시지 전송



1. 클라이언트로부터 특정 메시지를 메시지를 받음

2. strLen 변수를 통해 정상적으로 데이터를 받았는지 확인

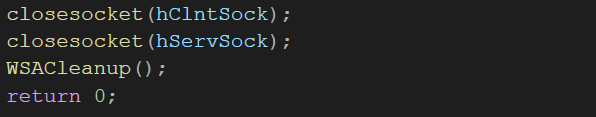
3. 정상적일 경우 클라이언트의 frame 의 seq, ack, pack\_num 정보를 받음

4. 현재 열려 있는 window의 범위를 출력

5. 클라이언트로부터 어떤 메시지가 왔고 받은 프레임의 seq, ack, pack\_num 을 출력

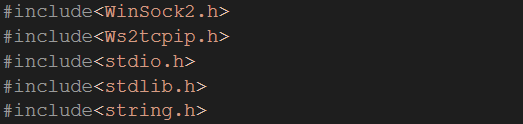
(pdf의 A one-bit Sliding Window Protocol 모델을 기준으로 프레임의 정보를 출력함)

6. 클라이언트 측에 데이터가 성공적으로 도착했다는 ack를 보냄



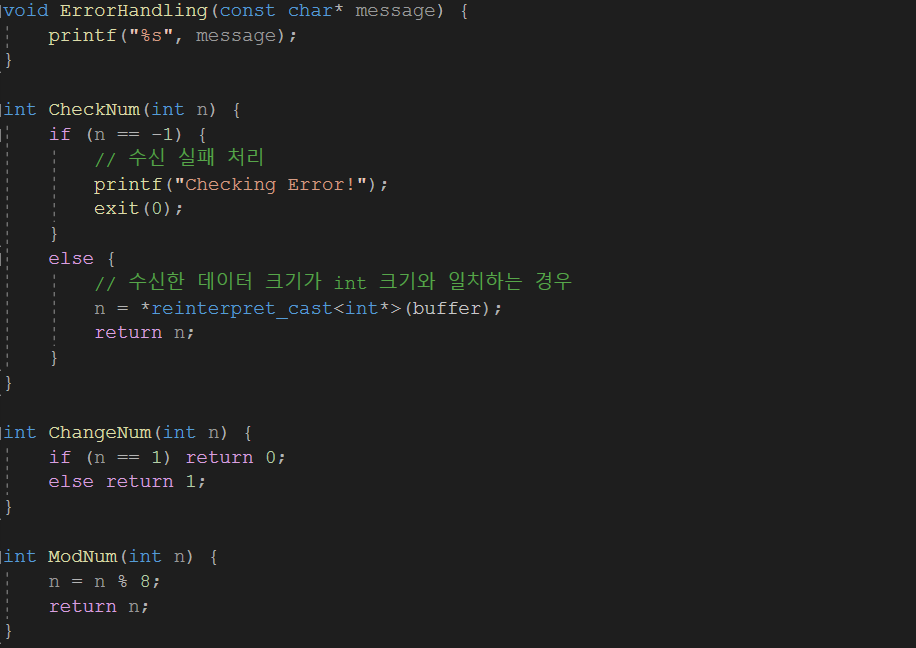
연결이 종료되면 각 소켓을 닫고 자원을 반환 후 프로그램 종료

**<client.cpp>**



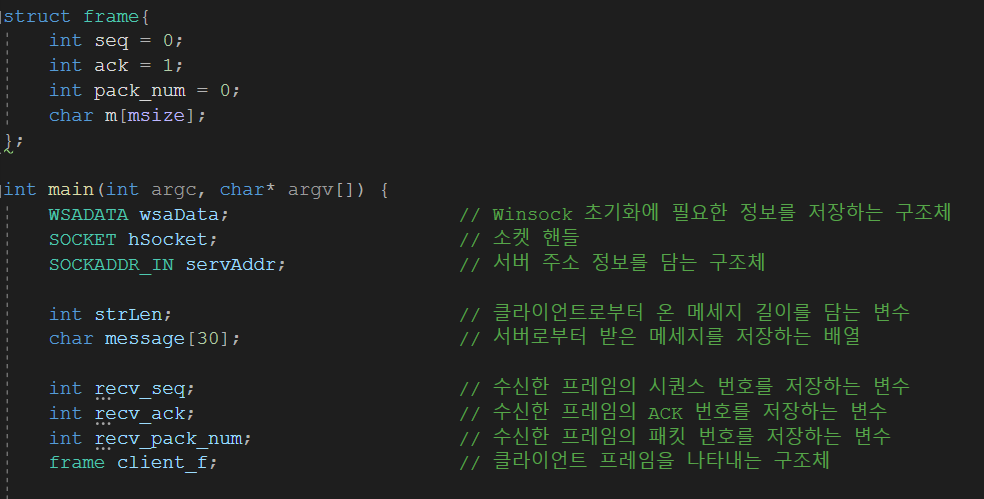
『server.cpp에서 설명한 부분은 생략』

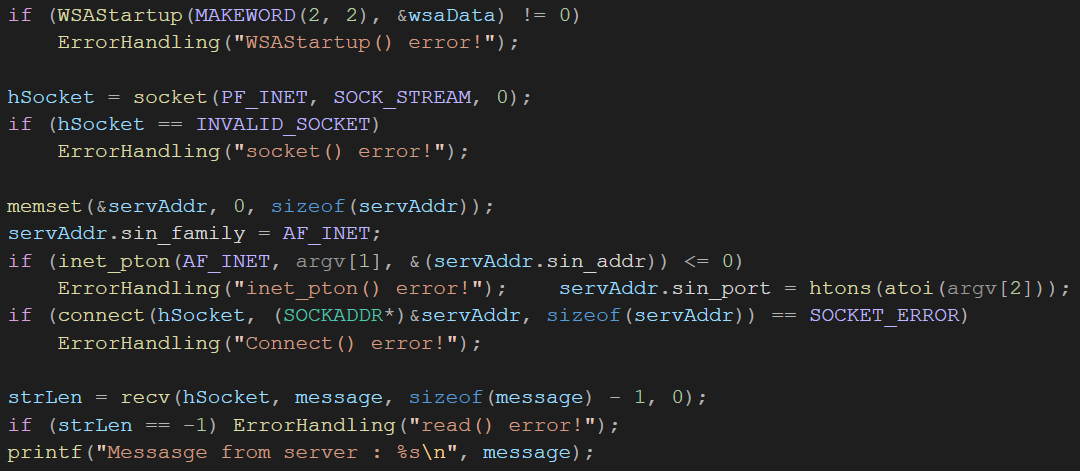
<Ws2tcpip.h> - inet\_pton() : IP 주소를 텍스트에서 바이너리 형식으로 변환



『server.cpp에서 설명한 부분은 생략』

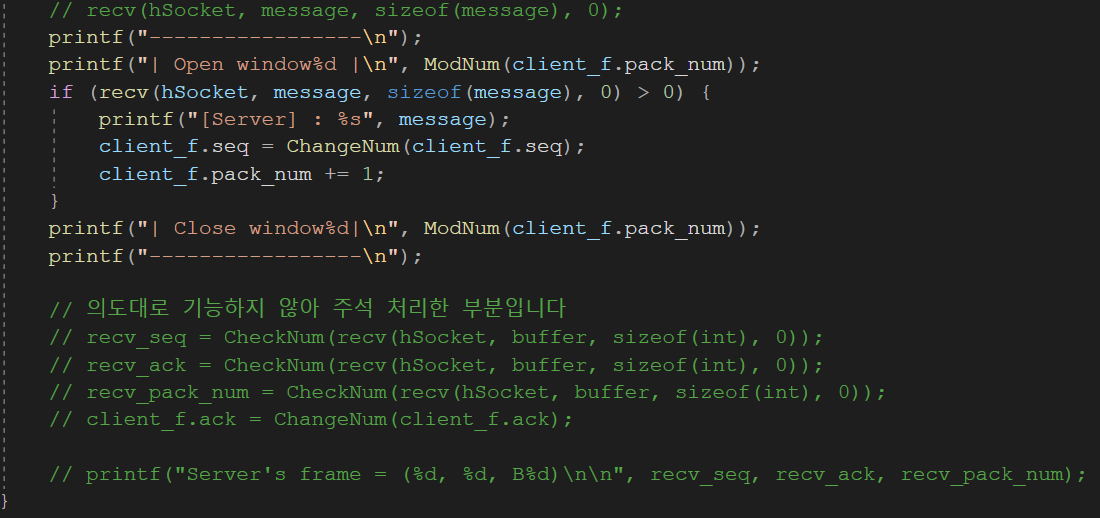
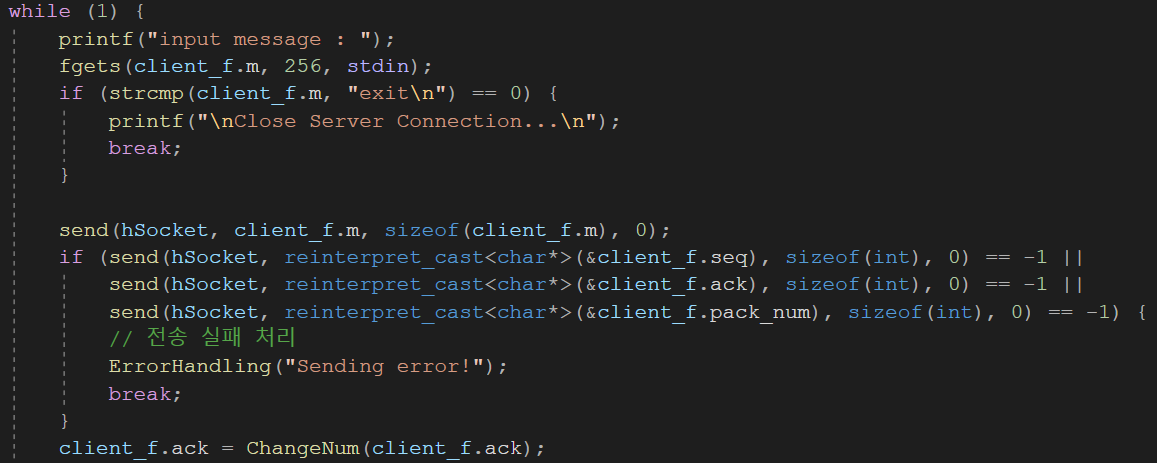
ChangeNum() : 데이터를 보낸 후 seq와, 받은 후의 ack 값을 변환시키기 위한 함수





『server.cpp에서 설명한 부분은 생략』

inet\_pton() 함수를 통해 IP주소와 PORT 번호를 받아 servAddr에 데이터를 삽입



1. fgets() 함수를 통해 클라이언트의 frame 에 메시지를 저장하고 해당 메시지가 “exit” 인지 확인

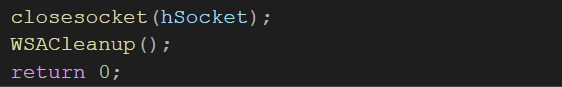
2. “exit”라면 연결을 끊고 프로그램 종료

3. 아닐 경우 해당 메시지를 서버로 전송 + 클라이언트의 seq, ack, pack\_num을 char 형식으로 전환해 전송

4. 전송 후, ChangeNum() 함수를 통해 ack를 변경

5. Window open / close 상태를 출력하고 서버로부터 ack를 받았을 경우 seq와 pack\_num 값 변화 및 증가

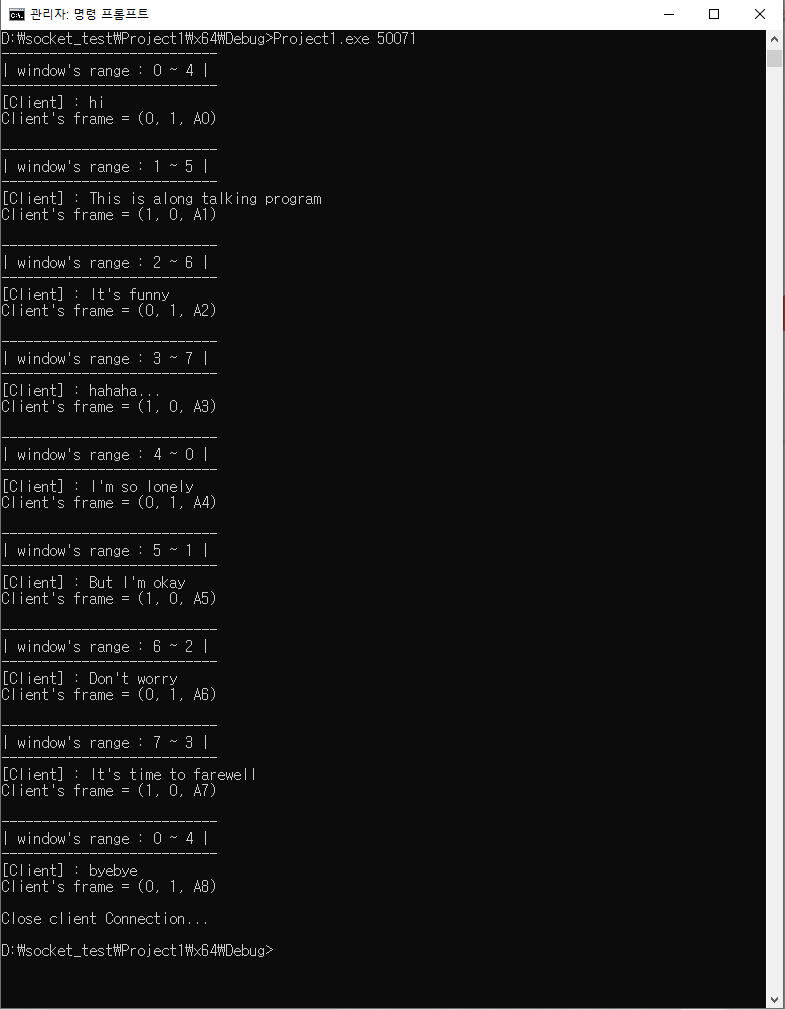
# 6. 원래 계획은 서버에서 보낸 frame의 seq, ack, pack\_num 을 출력하는 과정을 거치려 하였으나 정상적으로 작동하지 않아 주석처리 함



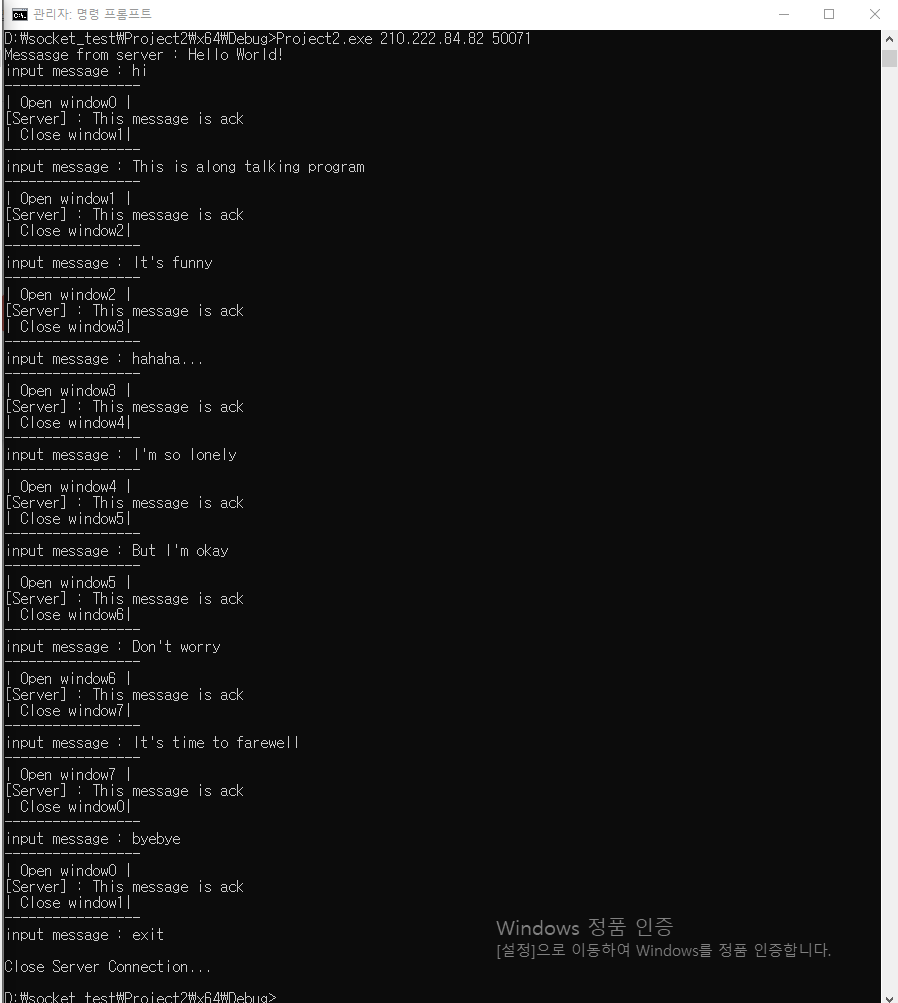
server.cpp 와 동일

**<결과 화면>**

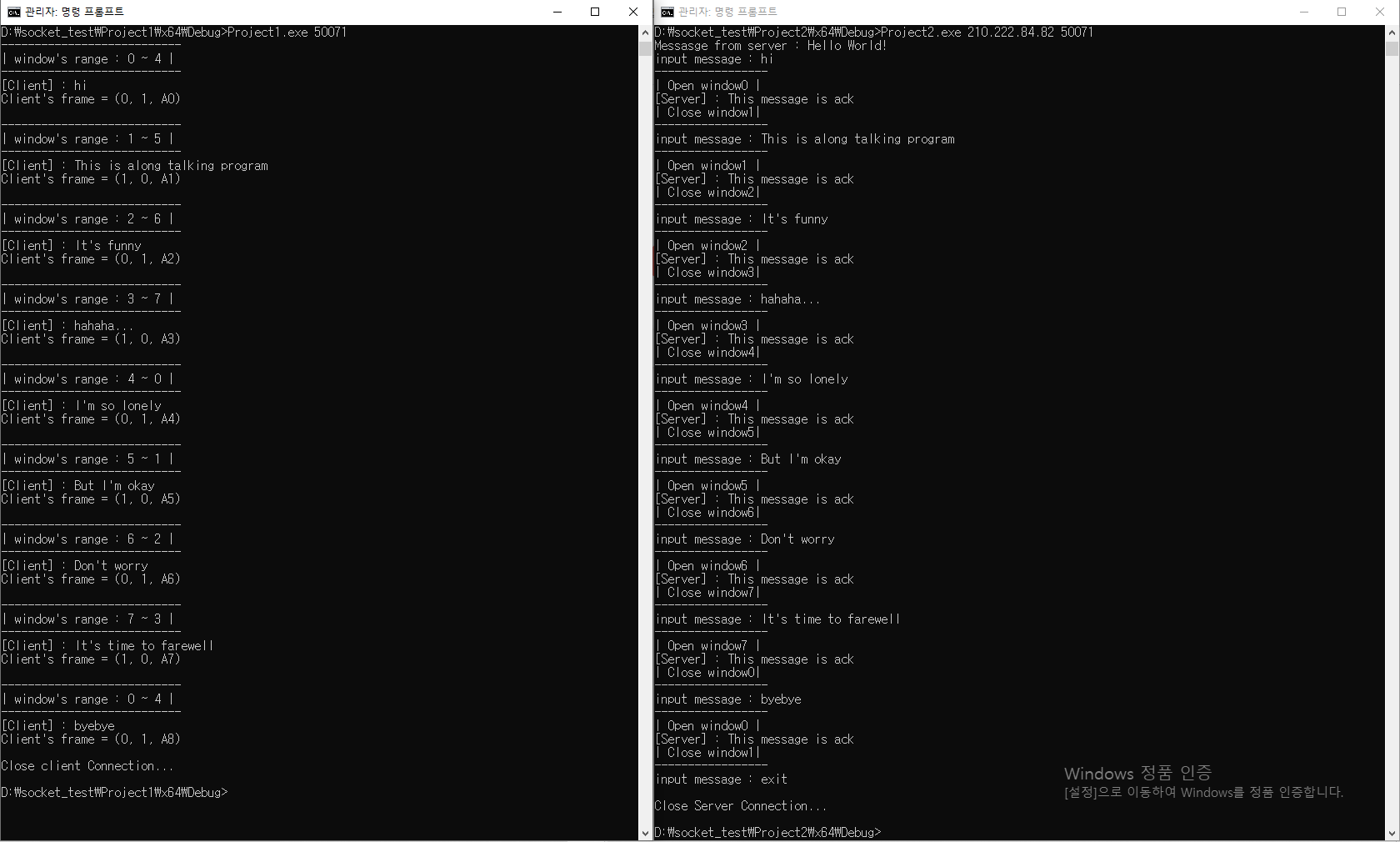
- server.cpp



- client.cpp



- 동시화면



**<과제 후기>**

처음 AWS를 접했을 때에는 사용법을 몰라 여러 자료들을 찾아보았다. 그 과정을 통해 AWS라는 서비스가 어떤 것인지 알게 되었고 굉장히 유용한 서비스인 것을 알게 되었다. 인스턴스 생성도 해보고 인증키 방식에 대해서도 알게 되었다. 비록 AWS를 통해 프로그램을 실행시켜 확인하지 못 하였지만 클라우드 서버 프로그래밍, 소켓 통신에 대해 알게 된 시간을 가질 수 있었다. Sliding window protocol의 작동 원리에 대해 생각해 보는 시간도 가졌으며, 이를 구현해 보는 재미도 느낄 수 있어 의미있는 시간이였다고 생각한다.