틱택토 보고서

185482 김기태

1. 서론

프로젝트 목적 및 배경 : 4주차까지 배운 내용에 대한 실습을 위해 진행
목표 : Tic Tac Toe 게임 구현

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

기능별 구현사항 :

< 1. 보드판 초기화 >

```
// 보드판 초기화
for (x = 0; x < numCell; x++)
{
    for (y = 0; y < numCell; y++)
    {
        board[x][y] = ' ';
    }
}
// 여기서는 board[x][y]의 값을 공란으로 초기화해주기 위한 이중 for문이다.
```

▣ 입력
• x = 좌표 x 값
• y = 좌표 y 값
• numCell = 가로/세로 칸 개수
▣ 결과
• 모든 보드판(board 2차원 배열 전체)을 공란으로 초기화한다.
▣ 설명
• 이중 for문을 통해서 읽어들인다.

< 2. 초기값 설정 및 유저 차례 지정 >

```
// 초기값 설정
int k = 0;
char currentUser = 'X';
// k값은 회차를 의미한다. 초기값은 0이다.
// currentUser의 기본값은 X이기에 X가 먼저 시작한다.

// While문 무한반복
while (true) // 이 while문은 break로 빠져나오지 않는 이상 무한 반복이다.
{
    // 누구 차례인지 출력
    switch (k % 2)
    {
    case 0:
        cout << k % 2 + 1 << "번 유저(X)의 차례입니다 -> ";
        currentUser = 'X';
        break;
    case 1:
        cout << k % 2 + 1 << "번 유저(0)의 차례입니다 -> ";
        currentUser = '0';
        break;
    }
    // 여기서는 유저의 차례를 서술하고 있다.
    // 유저의 번호는 k값을 2로 나눈 나머지에 1을 더한 값으로 출력되도록 한다.
```

▣ 입력

• k = 차례를 나타내는 정수 값(뒷부분에 k++로 계속 증가한다.)이다.

▣ 결과

• 현재 누구 차례인지 출력한다.

▣ 설명

• while (true)로 인해 무한반복을 수행한다.

• switch (k % 2)를 사용하여 k의 값을 2로 나눈 나머지에 따라 현재의 유저가 몇 번인지 표시한다.

• currentUser 변수는 현재 차례의 유저를 저장하기 위해 사용한다.

< 3. 좌표 입력받기 >

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

▣ 입력

• x = 사용자가 입력하는 x좌표 값

• y = 사용자가 입력하는 y좌표 값

▣ 결과

• 사용자로부터 x, y 좌표 값을 입력받는다.

▣ 설명

• cin으로 사용자로부터 x와 y의 좌표 값을 순서대로 입력받는다.

< 4. 좌표 유효성 체크 >

```
// 좌표 유효성 체크
if (x >= numCell || y >= numCell)
{
    cout << x << ", " << y << ": ";
    cout << " x와 y 둘 중 하나가 칸을 벗어납니다." << endl;
    continue;
}
// numCell은 3으로 지정되어 있고, 좌표값은 0, 1, 2만 존재하기에
// 3을 넘는 값은 해당 문구를 출력하고 게임이 계속된다.
if (board[x][y] != ' ')
{
    cout << x << ", " << y << "이미 돌이 차 있습니다." << endl;
    continue;
}
// 이 부분은 해당 지점에 이미 돌이 차 있을 때 다시 놓을 수 없을 때이다.
// 이는 board[x][y]가 공란이 아님을 확인한 뒤 해당 문구를 출력하고 계속한다.
```

▣ 입력

• x = 사용자가 입력한 x 좌표 값

• y = 사용자가 입력한 y 좌표 값

• numCell = 가로/세로 칸 개수

• board[][] = 게임 보드의 상태를 나타내는 2차원 배열

▣ 결과

• 사용자가 입력한 좌표의 유효성을 체크한 후, 유효하지 않은 경우 해당 이유를 출력한다.

• 유효하지 않은 경우 while문의 초반으로 이동한다.

▣ 설명

• 첫 번째 if문은 사용자가 입력한 좌표가 게임 판을 벗어났는지를 체크한다. 만약 x, y 중 하나라도 numCell 이상인 경우, 해당 좌표는 게임 판 범위 밖에 있음을 사용자에게 공지한다.

• 두 번째 if문은 사용자가 입력한 좌표에 이미 표기가 되어 있는지를 확인하고, 공백이 아닌 경우 그 사실을 사용자에게 공지한다.

• 유효성 체크에서 문제가 발생한 경우 continue 문을 사용하여 while문 도입부로 되돌아간다.

< 5. 보드 판 출력기 >

```
// 입력받은 좌표에 현재 유저의 돌 놓기
board[x][y] = currentUser;


// 보드 판 출력기
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;
// 여기서는 맵을 출력한다. 맵은 현재 상태를 그대로 출력시켜 준다.
// 이중 for문에서는 벽과 구분선을 출력하고 있다.
// board[i][j]은 현재 값이 차 있으면(누군가가 값을 넣었으면) 출력되며
// 공란만 들어있을 경우에는 공란이 그대로 출력된다.
```

▣ 입력

• numCell = 가로/세로 칸 개수

• board[][] = 게임 보드의 상태를 나타내는 2차원 배열

▣ 결과

• 게임 보드의 현재 상태를 화면에 출력한다.

▣ 설명

• 이중 for문을 사용하여 각 행을 순회하며, 각 행의 시작에서는 ---|---|--- 로 테두리를 출력한다.

• 안쪽 for문에는 각 행의 각 열을 순회하며, 각 칸마다 board 배열의 해당 좌표의 값을 출력하고, 마지막 칸이 아닌 경우에는 칸 사이의 구분용 바(|)를 출력한다.

• 마지막 칸을 출력한 후, 열 루프를 벗어나면 endl로 줄바꿈을 실행하여 다음 행으로 넘어간다.

• 모든 행을 출력한 후 마지막으로 ---|---|--- 로 테두리를 또 출력한다.

< 6. 승리 변수 >

```
// 승리 변수
bool isOwin = false;
bool isXwin = false;
```

▣ 입력

• isOwin, isXwin = 승리 상태 표기용 bool변수.

▣ 결과

• 승리 상태를 나타내는 두 개의 불 변수를 false로 초기화한다.

▣ 설명

• 양 변수는 각각 O와 X의 승리를 나타낸다. 초기에는 전부 false로 표기되지만 후에 승리시 true로 바뀐다.

< 7. 세로 승리 판독기 >

```
// 세로 승리 판독기
for (int j = 0; j < numCell; j++)
{
    int colOscore = 0;
    int colXscore = 0;
    for (int i = 0; i < numCell; i++)
    {
        if (board[i][j] == 'O')
        {
            colOscore++;
            if (colOscore == numCell)
            {
                cout << "세로에 모두 돌이 놓였습니다!: ";
                isOwin = true;
            }
        }
        if (board[i][j] == 'X')
        {
            colXscore++;
            if (colXscore == numCell)
            {
                cout << "세로에 모두 돌이 놓였습니다!: ";
                isXwin = true;
            }
        }
    }
```

▣ 입력

• numCell = 가로/세로 칸 개수

• board[][] = 게임 보드의 상태를 나타내는 2차원 배열

▣ 결과

• 보드의 각 세로(열)를 검사하여 승리 조건을 만족하는지 판독하고, 승리 조건을 만족하면 해당 유저의 승리 변수를 true로 설정하고 승리 메시지를 출력한다.

▣ 설명

• 외부 for문은 각 열을 순회한다.

• 각 열의 시작에서 O와 X의 점수를 나타내는 colOscore와 colXscore를 0으로 초기화한다.

• 내부 for문은 해당 열의 각 행을 순회하며, 해당 칸에 O가 놓여 있으면 colOscore를 1 증가시키고, 해당 칸에 X가

놓여 있으면 colXscore를 1 증가시킨다.
• 한쪽 유저의 점수가 numCell과 같아지면, 해당 유저가 해당 열에서 승리한다.

< 8. 가로 승리 판독기 >

```
// 가로 승리 판독기
for (int i = 0; i < numCell; i++)
{
    int rowOscore = 0;
    int rowXscore = 0;
    for (int j = 0; j < numCell; j++)
    {
        if (board[i][j] == 'O')
        {
            rowOscore++;
            if (rowOscore == numCell)
            {
                cout << "가로에 모두 돌이 놓였습니다!: ";
                isOwin = true;
            }
        }
        if (board[i][j] == 'X')
        {
            rowXscore++;
            if (rowXscore == numCell)
            {
                cout << "가로에 모두 돌이 놓였습니다!: ";
                isXwin = true;
            }
        }
    }
```

▣ 입력
• numCell = 가로/세로 칸 개수
• board[][] = 게임 보드의 상태를 나타내는 2차원 배열
▣ 결과
• 보드의 각 가로(행)를 검사하여 승리 조건을 만족하는지 판독하고, 승리 조건을 만족하면 해당 유저의 승리 변수를 true로 설정하고 승리 메시지를 출력한다.
▣ 설명
• 외부 for문은 각 행을 순회한다.
• 각 행의 시작에서 O와 X의 점수를 나타내는 rowOscore와 rowXscore를 0으로 초기화한다.
• 내부 for문은 해당 행의 각 열을 순회하며, 해당 칸에 O가 놓여 있으면 rowOscore를 1 증가시키고, 해당 칸에 X가 놓여 있으면 rowXscore를 1 증가시킨다.
• 각 유저의 점수가 numCell과 같아지면, 해당 유저가 해당 행에서 승리한다.

< 9. 좌측 대각선 승패 판독기 >

```
// 좌측 대각선 승패 판독기
int diaLOscore = 0;
int diaLXscore = 0;
int diaROscore = 0;
int diaRXscore = 0;
for (int i = 0; i < numCell; i++)
{
    if (board[i][i] == 'O')
    {
        diaLOscore++;
        if (diaLOscore == numCell)
        {
            cout << "왼쪽 대각선에 모두 돌이 놓였습니다!: ";
            isOwin = true;
        }
    }
    if (board[i][i] == 'X')
    {
        diaLXscore++;
        if (diaLXscore == numCell)
        {
            cout << "왼쪽 대각선에 모두 돌이 놓였습니다!: ";
            isXwin = true;
        }
    }
}
```

▣ 입력

• numCell = 가로/세로 칸 개수

• board[][] = 게임 보드의 상태를 나타내는 2차원 배열

▣ 결과

• 보드의 왼쪽 대각선을 검사하여 승리 조건을 만족하는지 판독하고, 승리 조건을 만족하면 해당 유저의 승리 변수를 true로 설정하고 승리 메시지를 출력한다.

▣ 설명

• diaLOscore와 diaLXscore 변수는 각각 O와 X의 왼쪽 대각선에 놓인 돌의 수를 나타낸다.

• for문을 사용하여 왼쪽 대각선에 있는 각 칸을 순회하되, 명심해야 할 것은 왼쪽 대각선의 특성상, 행과 열의 인덱스가 동일하다는 것이다. 따라서 board[i][i]를 사용하여 대각선의 각 칸을 확인한다. 해당 칸에 O가 놓여 있으면 diaLOscore를 1 증가시키고, 해당 칸에 X가 놓여 있으면 diaLXscore를 1 증가시킨다.

• 각 유저의 점수가 numCell과 같아지면, 해당 유저가 왼쪽 대각선에서 승리한다.

< 10. 우측 대각선 승패 판독기 >

```
// 우측 대각선 승패 판독기
for (int i = 0; i < numCell; i++)
{
    if (board[i][numCell - 1 - i] == 'O')
    {
        diaROscore++;
        if (diaROscore == numCell)
        {
            cout << "오른쪽 대각선에 모두 돌이 놓였습니다!: ";
            isOwin = true;
        }
    }
    if (board[i][numCell - 1 - i] == 'X')
    {
        diaRXscore++;
        if (diaRXscore == numCell)
        {
            cout << "오른쪽 대각선에 모두 돌이 놓였습니다!: ";
            isXwin = true;
        }
    }
}
```

▣ 입력

• numCell = 가로/세로 칸 개수

• board[][] = 게임 보드의 상태를 나타내는 2차원 배열

▣ 결과

• 보드의 오른쪽 대각선을 검사하여 승리 조건을 만족하는지 판독하고, 승리 조건을 만족하면 해당 유저의 승리 변수를 true로 설정하고 승리 메시지를 출력한다.

▣ 설명

• diaROscore와 diaRXscore 변수는 각각 O와 X의 오른쪽 대각선에 놓인 돌의 수를 나타낸다.

• for문을 사용하여 오른쪽 대각선에 있는 각 칸을 순회하되, 오른쪽 대각선의 특성상, 행의 인덱스는 증가하면서, 열의 인덱스는 감소하기 때문에, board[i][numCell - 1 - i]를 사용하여 대각선의 각 칸을 참조한다. 해당 칸에 O가 놓여 있으면 diaROscore를 1 증가시키고, 해당 칸에 X가 놓여 있으면 diaRXscore를 1 증가시킨다.

• 각 유저의 점수가 numCell과 같아지면, 해당 유저가 오른쪽 대각선에서 승리한다.

< 11. 승리 문구 및 반복문 탈출기 >

```
// 승리 문구 및 반복문 탈출기
if (isOwin == true)
{
    cout << k % 2 + 1 << "번 유저(" << currentUser << ")의 승리입니다!" << endl;
    cout << "종료합니다." << endl;
    break;
}
if (isXwin == true)
{
    cout << k % 2 + 1 << "번 유저(" << currentUser << ")의 승리입니다!" << endl;
    cout << "종료합니다." << endl;
    break;
}
// 이 부분은 승패 시에 반복문 탈출을 위한 부분이다.
// 0건 X건 한쪽이 이기면 break문으로 반복문을 탈출한다.
```

▣ 입력

• isOwin = 'O' 유저의 승리 상태를 나타내는 불 변수

• isXwin = 'X' 유저의 승리 상태를 나타내는 불 변수

• k = 차례를 나타내는 정수 값

• currentUser = 현재 턴의 유저

▣ 결과

• 승리한 유저의 승리 메시지 출력 후 게임이 종료된다.

▣ 설명

• 두 if문은 특정 유저가 승리했는지를 판별합니다. 둘 중 하나의 변수가 true일 경우, 승리한 유저의 번호와 승리 메시지를 출력하고 break 문을 사용하여 반복문을 이탈한다.

< 12. 꽉 찼는지 여부 확인기 >

```
// 꽉 찼는지 여부 확인기
int full = 0;
for (int i = 0; i < numCell; i++)
{
    for (int j = 0; j < numCell; j++)
    {
        if (board[i][j] != ' ')
        {
            full++;
        }
    }
}
if (full == numCell * numCell)
{
    cout << "모든 칸이 다 찼습니다. 종료합니다." << endl;
    break;
}
k++;
// 이 부분은 모든 칸이 찼는지 검증을 위한 이중 for문이다.
// 검증 결과 9칸 모두가 차면 break를 통해 while문을 빠져나온다.
// 이게 작동하면 결과적으로 승자 없이 게임이 종료된다.
// 이 부분을 마지막에 두는 이유는 마지막에도 승자가 결정될 수 있기 때문이다.
// 마지막의 k++는 회차 종료 카운트이다.
```

▣ 입력

• numCell = 가로/세로 칸 개수

• board[][] = 게임 보드의 상태를 나타내는 2차원 배열

▣ 결과

• 보드가 모두 찼는지 확인하고, 모든 칸이 찼다면 메시지를 출력 후 종료한다.

▣ 설명

• full 변수는 게임 보드의 빈 칸이 아닌 위치의 수를 나타낸다.

• 이중 for문을 사용하여 게임 보드의 모든 칸을 순회한다. 해당 칸에 돌이 표기되어 있다면, full 변수의 값을 1 증가시킨다. 모든 칸을 순회한 후, full 변수의 값이 numCell * numCell과 동일한지 확인한다. 이 값이 동일하다는 것은 게임 보드의 모든 칸이 빈 칸이 아니라는 것을 의미하므로, 승패 없이 break 문을 사용하여 게임을 종료한다.

• 맨 마지막 부분에 두어야 마지막에 승부가 나도 강제 종료되지 않는다.

4. 테스트

기능별 테스트 결과 :

① 누구의 차례인지 출력

```
1번 유저(X)의 차례입니다 -> (x, y) 좌표를 입력하세요 :
```

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

② 좌표 입력 받기

```
2번 유저(O)의 차례입니다 -> (x, y) 좌표를 입력하세요 : 0 1
```

③ 입력 받은 좌표 유효성 체크

```
1번 유저(X)의 차례입니다 -> (x, y) 좌표를 입력하세요 : 2 3
2, 3:  x와 y 둘 중 하나가 칸을 벗어납니다.
1번 유저(X)의 차례입니다 -> (x, y) 좌표를 입력하세요 : 0 1
0, 1이미 돌이 차 있습니다.
1번 유저(X)의 차례입니다 -> (x, y) 좌표를 입력하세요 :
```

④ 좌표에 O / X 놓기

```
---|---|---
X  |O  |
---|---|---
   |   |
---|---|---
   |   |
---|---|---
```

⑤ 현재 보드판 출력

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

⑥ 빙고 시 승자 출력 후 종료

```
1번 유저(X)의 차례입니다 -> (x, y) 좌표를 입력하세요 : 2 2
---|---|---
X  |O  |
---|---|---
   |X  |O
---|---|---
   |   |X
---|---|---
왼쪽 대각선에 모두 돌이 놓였습니다!: 1번 유저(X)의 승리입니다!
종료합니다.
```

⑦ 모든 칸이 찼으면 종료

```
---|---|---
X  |0  |X
---|---|---
0  |0  |X
---|---|---
X  |X  |0
---|---|---
모든 칸이 다 찼습니다. 종료합니다.
```

최종 테스트 스크린샷 :

< 꽉 찬 경우(무승부) >

```
1번 유저(X)의 차례입니다 -> (x, y) 좌표를 입력하세요 : 0 1
---|---|---
   |X  |
---|---|---
   |   |
---|---|---
   |   |
---|---|---
2번 유저(O)의 차례입니다 -> (x, y) 좌표를 입력하세요 : 0 0
---|---|---
O  |X  |
---|---|---
   |   |
---|---|---
   |   |
---|---|---
1번 유저(X)의 차례입니다 -> (x, y) 좌표를 입력하세요 : 1 0
---|---|---
O  |X  |
---|---|---
X  |   |
---|---|---
   |   |
---|---|---
2번 유저(O)의 차례입니다 -> (x, y) 좌표를 입력하세요 : 1 1
---|---|---
O  |X  |
---|---|---
X  |O  |
---|---|---
   |   |
---|---|---
1번 유저(X)의 차례입니다 -> (x, y) 좌표를 입력하세요 : 2 0
---|---|---
O  |X  |
---|---|---
X  |O  |
---|---|---
X  |   |
---|---|---
2번 유저(O)의 차례입니다 -> (x, y) 좌표를 입력하세요 : 2 1
---|---|---
O  |X  |
---|---|---
X  |O  |
---|---|---
X  |O  |
---|---|---
1번 유저(X)의 차례입니다 -> (x, y) 좌표를 입력하세요 : 2 2
```

```
---|---|---
O  |X  |
---|---|---
X  |O  |
---|---|---
X  |O  |X
---|---|---
2번 유저(O)의 차례입니다 -> (x, y) 좌표를 입력하세요 : 0 2
---|---|---
O  |X  |O
---|---|---
X  |O  |
---|---|---
X  |O  |X
---|---|---
1번 유저(X)의 차례입니다 -> (x, y) 좌표를 입력하세요 : 1 2
---|---|---
O  |X  |O
---|---|---
X  |O  |X
---|---|---
X  |O  |X
---|---|---
모든 칸이 다 찼습니다. 종료합니다.
```

< 가로 승리 경우 >

```
1번 유저(X)의 차례입니다 -> (x, y) 좌표를 입력하세요 : 0 0
---|---|---
X  |   |
---|---|---
   |   |
---|---|---
   |   |
---|---|---
2번 유저(O)의 차례입니다 -> (x, y) 좌표를 입력하세요 : 1 0
---|---|---
X  |   |
---|---|---
O  |   |
---|---|---
   |   |
---|---|---
1번 유저(X)의 차례입니다 -> (x, y) 좌표를 입력하세요 : 0 1
---|---|---
X  |X  |
---|---|---
O  |   |
---|---|---
   |   |
---|---|---
2번 유저(O)의 차례입니다 -> (x, y) 좌표를 입력하세요 : 2 0
---|---|---
X  |X  |
---|---|---
O  |   |
---|---|---
O  |   |
---|---|---
1번 유저(X)의 차례입니다 -> (x, y) 좌표를 입력하세요 : 0 2
---|---|---
X  |X  |X
---|---|---
O  |   |
---|---|---
O  |   |
---|---|---
가로에 모두 돌이 놓였습니다!: 1번 유저(X)의 승리입니다!
종료합니다.
```

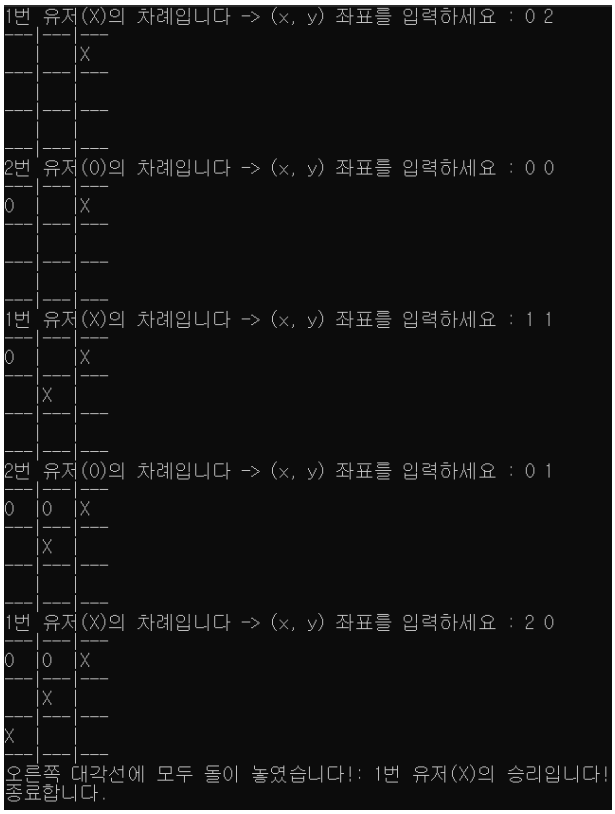
< 세로 승리 경우 >

```
1번 유저(X)의 차례입니다 -> (x, y) 좌표를 입력하세요 : 0 0
---|---|---
X  |   |
---|---|---
   |   |
---|---|---
   |   |
---|---|---
2번 유저(O)의 차례입니다 -> (x, y) 좌표를 입력하세요 : 0 1
---|---|---
X  |O  |
---|---|---
   |   |
---|---|---
   |   |
---|---|---
1번 유저(X)의 차례입니다 -> (x, y) 좌표를 입력하세요 : 1 0
---|---|---
X  |O  |
---|---|---
X  |   |
---|---|---
   |   |
---|---|---
2번 유저(O)의 차례입니다 -> (x, y) 좌표를 입력하세요 : 0 2
---|---|---
X  |O  |O
---|---|---
X  |   |
---|---|---
   |   |
---|---|---
1번 유저(X)의 차례입니다 -> (x, y) 좌표를 입력하세요 : 2 0
---|---|---
X  |O  |O
---|---|---
X  |   |
---|---|---
X  |   |
---|---|---
세로에 모두 돌이 놓였습니다!: 1번 유저(X)의 승리입니다!
종료합니다.
```

< 좌대각 승리 경우 >

```
1번 유저(X)의 차례입니다 -> (x, y) 좌표를 입력하세요 : 0 0
---|---|---
X  |   |
---|---|---
   |   |
---|---|---
   |   |
---|---|---
2번 유저(O)의 차례입니다 -> (x, y) 좌표를 입력하세요 : 0 1
---|---|---
X  |O  |
---|---|---
   |   |
---|---|---
   |   |
---|---|---
1번 유저(X)의 차례입니다 -> (x, y) 좌표를 입력하세요 : 1 1
---|---|---
X  |O  |
---|---|---
   |X  |
---|---|---
   |   |
---|---|---
2번 유저(O)의 차례입니다 -> (x, y) 좌표를 입력하세요 : 0 2
---|---|---
X  |O  |O
---|---|---
   |X  |
---|---|---
   |   |
---|---|---
1번 유저(X)의 차례입니다 -> (x, y) 좌표를 입력하세요 : 2 2
---|---|---
X  |O  |O
---|---|---
   |X  |
---|---|---
   |   |X
---|---|---
왼쪽 대각선에 모두 돌이 놓였습니다!: 1번 유저(X)의 승리입니다!
종료합니다.
```

< 우대각 승리 경우 >



5. 결과 및 결론

프로젝트 결과 : 시행착오가 많았는데 어떻게든 작동된다.

느낀 점 : 보드판이 커지는 경우에도 어떻게든 작동되는 구조를 생각하고 짜다 보니까 코딩을 잘 모르기에 시간이 꽤 걸렸다. 수업 때 만들다 만 코드 잔해들을 회수해서 어떻게든 만들어 보았는데 제대로 된 건지는 모르겠어도 아무튼 작동하니 다행이다.