

## 1. 서론

- 1) 프로젝트 목적 및 배경: 4주차까지 배운 내용 (입력, 출력, 데이터 타입, 배열 등)에 대한 실습을 위해 진행
- 2) 목표: Tic Tac Toe 게임 구현 (3 x 3 의 빙고판에서 한 행, 열, 대각선을 같은 모양으로 채우는 게임)

## 2. 요구사항

### 1) 기능 요구 사항

- 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

### 2) 사용자 요구 사항 : 두 명의 사용자가 번갈아가며 O와 X를 놓기

### 3. 설계 및 구현

#### 1. 기능 별 구현 사항

##### ① 누구의 차례인지 출력

```
int x, y;  
char winner = ' '; //승리한 유저 변수
```

```
int k = 0;  
char currentUser = 'X';  
//유저 차례 출력  
while(winner == ' '){ //승자가 생길 때 while문 탈출  
    switch(k % 2){ //몇 라운드인지에 따라 차례 바꾸기  
        case 0:{ //한 차례마다 k가 1씩 증가 -> 나머지가 0, 1 반복됨  
            cout << "첫 번째 유저(X)의 차례입니다. -> ";  
            currentUser = 'X';  
            break;  
        }  
        case 1:{  
            cout << "두 번째 유저(O)의 차례입니다. -> ";  
            currentUser = 'O';  
            break;  
        }  
    }  
}
```

```
k++;
```

승자가 생길 때 까지(winner 변수에 'O' 또는 'X'가 대입될 때까지)반복하는 while문 작성

While 블록의 한 루프마다 증가하는 정수형 변수 k 선언 -> 각 루프마다 k % 2는 0과 1을 반복하므로 플레이어 X를 선공으로 두고 0일 때 X, 1일 때 O의 차례로 설정

현재 차례인 유저를 문자형 변수 currentUser에 저장

##### ② 좌표 입력 받기

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

Cin 명령어를 통해 x, y좌표 입력받기

### ③ 입력 받은 좌표 유효성 체크

```
const int numCell = 3;  
char board[numCell][numCell]{};
```

```
//보드 초기화  
for (x = 0; x < numCell; x++){  
    for (y = 0; y < numCell; y++){  
        board[x][y] = ' ';    }  
}
```

```
//좌표 입력  
cout << "(x, y) 좌표를 입력하세요 : ";  
cin >> x >> y;  
//좌표 유효성 검사  
if (x >= numCell || y >= numCell){  
    cout << x << ", " << y << ": ";  
    cout << "x와 y 둘 중 하나가 칸을 벗어났습니다." << endl;  
    continue;  
}  
if (board[x][y] != ' '){  
    cout << "(" << x << ", " << y << ")에 이미 돌이 차있습니다." << endl;  
    continue;  
}
```

보드판의 크기를 의미하는 상수 numCell 선언 (틱택토 게임은 크기가 3)

보드판을 의미하는 numCell 크기의 2차원 배열 변수 board[numCell][numCell] 선언  
각 원소를 ' '로 초기화 (해당 칸은비어있음을 의미)

x와 y좌표가 각각 판의 크기인 numCell을 넘으면 칸을 벗어났다는 문구 출력

(x, y)좌표가 비어있음을 의미하는 문자 ' '가 아니라면 이미 돌이 놓여있음을 의미  
하므로 돌이 이미 차있다는 문구 출력

두 검사에 걸리면 while문 처음으로 돌아가는 continue 명령어 작성

#### ④ 좌표에 O / X 놓기

```
board[x][y] = currentUser;
```

유효성 검사를 통과하였으므로 좌표 위치를 의미하는 board[x][y]에 currentUser 문자  
대입 (O 또는 X)

#### ⑤ 현재 보드판 출력

```
//현재 보드 판 출력
for (int i = 0; i < numCell ; i++){ //몇 행 출력하는지 (3번 반복)
    cout << "---|---|---" << endl;
    for (int j = 0; j < numCell; j++){ //몇 열 출력하는지 (3번 반복)
        cout << " ";
        cout << board[i][j];
        if (j == numCell - 1){ //마지막 열이라면 오른쪽 벽 출력 x
            break;
        }
        cout << " |";
    }
    cout << endl;
}
cout << "---|---|---" << endl;
k++;
```

보드판의 각 칸을 지칭하기 위해 중첩 for문을 사용하여 board[0][0] ~ board[2][2]  
를 출력

칸을 구분하기 위해 각 j반복마다 벽 |를 출력

이후 한 차례가 끝났으므로 차례를 구분하는 k에 1을 더함

## ⑥ 빙고 시 승자 출력 후 종료

한 열과 행에 3(numCell)개가 놓여있음을 먼저 검사

```
//빙고 검사
if (k >= (2 * numCell) - 1){ //한 유저가 둘 3개 이상을 놓았을 시 빙고 검사
    //행,열 승리 검사
    for (int i = 0; i < numCell; i++){
        int rCountO = 0; //한 행에 같은 모양 개수 검사
        int rCountX = 0;
        int cCountO = 0; //한 열에 같은 모양 개수 검사
        int cCountX = 0;
```

효율을 위해 한 유저가 numCell개 만큼 놓았음을 검사((2 \* numCell - 1)과 같음)한 후 한 행과 열에 같은 모양의 개수를 담는 정수형 변수 rCountO, rCountX, cCountO, cCountX 선언

```
for (int j = 0; j < numCell; j++){
    if (board[i][j] == 'X') //행 카운트를 위한 if문
        rCountX++;
    else if (board[i][j] == 'O')
        rCountO++;
```

i와 j를 통한 중첩 반복문을 통해 board[i][j]로 board[0][0] ~ board[0][2]를 확인하여 X가있으면 rCountX 1증가, O가 있으면 rCountO 1증가

```
    if (board[j][i] == 'X') //열 카운트를 위한 if문
        cCountX++;
    else if (board[j][i] == 'O')
        cCountO++;
}
```

마찬가지로 동시에 열 검사를 위해 board[j][i]로 board[0][0] ~ board[2][0]으로 확인하여 X가 있으면 cCountX 1증가, O가 있으면 cCountO 1증가

```

//한 행과 열에 o,x가 3개일 경우 검사
if (rCountO == numCell){ //한 행에 o가 3개일 경우
    winner = 'O';
    cout << "가로로 3개가 놓였습니다. " << winner << "의 승리입니다."<< endl;
    break;
}
else if (rCountX == numCell){ //한 행에 x가 3개일 경우
    winner = 'X';
    cout << "가로로 3개가 놓였습니다. " << winner << "의 승리입니다."<< endl;
    break;
}
else if (cCountO == numCell){ //한 열에 o가 3개일 경우
    winner = 'O';
    cout << "세로로 3개가 놓였습니다. " << winner << "의 승리입니다."<< endl;
    break;
}
else if (cCountX == numCell){ //한 열에 x가 3개일 경우
    winner = 'X';
    cout << "세로로 3개가 놓였습니다. " << winner << "의 승리입니다."<< endl;
    break;
}
}

```

rCountX, rCountO, cCountX, cCountO가 판의 크기인 numCell과 같다면 한 열, 행이 같은 모양으로 놓여져있다는 것을 의미하므로 승자를 나타내는 문자형 변수 winner에 'X' 또는 'O' 대입. 어떻게 이겼는지 문구 작성 후 for문 break (while문 종료)

```

else { //둘 다 아니면 count변수 초기화 후 다음 행, 열로 이동
    rCountO = 0;
    rCountX = 0;
    cCountO = 0;
    cCountX = 0;
}
}

```

카운트 변수 4개 모두 0이 아니라면 카운트 변수 모두 초기화 후 다음 i반복분 루프로 이동

```
//대각 승리 검사
int countX = 0; //대각 x카운트
int countO = 0; //대각 o카운트
```

대각선으로 승리했는지 검사 시작

대각으로 X, O로 찾는지 카운트하는 정수형 변수 countX, countO 선언

```
//왼 -> 오 대각 빙고 검사
for (int i = 0; i < numCell; i++){
    if (board[i][i] == 'X')
        countX++;
    else if (board[i][i] == 'O')
        countO++;
}
```

왼쪽위 -> 오른쪽아래 대각으로 3개가 놓였는지 확인 시작

해당 대각선은 [0][0], [1][1]과 같이 board[i][i]에 O 또는 X가 있는지 확인 후 있다면 각각의 카운트 변수에 1을 더함

```
if (countX == numCell || countO == numCell){
    if (countX == numCell) //대각에 x가 3개면 x승리
        winner = 'X';
    else if (countO == 3) //대각에 o가 3개면 y승리
        winner = 'O';
    cout << "왼쪽에서 오른쪽 아래 대각선으로 " << winner << "들이 놓였습니다. " << winner << "유저의 승리입니다." << endl;
}
```

카운트 변수 둘 중 하나가 numCell만큼, 즉 같은 모양이 대각선에 채워져있다면 winner에 O또는 X를 대입하고 대각선으로 놓였다는 문구 출력 (while문 종료)

```
//오 -> 왼 대각 빙고 검사
countX = 0;
countO = 0;
for (int i = 0; i < numCell; i++){
    if (board[i][numCell - i - 1] == 'X')
        countX++;
    else if (board[i][numCell - i - 1] == 'O')
        countO++;
}
```

오른쪽 위 -> 왼쪽 아래 대각으로 3개가 놓였는지 확인 시작

카운트 변수 초기화. 해당 대각선은 [0][2], [1][1], [2][0]과 같이 board[i][numCell - i - 1]에 O, X가 있는지 확인 후 있다면 각각의 카운트 변수에 1을 더함



```

if (countX == numCell || countO == numCell){
    if (countX == numCell) //대각에 x가 3개면 x승리
        winner = 'X';
    else if (countO == numCell) //대각에 o가 3개면 y승리
        winner = 'Y';
    cout << "오른쪽에서 왼쪽 아래 대각선으로 " << winner << "들이 놓였습니다. " << winner << "유자의 승리입니다." << endl;
}

```

이 전에 했던 조건과 같이 countX, countO 변수가 numCell과 같아지면 대각선에 같은 모양이 채워져있다는 것을 의미하므로 winner에 O, X 대입 후 대각선으로 놓였다는 문구 출력(while문 종료)

## ⑦ 모든 칸이 찼으면 종료

```

if (k == numCell * numCell){
    cout << "판이 꽉 찼으나 승자가 없습니다. 무승부입니다." << endl;
    winner = 'A';
}

```

플레이어의 차례를 나타내는 k가 판의 칸 개수만큼 증가하면 판이 꽉 찼음을 의미하므로 k가 numCell \* numCell과 같다면 winner에 아무런 문자를 대입하며 while 문 종료 후 프로그램 종료



## 4. 테스트

### 1. 기능 별 테스트 결과

- ① 누구의 차례인지 출력
- ② 좌표 입력 받기

첫 번째 유저(x)의 차례입니다. -> (x, y) 좌표를 입력하세요 : 1 1

- ③ 입력 받은 좌표 유효성 체크

두 번째 유저(o)의 차례입니다. -> (x, y) 좌표를 입력하세요 : 1 1  
(1, 1)에 이미 돌이 차있습니다.  
두 번째 유저(o)의 차례입니다. -> (x, y) 좌표를 입력하세요 : 1 4  
1, 4: x와 y 둘 중 하나가 칸을 벗어났습니다.

- ④ 좌표에 O / X 놓기
- ⑤ 현재 보드판 출력

---	---	---
---	---	---
---	X	---
---	---	---
---	---	---

- ⑥ 빙고 시 승자 출력 후 종료

---	---	---
X	O	---
---	---	---
X	X	X
---	---	---
O	---	O
---	---	---

가로로 3개가 놓였습니다. x의 승리입니다.

- ⑦ 모든 칸이 찼으면 종료

---	---	---
X	O	X
---	---	---
O	O	X
---	---	---
X	X	O
---	---	---

판이 꽉 찼으나 승자가 없습니다. 무승부입니다.

## 2. 최종 테스트 스크린샷

```
첫 번째 유저(x)의 차례입니다. -> (x, y) 좌표를 입력하세요 : 0 1
---|---|---
   | x |   |
---|---|---
   |   |   |
---|---|---
   |   |   |
---|---|---
두 번째 유저(o)의 차례입니다. -> (x, y) 좌표를 입력하세요 : 0 0
---|---|---
 0 | x |   |
---|---|---
   |   |   |
---|---|---
   |   |   |
---|---|---
첫 번째 유저(x)의 차례입니다. -> (x, y) 좌표를 입력하세요 : 1 1
---|---|---
 0 | x |   |
---|---|---
   | x |   |
---|---|---
   |   |   |
---|---|---
두 번째 유저(o)의 차례입니다. -> (x, y) 좌표를 입력하세요 : 1 2
---|---|---
 0 | x |   |
---|---|---
   | x | 0 |
---|---|---
   |   |   |
---|---|---
첫 번째 유저(x)의 차례입니다. -> (x, y) 좌표를 입력하세요 : 2 1
---|---|---
 0 | x |   |
---|---|---
   | x | 0 |
---|---|---
   | x |   |
---|---|---
세로로 3개가 놓였습니다. x의 승리입니다.
```

## 5. 결과 및 결론

**1. 프로젝트 결과 :** 반복분, 조건문, 2차원 배열 등을 이용하여 Tic Tac Toe 게임을 만들었음.

**2. 느낀 점 :** 기본적인 구조가 제공되어 있기에 프로그램을 쉽게 짤 수 있었지, 아예 썬 판으로 코드를 짰으면 상당히 오랜시간이 걸렸으며 비효율적이었을 것 같음. 지금까지 배운 c++ 명령어를 사용해 직접 알고리즘을 짜보니 생각보다 재밌고 유용한 프로젝트였다고 생각함. 시험도 이렇게 프로그램 짜는 느낌으로 나올 것 같아 알고리즘에 대해 더욱 공부해야겠다고 느낌.