c++ 프로그래밍 및 실습

체스 게임

진척 보고서 #3

제출일자: 2024.12.15
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:

- 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:

- 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 1로해당 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 옳지않다는 것을 사용자에게 알려주고 flase 리턴
- 폰이 상대 기물을 대각선으로만 잡을 수 있는 게임 규칙을 if문을 통해 구현
- 기물이 움직일 때 다른 기물들에 의해 막혀있는 곳으로 움직이려고 할 시 "해당 경로가 막혀있습니다" 출력하고 false를 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 ChessRule(Piece &piece, int plus_x, int plus_y)
{
    int original_x = piece.GetX();
    int original_y = piece.GetY();
    int new_x = original_x + plus_x;
    int new_y = original_y + plus_y;

    string name = piece.GetName();
    if (name.find("폰") != string::npos)
    {
        if (piece.GetTeamNum() == 0)
        {
            if (!(plus_x == 0 && (plus_y == 1 || (original_y == 1 && plus_y == 2))))
            {
                if (board[original_y + 1][original_x + 1] == 2 || board[original_y + 1][original_x - 1] == 2)
                {
                    if (plus_y == 1 && (plus_x == 1 || plus_x == -1))
                    {
                        return true;
                    }
                }
                else
                {
                    cout << "폰의 움직임이 바르지 않습니다.\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";
```

```
                {
                    cout << "해당 경로가 막혀있습니다\n";
                    return false;
                }
            }
            if ((plus_y == 1 && board[original_y + 1][original_x] == 2) || (plus_y == 2 && board[original_y + 2][original_x] == 2))
            {
                cout << "폰은 직선으로 상대 기물을 잡을 수 없습니다.\n";
                return false;
            }
        }
    }
    else
    {
        if (!(plus_x == 0 && (plus_y == -1 || (original_y == 6 && plus_y == -2))))
        {
            if (board[original_y - 1][original_x + 1] == 1 || board[original_y - 1][original_x - 1] == 1)
            {
                if (plus_y == -1 && (plus_x == 1 || plus_x == -1))
                {
                    return true;
                }
            }
            else
            {
                cout << "폰의 움직임이 바르지 않습니다.\n";
                return false;
            }
        }
        else
        {
```

```
        }
        else
        {
            for (int i = -1; i > plus_y; i--)
            {
                if (board[original_y + i][original_x] != 0)
                {
                    cout << "해당 경로가 막혀있습니다\n";
                    return false;
                }
            }

            if ((plus_y == -1 && board[original_y - 1][original_x] == 1) || (plus_y == -2 && board[original_y - 2][original_x] == 1))
            {
                cout << "폰은 직선으로 상대 기물을 잡을 수 없습니다.\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 (plus_x == 0)
        {
            if (plus_y > 0)
```

```
        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;
                    }
                }
            }
        }
    }
    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)
```

```
else if (name.find("비숍") != string::npos)
{
    if (abs(plus_x) != abs(plus_y))
    {
        cout << "비숍의 움직임이 바르지 않습니다.\n";
        return false;
    }
    else
    {
        if (plus_x > 0)
        {
            if (plus_y > 0)
            {
                for (int i = 1; i < plus_y; i++)
                {
                    for (int j = 1; j < plus_x; j++)
                    {

                        if (board[original_y + i][original_x + j] != 0)
                        {
                            cout << "해당 경로가 막혀있습니다.\n";
                            return false;
                        }
                    }
                }
            }
            else
            {
                for (int i = 0; i > plus_y; i--)
                {
                    for (int j = 0; j < plus_x; j++)
                    {
```

```
else if (name.find("비숍") != string::npos)
    else
        if (plus_x > 0)
            else
                    for (int j = 0; j < plus_x; j++)
                    {

                        if (board[original_y + i][original_x + j] != 0)
                        {
                            cout << "해당 경로가 막혀있습니다.\n";
                            return false;
                        }
                    }
                }
            }
        }
        else
        {
            if (plus_y > 0)
            {
                for (int i = 0; i < plus_y; i++)
                {
                    for (int j = 0; j > plus_x; j--)
                    {

                        if (board[original_y + i][original_x + j] != 0)
                        {
                            cout << "해당 경로가 막혀있습니다.\n";
                            return false;
                        }
                    }
                }
            }
            else
            {
                for (int i = 0; i > plus_y; i--)
```

```
                {
                    for (int i = 0; i > plus_y; i--)
                    {
                        for (int j = 0; j > plus_x; j--)
                        {

                            if (board[original_y + i][original_x + j] != 0)
                            {
                                cout << "해당 경로가 막혀있습니다.\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;
```

```
else if (abs(plus_x) == abs(plus_y))
{
    if (plus_x > 0)
    {
        if (plus_y > 0)
        {
            for (int i = 0; i < plus_y; i++)
            {
                for (int j = 0; j < plus_x; j++)
                {

                    if (board[original_y + i][original_x + j] != 0)
                    {
                        cout << "해당 경로가 막혀있습니다.\n";
                        return false;
                    }
                }
            }
        }
        else
        {
            for (int i = 0; i > plus_y; i--)
            {
                for (int j = 0; j < plus_x; j++)
                {

                    if (board[original_y + i][original_x + j] != 0)
                    {
                        cout << "해당 경로가 막혀있습니다.\n";
                        return false;
                    }
                }
            }
```

```
    else
    {
        if (plus_y > 0)
        {
            for (int i = 0; i < plus_y; i++)
            {
                for (int j = 0; j > plus_x; j--)
                {

                    if (board[original_y + i][original_x + j] != 0)
                    {
                        cout << "해당 경로가 막혀있습니다.\n";
                        return false;
                    }
                }
            }
        }
        else
        {
            for (int i = 0; i > plus_y; i--)
            {
                for (int j = 0; j > plus_x; j--)
                {

                    if (board[original_y + i][original_x + j] != 0)
                    {
                        cout << "해당 경로가 막혀있습니다.\n";
                        return false;
                    }
                }
```

```
else if (plus_x == 0 || plus_y == 0)
{
    if (plus_x == 0)
    {
        if (plus_y > 0)
        {
            for (int i = 0; i < plus_y; i++)
            {
                if (board[original_y + i][original_x] != 0)
                {
                    cout << "해당 경로가 막혀있습니다.\n";
                    return false;
                }
            }
        }
        else
        {
            for (int i = 0; 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 = 0; i < plus_x; i++)
                    {
                        if (board[original_y][original_x + i] != 0)
                        {
                            cout << "해당 경로가 막혀있습니다.\n";
                            return false;
                        }
                    }
                }
                else
                {
                    for (int i = 0; i > plus_x; i--)
                    {
                        if (board[original_y][original_x + i] != 0)
                        {
                            cout << "해당 경로가 막혀있습니다.\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

출력:

- 현재 기물이 가지고 있는 x,y값을 바탕으로 체스판을 출력

설명

- 2중 for문을 만들어 체스판을 출력
- 2중 for문으로 체스판을 출력하려면 해당 칸에 기물이 있는지 확인하고, 있다면 기물의 이름을, 없다면 빈 칸을 출력해야함, 이를 위해 2중 for문 안에 또 2중 for문을 돌려 chess_piece 객체 배열 원소에 접근해 x,y값을 받아오고 출력중인 체스판의 좌표와 일치하면 해당 배열 원소의 name값을 출력했다. name값을 출력할 때 죽어있는 체스 기물을 체스판에 출력하면 안되니까 life값이 1인 기물만 name값을 출력하게 했다.

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

```
void PrintBoard(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].GetX();
                    y = chess_piece[q][w].GetY();
                    if (x == j && y == i && chess_piece[q][w].GetLife() == 1)
                    {
                        chess_piece[q][w].PrintName();
                        found = true;
                        break;
                    }
                }
                if (found)
                    break;
            }
            if (!found)
                cout << "      ";
        }
        cout << "|" << endl;
    }
    cout << "|------|------|------|------|------|------|------|------|" << endl;
}
```

(3) 세부기능 3: 사용자가 이동을 원하는 말을 선택 하면 이동 가능 경로 시각화

입출력

입력:

- movable: 사용자가 선택한 기물의 이동가는 경로를 바고싶은지에 대한 의사가 저장된 변수
- chess_piece: 기물들의 정보를 저장해 놓은 객체배열
- a: MovableChessRule, MovableBoardCheck 함수에 plus_x 부분으로 들어가는 값
- b: MovableChessRule, MovableBoardCheck 함수에 plus_y 부분으로 들어가는 값

출력:

- 해당 기물이 움직일 수 있는 x,y 증가량을 시각화

설명

- 위 기능을 사용할지 사용자에게 물어보고 사용자가 1을 입력했을 때 조건문을 활용해 해당 기능을 수행한다.
- 위 기능을 구현하기 위해 MovablChessRule 함수와 MovableBoardCheck 함수를 만들었다.
  이 2 함수는 ChessRule 함수와 BoardCheck함수를 수정해서 만든 함수로 이 두 함수 내에서 프린트하는 명령을 빼고 bool 형식의 리턴값만을 반환한다. ChessRule함수는 위에서 소개했듯 해당 기물의 움직임이 체스 규칙상 올바른지를 확인하고, BoardCheck 함수는 뒤에 자세한 설명이 나오지만 원하는 좌표로 기물을 이동시켰을 때 이동된 좌표가 빈칸인지 기물이 있는 자리인지에 따라 bool 값을 반환하는 함수이다. BoardCheck 함수는 올바른 이동이었다면 true를 반환하고 기물의 좌표를 옮겨주고
  잘못된 이동이었다면 기물의 좌표를 원래 있던 곳에 놔두고 false를 반환하지만 MoableBoardCheck 함수에서는 어떤 상황에서든 기물의 x,y,값을 기존 값으로 놔두고 true, false만 반환한다.
- MovableChessRule 함수와 MovabeBoardCheck 함수를 모두 통과한 기물과 x,y 증가량은 해당 기물이 움직일 수 있는 좌표에 해당한다.
- 플레이어가 움직이고 싶은 기물 하나를 고른 후 2중 for문을 돌려 체스판 내에서 움직일 수 있는 모든 x,y 증가량을 넣고 위 두 함수를 돌린다. 2함수 모두 true 값이 나왔다면 해당 x증가량과 y 증가량은 해당 기물이 움직일 수 있는 좌표라 판단하고 플레이어에게 알려준다.

적용된 배운 내용

- 객체 배열 (10주차)
- 조건문 (4주차)
- 2중 for문 (4주차)
- 함수 (6주차)
- 객체 주소를 매개변수로 넘기기 (11주차)
- 캡슐화 (9주차)

코드 스크린샷

```
{
    int movable;
    cout << chess_piece[i][j].GetName() << "이 이동할 수 있는 좌표증가량을 알고싶다면 1을 입력하고, 필요하지 않다면 0을 입력하시오" << endl;
    cin >> movable;
    if (movable == 1)
    {
        for (int a = -7; a < 8; a++)
        {
            for (int b = -7; b < 8; b++)
            {
                if (MovableChessRule(chess_piece[i][j], a, b))
                {   if(MovableBoardCheck(chess_piece, chess_piece[i][j], a, b))
                    {
                        cout << "x증가량:" << a << " y증가량:" << b << endl;
                    }

                }
            }
        }
    }

    cout << chess_piece[i][j].GetName() << "입니다\n원하는 x좌표 증가량과 y좌표 증가량을 입력하시오(x,y축은 0부터 시작합니다. x축은 오른쪽이
```

(4) 세부기능 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 선언되어있음, 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 객체의life를 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 게임을 선택했다면 while문을 이용해 무한루프
- make_piece, naming_piece, board_copy 함수를 사용해 객체 배열의 원소 각각에 알맞은값을 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
- 조건문을 통해 누구의 턴인지에 따라 코드를 따로 작성함
- 옮기고 싶은 기물의 x,y 좌표를 받고 해당 좌표가 올바른지 체크
- 옮기고 싶은 기물의 x,y값을 -2 -2로 입력하면 항복
- 해당 좌표에 해당하는 기물의 이름을 알려주고 원하는 x, y 증가량 받기
- 기물을 잘못 선택했다면 x,y증가량에 -10 -10을 넣으면 기물을 다시 선택할 수 있음
- 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
- 무한 루프 끝에서 양팀 킹에 해당하는 객체에 접근해 life값을 확인, king의 life값이 0이라면 king이 죽은 것 이므로 상대 player가 승리하게 됨

- 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)

```
if (choice == 2)
}
while (1)
{
    int turn;
    int x, y;
    int moving_x, moving_y;

    PrintBoard(chess_piece);
    if ((k % 2) == 1)
    {
        cout << "1번째 플레이어의 차례입니다.\n";
        turn = 0;
    }
    else
    {
        cout << "2번째 플레이어의 차례입니다\n";
        turn = 1;
    }
    cout << "움기고 싶은 기물의 x,y좌표를 입력하시오 (x축은 오른쪽이 +, y축은 아래쪽이 +,좌표는 0부터 시작합니다, x,y에 -2를 입력하면 항복입니다.)\n";
    cin >> x >> y;
    if (x == -2 && y == -2)
    {
        if (turn == 0)
        {
            cout << "2번 플레이어가 승리하셨습니다.";
            return 1;
        }
        else
        {
            cout << "1번 플레이어가 승리하셨습니다.";
            return 1;
        }
    }
```

(1번 player차례)

```
    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].GetX() && y == chess_piece[i][j].GetY())
            {
                if (turn == 0)
                {
                    if (turn != chess_piece[i][j].GetTeamNum())
                    {
                        cout << "상대의 기물입니다. 다시 입력하십시오\n";
                        k++;
                    }
                    else
                    {
                        int movable;
```

```
else
{
    int movable;
    cout << chess_piece[i][j].GetName() << "이 이동할 수 있는 좌표증가량을 알고싶다면 1을 입력하고, 필요하지 않다면 0을 입력하시오" << endl;
    cin >> movable;
    if (movable == 1)
    {
        for (int a = -7; a < 8; a++)
        {
            for (int b = -7; b < 8; b++)
            {
                if (MovableChessRule(chess_piece[i][j], a, b))
                {   if(MovableBoardCheck(chess_piece, chess_piece[i][j], a, b))
                    {
                        cout << "x증가량:" << a << " y증가량:" << b << endl;
                    }

                }
            }
        }
    }

    cout << chess_piece[i][j].GetName() << "입니다\n원하는 x좌표 증가량과 y좌표 증가량을 입력하시오(x,y축은 0부터 시작합니다. x축은 오른쪽이 +, y축은 아래
    cin >> moving_x >> moving_y;
```

```
                while (1)
                {
                    if (moving_x == -10 && moving_y == -10)
                    {
                        k++;
                        break;
                    }
                    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 (만약 다른 말을 입력하고 싶다면 x,y에 -10을 입력하세요)";
                            cin >> moving_x >> moving_y;
                            break;
                        }
                        if (check == 1)
                        {
                            break;
                        }
                    }
                    else
                    {
                        cout << "다시 입력하세요 (만약 다른 말을 입력하고 싶다면 x,y에 -10을 입력하세요)";
                        cin >> moving_x >> moving_y;
                    }
                }

                BoardCopy(chess_piece);
            }
```

(2번 player 차례)

```
else
{
    if (turn != chess_piece[i][j].GetTeamNum())
    {
        cout << "상대의 기물입니다. 다시 입력하십시오\n";
        k++;
    }
    else
    {
        int movable;
        cout << chess_piece[i][j].GetName() << "이 이동할 수 있는 좌표증가량을 알고싶다면 1을 입력하고, 필요하지 않다면 0을 입력하시
        cin >> movable;
        if (movable == 1)
        {
            for (int a = -7; a < 8; a++)
            {
                for (int b = -7; b < 8; b++)
                {
                    if (MovableChessRule(chess_piece[i][j], a, b))
                    {   if(MovableBoardCheck(chess_piece, chess_piece[i][j], a, b))
                        {
                            cout << "x:" << a << " y:" << b << endl;
                        }

                    }
                }
            }
        }
        cout << chess_piece[i][j].GetName() << "입니다\n원하는 x좌표 증가량과 y좌표 증가량을 입력하시오(x,y축은 0부터 시작합니다. x
        cin >> moving_x >> moving_y;


                while (1)
                {
                    if (moving_x == -10 && moving_y == -10)
                    {
                        k++;
                        break;
                    }
                    if (ChessRule(chess_piece[i][j], moving_x, moving_y))
                    {
                        while (!BoardCheck(chess_piece, chess_piece[i][j], moving_x, moving_y))
                        {
                            cout << "다시 입력하세요 (만약 다른 말을 입력하고 싶다면 x,y에 -10을 입력하세요)";
                            cin >> moving_x >> moving_y;
                            break;
                        }
                        break;
                    }
                    else
                    {
                        cout << "다시 입력하세요 (만약 다른 말을 입력하고 싶다면 x,y에 -10을 입력하세요)";
                        cin >> moving_x >> moving_y;
                    }
                }

                BoardCopy(chess_piece);
            }
        }
    }
}
```

```
        if (chess_piece[0][12].GetLife() == 0)
        {
            cout << "2째 플레이어가 승리하셨습니다.";
            break;
        }
        else if (chess_piece[1][12].GetLife() == 0)
        {
            cout << "1번째 플레이어가 승리하셨습니다";
            break;
        }
        k++;
    }

    return 0;
}
```

```
    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) 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
- choice_num: 사용자가 원하는 기능을 고를 수 있게 입력을 저장하는 변수

출력:
- 출력값 없음

설명
- 사용자가 체스 퀴즈 제작을 고르면 객체 배열에 접근해 말들의 x,y좌표를 모드 -1로 설정, life값 0으로 설정, 2차원 배열 board의 모든 원소값을 0으로 설정해 비어있는 보드판 만들기
- 사용자가 기물 추가 or 게임 실행에 해당하는 1을 누르면 체스퀴즈 툴 실행
- while문을 이용해 무한루프를 만들고 계속해서 기물과 좌표를 선택
- 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
- 놓고싶은 기물을 잘못 골랐다면 x_dot, y_dot에 -1 -1을 넣으면 기물을 다시 고를 수 있음
- 저장된 x,y 값을 choiece_piece의 x,y 값으로 설정
- 잘못된 x,y값을 입력하면 무한루프로 다시 받기
- choice_piece의 life 값을 1로 설정
- print_board함수를 실행해 현재 보드판을 보여주고 다시 무한루프
- 만약 사용자가 기물을 고르는 부분에서 0을 입력하였다면 체스판이 완성되었다고 해석하고 king 2개가 있는지 확인 후 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)
- while문(4주차)
- do-while문(4주차)
- 2차원 배열(5주차)
- 포인터를 통해 객체 넘기기 (11주차)
- 객체(9주차)
- 함수(6주차)

코드 스크린샷

```
do
{
    cout << "1.체스 게임\n2.체스 퀴즈 제작\n 입력하시오" << endl;
    cin >> choice;
    if (choice != 1 && choice != 2)
    {
        cout << "잘못된 입력값입니다." << endl;
        continue;
    }
    break;
} while (1);

if (choice == 2)
{

    for (int i = 0; i < 2; i++)
    {
        for (int j = 0; j < 16; j++)
        {
            chess_piece[i][j].SetX(-1);
            chess_piece[i][j].SetY(-1);
            chess_piece[i][j].SetLife(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)
{
    int num;
    int x_dot, y_dot;
    int play_game = 0;
    int choice_num;
    Piece *choice_piece;
    cout << "1. 기물 추가 or 게임 실행\n2. 기물 삭제\n3.게임 저장\n4.저장된 게임 보기\n";
    cin >> choice_num;
    if (choice_num == 1)
    {
        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)
            {
                for (int i = 0; i < 7; i++)
                {
                    if (chess_piece[0][i].GetLife() == 0)
                    {
                        choice_piece = &chess_piece[0][i];
                        break;
                    }
                }
                break;
            }
            else if (num == 2)
            {
                if (chess_piece[0][8].GetLife() == 1)
                {
```

```
                if (chess_piece[0][8].GetLife() == 1)
                {
                    choice_piece = &chess_piece[0][15];
                }
                else
                {
                    choice_piece = &chess_piece[0][8];
                }

                break;
            }
            else if (num == 3)
            {
                if (chess_piece[0][9].GetLife() == 1)
                {
                    choice_piece = &chess_piece[0][14];
                }
                else
                {
                    choice_piece = &chess_piece[0][9];
                }

                break;
            }
            else if (num == 4)
            {
                if (chess_piece[0][10].GetLife() == 1)
                {
                    choice_piece = &chess_piece[0][13];
                }
                else
                {
```

```
        }
        else
        {
            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)
    {
        for (int i = 0; i < 8; i++)
        {
            if (chess_piece[1][i].GetLife() == 0)
            {
                choice_piece = &chess_piece[1][i];
                break;
            }
        }
        break;
    }
    else if (num == 8)
    else if (num == 8)
    {
        if (chess_piece[1][8].GetLife() == 1)
        {
            choice_piece = &chess_piece[1][15];
        }
        else
        {
            choice_piece = &chess_piece[1][8];
        }

        break;
    }
    else if (num == 9)
    {
        if (chess_piece[1][9].GetLife() == 1)
        {
            choice_piece = &chess_piece[1][14];
        }
        else
        {
            choice_piece = &chess_piece[1][9];
        }

        break;
    }
    else if (num == 10)
    {
        if (chess_piece[1][10].GetLife() == 1)
        {
```

```
        }
        else if (num == 10)
        {
            if (chess_piece[1][10].GetLife() == 1)
            {
                choice_piece = &chess_piece[1][13];
            }
            else
            {
                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)
        {
            play_game = 1;
            break;
        }
        else
        {
    else
    {
        cout << "잘못 입력하셨습니다.";
    }
}
if (play_game == 1)
{
    if (chess_piece[0][12].GetLife() == 0 || chess_piece[1][12].GetLife() == 0)
    {
        cout << "king 기물이 없기에 게임을 진행할 수 없습니다." << endl;
        continue;
    }
    BoardCopy(chess_piece);
    break;
}

do
{
    cout << "기물을 놓고싶은 x좌표와 y좌표를 입력해 주세요(좌표는 0부터 시작합니다),(말을 잘못 고르셨다면 x,y값에 -1을 넣으세요)" << endl;
    cin >> x_dot >> y_dot;
    if (x_dot < 0 || x_dot > 8 || y_dot < 0 || y_dot > 8 || (board[y_dot][x_dot] != 0))
    {
        cout << "잘못된 좌표값입니다." << endl;
        continue;
    }
    break;
} while (1);

if (x_dot == -1 && y_dot == -1)
{
    continue;
}
    }
    choice_piece->SetX(x_dot);
    choice_piece->SetY(y_dot);
    choice_piece->SetLife(1);
    BoardCopy(chess_piece);
    PrintBoard(chess_piece);
}
```

(2) 세부 기능 8: 잘못 놓여진 체스말 삭제

- 입출력

입력(변수):
- delete_x, delete_y: 삭제하고 싶은 기물의 x,y 좌표를 저장하는 변수
- board : 보드판에 해당하는 2차원 배열
- chess_piece: 기물들의 정보가 저장된 객체 배열

출력:
- 출력값 없음

설명
- 사용자가 체스말 삭제 기능에 해당하는 2번을 입력하면 해당 기능 실행
- 삭제하고 싶은 말위 좌표를 사용자로부터 받고delete_x, delete_y에 저장, 해당 좌표가 유효한지 검사
- 해당 좌표에 말이 없거나 유효하지 않은 좌표라면 왜 잘못됐는지 사용자에게 알려주고 다시 원하는 기능 입력받기
- 유효한 좌표라면 chess_piece의 모든 객체에 접근해 x,y값이 delete_x와delete_y와 일치하는 말을 찾고 해당 말의 x,y값을 -1로, life값을 0으로 설정
- BoardCopy함수를 통해 체스판을 동기화 하고 PrintBoard 함수를 통해 원하는 기물이 삭제된 체스판을 출력

적용된 배운 내용

- 조건문 (4주차)
- do-while문 (4주차)
- continue,break (4주차)
- while문 (4주차)
- 객체 배열 (10주차)
- 캡슐화 (9주차)
- 객체 (9주차)
- 함수(6주차)
- 배열(5주차)
- 배열 객체 주소를 매개변수로 넘기기 (11주차)

- 코드 스크린샷

```
else if(choice_num==2)
{
    int delete_x, delete_y;
    do
    {
        cout << "삭제하고 싶은 기물의 좌표를 입력하시오(x축은 오른쪽이 +, y축은 아래쪽이 +,좌표는 0부터 시작합니다)" << endl;
        cin >> delete_x >> delete_y;
        if (delete_x < 0 || delete_x > 8 || delete_y < 0 || delete_y > 8)
        {
            cout << "잘못된 좌표값입니다." << endl;
            continue;
        }
        break;
    } while (1);
    if (board[delete_y][delete_x] == 0)
    {
        cout << "해당 위치에는 말이 없습니다." << endl;
    }
    else
    {
        for (int i = 0; i < 2; i++)
        {
            for (int j = 0; j < 16; j++)
            {
                if (chess_piece[i][j].GetX() == delete_x && chess_piece[i][j].GetY() == delete_y)
                {
                    chess_piece[i][j].SetX(-1);
                    chess_piece[i][j].SetY(-1);
                    chess_piece[i][j].SetLife(0);
                    BoardCopy(chess_piece);
                    PrintBoard(chess_piece);
                    break;
                }
```

(4) 체스 퀴즈 저장 및 삭제

(1) 세부 기능 1: 체스 퀴즈 저장하기

- 입출력 입력(변수):
  - data_name: 저장할 데이터의 이름을 저장하는 변수
  - choice_num: 사용자가 원하는 기능을 저장하는 변수
  - os : 오픈한 파일에 해당하는 객체
  - chess_piece: 기물들의 정보가 저장된 객체배열

출력:
  - 출력값 없음

설명
- 사용자가 체스말 삭제 기능에 해당하는 3번을 입력하면 해당 기능 실행
- 저장할 데이터를 입력하고 data_name에 저장
- chess_data.txt 파일을 이어쓰기 모드로 열고 해당 파일을 의미하는 os 객체 설정
- os에 데이터 이름을 저장하고 줄 바꾸기
- chess_piece배열 원소들에 접근해서 x,y,life값을 txt파일에 저장
- 읽을 때 용이하기 위해 원소 하나 하나마다 줄 바꿈 문자를 넣어줌
- 다 저장하고 나면 close 함수로 파일을 닫아줌

적용된 배운 내용

- 조건문 (4주차)
- 반복문 (4주차)
- 객체 배열 (10주차)
- 캡슐화 (9주차)
- 객체 (9주차)
- 함수(6주차)
- 파일 출력(14주차)

- 코드 스크린샷

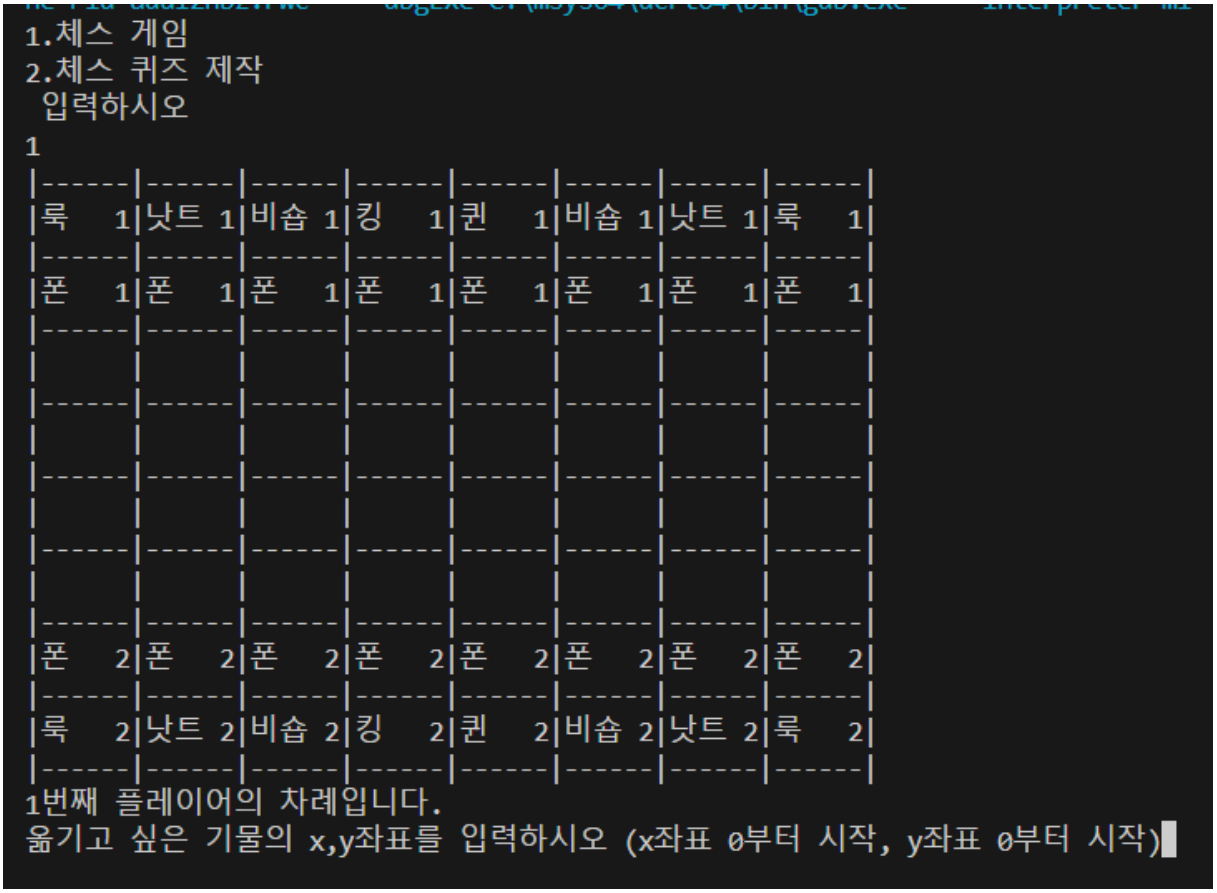
```
else if(choice_num==3)
{   string data_name;
    cout<<"저장할 데이터의 이름을 입력하시오"<<endl;
    cin >> data_name;
    ofstream os{"chess_data.txt", ios::app};
    if(!os)
    {
        cerr<<"파일 오픈에 실패하셨습니다."<<endl;
        exit(1);
    }
    os<<"data:"<<data_name<<"\n";
    for(int i=0;i<2;i++)
    {
        for(int j=0;j<16;j++)
        {
            os << chess_piece[i][j].GetLife() << " " << chess_piece[i][j].GetX() << " " << chess_piece[i][j].GetY() << " "<<endl
        }
    }
    os << "\n";
    os.close();
}
```

2) 테스트 결과

(1) 사용자가 원하는 기능 판단

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

- 테스트 결과 스크린샷

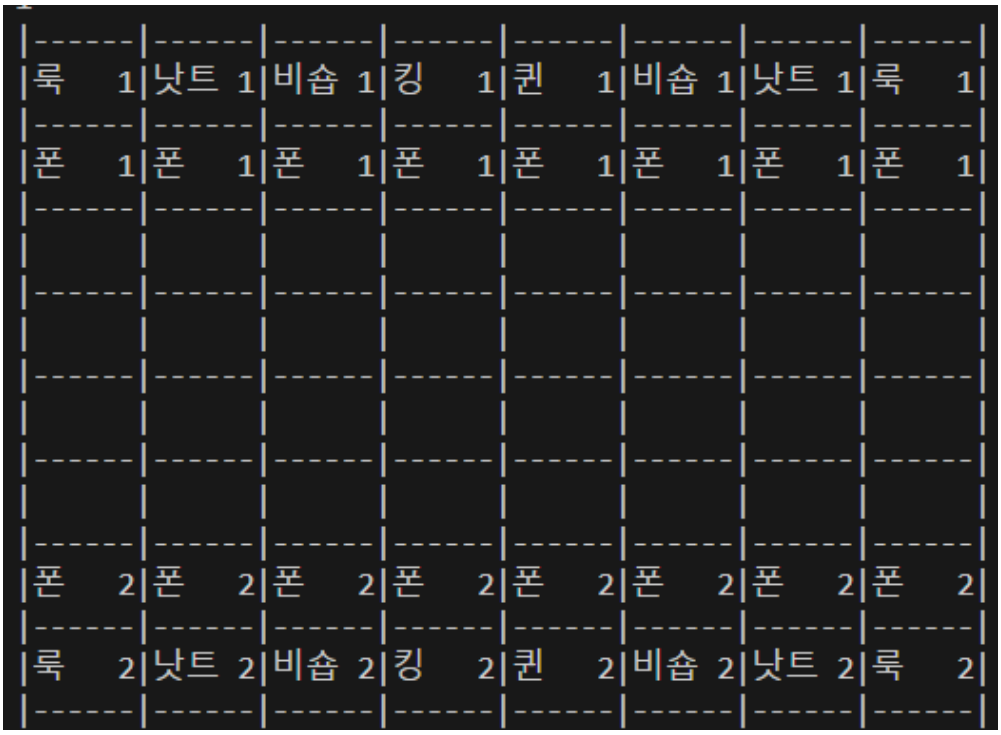


(2) player가 2명일 때 기존의 체스게임 실행

세부 기능 1: 체스판 출력

- 설명: 2차원 배열의 각 인덱스를 출력해 체스판을 시각화

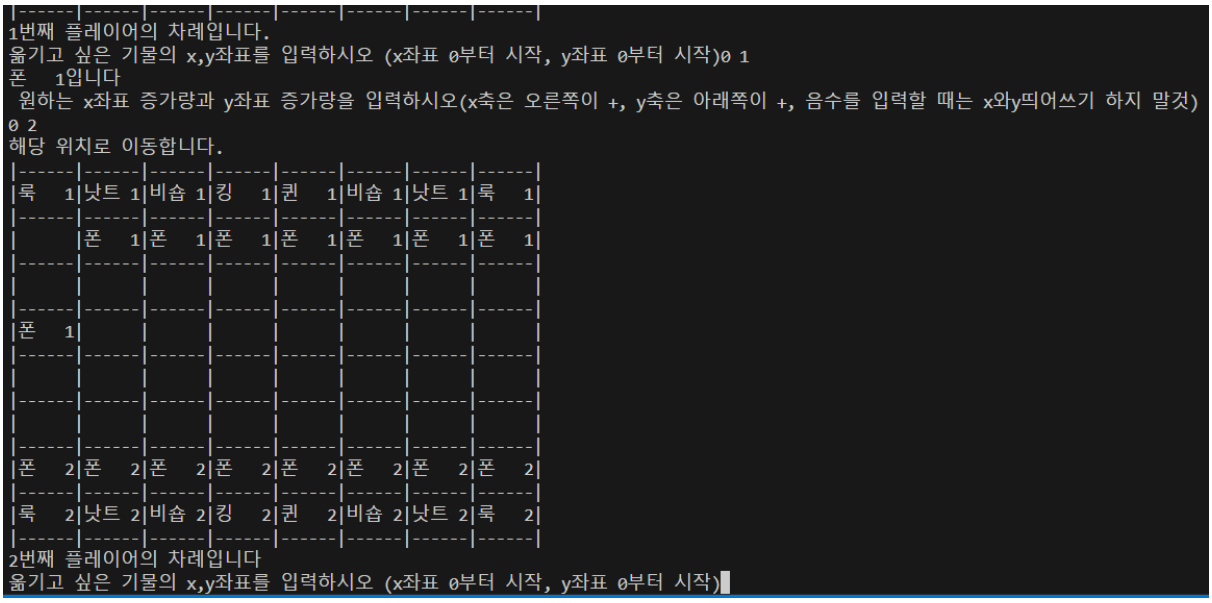
- 테스트 결과 스크린샷

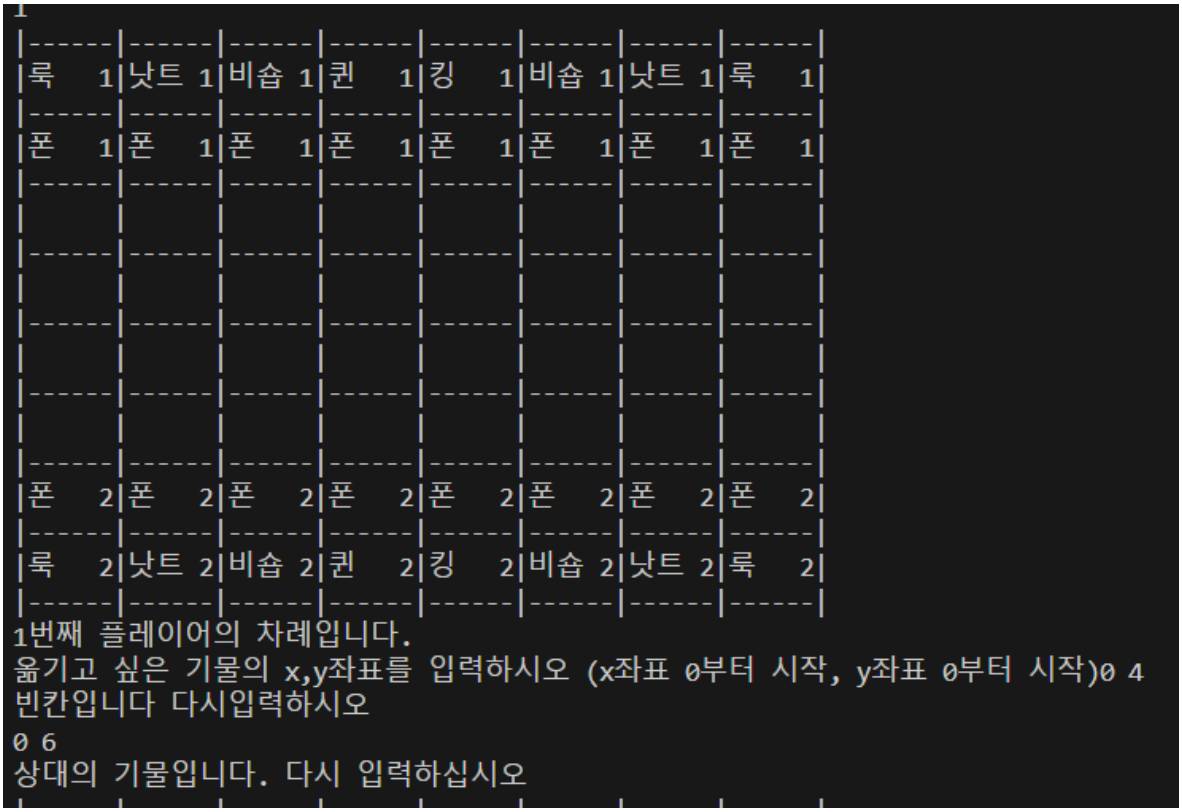


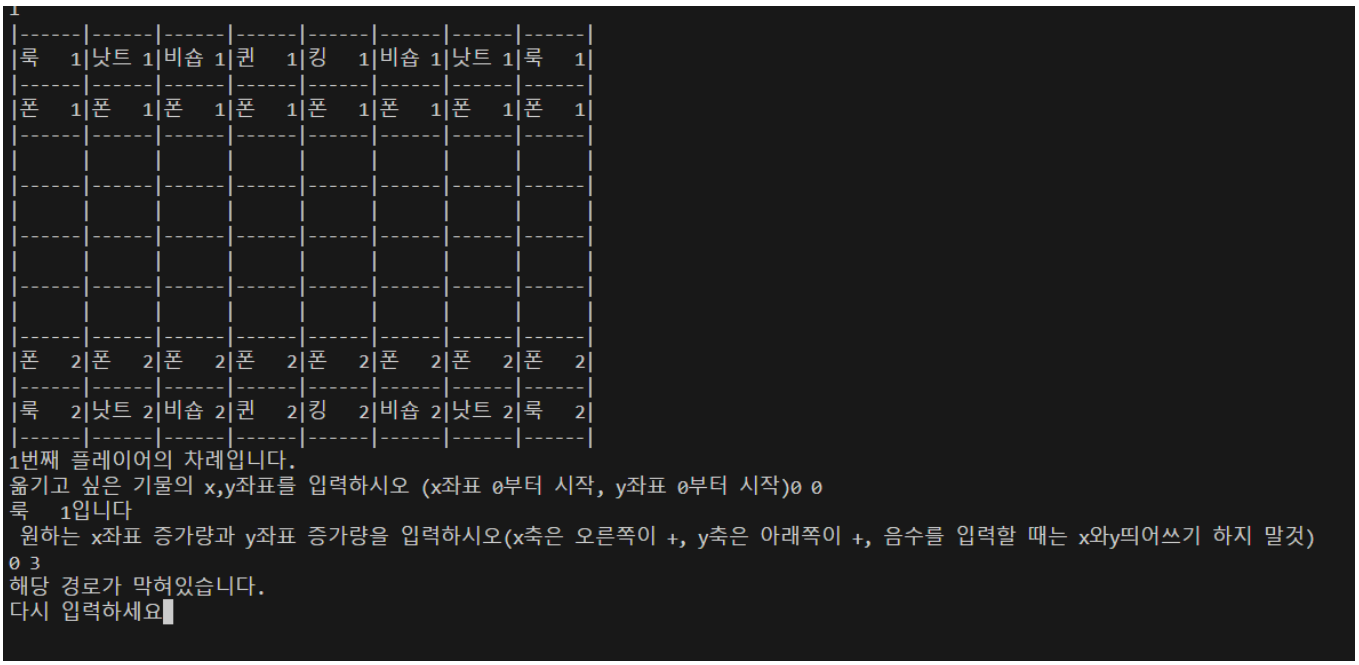
세부 기능 2: 사용자가 원하는 말을 이동하면 그 결과를 시각화 하여 출력

- 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

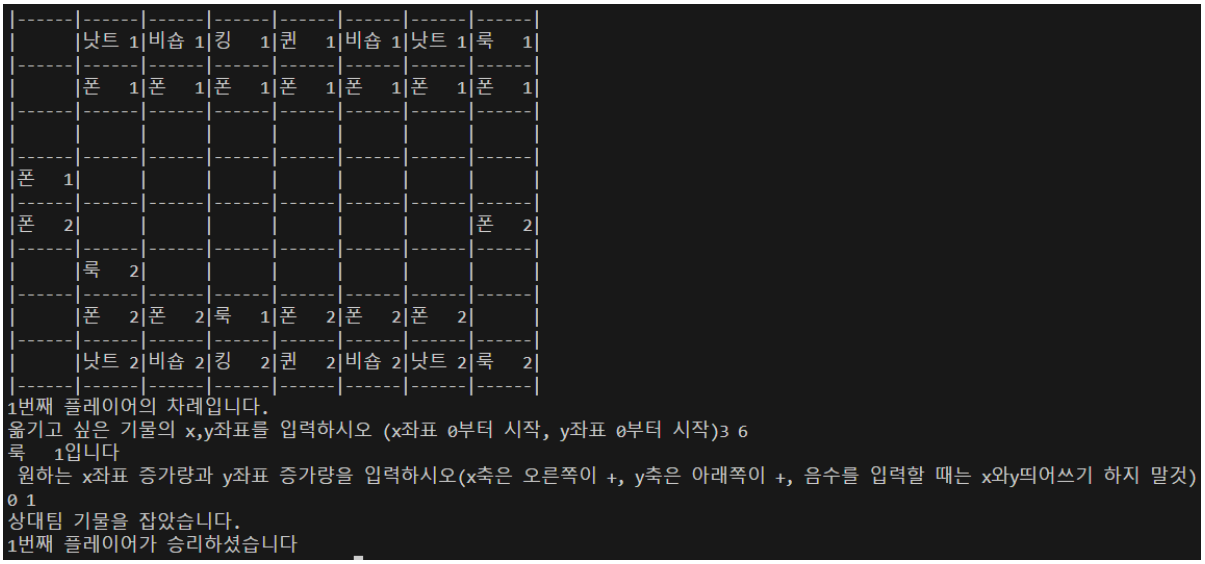

1
룩 1 낫트 1 비숍 1 퀸 1 킹 1 비숍 1 낫트 1 룩 1
폰 1 폰 1 폰 1 폰 1 폰 1 폰 1 폰 1 폰 1
폰 2 폰 2 폰 2 폰 2 폰 2 폰 2 폰 2 폰 2
룩 2 낫트 2 비숍 2 퀸 2 킹 2 비숍 2 낫트 2 룩 2
1번째 플레이어의 차례입니다.
옮기고 싶은 기물의 x,y좌표를 입력하시오 (x좌표 0부터 시작, y좌표 0부터 시작)0 0
룩 1입니다
원하는 x좌표 증가량과 y좌표 증가량을 입력하시오(x축은 오른쪽이 +, y축은 아래쪽이 +, 음수를 입력할 때는 x와y띄어쓰기 하지 말것)
0 3
해당 경로가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

세부 기능 3: 체스 게임의 룰 적용

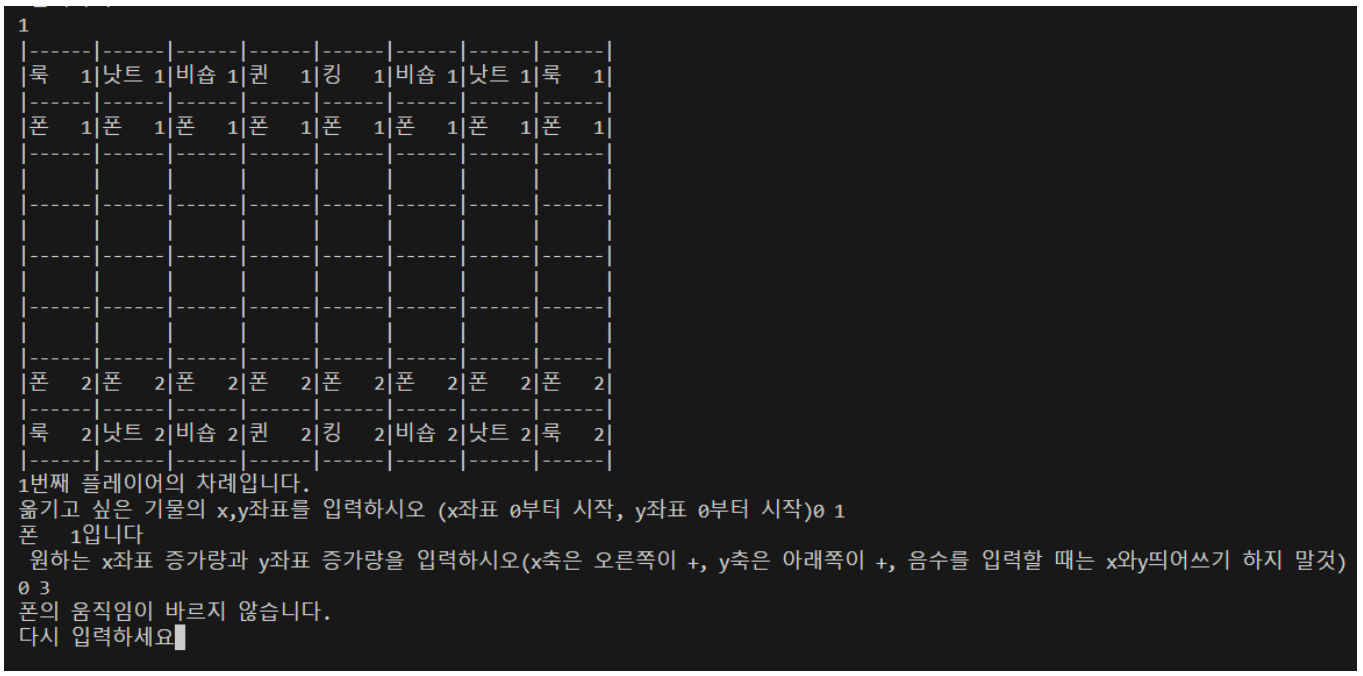
- 설명: 체스게임에는 특정 상황이 말의 기능이 평소와 달라지는 룰이 있기에 그 룰들을 함수화 하여 적용

- 테스트 결과 스크린샷

(킹이 죽으면 게임이 끝남)



(기물별로 움직이는 방식이 다름)

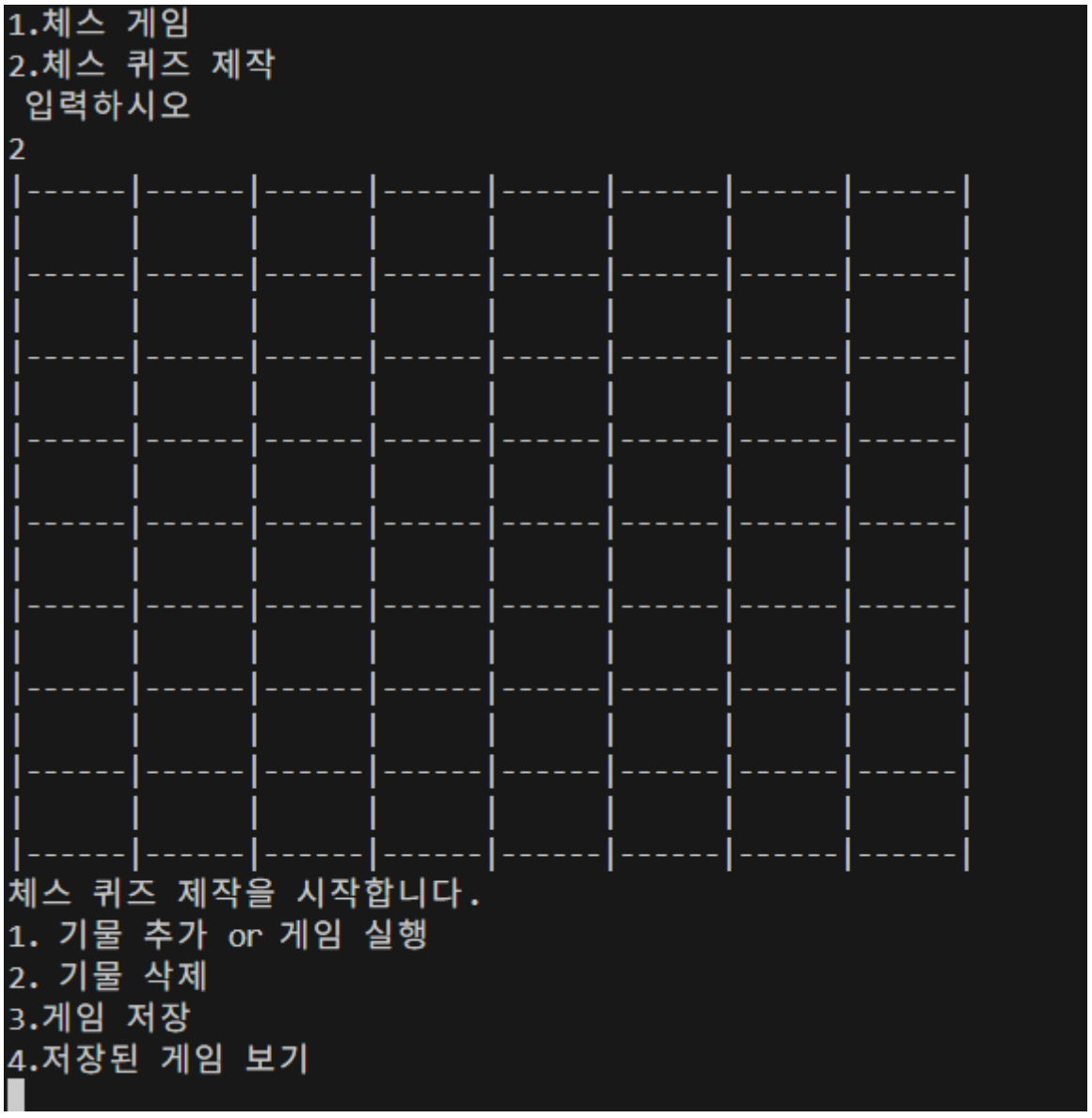


(3) 체스 퀴즈 툴

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

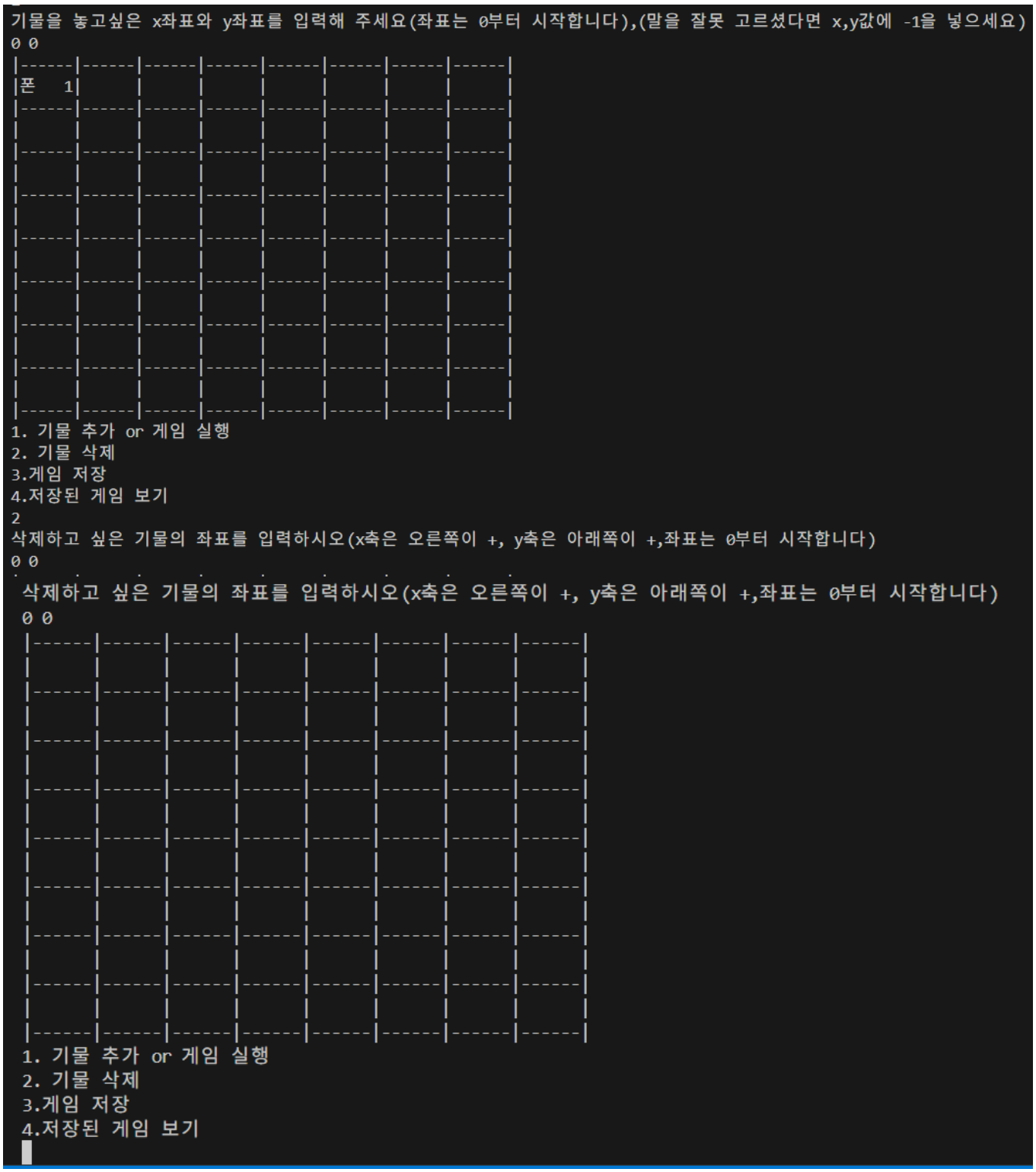


```
6
기물을 놓고싶은 x좌표와 y좌표를 입력해 주세요0 4
|------|------|------|------|------|------|------|------|
|      |      |      |      |      |      |      |      |
|------|------|------|------|------|------|------|------|
|      |      |      |      |      |      |      |      |
|------|------|------|------|------|------|------|------|
|      |      |      |      |      |      |      |      |
|------|------|------|------|------|------|------|------|
|      |      |      |      |      |      |      |      |
|------|------|------|------|------|------|------|------|
|      |      |      |      |      |      |      |      |
|------|------|------|------|------|------|------|------|
|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)
```

세부 기능 8: 잘못 놓여진 체스말 삭제

- 설명: : 체스판에 원하는 말을 놓다가 말이 잘못 놓여진다면 해당 말을 삭제하는 기능

- 테스트 결과 스크린샷

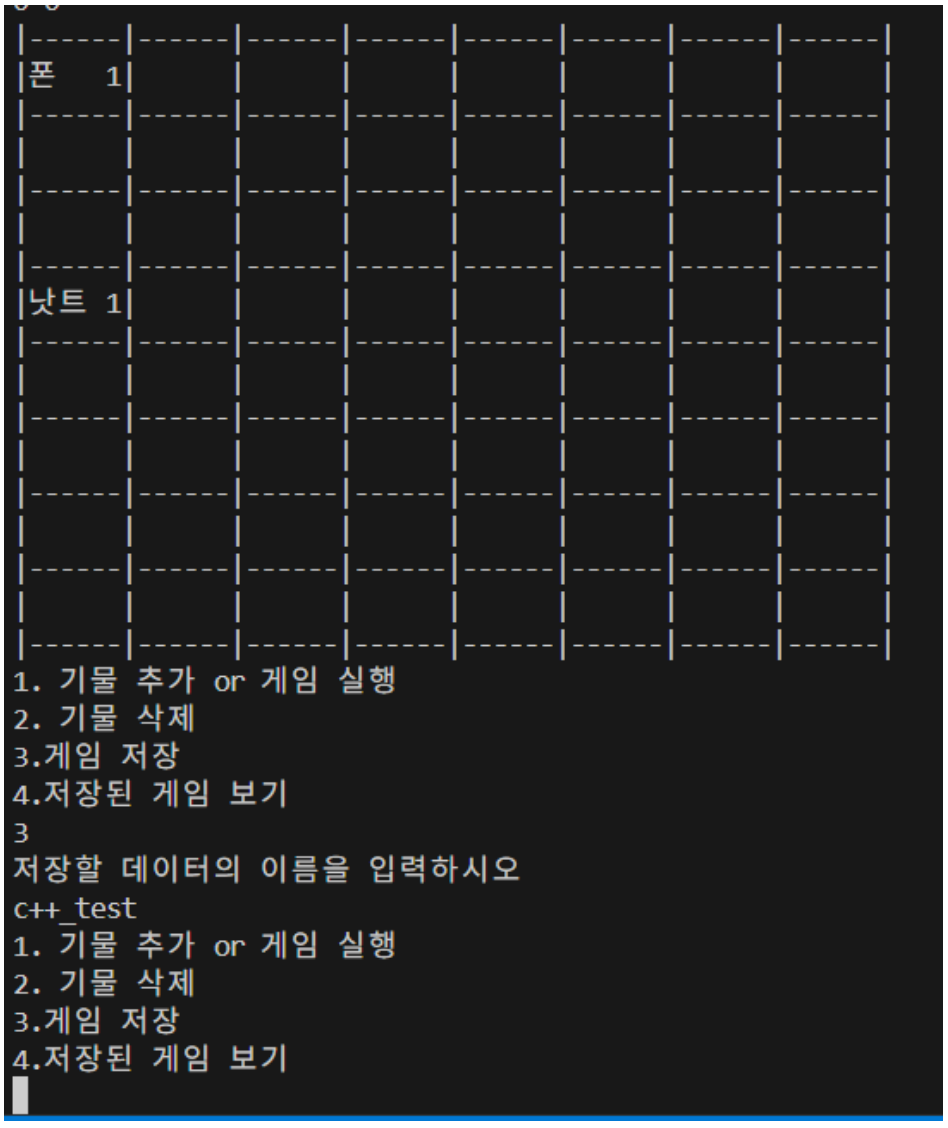


(3) 체스 퀴즈 저장 및 삭제

세부 기능 1: 만들어진 체스 퀴즈 저장하기

- 설명: : 체스판에 원하는 말을 놓다가 말이 잘못 놓여진다면 해당 말을 삭제하는 기능

- 테스트 결과 스크린샷



(txt파일)

```
data:c++_test
1 0 0
0 -1 -1
0 -1 -1
0 -1 -1
0 -1 -1
0 -1 -1
0 -1 -1
0 -1 -1
0 -1 -1
1 0 3
0 -1 -1
0 -1 -1
0 -1 -1
0 -1 -1
0 -1 -1
0 -1 -1
0 -1 -1
0 -1 -1
0 -1 -1
0 -1 -1
0 -1 -1
0 -1 -1
0 -1 -1
0 -1 -1
0 -1 -1
0 -1 -1
0 -1 -1
0 -1 -1
0 -1 -1
0 -1 -1
0 -1 -1
```

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 1 - 세부 기능 1 : 사용자가 원하는 입력을 받아 해당 함수 실행

- 이전: 사용자가 체스게임을 할지 체스퀴즈를 만드는 툴을 실행할지 입력받고 그에 맞는 함수 실행

- 이후: 사용자가 원하는 기능을 입력받으면 해당 기능을 굳이 함수화 하지 않고, main에서 수행

- 사유: 해당 코드가 반복되지 않아 함수를 만들 필요성을 느끼지 못함

3) 기능 2 – 세부기능 4 : 사용자가 원하는 말을 이동하면 그 결과를 시각화 하여 출력

- 이전: 사용자가 원하는 말과 원하는 좌표를 입력하면 원하는 말이 해당 좌표로 이동

- 이후: 사용자가 원하는 말과 원하는 x,y 증가량을 입력하면 해당 말이 원하는 좌표로 이동

- 사유: 코드의 간결성을 위해 원하는 좌표 대신 x,y 증가량을 입력하는 것으로 수정

4) 기능 4 – 세부기능 1 만들어진 체스 퀴즈 저장하기

- 이전: 동적으로 객체를 생성해 사용자가 만든 체스판의 내용을 저장

- 이후: 파일 입출력을 통해 만든 체스판의 내용을 저장

- 사유: 해당 기능을 수행하기에 파일 입출력이 더 적합하다고 판단

5) 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

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		완료				
	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			완료			
중간보고서2 작성					완료		
기능4	세부기능1				완료		
	세부기능2				---------->		
	세부기능3				---------->		
중간보고서3 작성						완료	
기능 보완						---------->	
최종보고서 작성							---------->