

C++ 프로그래밍 및 실습

# 체스 게임

진척 보고서 #1

제출일자: 2024.11.17

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 힘들. 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 턴을 번갈아 가면서 말을 움직이게 해주고 1인용이라면 체스 퀴즈를 풀 수 있게 작동하는 것을 목표로 함
- 사용자가 체스말들의 초기 위치를 직접 설정할 수 있는 툴을 제공해 풀어보고 싶었던 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 한다는 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 헛갈릴 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 없었음, 이는 재밌는 체스 퀴즈나 체스게임 복기 내용을 사람이 기억하는데 한계가 있음 이 프로그램은 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 4: 체스 게임의 룰 적용

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

(6) 세부 기능 5: 게임 종료

- 설명: 왕이 죽거나 사용자가 항복이라고 입력하면 게임을 종료하는 기능

### 3) 기능 3 체스 퀴즈 톨

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

(1) 세부 기능 1: 원하는 말을 원하는 위치에 놓기

- 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 두었던 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 (예: 반복문, 조건문, 클래스, 함수, 포인터 등)

- 조건문

- 코드 스크린샷

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

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

## 2. 각 기물의 x,y 값 배정 부분

### (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: c

- class의 필드값으로 기물들 자신에 대한 정보를 저장하게 만들었다. 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 (예: 반복문, 조건문, 클래스, 함수, 포인터 등)
- 클래스
- 캡슐화
- 함수의 구현은 다른 파일에서 하기
- 변수 이름이 같을 때 this -> 사용
- 객체



- 코드 스크린샷

```
#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;
}

#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 team_num);
    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 (예: 반복문, 조건문, 클래스, 함수, 포인터 등)

- 객체 배열

- 2중 for문

- 조건문

- 캡슐화

- 함수

- 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 (예: 반복문, 조건문, 클래스, 함수, 포인터 등)

- 2중 for문

- 객체 배열

- 조건문

- 캡슐화

- 함수

- 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("나트 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("나트 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:

- ture: 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 움직일 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
- 조건문
- 객체의 원본을 파라미터로 넘기기
- math 함수
- 문자열
- 함수



- 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("나트") != string::npos) {
        if (!(abs(plus_x) == 2 && abs(plus_y) == 1) || (abs(plus_x) == 1 && abs(plus_y) == 2)) {
            cout << "나트의 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 만들
- 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 함수를 true로 바꾸고 2중 for문 빠져나옴, 끝까지 2중 for문을 돌려도 일치하는 x,y좌표가 없었다면 " "을 출력해 빈칸임을 보여줌
- 적용된 배운 내용 (예: 반복문, 조건문, 클래스, 함수, 포인터 등)
- 객체 배열
- 이중 for문을 이용한 보드판 출력
- private로 선언된 class 값을 함수로 꺼내 사용
- break
- 조건문
- 함수

- 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 3: 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 숫자로 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 살아있으면 black\_life를 1로 변경, 무한루프 끝에서 white\_color or balck\_color의 값이 여전히 0이면 king이 죽었다는 것으로 생각하고, 게임의 결과를 출력 및 break 문을 통해 게임을 끝냄
- 
- 적용된 배운 내용 (예: 반복문, 조건문, 클래스, 함수, 포인터 등)
  - 이중 for문
  - 조건문
  - 배열객체
  - 배열객체를 파라미터로 넘기기
  - 전역변수
  - 2차원 배열
  - 함수
  - while문을 이용해 무한 loop

- 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;
```

```
        }
        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 입력하십시오(x축은 오른쪽이 +, y축은 아래가 +)\n";
            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;
}

```

## 2) 테스트 결과

### (1) 사용자가 원하는 기능 판단

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

```
1.체스 게임
2.체스 퀴즈 제작
입력하시오
1
|-----|-----|-----|-----|-----|-----|-----|-----|
|룩  1|나트 1|비숍 1|킹   1|퀸   1|비숍 1|나트 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|나트 2|비숍 2|킹   2|퀸   2|비숍 2|나트 2|룩   2|
|-----|-----|-----|-----|-----|-----|-----|-----|
1번째 플레이어의 차례입니다.
옮기고 싶은 기물의 x,y좌표를 입력하시오 (x좌표 0부터 시작, y좌표 0부터 시작)
```

### (1) player가 2명일 때 기존의 체스게임 실행

#### 세부 기능 1: 체스판 출력

- 설명: 2차원 배열의 각 인덱스를 출력해 체스판을 시각화

룩 1	나트 1	비숍 1	킹 1	퀸 1	비숍 1	나트 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	나트 2	비숍 2	킹 2	퀸 2	비숍 2	나트 2	룩 2

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

```

1번째 플레이어의 차례입니다.
움기고 싶은 기물의 x,y좌표를 입력하십시오 (x좌표 0부터 시작, y좌표 0부터 시작)0 1
폰 1입니다
원하는 x좌표 증가량과 y좌표 증가량을 입력하십시오(x축은 오른쪽이 +, y축은 아래쪽이 +, 음수를 입력할 때는 x와y띄어쓰기 하지 말것)
0 2
해당 위치로 이동합니다.

|-----|-----|-----|-----|-----|-----|-----|-----|
|룩 1|나트 1|비숍 1|킹 1|퀸 1|비숍 1|나트 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|나트 2|비숍 2|킹 2|퀸 2|비숍 2|나트 2|룩 2|
|-----|-----|-----|-----|-----|-----|-----|-----|

2번째 플레이어의 차례입니다
움기고 싶은 기물의 x,y좌표를 입력하십시오 (x좌표 0부터 시작, y좌표 0부터 시작)

```

### 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)

```

|-----|-----|-----|-----|-----|-----|-----|-----|
|      |나트 1|비숍 1|킹 1|퀸 1|비숍 1|나트 1|룩 1|
|-----|-----|-----|-----|-----|-----|-----|-----|
|      |폰 1|폰 1|폰 1|폰 1|폰 1|폰 1|폰 1|
|-----|-----|-----|-----|-----|-----|-----|-----|
|      |      |      |      |      |      |      |      |
|-----|-----|-----|-----|-----|-----|-----|-----|
|폰 1|      |      |      |      |      |      |      |
|-----|-----|-----|-----|-----|-----|-----|-----|
|폰 2|      |      |      |      |      |      |폰 2|
|-----|-----|-----|-----|-----|-----|-----|-----|
|      |룩 2|      |      |      |      |      |      |
|-----|-----|-----|-----|-----|-----|-----|-----|
|      |폰 2|폰 2|룩 1|폰 2|폰 2|폰 2|      |
|-----|-----|-----|-----|-----|-----|-----|-----|
|      |나트 2|비숍 2|킹 2|퀸 2|비숍 2|나트 2|룩 2|
|-----|-----|-----|-----|-----|-----|-----|-----|
1번째 플레이어의 차례입니다.
옮기고 싶은 기물의 x,y좌표를 입력하시오 (x좌표 0부터 시작, y좌표 0부터 시작)3 6
룩 1입니다
원하는 x좌표 증가량과 y좌표 증가량을 입력하시오(x축은 오른쪽이 +, y축은 아래쪽이 +, 음수를 입력할 때는 x와y띄어쓰기 하지 말것)
0 1
상대팀 기물을 잡았습니다.
1번째 플레이어가 승리하였습니다

```

## 4. 계획 대비 변경 사항

### 1) 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

## **1) 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

## **1) 세부 기능 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



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		(현재 일부 미완성)-----→				
	세부기능5		완료				
	세부기능6		완료				

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