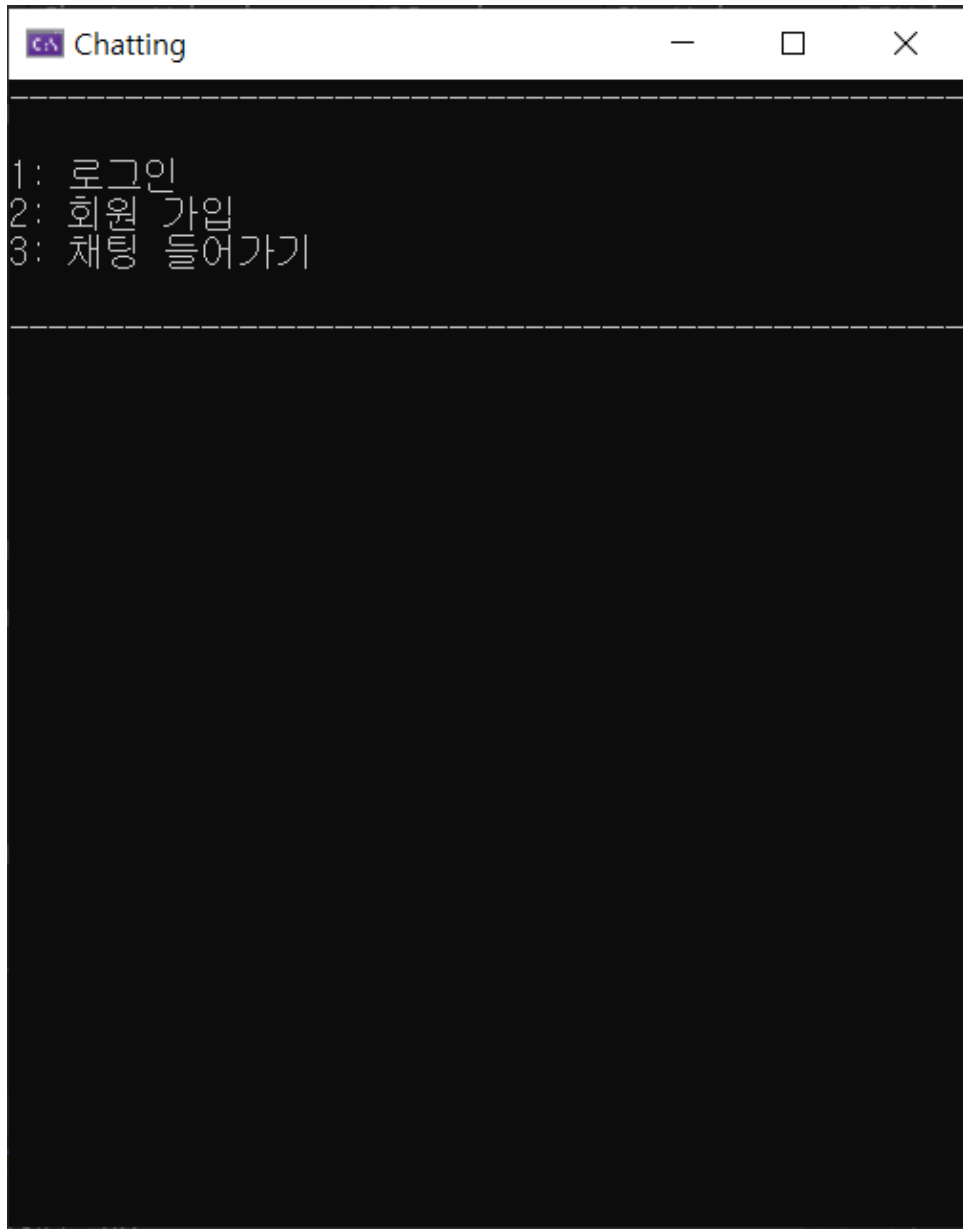
# Socket 통신
# 채팅 프로젝트

# 메인화면 UI 설정




메인화면

아래의 코드를 추가해 채팅하는 콘솔 창을
미리 설정한 크기로 콘솔창이 뜨게 설정하고,
이름을 " Chatting”으로 보일 수 있게끔 설정했습니다.

```
system("mode con cols=50 lines=30 | title Chatting");
```

## 비밀번호 암호화

| | id | pw | name |
|---|---|---|---|
| ▶ | bunny94 | 8127-18-811-72104-29-120124-2258101-43107 | 토끼 |
| * | NULL | NULL | NULL |

DB 서버에서 관리자가  비밀번호나, 아이디를 노출시킬 가능성이 있어
비밀번호 암호화를 구현했습니다.

비밀번호 암호화 라이브러리가 따로 존재하지만,
저희가 이번에 구현한 암호화 방식은 비트연산자를 사용해 간단히 암호화
하는 방식입니다.

클라이언트에서 암호화 후, 서버에 업로드하는 형식입니다.

# 암호화 구현 방법

```
const INT C1 = 52845;
const INT C2 = 22719;
const INT KEY = 78695;

class CCrypt
{
private:


public:
    CCrypt(void) {};
    ~CCrypt(void) {};


    static vector<int> Encrypt(char* source, char* destination, DWORD length) {
        DWORD i;
        int key = KEY;
        vector<int> temp;
        if (!source || !destination || length <= 0)
        {
            return temp;
        }
        for (i = 0; i < length; i++) {
            destination[i] = source[i] ^ key >> 8;
            key = (destination[i] + key) * C1 + C2;

            temp.push_back(destination[i]);
        }
        return temp;
    };
};
```

암호화 코드
C1, C2, KEY 값을 사용해 비트연산을 하게 됩니다.

Encrypt에 들어가는 인자로는 순서대로
(비밀번호, 결과값이 들어갈 변수, 비밀번호의 크기)를
입력 받습니다.

# 암호화 구현 방법

### 3. ^연산자

^| 연산은 두 개의 비트가 서로 다른 경우에 1을 반환하는 XOR 연산입니다. 따라서 다음의 연산결과를 보입니다.

| 연산 | 결과 |
|---|---|
| 0 ^ 0 | 0 |
| 0 ^ 1 | 1 |
| 1 ^ 0 | 1 |
| 1 ^ 1 | 0 |

^ 연산자로 임의의 수와 연산후,
>> 연산자로 오른쪽으로 8칸씩 이동시켜 수를 암호화 합니다!

### 5. << 연산자

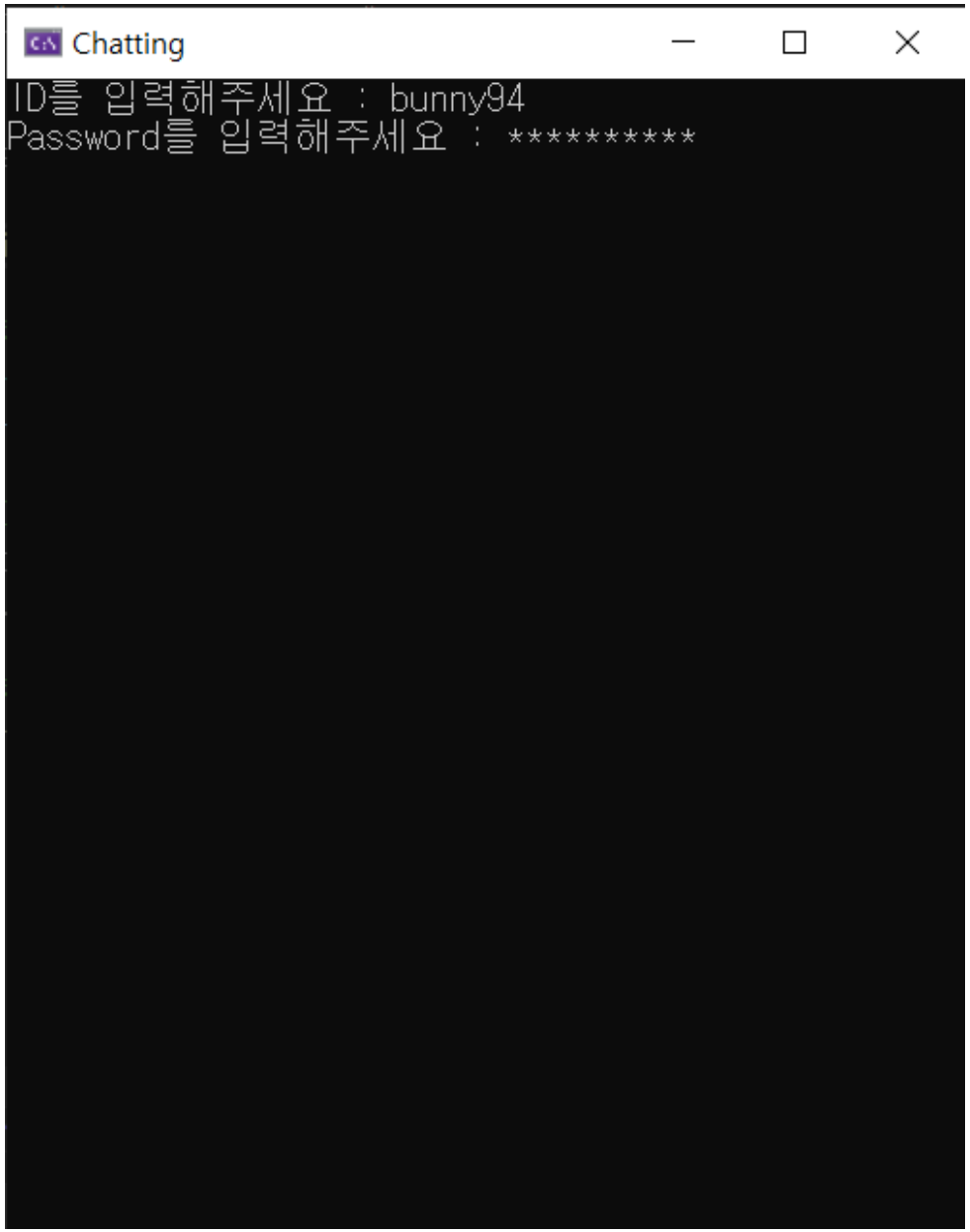
<< 연산자는 비트를 왼쪽으로 이동하는 shift 연산입니다.
num1 << num2
로 표현되며 num1의 비트 열을 num2칸씩 왼쪽으로 이동시킨다는 의미를 갖고 있습니다.

```c
#include <stdio.h>

int main(void)
{
    int num1 = 15;     // 00000000 00000000 00000000 00001111

    int result1 = num1 << 1;
    int result2 = num1 << 2;
    int result3 = num1 << 3;

    printf("1칸 이동 결과 : %d ₩n", result1);
    printf("2칸 이동 결과 : %d ₩n", result2);
    printf("3칸 이동 결과 : %d ₩n", result3);

    return 0;
}
```

```
선택 Microsoft Visual Studio 디버그 콘솔
1칸 이동 결과 : 30
2칸 이동 결과 : 60
3칸 이동 결과 : 120
```

```c
static vector<int> Encrypt(char* source, char* destination, DWORD length) {
    DWORD i;
    int key = KEY;
    vector<int> temp;
    if (!source || !destination || length <= 0)
    {
        return temp;
    }
    for (i = 0; i < length; i++) {
        destination[i] = source[i] ^ key >> 8;
        key = (destination[i] + key) * C1 + C2;

        temp.push_back(destination[i]);
    }
    return temp;
};
};
```

## 비밀번호 입력 보안




비밀번호 입력 시 *로 표시가 되게끔 구현했습니다.
스페이스바를 무시하고,
최대 10자리까지 입력할 수 있게
입력 제한 조건을 두었습니다.

# 비밀번호 입력 보안 방법

```
string PwCheck(string pw) {
    int cnt = 0;
    pw = "";
    while (1) {
        if (_kbhit()){
            int c = _getch();
            /*a += _getch();*/
            //엔터
            if (c == 13) {
                pw = PwEncrypt(pw);
                return pw;
            }
            //스페이스바
            else if (c == 32) {
                continue;
            }
            //백스페이스
            else if (c == 8) {
                if (cnt == 0) {
                    continue;
                }
                else {
                    //* 별지우기
                    cout << "\b \b";
                    pw.pop_back();
                    if (cnt != 0) {
                        cnt--;
                    }
                    continue;
                }
            }
            else {
                //10자리 이상 입력 제한
                if (cnt > 9) continue;
                pw += c;
                cnt++;
                cout << "*";
            }
        }
    }
}
```

kbhit()이라는 함수를 사용해
키보드의 입력 여부를  확인이 가능합니다.

getch()
어떤 키를 입력했는지 값을 가져옵니다.

위 두 함수를 사용해
아스키 코드를 사용해 특정 키를 입력할 때 마다 조건을 걸어
입력시 바로 *로 표시될 수 있게 구현했습니다.

# 시연