프로젝트형 실습
(project/week5/tic_tac_toe.cpp)

223310 주소연

1. 서론
프로젝트 목적 및 배경: 4주차까지 배운 내용에 대한 실습을 위해 진행한다.
목표: Tic Tac Toe 게임 구현이 목표이다.

2. 요구사항
사용자 요구사항: 두 명의 사용자가 번갈아가며 O와 X를 놓기
기능 요구사항: ① 누구의 차례인지 출력 ② 좌표 입력 받기 ③ 입력 받은 좌표 유효성 체크 ④ 좌표에 O / X 놓기 ⑤ 현재 보드판 출력 ⑥ 빙고 시 승자 출력 후 종료 ⑦ 모든 칸이 찼으면 종료

3. 설계 및 구현
기능 별 구현 사항: (요구사항 별 코드)
① 누구의 차례인지 출력

```
int k = 0;          // 누구 차례인지 체크하기 위한 변수
char currentUser; // 현재 유저의 돌을 저장하기 위한 문자 변수

while (true)
{
    // 1. 누구 차례인지 출력
    switch (k % 2)
    {
    case 0:
        cout << "첫번째 유저(X)의 차례입니다 -> ";
        currentUser = 'X';
        break;
    case 1:
        cout << "두번째 유저(O)의 차례입니다 -> ";
        currentUser = 'O';
        break;
    }
```

- 입력: k = 게임 턴 수를 나타내는 변수, currentUser = 현재 유저의 돌 (X 또는 O)
- 결과: k % 2의 결과에 따라 currentUser가 결정한다. k % 2 == 0: 첫 번째 유저(X)의 차례이며, "첫번째 유저(X)의 차례입니다 -> "라는 메시지를 출력한다. k % 2 == 1: 두 번째 유저(O)의 차례이며, "두번째 유저(O)의 차례입니다 -> "라는 메시지를 출력한다.
- 설명: k는 게임의 턴을 나타내며 매 턴마다 값이 증가한다. k % 2는 현재 턴이 짝수인지 홀수인지를 판단하여 짝수면 첫 번째 유저(X), 홀수면 두 번째 유저(O)의 차례가 된다.

② 좌표 입력 받기

```
// 2. 좌표 입력 받기
cout << "(x, y) 좌표를 입력하세요: ";
cin >> x >> y;
```

입력: x = 좌표 x 값 y = 좌표 y 값
결과: 사용자가 키보드로 입력한 값을 x와 y 변수에 저장한다.
설명: cin을 통해 사용자가 입력한 두 개의 좌표를 받아서 변수 x와 y에 저장한다.

③ 입력 받은 좌표 유효성 체크

```
// 3. 입력받는 좌표의 유효성 체크
if (x >= numCell || y >= numCell)
{
    cout << x << ", " << y << ": ";
    cout << "x와 y 둘 중 하나가 칸을 벗어납니다." << endl;
    continue;
}

if (board[x][y] != ' ')
{
    cout << x << ", " << y << ": 이미 돌이 차있습니다" << endl;
    continue;
}
```

입력: x = 좌표 x 값 y = 좌표 y 값 numCell = 가로/세로 칸 개수
결과: 칸을 놓을 수 없는 이유를 출력하고 출력 후 while문 초반으로 이동한다.
설명: 사용자가 입력한 좌표가 게임 판을 벗어나는지 if로 체크하고 사용자가 입력한 좌표에 돌이 이미 있는지 if로 체크한다.

④ 좌표에 O / X 놓기

```
// 4. 입력받는 좌표에 현재 유저의 돌 놓기
board[x][y] = currentUser;
```

입력: x = 좌표 x 값 y = 좌표 y 값 currentUser: 현재 유저의 돌
결과: 입력받은 좌표에 현재 유저 돌을 놓는다
설명: board[x][y]에 currentUser 값을 할당함으로써 선택한 곳에 현재 유저의 돌이 추가된다.

⑤ 현재 보드판 출력

```
// 5. 현재 보드 판 출력
for (int i = 0; i < numCell; i++)
{
    cout << "---|---|---" << endl;
    for (int j = 0; j < numCell; j++)
    {
        cout << " " << board[i][j] << " ";
        if (j == numCell - 1)
        {
            break;
        }
        cout << "|";
    }
    cout << endl;
}
cout << "---|---|---" << endl;
```

입력: numCell = 보드의 크기
결과: 현재 보드판 상태를 출력한다.
설명: 각 행을 출력하기 전에 "---|---|---"를 출력하여 행과 행 사이의 구분을 명확히 하고 board[i][j]를 사용하여 현재 위치에 놓인 돌을 출력한다. 만약 돌이 놓여 있지 않은 경우에는 공백이 출력된다.

⑥ 빙고 시 승자 출력 후 종료

```
// 6. 빙고되는지를 체크 (가로, 세로, 대각선)
bool win = false;
// 가로 체크
for (int i = 0; i < numCell; i++)
{
    if (board[i][0] == currentUser && board[i][1] == currentUser && board[i][2] == currentUser)
    {
        cout << "가로에 모두 돌이 놓였습니다.";
        win = true;
    }
}
// 세로 체크
for (int i = 0; i < numCell; i++)
{
    if (board[0][i] == currentUser && board[1][i] == currentUser && board[2][i] == currentUser)
    {
        cout << "세로에 모두 돌이 놓였습니다.";
        win = true;
    }
}

// 대각선 체크
if (board[0][0] == currentUser && board[1][1] == currentUser && board[2][2] == currentUser)
{
    cout << "왼쪽 위에서 오른쪽 아래 대각선에 모두 돌이 놓였습니다.";
    win = true;
}
if (board[0][2] == currentUser && board[1][1] == currentUser && board[2][0] == currentUser)
{
    cout << "오른쪽 위에서 왼쪽 아래 대각선에 모두 돌이 놓였습니다.";
    win = true;
}

// 승리 시 종료
if (win)
{
    cout << "축하합니다! 유저 " << currentUser << "가 승리했습니다!" << endl;
    break; // 게임 종료
}
```

입력: board, currentUser: 현재 플레이어
결과: 가로, 세로, 대각선 중 한 줄이 모두 같은 유저의 돌로 채워졌는지를 확인하여, 빙고가 이루어지면 승자를 결정하고 게임을 종료한다.
설명: 가로 체크는 동일한 행에 현재 유저의 돌이 모두 놓였는지 확인하고, 발견 시 메시지를 출력하고 세로 체크는 동일한 열에 현재 유저의 돌이 모두 놓였는지 확인하고, 발견 시 메시지를 출력한다. 대각선의 경우 대각선에 유저의 돌이 모두 놓였는지 확인하고 발견 시 대각선에 대한 메시지를 출력한다. 빙고가 확인되면 "축하합니다! 유저 X(O)가 승리했습니다!" 메시지를 출력하고, 게임을 종료한다.

⑦ 모든 칸이 찼으면 종료

```
// 7. 모든 칸이 찼는지 체크
bool isFull = true;
for (int i = 0; i < numCell; i++)
{
    for (int j = 0; j < numCell; j++)
    {
        if (board[i][j] == ' ')
        {
            isFull = false; // 빈칸이 있으면 false
        }
    }
}

// 모든 칸이 찼으면 종료
if (isFull)
{
    cout << "모든 칸이 찼습니다 종료합니다." << endl;
    break; // 게임 종료
}
```

입력: board

결과: 모든 칸이 차 있는지 확인한다 빈칸이 하나라도 있으면 false 빈칸이 없으면 true

설명: 이중 반복문을 사용해 board[i][j]를 순회하며, 빈칸이 있는지 체크한다. if (board[i][j] == ' ') 조건을 통해, 비어 있는 칸을 발견하면 즉시 isFull을 false로 설정하여 게임을 계속 진행한다. isFull이 true라면 게임판의 모든 칸이 채워져 더 이상 돌을 놓을 수 없을 경우이므로 메시지를 출력하고 break를 사용해 while 루프를 종료합니다.

4. 테스트

• (입력에 따라 원하는 결과나 나오는지 확인하는 과정)

- 기능 별 테스트 결과: (요구사항 별 스크린샷)

누구의 차레인지 출력

```
첫번째 유저(X)의 차례입니다 -> (x, y) 좌표를 입력하세요:
두번째 유저(O)의 차례입니다 -> (x, y) 좌표를 입력하세요:
```

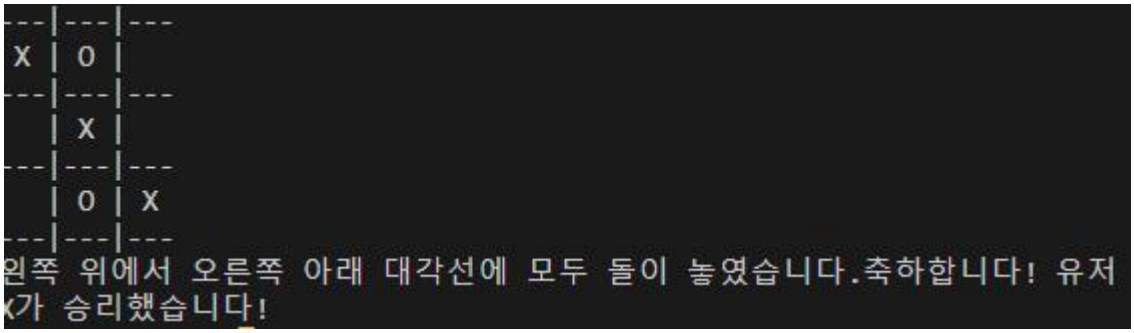
좌표 입력받기

```
두번째 유저(O)의 차례입니다 -> (x, y) 좌표를 입력하세요: 1 2
```

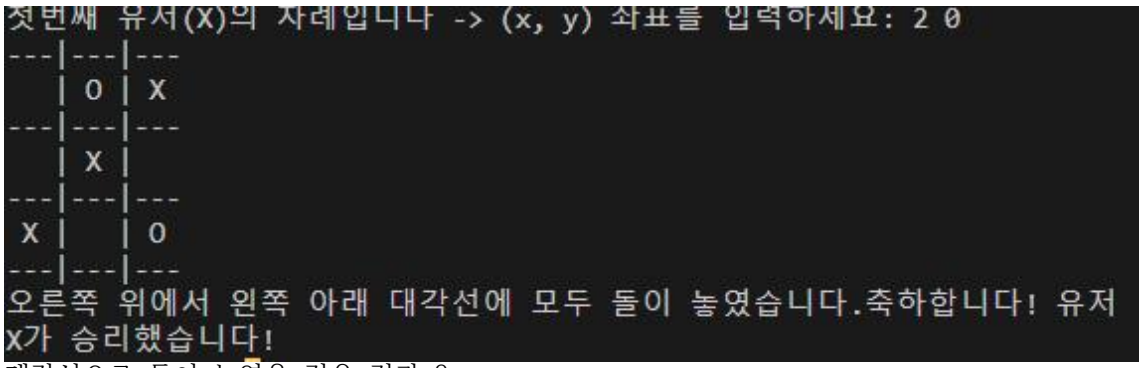
- 최종 테스트 스크린샷: (프로그램 전체 동작 스크린샷)

```
---|---|---
 O | X | O
---|---|---
 X | X | O
---|---|---
 X | O | X
---|---|---
모든 칸이 찼습니다 종료합니다.
```

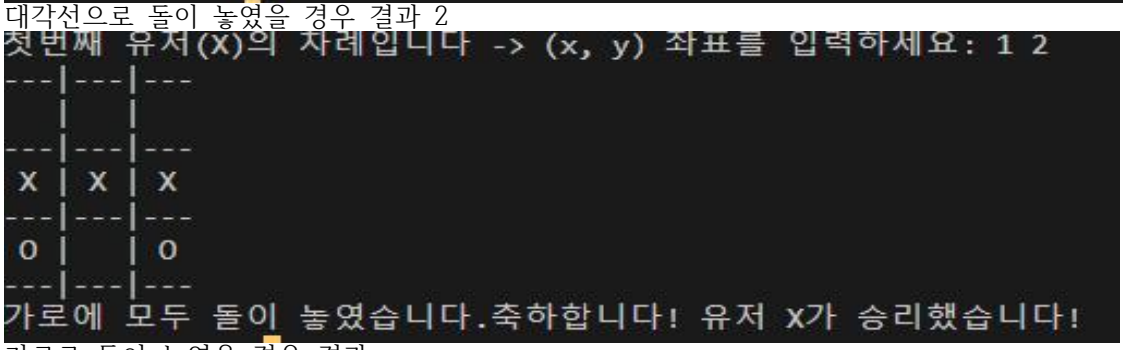
모든 칸이 찼을 경우 종료



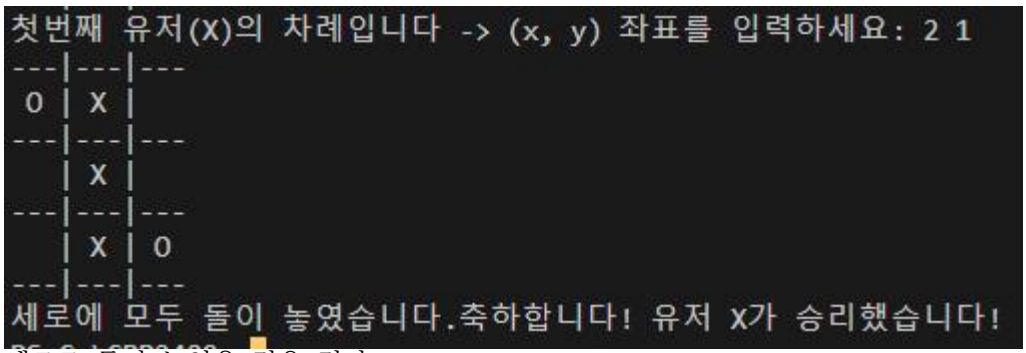
대각선으로 돌이 놓였을 경우 결과 1



대각선으로 돌이 놓였을 경우 결과 2



가로로 돌이 놓였을 경우 결과



세로로 돌이 놓였을 경우 결과

5. 결과 및 결론

프로젝트 결과: Tic Tac Toe 기본 코드에다가 빙고 성공 시 승자 출력 후 종료하는 코드와 모든 칸이 찼으면 종료하는 코드를 더해 Tic Tae Toe 게임을 만들었다.

느낀 점: 수업시간에 하기에는 시간이 부족한 것 같다.