214456 박진관

1. 서론

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

2. 요구사항

1) 사용자 요구사항: 두 명의 사용자가 번갈아가며 O와 X를 놓기
2) 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

① 누구의 차례인지 출력

```
// 차례 결정
int k = 0;
char currentUser = 'X';
while(true) {
    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;
    }
```

1. 입력

- k = 진행된 턴을 기록, 턴을 2로 나눈 나머지로 플레이어 차례 결정
- currentUser = 유저가 두는 말을 저장.

2. 결과 : 정상적으로 첫번째 유저부터 턴이 시작됨 .턴이 교체되면서 플레이어의 차례도 교체되는 것을 확인.

3. 설명 : k%2을 스위치 조건문으로 이용하여 플레이어 턴을 결정.
    플레이어가 교체될 때 마다 currentUser에 저장된 문자도 교체됨.

214456 박진관

② 좌표 입력 받기

```
// 좌표 입력 이 때 y와 x를 바뀌어서 x,y 좌표 바뀌는 것을 수정
cout << "(x,y) 좌표를 입력하세요: " ;
cin >> y >> x;
```

1. 입력

- x,y = 두는 말의 좌표를 결정.

2. 결과 : x, y 좌표를 받음.

3. 설명 : x, y를 바꿔서 y를 먼저 입력받아 보드판에 반대로 y,x로 설치되는 것을 방지.

③ 입력 받은 좌표 유효성 체크

```
// 좌표 유효성 체크
if (x >= numCell || y>= numCell) {
    cout<< y << ', ' << x << ": " ;
    cout<< "x 와 y 둘 중 하나가 칸을 벗어났습니다." << endl;
    continue;
}
if (board[x][y] != ' ' ) {
    cout << y << ' ' << x << ": 이미 돌이 놓아져있습니다." << endl;
    continue;
}
```

1. 입력

- numCell : 3으로 상수로 설정. 보드판의 길이 + 높이를 표시.

2. 결과: x,y를 보드판 범위 밖으로 설정했더니 칸을 벗어났다는 메세지 출력 후 게임 지속.
보드판 위에 이미 말이 올려져있는 곳을 설정했더니 이미 돌이 놓아져있다는 메세지 출력 후 게임 지속됨.

3. 설명: 조건문으로 보드판 범위 numCell의 수 보다 큰 값을 입력하면 칸을 벗어났다는 메세지 출력 함.
x,y를 입력했을 때 그 배열의 내용이 빈 칸이 아니면 이미 돌이 놓아져 있다고 출력.
②번과 마찬가지로 x,y가 바뀐 상태이므로 여기서도 위치를 바꿈.

214456 박진관

④ 좌표에 O / X 놓기, ⑤ 현재 보드판 출력

```
//보드판 만들기. 반복문이 굳이 2개나 반복될 필요가 없어 수정
for (int i = 0; i < numCell; i++) {
    cout << " " << board[i][0] << " | " << board[i][1] << " | " << board[i][2] << endl;
    if (i < numCell - 1) {
        cout << "---|---|---" << endl;
    }
}
```

1. 결과 : 보드판이 정상적으로 완성됨. x,y좌표 값을 입력하면 그에 맞는 좌표에 말을 놓음.

2. 설명 : 한 줄에 board[i][0] | board[i][1] | board[i][2]를 출력하고 그 밑줄에
---|---|---를 출력. ---|---|---는 맨 아래까지 그을 필요는 없으므로 조건문으로 2번만 출력하게 만듬.
이 과정을 반복문으로 반복하면 3x3 보드판이 만들어짐
이때 board[i][0] 출력은 좌표 유효성 체크 후 board에 저장되도록 설정해놔서 자신이 설정한 좌표를 제외하면 다 빈칸이 출력됨.

⑥ 빙고 시 승자 출력 후 종료

```
//승리 코드
bool isWin = false;
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!";
        isWin = true;
    }
    if ( board[0][i] == currentUser && board[1][i] == currentUser && board[2][i] == currentUser) {
        cout<< "세로에 모두 돌이 놓였습니다!";
        isWin = true;
    }
}
if ( board[0][0] == currentUser && board[1][1] == currentUser && board[2][2] == currentUser) {
        cout<< "왼쪽 위에서 오른쪽 아래 대각선으로 모두 돌이 놓였습니다!";
        isWin = true;
    }
if ( board[0][2] == currentUser && board[1][1] == currentUser && board[2][0] == currentUser) {
        cout<< "오른쪽 위에서 왼쪽 아래 대각선으로 모두 돌이 놓였습니다!";
        isWin = true;
    }

if (isWin == true){
    cout << k % 2 + 1 <<"번 유저( " << currentUser << ")의 승리입니다!" <<endl;
    cout << "종료합니다!" << endl;
    break;
}
```

1. 결과 : 모든 경우가 정상적으로 출력되고 종료되는것을 확인.
2. 설명 : bool isWin으로 게임의 승리를 체크하는 변수를 만듬. 가로는 board[i][0,1,2] 세로는 board[0,1,2][i]가 같은 문자면 승리하도록 설정. 대각선은 특이한 경우이므로 위치를 직접 지명하여 조건문을 세움.

⑦ 모든 칸이 찼으면 종료

```
// 칸이 다 찼는지 체크
int checked = 0;
for (int i = 0; i <numCell; i++) {
    for (int j = 0 ; j <numCell; j++){
        if(board[i][j] == ' '){
            checked++;
        }
    }
}
if (checked== 0) {
    cout << "모든 칸이 다 찼습니다. 종료합니다" <<endl;
    break;
}
```

1. 결과: 모든 좌표에 말을 놓으면 출력이 정상적으로 되고 종료되는 것을 확인.

2. 설명: 매 턴마다 board 배열 내용을 확인. 빈 칸이 하나라도 있으면 checked가 값이 하나라도 올라서 종료가 안되나, 빈칸이 없을 시 checked=0 이 되므로 아래 조건문이 정상적으로 실행됨.

4. 테스트

① 누구의 차례(O)인지 출력, ② 좌표 입력 받기

```
-Out-dmcputoj.0q3' '--stderr=Microsoft-MIEngine-Error-an2or
첫번째 유저(x)의 차례입니다. -> (x,y) 좌표를 입력하세요:
```

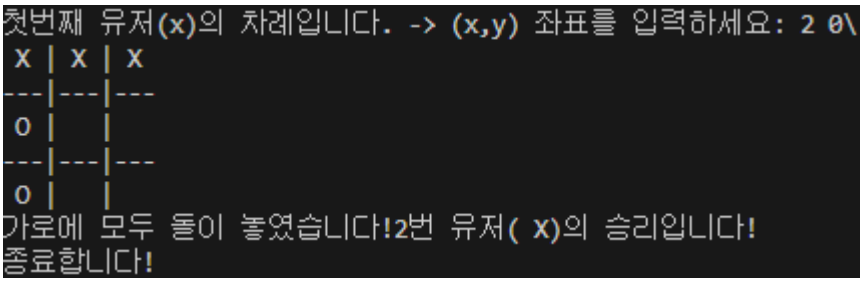
③ 입력 받은 좌표 유효성 체크

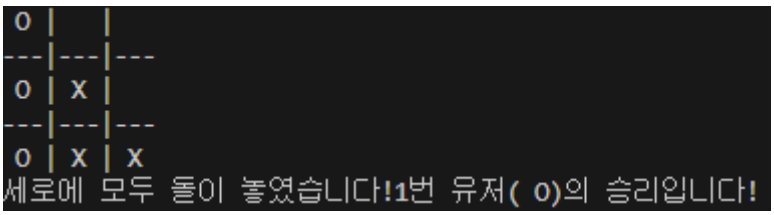
```
첫번째 유저(x)의 차례입니다. -> (x,y) 좌표를 입력하세요: 0 0
 X |   |
---|---|---
   |   |
---|---|---
   |   |
두번째 유저(o)의 차례입니다. -> (x,y) 좌표를 입력하세요: 0 0
0 0: 이미 돌이 놓아져있습니다.
두번째 유저(o)의 차례입니다. -> (x,y) 좌표를 입력하세요:
```

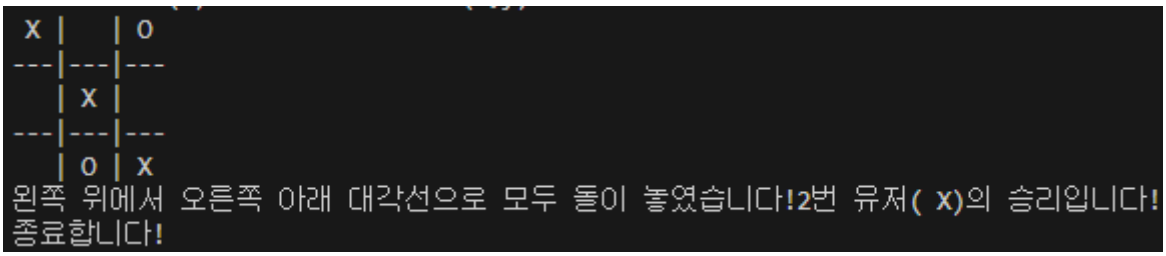
① 누구의 차례(O)인지 출력, ④ 좌표에 O / X 놓기 ⑤ 현재 보드판 출력

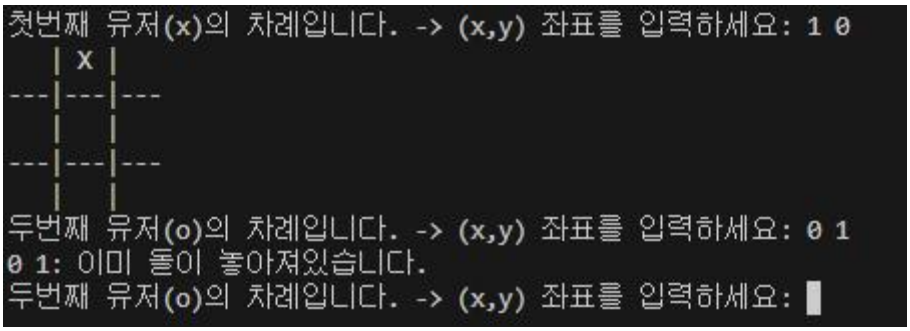
```
두번째 유저(o)의 차례입니다. -> (x,y) 좌표를 입력하세요: 0 1
 X |   |
---|---|---
 O |   |
---|---|---
   |   |
첫번째 유저(x)의 차례입니다. -> (x,y) 좌표를 입력하세요:
```

⑥ 빙고 시 승자 출력 후 종료

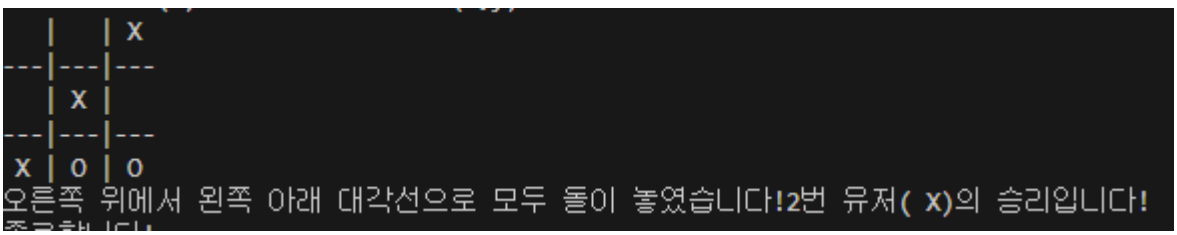






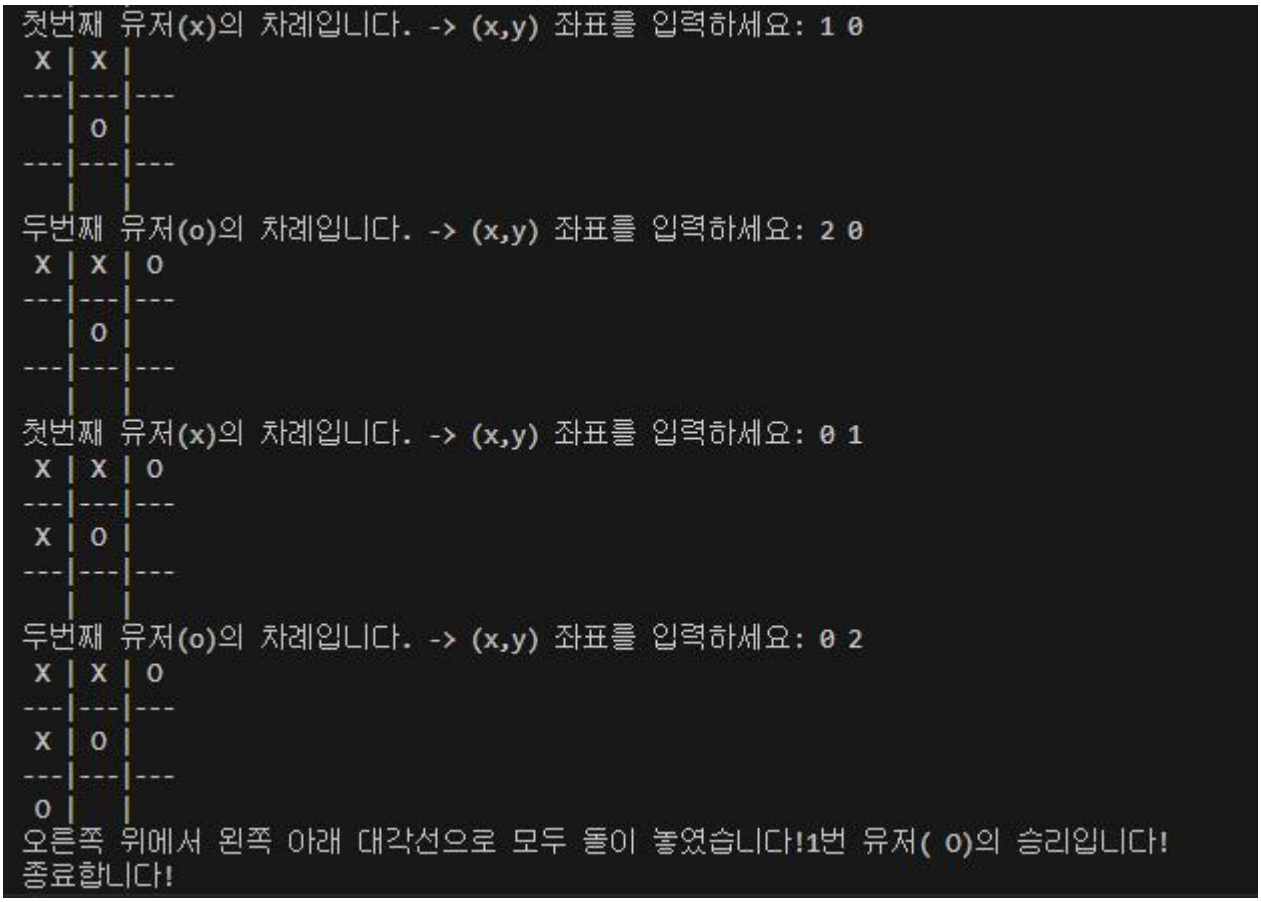


⑦ 모든 칸이 찼으면 종료



214456 박진관

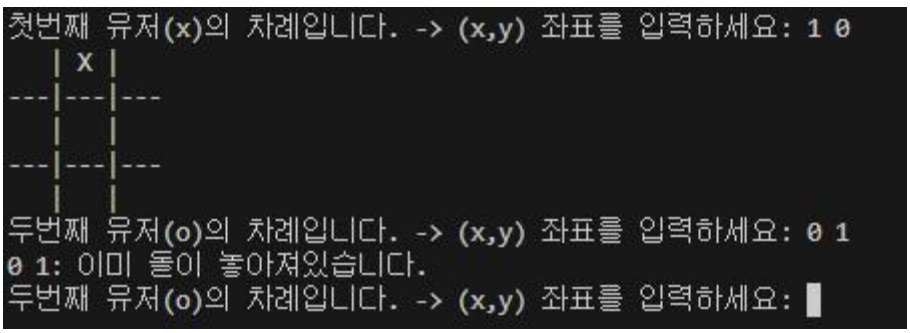
프로그램 전체 동작



5. 결과 및 결론:

1) 결과: Tic Tac Toe 게임을 만들었음

2. 느낀 점: 예제에서는 x,y를 입력하면 y,x로 적용되길래 똑바로 적용되게 코드를 수정했다, 근데 수정했더니 코드가 계속 꼬였다.



1,0을 입력했더니 board에 0,1에 들어가거나 계속 x,y가 바뀌는 오류가 계속 나왔었다. 이것 때문에 너무 골치가 너무 아파서 수강시간 후 계속 코드를 수정하는 단계를 거쳤다.