c++ 프로그래밍 및 실습

체스 게임

진척 보고서 #2

제출일자: 2024.11.29
제출자명: 기도현
제출자학번: 214953

1. 프로젝트 목표

1) 배경 및 필요성

- 체스가 두뇌 개발과 치매 예방에 효과적이라는 보고가 많지만 주변에서 같이 체스를 둘 사람이 많지 않음

- 어린시절 학교에 있는 체스판으로 체스를 즐겨 했지만 요즘은 체스판을 찾아보기 힘듦. 체스판의 부피가 크기 때문에 가지고 다니는 것도 부담임

- 인터넷에서 풀어보고 싶은 체스 퀴즈를 풀어 보거나 게임을 복기할 때 사용할 툴이 마땅치 않음

- 내가 만든 체스 퀴즈와 복기 내용을 저장할 장소가 마땅치 않음

2) 프로젝트 목표

- 플레이어의 선택에 따라 1인용으로 할지 2인용으로 할지 결정, 2인용이라면 턴 을 번갈아 가면서 말을 움직이게 해주고 1인용이라면 체스 퀴즈를 풀 수 있게 작 동하는 것을 목표로 함

- 사용자가 체스말들의 초기 위치를 직접 설정할 수 있는 툴을 제공해 풀어보고 싶던 체스 퀴즈도 풀어보고 체스게임을 복기할 수 있는 기능을 목표로 함

- 내가 만든 체스 퀴즈를 저장하고 원할 때 다시 꺼내어 풀어볼 수 있는 기능을 목표로 함

3) 차별점

- 기존의 체스 프로그램은 게임을 시작하면 말 위치의 초기값이 정해져 있고 이를 수정할 수 없음. 이는 원하는 체스 퀴즈를 만들 수 없고, 체스 게임을 복기할 때 처음부터 두어야 하다는 불편함이 있음. 이 프로그램은 체스 퀴즈를 만들 수 있는 툴을 제공해 살아있는 말의 종류와 수, 위치를 수정하고 게임을 시작할 수 있다는 것에 기존 프로그램과 차별점이 있음

- 기존의 체스 프로그램은 말을 고르고 좌표를 고르면 해당 좌표로 이동하는 방식을 취함. 이는 체스게임 초보자라면 말이 갈 수 있는 길을 헷갈릴 수 있다는 단점이 있음. 이 프로그램은 말을 선택하면 갈 수 있는 길을 알려준다는 점에서 기존 프로그램과 차별점이 있음

- 기존의 체스 프로그램은 내가 만든 체스 퀴즈를 저장할 수 없음, 이는 재밌는 체스 퀴즈나 체스게임 복기 내용을 사람이 기억하는데 한계가 있음 이 프로그 램은 내가 만든 상황을 저장하여 다시 꺼내어 볼 수 있다는 점에서 기존 프로그램과 차별점이 있음

2. 기능 계획

1) 기능 1 사용자가 원하는 기능 판단

- 설명: 사용자의 선택에 따라 체스게임도 할 수 있고 체스 퀴즈도 만들 수 있음

(1) 세부 기능 1: 사용자가 원하는 기능을 입력 받아 그에 해당하는 함수 실행

- 설명: 체스게임을 입력한다면 체스게임 함수를, 체스 퀴즈를 입력한다면 체스 퀴즈 함수를 실행

2) 기능 2 player가 2명일 때 기존의 체스게임 실행

- 설명: player가 2명이라면 턴을 번갈아 가며 말을 옮겨 체스게임을 할 수 있게 해줌

(1) 세부 기능 1: 말 별로 class를 만들어 관리

- 설명: 체스에는 여러 종류의 말이 있기 때문에 말 별로 class를 만들어 코드를

깔끔하게 작성

(2) 세부 기능 2: 체스판 출력
- 설명: 2차원 배열의 각 인덱스를 출력해 체스판을 시각화

(3) 세부 기능 3: 사용자가 이동을 원하는 말을 선택 하면 이동 가능 경로 시각화
- 설명: 이동을 원하는 말을 입력하면 그 말이 움직일 수 있는 좌표를 시각화 하여 체스판에 점을 찍어 보여줌

(4) 세부 기능 4: 사용자가 원하는 말을 이동하면 그 결과를 시각화 하여 출력

- 설명: 사용자가 원하는 말 원하는 좌표를 선택하면 그 좌표로 말을 이동하고 그 결과를 보드판 출력을 통해 시각화

(5) 세부 기능 5: 체스 게임의 룰 적용

- 설명 : 체스게임에는 특정 상황이 말의 기능이 평소와 달라지는 룰이 있기에 그 룰들을 함수화 하여 적용

(6) 세부 기능 6: 게임 종료

- 설명: 왕이 죽거나 사용자가 항복이라고 입력하면 게임을 종료하는 기능

3) 기능 3 체스 퀴즈 툴

- 설명: 사용자가 원하는 말들을 원하는 위치에 놓고 게임을 시작할 수 있게 해주는 기능

(1) 세부 기능 1: 원하는 기물을 원하는 위치에 놓고 게임 실행

- 설명: 원하는 말과 위치를 확인해 해당 위치에 말을 놓고 보드판을 출력

(2)~(7) 세부기능 2~7: 기능 2의 세부기능 1~와 동일

(8) 세부기능 8: 잘못 놓여진 체스말 삭제

- 설명: 체스판에 원하는 말을 놓다가 말이 잘못 놓여진다면 해당 말을 삭제

4) 기능 4 체스 퀴즈 저장 및 삭제

- 설명: 사용자가 만든 체스판 상황을 저장 및 삭제할 수 있는 기능을 만들어 사용자가 원할 때 만들어 두던 체스판 상황을 꺼내어 볼 수 있는 기능

(1) 세부 기능 1: 만들어진 체스 퀴즈 저장하기

- 설명: 동적으로 객체를 생성해 사용자가 만든 체스판의 내용을 저장

(2) 세부 기능 2: 저장된 체스 퀴즈 풀어보기

- 설명: 저장된 객체를 매개변수로 하는 함수호출을 통해 저장된 체스판을 출력

(3) 세부기능 3: 체스 퀴즈 삭제

- 설명: 저장된 객체를 삭제하는 방식으로 사용자가 잘못 만든 체스판을 삭제할

수 있음

3. 진척사항

1) 기능 구현

(1) 사용자가 원하는 기능 판단

- 입출력

  입력(변수):

- choice: 체스 게임 or 체스 퀴즈 제작 중 선택한 번호를 저장하는 변수

- 설명

- cin 명령어로 1or 2를 받아 기능을 선택하고 if문을 통해 해당 기능 수행

- 적용된 배운 내용

- 조건문 (4주차)

- 코드 스크린샷

```
cout<<"1.체스 게임\n2.체스 퀴즈 제작\n 입력하시오"<<endl;
cin>>choice;

if(choice==1){
while(1){
```

(2) player가 2명일 때 기존의 체스게임 실행

(1) 세부 기능 1: 말 별로 class를 만들어 관리

1. class 부분

- 입출력

입력(변수):

- x,y: 해당 기물의 좌표값을 저장

- team_num: 어떤 팀의 기물인지 저장

- life: 해당 기물이 살아있는지 저장

-name: 해당 기물의 이름을 저장

-출력:

- get_?() 함수의 출력:

private로 선언된 ?변수를 리턴해 private로 선언된 변수들의 값을 읽을 수 있게 해준다.

- 설명

- class의 필드값으로 기물들 자신에 대한 정보를 저장하게 만든다. life의 초기값은 1로 해당 체스말의 초기값은 살아있다는 뜻

- piece class 하나로 name 값을 다르게 하여 다른 말로 사용

- 데이터 은닉과 실행속도 향상을 위해 캡슐화 하고 다른 cpp 파일에서 함수 구현
- get함수와 set 함수를 통해 private로 선언된 변수들에 접근

- 적용된 배운 내용
- 클래스 (9주차)
- 캡슐화 (9주차)
- 함수의 구현은 다른 파일에서 하기 (9주차)
- 변수 이름이 같을 때 this -> 사용 (11주차)
- 객체 (9주차)
- 함수 (6주차)

- 코드 스크린샷

```cpp
#include "chess_piece.h"

int Piece::get_x(){
    return x;
}
int Piece::get_y(){
    return y;
}
int Piece::get_team_num(){
    return team_num;
}
int Piece::get_life(){
    return life;
}
string Piece::get_name(){
    return name;
}
void Piece::set_x(int x){
    this->x=x;
}
void Piece::set_y(int y){
    this->y=y;
}
void Piece::set_life(int life){
    this->life=life;
}
void Piece::set_team_num(int team_num){
    this->team_num=team_num;
}
void Piece::set_name(string name){
        this->name=name;
    }
void Piece::print_name(){
        cout<<name;
    }
```

```cpp
#include <iostream>
#include <string>
using namespace std;

class Piece{
private:
    int x,y;
    int team_num;
    int life=1;
    string name;

public:
    int chess_num;
    int get_x();
    int get_y();
    int get_team_num();
    int get_life();
    string get_name();
    void set_x(int x);
    void set_y(int y);
    void set_life(int life);
    void set_team_num(int life);
    void set_name(string name);
    void print_name();
};
```

2. 각 기물들의 x,y 좌표값 설정 부분

- 입출력

  입력:

- chess_piece: Piece class의 객체를 객체 배열로 생성

- 설명

- 객체 배열 전체를 파라미터로 받아옴

- [0][?] 부분은 0 팀, [1][?] 부분은 1팀으로 이중 for문과 조건문을 활용해 작성

- 각 인덱스마다 어떤 기물을 맡을지를 정했다 가정하고 처음 초기 위치의 x,y값을 이중 for문을 활용해 입력

- 적용된 배운 내용

- 객체 배열 (10주차)

- 2중 for문 (4주차)

- 조건문 (4주차)

- 캡슐화 (9주차)

- 함수 (6주차)

- 코드 스크린샷

```
Piece chess_piece[2][16];
```

```
void make_piece(Piece chess_piece[2][16]){
    for(int i=0;i<2;i++){
        for(int j=0;j<16;j++){
            if(i==0){
                chess_piece[i][j].set_team_num(0);
            }
            else{
                chess_piece[i][j].set_team_num(1);
            }
            if(chess_piece[i][j].get_team_num()==0){
                if(j<8){
                    chess_piece[i][j].set_x(j);
                    chess_piece[i][j].set_y(1);
                }
                else{
                    chess_piece[i][j].set_x(j-8);
                    chess_piece[i][j].set_y(0);
                }
            }

            if(chess_piece[i][j].get_team_num()==1){
                if(j<8){
                    chess_piece[i][j].set_x(j);
                    chess_piece[i][j].set_y(6);
                }
                else{
                    chess_piece[i][j].set_x(j-8);
                    chess_piece[i][j].set_y(7);
                }
            }
        }
    }

}
```

3. 각각 기물의 해당하는 배열 객체의 원소들에게 이름 붙여주기

- 입출력

  입력:

 - chess_piece: Piece class 배열객체

 - 설명

- [0][?] 부분은 0 팀, [1][?] 부분은 1팀으로 make_piece 함수에서 저장함, 각 배열 인덱스마다 해당 인덱스가 어떤 역할을 할 것인지 가정해놓고 코드를 작성함

- 2중 for문을 활용해 각 배열객체 원소 하나하나 기물 이름+ 플레이어 이름에 해당하는 문자열을 Piece class의 name 변수에 저장

- 적용된 배운 내용

- 2중 for문 (4주차)
- 객체 배열 (10주차)
- 조건문 (4주차)
- 캡슐화 (9주차)
- 함수 (6주차)

- 코드 스크린샷
-

```
void naming_piece(Piece chess_piece[2][16]){
    for(int i=0;i<2;i++){
        for(int j=0;j<16;j++){
            if(i==0){
                if(chess_piece[i][j].get_y()==1){
                    chess_piece[i][j].set_name("폰   1");
                }
                else if(chess_piece[i][j].get_x()==0||chess_piece[i][j].get_x()==7){
                    chess_piece[i][j].set_name("룩   1");
                }
                else if(chess_piece[i][j].get_x()==1||chess_piece[i][j].get_x()==6){
                    chess_piece[i][j].set_name("낫트 1");
                }
                else if(chess_piece[i][j].get_x()==2||chess_piece[i][j].get_x()==5){
                    chess_piece[i][j].set_name("비숍 1");
                }
                else if(chess_piece[i][j].get_x()==3){
                    chess_piece[i][j].set_name("킹   1");
                }
                else{
                    chess_piece[i][j].set_name("퀸   1");
                }

            }
            else{
                if(chess_piece[i][j].get_y()==6){
                    chess_piece[i][j].set_name("폰   2");
                }
                else if(chess_piece[i][j].get_x()==0||chess_piece[i][j].get_x()==7){
                    chess_piece[i][j].set_name("룩   2");
                }
                else if(chess_piece[i][j].get_x()==1||chess_piece[i][j].get_x()==6){
                    chess_piece[i][j].set_name("낫트 2");
                }
                else if(chess_piece[i][j].get_x()==2||chess_piece[i][j].get_x()==5){
                    chess_piece[i][j].set_name("비숍 2");
                }
                else if(chess_piece[i][j].get_x()==3){
                    chess_piece[i][j].set_name("킹   2");
                }
                else{
                    chess_piece[i][j].set_name("퀸   2");
                }

            }
        }
    }

}
```

4. 기물 이름에 따라 기물의 움직임이 달라지는 부분

- 입출력

입력:

- piece: class 객체 하나의 주소
- plus_x: 사용자가 원하는 x방향 증가량
- plus_y: 사용자가 원하는 y방향 증가량

출력:

- true: 기물의 움직임이 올바르다
- flase: 기물의 움직임이 올바르지 않다.

- 설명:
- 함수의 파라미터 값으로 객체의 주소와 사용자가 원하는 x,y증가량을 받음
- 원래 기물이 있던 장소의 x,y값을 original_x, original_y로 저장
- 이동한 기물의 x,y값을 new_x, new_y로 저장
- 기물이 이동했을 때 체스판 범위 밖이라면 체스판을 나갔다고 알려주고 false 리턴
- find함수를 통해 기물의 종류를 판단
- 해당 기물이 움직을 수 있는 x,y증가량을 조건문을 통해 판단한 후 옳지 않으면 옳지 않다는 것을 사용자에게 알려주고 flase 리턴
- 해당 기물의 움직임이 올바르면 true를 리턴

- 적용된 배운 내용 (예: 반복문, 조건문, 클래스, 함수, 포인터 등)
- 문자열 함수 (2주차& 6주차)
- 조건문 (4주차)
- 객체의 주소를 파라미터로 넘기기 (11주차)
- math 함수 (4주차에 구두로 설명)
- 문자열 (2주차)
- 함수 (6주차)

- 코드 스크린샷

```
bool chess_rule(Piece &piece, int plus_x, int plus_y) {
    int original_x = piece.get_x();
    int original_y = piece.get_y();
    int new_x = original_x + plus_x;
    int new_y = original_y + plus_y;


    if (new_x < 0 || new_x > 7 || new_y < 0 || new_y > 7) {
        cout << "기물의 움직임이 체스판 범위를 벗어났습니다.\n";
        return false;
    }


    string name = piece.get_name();
    if (name.find("폰") != string::npos) {
        if (piece.get_team_num() == 0) {
            if (!(plus_x == 0 && (plus_y == 1 || (original_y == 1 && plus_y == 2)))) {
                cout << "폰의 움직임이 바르지 않습니다.\n";
                return false;
            }
        } else {
            if (!(plus_x == 0 && (plus_y == -1 || (original_y == 6 && plus_y == -2)))) {
                cout << "폰의 움직임이 바르지 않습니다.\n";
                return false;
            }
        }
    } else if (name.find("룩") != string::npos) {
        if (!(plus_x == 0 || plus_y == 0)) {
            cout << "룩의 움직임이 바르지 않습니다.\n";
            return false;
        }
```

```
    } else if (name.find("낫트") != string::npos) {
        if (!((abs(plus_x) == 2 && abs(plus_y) == 1) || (abs(plus_x) == 1 && abs(plus_y) == 2))) {
            cout << "낫트의 움직임이 바르지 않습니다.\n";
            return false;
        }
    } else if (name.find("비숍") != string::npos) {
        if (abs(plus_x) != abs(plus_y)) {
            cout << "비숍의 움직임이 바르지 않습니다.\n";
            return false;
        }
    } else if (name.find("킹") != string::npos) {
        if (abs(plus_x) > 1 || abs(plus_y) > 1) {
            cout << "킹의 움직임이 바르지 않습니다.\n";
            return false;
        }
    } else if (name.find("퀸") != string::npos) {
        if (!(abs(plus_x) == abs(plus_y) || plus_x == 0 || plus_y == 0)) {
            cout << "퀸의 움직임이 바르지 않습니다.\n";
            return false;
        }
    }

    return true;
}
```

(2) 세부 기능 2 체스판 출력

- 입출력

입력:

- chess_piece: Piece 클래스의 객체들을 객체 배열 만듬

- found : 해당 좌표에 체스 기물이 있는지 확인하는 변수

- 설명

- main 함수에서 체스게임을 고르면 체스 말들을 고르고 바로 board_copy 함수 실행

- 2중 for문을 만들어 체스판 모양을 출력

- 해당 칸에 기물이 있는지 2중 for문 안에서 2중 for문을 돌려 객체 배열의 원소들 각각의 x,y좌표를 확인, 이 x,y좌표와 현재 출력하는 보드판의 x,y좌표가 일치하면 found 함 수를 true로 바꾸고 2중 for문 빠져나옴, 끝까지 2중 for문을 돌려도 일치하는 x,y좌표가 없♘다면" "을 출력해 빈칸임을 보여줌

- 적용된 배운 내용

- 객체 배열 (10주차)

- 이중 for문을 이용한 보드판 출력 (4주차)

- private로 선언된 class 값을 함수로 꺼내 사용 (9주차)

- break (4주차)

- 조건문 (4주차)

- 함수 (6주차)

- 코드 스크린샷

```
while(1){
    int turn;
    int x,y;
    int moving_x,moving_y;
    white_life=0;
    black_life=0;

    print_board(chess_piece);
    if((k%2)==1){
```

```
void print_board(Piece chess_piece[2][16]) {
    int x, y;
    for (int i = 0; i < 8; i++) {
        cout << "|------|------|------|------|------|------|------|------|" << endl;
        for (int j = 0; j < 8; j++) {
            cout << "|";
            bool found = false;
            for (int q = 0; q < 2; q++) {
                for (int w = 0; w < 16; w++) {
                    x = chess_piece[q][w].get_x();
                    y = chess_piece[q][w].get_y();
                    if (x == j && y == i) {
                        chess_piece[q][w].print_name();
                        found = true;
                        break;
                    }
                }
                if (found) break;
            }
            if (!found) cout << "      ";
        }
        cout << "|" << endl;
    }
    cout << "|------|------|------|------|------|------|------|------|" << endl;
}
```

(3) 세부 기능 4,5: 사용자가 원하는 말을 이동하면 그 결과를 시각화 하여 출력

- 입출력

입력:

(main 함수 부분)

- k: 계속 %2를 하면서 한번씩 턴을 진행하게 해주는 변수

- white_life: 1번 플레이어의 목숨

- black_life: 2번 플레이어의 목숨

- moving_x: 해당 기물에 대해 원하는 x 증가량

- moving_y: 해당 기물에 대해 원하는 y 증가량

(board_copy 함수 부분)

- chess_piece: Piece 클래스 객체 배열

- 전역 변수 board 배열

(kill 함수 부분)

- chess_piece: 객체 배열

- piece: Piece객체의 원본

- team_num: 어떤 팀의 기물인지 저장하는 값

(board_check)

- chess_piece: 객체 배열

- piece: Piece객체의 원본

- plus_x: 해당 기물에 대해 원하는 x 증가량

-plus_y: 해당 기물에 대해 원하는 y 증가량

출력:

(board_copy 함수 부분)

- 출력 x

(kill 함수 부분)

- 출력 x

(board_check 함수 부분)

- true: 해당 기물의 이동한 위치가 올바른 장소일 때

-false: 해당 기물의 이동한 위치에 아군 기물이 있을 때

- 설명

(board_copy 함수 부분)

- 해당 cpp 파일 맨 앞에 체스판에 해당하는 board[8][8] 배열이 전역변수로 선언되어 있음, 모든 배열 원소들은 0으로 초기화

- 모든 체스 기물이 들어있는 배열 객체를 받아 이중 for문으로 접근

- 각각 원소 객체의 x, y, team_num값을 받아와 x, y좌표에 해당하는 배열 인덱스에 team_num이 0 이면 1을 1이면 2를 대입

- 이렇게 함으로써 현재 기물들의 x, y 좌표를 board 배열과 동기화 할 수 있음

(kill 함수 부분)

- kill 함수는 기물이 도착한 곳에 상대팀 기물이 있을 때 호출되는 함수임

- 파라미터로 전달받은 Piece 객체가 상대팀 기물을 잡은 것임

- 원래 그 좌표에 있던 상대팀 객체를 for문을 통해 찾음, 객체를 찾았다면 해당 객체의 life를 0으로 하고 그 객체의 x, y 값을 -1로 변경, 그리고 board 배열의 해당 x,y 좌표에 해당하는 배열 원소에 잡은 팀의 숫자를 입력

(board_check 함수 부분)

- 파라미터로 받은 plus_x와 plus_y를 이용해 원래 좌표값과 이동 후 좌표값을 따로 저장

- 새로운 좌표값으로 파라미터에서 받은 객체의 좌표값을 변경

- 조건문을 이용해 이동한 위치에 같은팀 기물이 있으면 같은 팀 기물이 있고 플레이어에게 알려준 후 원래 x, y 좌표로 돌아가고 false를 반환

- 조건문을 이용해 이동한 위치에 상대팀 기물이 있다면 상대팀 기물을 잡았다고 알려주고 kill()함수를 호출, true를 반환

- 조건문을 이용해 이동한 위치가 비어있다면 해당 위치로 이동한다고 플레이어에게 알려주고 true를 반환

(main 함수 부분)

- Piece 클래스 타입의 2차원 배열을 선언

- if문을 이용해 체스게임을 할지 체스 퀴즈를 할지 선택

- 체스 퀴즈를 선택했다면 while문을 이용해 무한루프

- make_piece, naming_piece, board_copy 함수를 사용해 객체 배열의 원소 각각에 알맞은 값을 넣고 board 배열에 동기화

- print_board 함수를 통해 보드를 보여주고 시작

- 무한루프 마지막에 k++를 해주고 계속 코드 시작에 k%2를 해줘서 턴을 번갈아 가며 게임을 진행

- 옮기고 싶은 기물의 x,y 좌표를 받고 해당 좌표가 올바른지 체크

- 해당 좌표에 해당하는 기물의 이름을 알려주고 원하는 x, y 증가량 받기

- 해당 증가량을 통해 이동한 좌표값이 올바른지 chess_rule 함수와 board_check 함수를 통해 확인하고 좌표 이동

- 좌표 이동후 board_copy 함수를 이용해 board 배열에 동기화

- white_life와 black_life를 무한루프 처음에 0으로 초기화, 무한루프 끝에서 각 팀의 king에 해당하는 객체에 접근 0팀의 킹이 살아있으면 white_life를 1로 변경, 1팀의 킹이 살아 있으면 black_life를 1로 변경, 무한루프 끝에서 white_life or balck_life의 값이 여전히 0이면 king이 죽🐴다는 것으로 생각하고, 게임의 결과를 출력 및 break 문을 통해 게임을 종료함

- 적용된 배운 내용 (예: 반복문, 조건문, 클래스, 함수, 포인터 등)

- 이중 for문 (4주차)

- 조건문 (4주차)

- 배열객체 (10주차)

- 배열객체의 주소를 파라미터로 넘기기 (11주차)

- 전역변수 (2주차)

- 2차원 배열 (5주차)

- 함수 (6주차)

- while문을 이용해 무한 loop (4주차)

- 코드 스크린샷

(전역변수)

```
int board[8][8]={0};
```

(board_copy 함수 부분)

```
void board_copy(Piece chess_piece[2][16]){
    int x,y;
    for(int i=0;i<2;i++){
        for(int j=0;j<16;j++){
            x=chess_piece[i][j].get_y();
            y=chess_piece[i][j].get_x();
            if(chess_piece[i][j].get_team_num()==0){
                board[x][y]=1;}
            else{
                board[x][y]=2;
            }
        }
    }
}
```

(kill 함수 부분)

```
void kill(Piece chess_piece[2][16],Piece &piece,int team_num){
    int x=piece.get_x();
    int y=piece.get_y();
    for(int i=0;i<16;i++){
        if(team_num==0){
            if(chess_piece[1][i].get_x()==x&&chess_piece[1][i].get_y()==y){
                chess_piece[1][i].set_life(0);
                chess_piece[1][i].set_x(-1);
                chess_piece[1][i].set_y(-1);
                board[y][x]=1;
            }
        }
        else{
            if(chess_piece[0][i].get_x()==x&&chess_piece[0][i].get_y()==y){
                chess_piece[0][i].set_life(0);
                chess_piece[0][i].set_x(-1);
                chess_piece[0][i].set_y(-1);
                board[y][x]=2;
            }
        }
    }
}
```

(board_check 함수 부분)

```
bool board_check(Piece chess_piece[2][16],Piece &piece,int plus_x,int plus_y){
    int original_x = piece.get_x();
    int original_y = piece.get_y();
    int new_x = original_x + plus_x;
    int new_y = original_y + plus_y;
    piece.set_x(new_x);
    piece.set_y(new_y);
   int x=new_x;
   int y=new_y;
        if(piece.get_team_num()==0){
            if(board[y][x]==1){
                cout<<"해당 위치에 같은 팀 기물이 있습니다.\n";
                piece.set_x(original_x);
                piece.set_y(original_y);
                return false;
            }
            else if(board[y][x]==2){
                cout<<"상대팀 기물을 잡았습니다.\n";
                kill(chess_piece,piece,0);
                board[original_y][original_x]=0;
                return true;
            }
            else{
                cout<<"해당 위치로 이동합니다.\n";
                board[original_y][original_x]=0;
                return true;           //빈칸일 때
            }
```

```
        else{
            if(board[y][x]==2){
                cout<<"해당 위치에 같은 팀 기물이 있습니다.\n";
                cout<<x<<" "<<y<<endl;
                piece.set_x(original_x);
                piece.set_y(original_y);
                return false;
            }
            else if(board[y][x]==1){
                cout<<"상대팀 기물을 잡았습니다.\n";
                kill(chess_piece,piece,1);
                board[original_y][original_x]=0;
                return true;
            }
            else{
                cout<<"해당 위치로 이동합니다.\n";
                board[original_y][original_x]=0;
                return true;           //빈칸일 때
            }

        }
}
```

(main 함수 부분)

```
int main(){

    int k=1;
    int white_life;
    int black_life;

    Piece chess_piece[2][16];

    int choice;
    cout<<"1.체스 게임\n2.체스 퀴즈 제작\n 입력하시오"<<endl;
    cin>>choice;

    if(choice==1){
    make_piece(chess_piece);
    naming_piece(chess_piece);
    board_copy(chess_piece);
    while(1){
        int turn;
        int x,y;
        int moving_x,moving_y;
        white_life=0;
        black_life=0;

        print_board(chess_piece);
        if((k%2)==1){
            cout<<"1번째 플레이어의 차례입니다.\n";
            turn=0;
        }
        else{
            cout<<"2번째 플레이어의 차례입니다\n";
            turn=1;
        }
        cout<<"옮기고 싶은 기물의 x,y좌표를 입력하시오 (x좌표 0부터 시작, y좌표 0부터 시작)";
        cin>>x>>y;
```

```
while(1){
    else{
    cout<<"옮기고 싶은 기물의 x,y좌표를 입력하시오 (x좌표 0부터 시작, y좌표 0부터 시작)";
    cin>>x>>y;
    while(board[y][x]==0||(x<0||x>7||y<0||y>7)){
        cout<<"빈칸입니다 다시입력하시오"<<endl;
        cin>>x>>y;
    }



    for(int i=0;i<2;i++){
        for(int j=0;j<16;j++){
            if(x==chess_piece[i][j].get_x()&&y==chess_piece[i][j].get_y()){
                if(turn==0){
                    if(turn!=chess_piece[i][j].get_team_num()){
                        cout<<"상대의 기물입니다. 다시 입력하십시오\n";
                        k++;
                        }
                    else{
                            cout<<chess_piece[i][j].get_name()<<"입니다\n 원하는 x좌표 증가량과 y좌표 증가량을 입력하시오(x축은 오른쪽이 +, y축은
                            cin>>moving_x>>moving_y;
                            while(!chess_rule(chess_piece[i][j],moving_x,moving_y)){
                                cout<<"다시 입력하세요";
                                cin>>moving_x>>moving_y;
                                continue;
                            }
                            while(!board_check(chess_piece,chess_piece[i][j],moving_x,moving_y)){
                                cout<<"다시 입력하세요";
                                cin>>moving_x>>moving_y;
                                continue;
                            }

                            board_copy(chess_piece);
```

```
            else{
                if(turn!=chess_piece[i][j].get_team_num()){
                cout<<"상대의 기물입니다. 다시 입력하십시오\n";
                k++;
                }
                else{
                    cout<<chess_piece[i][j].get_name()<<"입니다\n 원하는 x좌표 증가량과 y좌표 증가량을 입력하시오(x축은 오른쪽이 +, y축은 아래
                    cin>>moving_x>>moving_y;
                    while(!chess_rule(chess_piece[i][j],moving_x,moving_y)){
                        cout<<"다시 입력하세요";
                        cin>>moving_x>>moving_y;
                        continue;
                    }
                    while(!board_check(chess_piece,chess_piece[i][j],moving_x,moving_y)){
                        cout<<"다시 입력하세요";
                        cin>>moving_x>>moving_y;
                        continue;
                    }
                    board_copy(chess_piece);
```

```
            if(chess_piece[0][11].get_life()==1){
                white_life=1;
            }

            if(chess_piece[1][11].get_life()==1){
                black_life=1;
            }

        if(white_life==0){
            cout<<"2째 플레이어가 승리하셨습니다.";
            break;
        }
        else if(black_life==0){
            cout<<"1번째 플레이어가 승리하셨습니다";
            break;
        }
        k++;
    }


    }
    return 0;
}
```

(3) 세부 기능 3: 사용자가 이동을 원하는 말을 선택 하면 이동 가능 경로 시각화 부분수정

- 입출력

입력(변수):
- 입력 부분 수정하지 않음

출력:
- 출력 부분 수정하지 않음

- 설명

- 원래 내 코드는 내가 선택한 기물의 움직임이 체스 규칙에 맞는지 확인하긴 위한 ChessRule함수+while문과 움직인 자리가 빈칸인지, 상대팀 기물이 있는지, 우리팀 기물이 있는지를 확인하고 상황에 맞게 처리하는BoardCheck함수+while문을 따로따로 작성함
이 코드의 문제점은 코드상 위에있던 ChessRule함수를 통과하고 BoardCheck 함수를 통과하지 못해 다시 plus_x, plus_y값을 입력받는 상황이 오면 이 plus_x, plus_y값은 ChessRule함수의 확인을 받지 못함
- 이를 해결하기 위해 ChessRule함수를 위한 while문 안에 BoardCheck를 위한 while문을 넣어서 문제를 해결함

- BoardCheck 함수를 위한 while문을 나왔을 때 현재 plus_x, plus_y가 ChessRule 함수를 통과했는지 모르는 문제가 있음, 해당 문제를 해결하기 위해 check 변수를 넣어 BoardCheck 함수를 통과하지 못해 plus_x, plus_y값을 다시 받았을 경우 check값을 0으로 설정해 ChessRule함수에 들어가게 함

- 적용된 배운 내용

- 새롭게 추가된 개념 없음

- 코드 스크린샷

(수정 전)

```
cin>>moving_x>>moving_y;
while(!chess_rule(chess_piece[i][j],moving_x,moving_y)){
    cout<<"다시 입력하세요";
    cin>>moving_x>>moving_y;
    continue;
}
while(!board_check(chess_piece,chess_piece[i][j],moving_x,moving_y)){
    cout<<"다시 입력하세요";
    cin>>moving_x>>moving_y;
    continue;
}
```

(수정 후)

```
while (1)
{
    int check = 1;
    if (ChessRule(chess_piece[i][j], moving_x, moving_y))
    {
        while (!BoardCheck(chess_piece, chess_piece[i][j], moving_x, moving_y))
        {
            check = 0;
            cout << "다시 입력하세요";
            cin >> moving_x >> moving_y;
            break;
        }
        if (check == 1)
        {
            break;
        }
    }
    else
    {
        cout << "다시 입력하세요";
        cin >> moving_x >> moving_y;
    }
}
```

(3) 세부 기능 3: 사용자가 이동을 원하는 말을 선택 하면 이동 가능 경로 시각화 기능에서 기물이 움직일 때 이동경로에 다른 기물이 있다면 지나갈 수 없게 제한

- 입출력

입력(변수):
- piece : Piece class 객체 하나의 주소
- plus_x: 사용자가 원하는 기물의 x축 증가량
- plus_y: 사용자가 원하는 기물의 y축 증가량
- original_x: 기물의 원래 x값
- original_y: 기물의 원래 y값
- new_x: 기물의 원래 x에 x축 증가량을 더한 값
- new_y: 기물의 원래 y값에 y축 증가량을 더한 값

출력:
- true: 기물의 움직임이 바르다
- false: 기물의 움직임이 바르지 않다.

- 설명

- 기물에 따라 움직일 수 있는 x축 증가량과 y축 증가량이 정해져 있음
- 원래 ChessRule 함수에서는 기물 이름에 정해져 있는 기물의 움직임이, plus_x와 plus_y와 일치하는지 확인하는 작업만 수행함, 이 작업만 수행하면 가는 길이 막혀있어도 도착지만 빈칸이면 이동이 가능하다는 문제가 있음
- else문과 각 기물에 맞는 for문을 추가하여 기물 이름에 맞는 움직임이었다 하더라도 가는 길이 다른 기물에 의해 막혀있다면 false를 반환하게 하여 해결함
- 하나의 축으로만 이동한다면 그냥 for문을 x,y축 모두 이동한다면 2중 for문을 사용하여 가는 길목 하나하나 막고있는 다른 기물이 없는지 확인함

- 적용된 배운 내용
- 조건문(4주차)
- for문 (4주차)
- 2차원 배열 (5주차)
-함수 (6주차)

- 코드 스크린샷

(룩을 대표로 스크린샷 찍음)

```
else if (name.find("룩") != string::npos)
{
    if (!(plus_x == 0 || plus_y == 0))
    {
        cout << "룩의 움직임이 바르지 않습니다.\n";
        return false;
    }
    else
    {
        if (plus_x == 0)
        {
            if (plus_y > 0)
            {
                for (int i = 1; i < plus_y; i++)
                {
                    if (board[original_y + i][original_x] != 0)
                    {
                        cout << "해당 경로가 막혀있습니다.\n";
                        return false;
                    }
                }
            }
            else
            {
                for (int i = -1; i > plus_y; i--)
                {
                    if (board[original_y + i][original_x] != 0)
                    {
                        cout << "해당 경로가 막혀있습니다.\n";
                        return false;
                    }
                }
            }
        }
        else
        {
            if (plus_x > 0)
            {
                for (int i = 1; i < plus_x; i++)
                {
                    if (board[original_y][original_x + i] != 0)
                    {
                        cout << "해당 경로가 막혀있습니다.\n";
                        return false;
                    }
                }
            }
            else
            {
                for (int i = -1; i > plus_x; i--)
                {
                    if (board[original_y][original_x + i] != 0)
                    {
                        cout << "해당 경로가 막혀있습니다.\n";
                        return false;
                    }
                }
            }
        }
    }
```

(3) 체스 퀴즈 툴

(1) 세부 기능 1: 원하는 기물을 원하는 위치에 놓고 게임 실행

- 입출력

입력(변수):
- num: 사용자가 입력한 숫자를 저장
- x_dot, y_dot: 사용자가 기물을 놓고싶은 위치 좌표
- play_game: 사용자가 체스 퀴즈를 다 만들었다는 정보를 저장할 변수, 초기값을 0으로 설정
- choice_piece: 사용자가 고른 기물의 주소를 저장
- chess_piece: Piece class의 객체 배열
- modify_num: 기물이 비어있는 보드판을 출력하기 위해 Piece class에 modify_num을 추가함
  modify_num은 1로 초기화 되어있음

출력:
- 출력값 없음

- 설명
- while문을 이용해 무한루프를 만들고 계속해서 기물과 좌표를 선택
- 사용자가 체스퀴즈 제작으 고르면 , 모든 보드판이 비어있다는 뜻으로 chess_piece 배열의 모든 원소의 modify_num을 0으로 변경
- 사용자에게 기물과 팀을 고르게 하고 그 정보를 num 변수에 저장
- 조건문을 활용해 num 변수에 저장된 값을 토대로 알맞은 chess_piece배열 원소에 접근, 그 원소의 주소를 choice_piece에 저장
- 기물을 놓고싶은 주소를 받아 x_dot, y_dot에 저장
- 저장된 x,y 값을 choiece_piece의 x,y 값으로 설정
- choice_piece의 modify_num값을 1로 설정
- print_board함수를 실행해 현재 보드판을 보여주고 다시 무한루프
- 만약 사용자가 기물을 고르는 부분에서 0을 입력하였다면 체스판이 완성되었다고 해석하고 조건문과 for문을 통해 아직까지 선택되지 않은 기물들, 즉 modify_num값이 0인 기물들의 x,y,life를 죽어있는 말과 같이 설정(x,y=-1, life=0) + play_game을 1로 설정
- 조건문을 통해 play_game이 1이라면 BoardCopy함수를 실행해 현재 기물들의 상태를 board판에 동기화 시키고 체스판을 만드는 while문을 나가 기능 2에 해당하는체스게임을 내가 만든 체스판 위에서 실행

- 적용된 배운 내용
  - 조건문(4주차)
  - for문 (4주차)
  - 2차원 배열(5주차)
  - 포인터를 통해 객체 넘기기 (11주차)
  - 객체(9주차)
  - 함수(6주차)

- 코드 스크린샷

```
int choice;
cout << "1.체스 게임\n2.체스 퀴즈 제작\n 입력하시오" << endl;
cin >> choice;

if (choice == 2)
{

    for (int i = 0; i < 2; i++)
    {
        for (int j = 0; j < 16; j++)
        {
            chess_piece[i][j].SetModify(0);
        }
    }
    for(int i=0;i<8;i++)
    {
        for(int j=0;j<8;j++)
        {
            board[i][j]==0;
        }
    }
    PrintBoard(chess_piece);
    cout << "체스 퀴즈 제작을 시작합니다." << endl;

    while (1)
    {
```

```
while (1)
{
    int num;
    int x_dot, y_dot;
    int play_game = 0;
    Piece *choice_piece;
    cout << "놓고싶은 기물과 팀에 해당하는 번호를 입력하시오 (만들어진 체스판에서 체스를 실행하고 싶다면 0을 입력하시오)" << endl;
    cout << "1.폰 1\n2.룩 1\n3.낫트 1\n4.비숍 1\n5.퀸 1\n6.킹 1\n7.폰 2\n8.룩 2\n9.나이트 2\n10.비숍 2\n11.퀸 2\n12.킹 2\n";
    cin >> num;

    while (1)
    {
        if (num == 1)
        {
            choice_piece = &chess_piece[0][1];
            break;
        }
        else if (num == 2)
        {
            choice_piece = &chess_piece[0][8];
            break;
        }
        else if (num == 3)
        {
            choice_piece = &chess_piece[0][9];
            break;
        }
        else if (num == 4)
        {
            choice_piece = &chess_piece[0][10];
            break;
```

```
    }
    else if (num == 5)
    {
        choice_piece = &chess_piece[0][11];
        break;
    }
    else if (num == 6)
    {
        choice_piece = &chess_piece[0][12];
        break;
    }
    else if (num == 7)
    {
        choice_piece = &chess_piece[1][1];
        break;
    }
    else if (num == 8)
    {
        choice_piece = &chess_piece[1][8];
        break;
    }
    else if (num == 9)
    {
        choice_piece = &chess_piece[1][9];
        break;
    }
    else if (num == 10)
    {
        choice_piece = &chess_piece[1][10];
        break;
```

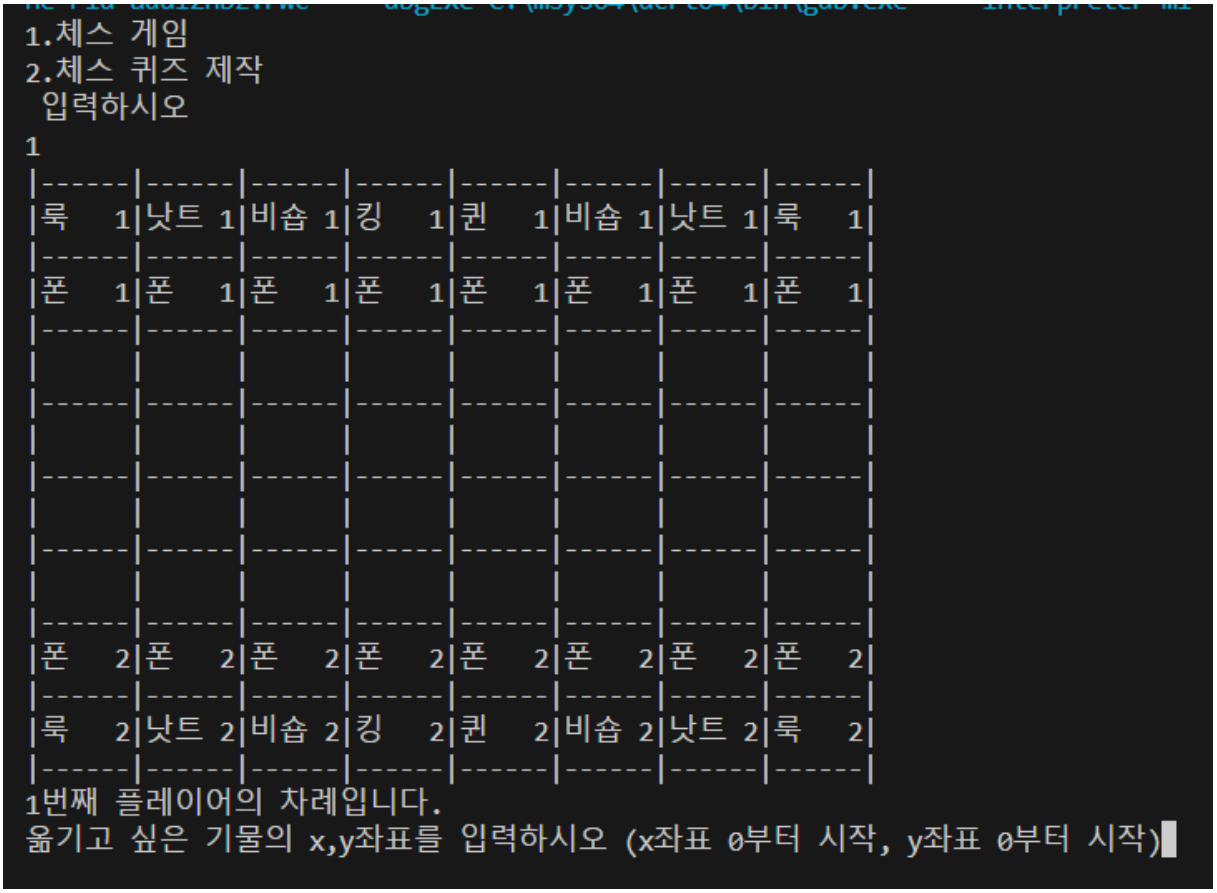
```
else if (num == 11)
{
    choice_piece = &chess_piece[1][11];
    break;
}
else if (num == 12)
{
    choice_piece = &chess_piece[1][12];
    break;
}
else if (num == 0)
{
    for (int i = 0; i < 2; i++)
    {
        for (int j = 0; j < 16; j++)
        {
            if (chess_piece[i][j].GetModify() == 0)
            {
                chess_piece[i][j].SetLife(0);
                chess_piece[i][j].SetX(-1);
                chess_piece[i][j].SetY(-1);
            }
        }
    }
    play_game = 1;
    break;
}
else
            }
            else
            {
                cout << "잘못 입력하셨습니다.";
            }
        }
        if (play_game == 1)
        {
            BoardCopy(chess_piece);

            break;
        }

        cout << "기물을 놓고싶은 x좌표와 y좌표를 입력해 주세요";
        cin >> x_dot >> y_dot;
        choice_piece->SetX(x_dot);
        choice_piece->SetY(y_dot);
        choice_piece->SetModify(1);
        PrintBoard(chess_piece);
    }
}
```

2) 테스트 결과

(1) 사용자가 원하는 기능 판단

- 설명: 사용자의 선택에 따라 체스게임도 할 수 있고 체스 퀴즈도 만들 수 있음

- 테스트 결과 스크린샷

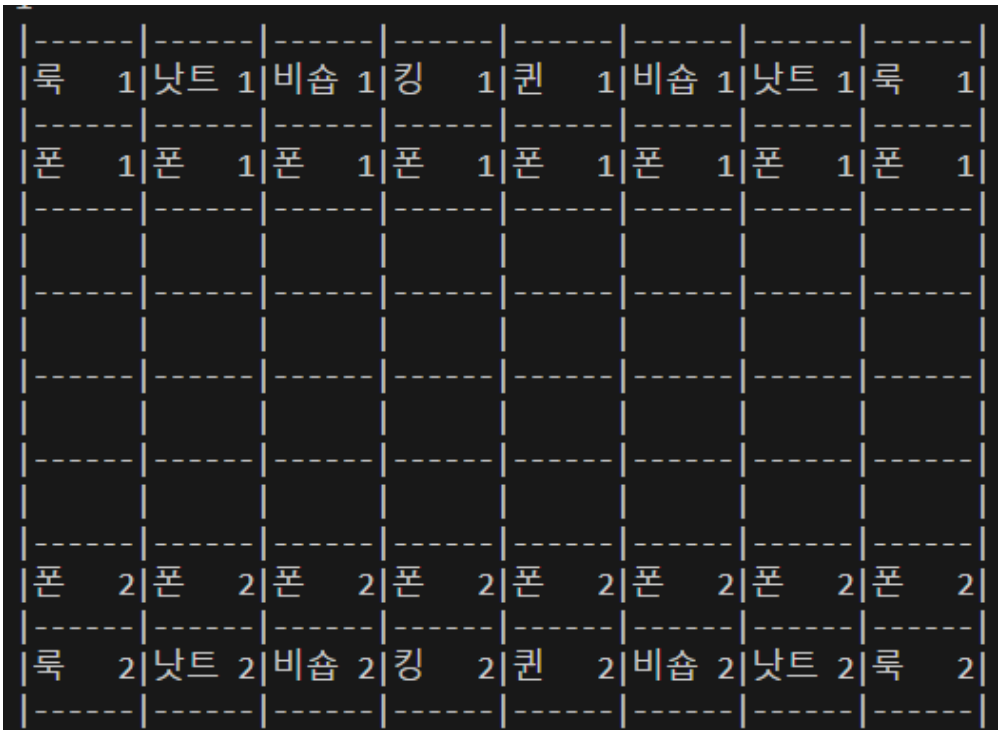


(2) player가 2명일 때 기존의 체스게임 실행

세부 기능 1: 체스판 출력

- 설명: 2차원 배열의 각 인덱스를 출력해 체스판을 시각화

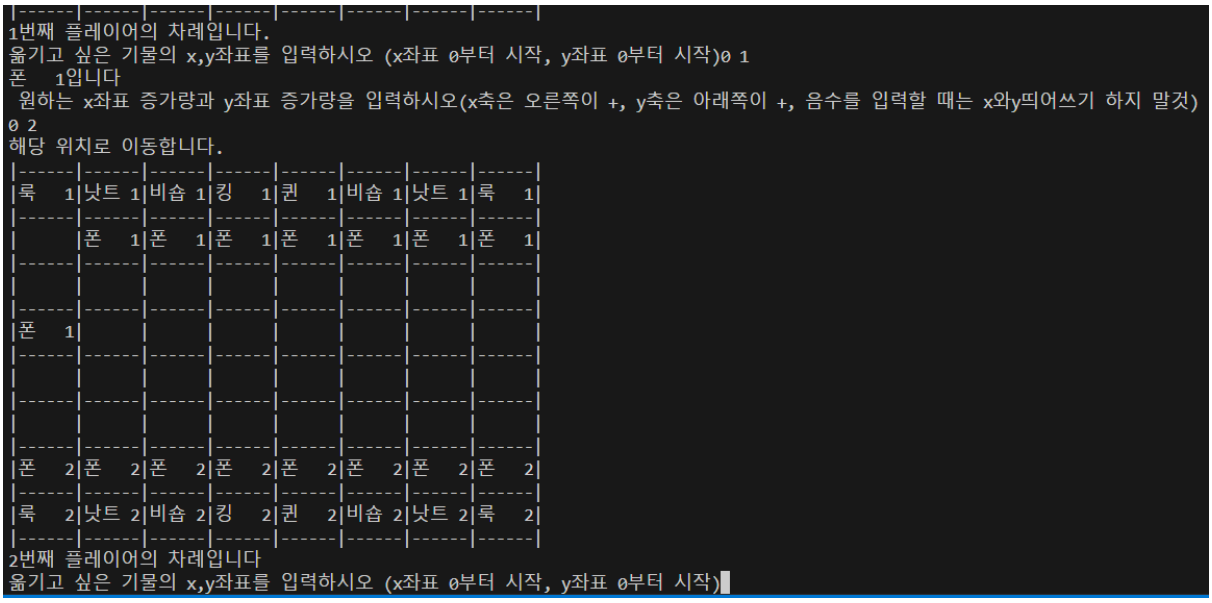
- 테스트 결과 스크린샷

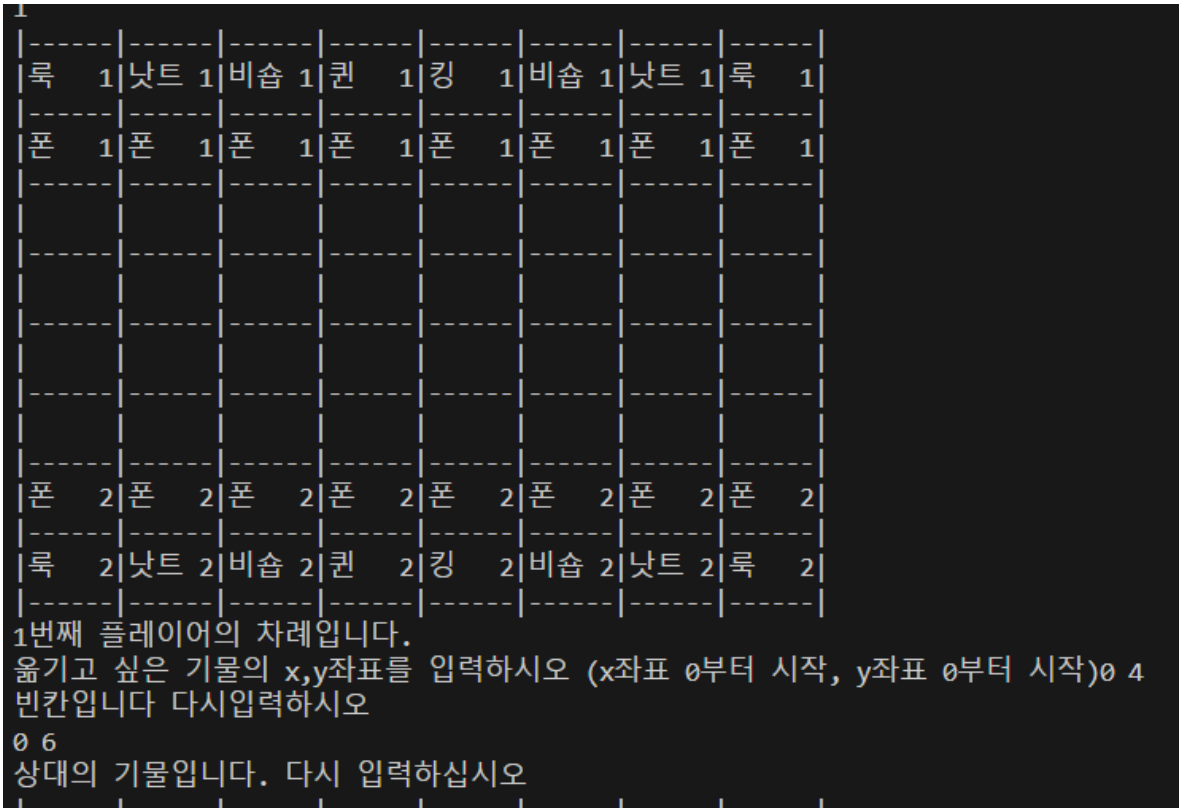


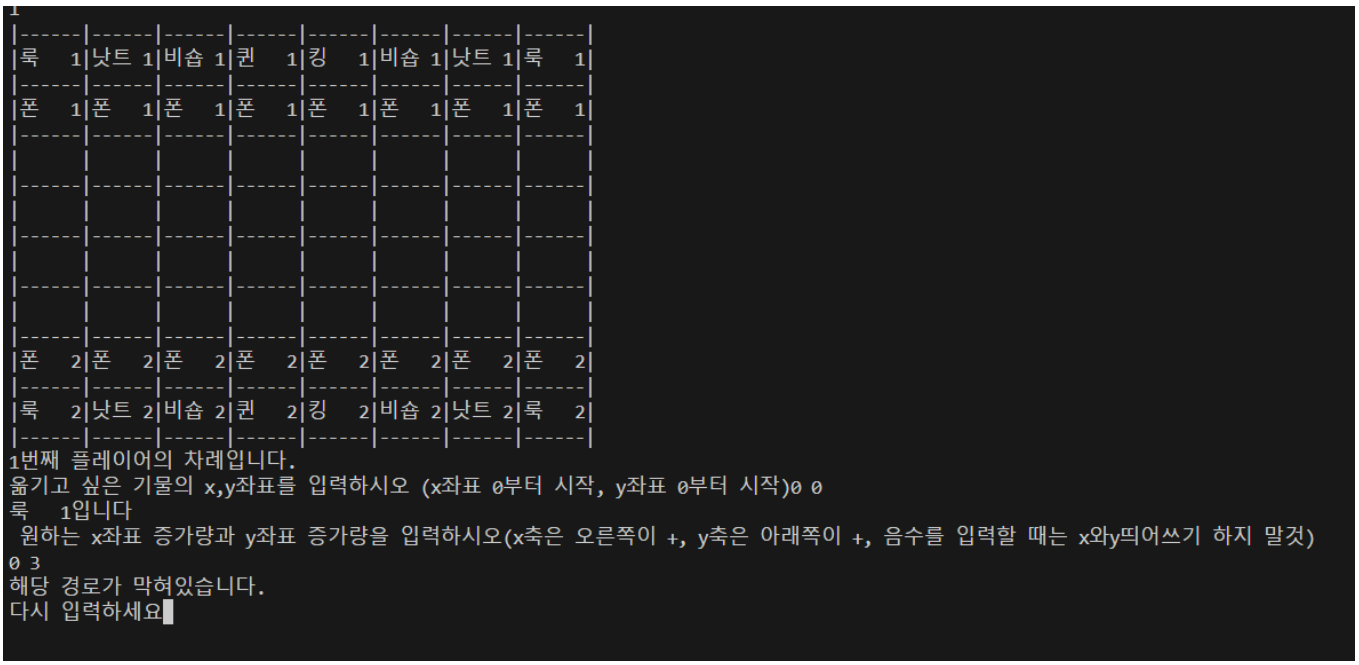
세부 기능 2: 사용자가 원하는 말을 이동하면 그 결과를 시각화 하여 출력

- 설명: 사용자가 원하는 말의 원하는 좌표를 선택하면 그 좌표로 말을 이동하고 그 결과를 보드판 출력을 통해 시각화

- 테스트 결과 스크린샷




1
룩 1 낫트 1 비숍 1 퀸 1 킹 1 비숍 1 낫트 1 룩 1
폰 1 폰 1 폰 1 폰 1 폰 1 폰 1 폰 1 폰 1
폰 2 폰 2 폰 2 폰 2 폰 2 폰 2 폰 2 폰 2
룩 2 낫트 2 비숍 2 퀸 2 킹 2 비숍 2 낫트 2 룩 2
1번째 플레이어의 차례입니다.
옮기고 싶은 기물의 x,y좌표를 입력하시오 (x좌표 0부터 시작, y좌표 0부터 시작)0 4
빈칸입니다 다시입력하시오
0 6
상대의 기물입니다. 다시 입력하십시오

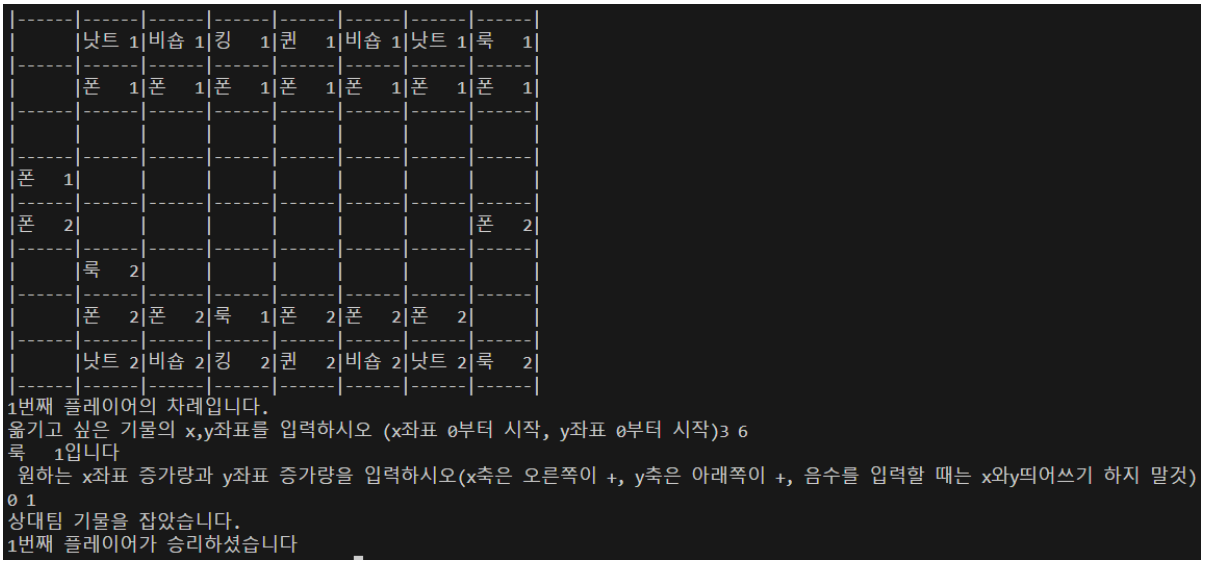

1
룩 1 낫트 1 비숍 1 퀸 1 킹 1 비숍 1 낫트 1 룩 1
폰 1 폰 1 폰 1 폰 1 폰 1 폰 1 폰 1 폰 1
폰 2 폰 2 폰 2 폰 2 폰 2 폰 2 폰 2 폰 2
룩 2 낫트 2 비숍 2 퀸 2 킹 2 비숍 2 낫트 2 룩 2
1번째 플레이어의 차례입니다.
옮기고 싶은 기물의 x,y좌표를 입력하시오 (x좌표 0부터 시작, y좌표 0부터 시작)0 0
룩 1입니다
원하는 x좌표 증가량과 y좌표 증가량을 입력하시오(x축은 오른쪽이 +, y축은 아래쪽이 +, 음수를 입력할 때는 x와y띄어쓰기 하지 말것)
0 3
해당 경로가 막혀있습니다.
다시 입력하세요

세부 기능 3: 체스 게임의 룰 적용

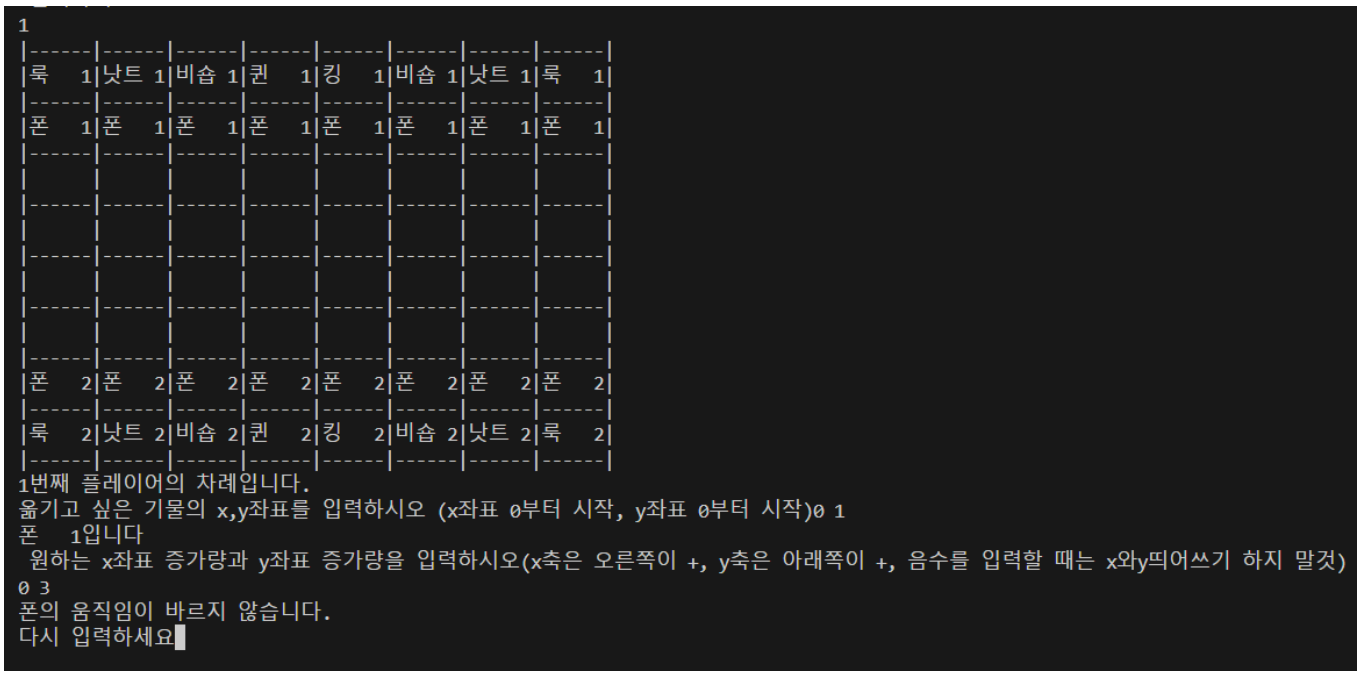
- 설명: 체스게임에는 특정 상황이 말의 기능이 평소와 달라지는 룰이 있기에 그 룰들을 함수화 하여 적용

- 테스트 결과 스크린샷

(킹이 죽으면 게임이 끝남)



(기물별로 움직이는 방식이 다름)

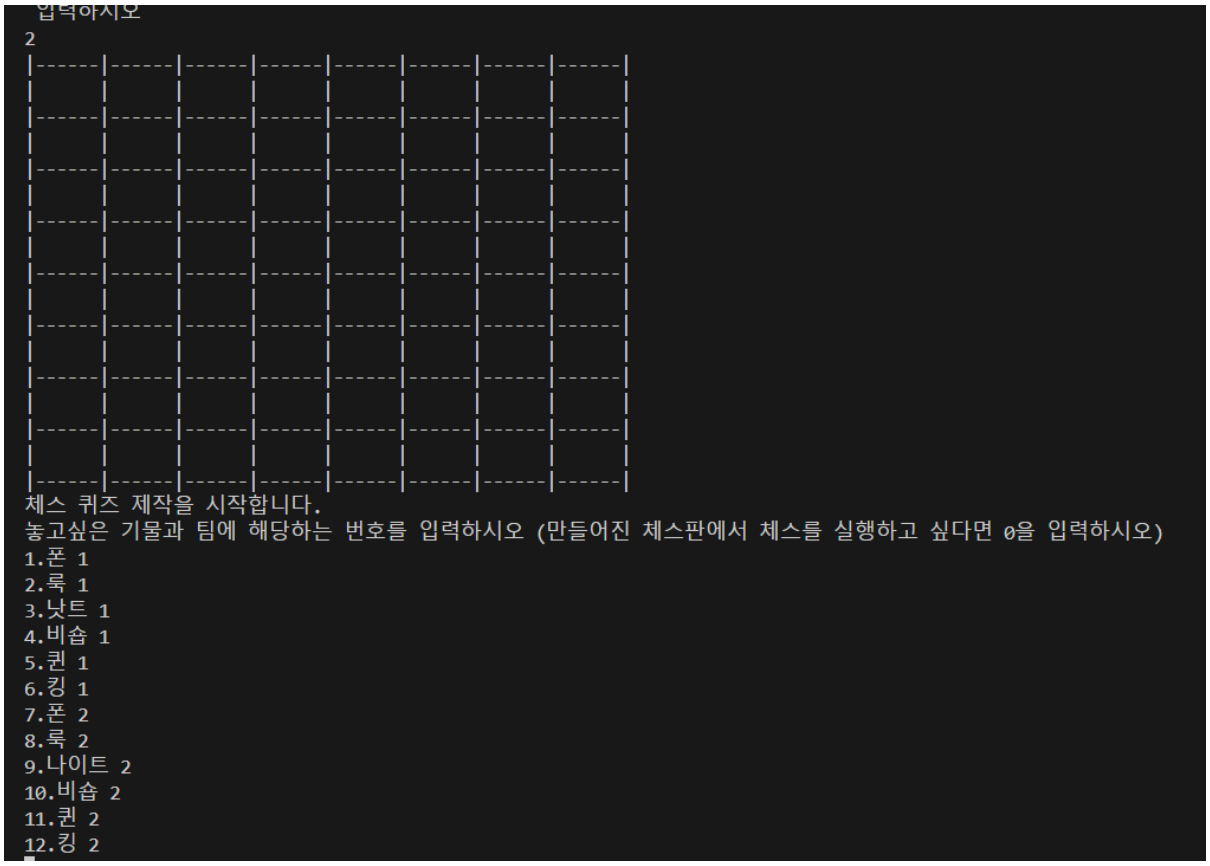


(3) 체스 퀴즈 툴

세부 기능 1: 원하는 말을 원하는 위치에 놓고 게임 실행

- 설명: 사용자가 원하는 말들을 원하는 위치에 놓고 게임을 시작할 수 있게 해주는 기능

- 테스트 결과 스크린샷



```
6
기물을 놓고싶은 x좌표와 y좌표를 입력해 주세요0 4
|------|------|------|------|------|------|------|------|
|      |      |      |      |      |      |      |      |
|------|------|------|------|------|------|------|------|
|      |      |      |      |      |      |      |      |
|------|------|------|------|------|------|------|------|
|      |      |      |      |      |      |      |      |
|------|------|------|------|------|------|------|------|
|      |      |      |      |      |      |      |      |
|------|------|------|------|------|------|------|------|
|      |      |      |      |      |      |      |      |
|------|------|------|------|------|------|------|------|
|킹   1|      |      |      |      |      |      |      |
|------|------|------|------|------|------|------|------|
|      |      |      |      |      |      |      |      |
|------|------|------|------|------|------|------|------|
|      |      |      |      |      |      |      |      |
|------|------|------|------|------|------|------|------|
|      |      |      |      |      |      |      |      |
|------|------|------|------|------|------|------|------|
|      |      |      |      |      |      |      |      |
|------|------|------|------|------|------|------|------|
놓고싶은 기물과 팀에 해당하는 번호를 입력하시오 (만들어진 체스판에서 체스를 실행하고 싶다면 0을 입력하시오)
1.폰 1
2.룩 1
3.낫트 1
4.비숍 1
5.퀸 1
6.킹 1
7.폰 2
8.룩 2
9.나이트 2
10.비숍 2
11.퀸 2
12.킹 2
12.킹 2
0
|------|------|------|------|------|------|------|------|
|      |      |      |      |      |      |      |      |
|------|------|------|------|------|------|------|------|
|      |      |      |      |      |      |      |      |
|------|------|------|------|------|------|------|------|
|      |      |      |      |      |      |      |      |
|------|------|------|------|------|------|------|------|
|      |      |      |      |      |      |      |      |
|------|------|------|------|------|------|------|------|
|      |      |      |      |      |      |      |      |
|------|------|------|------|------|------|------|------|
|킹   1|      |      |      |      |      |      |      |
|------|------|------|------|------|------|------|------|
|      |      |      |      |      |      |      |      |
|------|------|------|------|------|------|------|------|
|      |      |      |      |      |      |킹   2|      |
|------|------|------|------|------|------|------|------|
|      |      |      |      |      |      |      |      |
|------|------|------|------|------|------|------|------|
1번째 플레이어의 차례입니다.
옮기고 싶은 기물의 x,y좌표를 입력하시오 (x좌표 0부터 시작, y좌표 0부터 시작)
```

4. 계획 대비 변경 사항

1) 기능 2 - 세부 기능 1: 말 별로 class를 만들어 관리

- 이전: 같은 종류의 말별로 class 만들기

- 이후: piece class 하나 만들고 모든 말들이 사용하기

- 사유: class를 여러 개 만드는 것보다 piece class를 하나 만들고 name 필드를 추가하여 find 함수로 해당 객체가 어떤 종류의 말인지 판단하는 코드가 더 효율적이라 판단

2) 기능 2 - 세부 기능 4 : 체스 게임의 룰 적용

- 이전: 폰이 대각선으로 상대방을 잡는 것과, 기물을 넘을 수 있는 기물이 있고 넘을 수 없는 기물이 있는 체스 게임의 규칙을 11/17일까지 구현

- 이후: 12/15 ~ 12/22 기능 보완 기간에 구현

- 사유: 내가 구상했던 코드에 결점이 많아 이런 상황을 대비해 계획해 놓았던 기능 보완 기간에 해당 기능을 구현

3) 기능 2 – 세부기능 3 사용자가 원하는 말을 이동하면 그 결과를 시각화하여 출력

- 이전: 11/17일까지 해당 기능 구현

- 이후: 12/15 ~ 12/22 기능 보완 기간에 구현

- 사유: 세부기능 4번이 필요한 바탕이 돼야 가능한 기능임, 이 기능 또한 계획해 놓았던 기능 보완 기간에 해당 기간을 구현

4) 기능 2 – 세부기능 3 사용자가 이동을 원하는 말을 선택 하면 이동 가능 경로 시각화

- 이전: 11/17일까지 해당 기능 구현

- 이후: 12/15 ~ 12/22 기능 보완 기간에 구현

- 사유: 내가 구상했던 코드에 결점이 많아 이런 상황을 대비해 계획해 놓았던 기능 보완 기간에 해당 기능을 구현

4) 기능 3 – 세부기능 1 원하는 말을 원하는 위치에 놓기

- 이전: 세부기능의 이름이 "원하는 말을 원하는 위치에 놓기"

- 이후: 세부기능의 이름이 "원하는 기물을 원하는 위치에 놓고 게임 실행"

- 사유: 해당 기능을 더 잘 설명해주는 이름으로 수정

5. 프로젝트 일정

(진행한 작업과 진행 중인 작업 등을 표기)

업무		11/3	11/10	11/17	12/1	12/15	12/22
제안서 작성		완료					
기능1	세부기능1		완료				
기능2	세부기능1		완료				
	세부기능2		완료				
	세부기능3		----------→				
	세부기능4		완료				
	세부기능5		완료				
	세부기능6		완료				

업무		11/3	11/10	11/17	12/1	12/15	12/22
중간보고서1 작성			완료				
기능3	세부기능1			완료			
	세부기능2			완료			
	세부기능3			완료			
	세부기능4			완료			
	세부기능5			완료			
	세부기능6			완료			
	세부기능7			완료			
	세부기능8			---------->			
중간보고서2 작성					완료		
기능4	세부기능1				---------->		
	세부기능2				---------->		
	세부기능3				---------->		
중간보고서3 작성						---------->	
기능 보완						---------->	
최종보고서 작성							---------->