

# Tic Tac Toe 보고서

215118 독일지역학과 이규민

## 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
  - 보드판은 2차원 배열을 사용한다
  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

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

while (!gamewon && !boardFull) { // 빙고가 완성되거나, 모든 칸이 차지 않는 이상 계속 게임 코드 반복
    // 1. 누구 차례인지 출력
    switch (k % 2) {
        case 0:
            cout << k % 2 + 1 << "번 유저(X)의 차례입니다 -> "; // 홀수번 user = X
            currentUser = 'X';
            break;
        case 1:
            cout << k % 2 + 1 << "번 유저(O)의 차례입니다 -> "; // 짝수번 user = O
            currentUser = 'O';
    }
}
```

```
break;
}
```

- 입력
  - `k`: 현재 차례를 나타내는 변수.
  - `currentUser`: 현재 차례의 유저를 나타내는 변수.
- 결과
  - 현재 차례인 유저 번호와 돌(X 또는 O)를 출력합니다.
- 설명
  - `k`를 통해 현재 차례가 홀수 번호인지 짝수 번호인지를 판별하고, 그에 따라 'X' 또는 'O'를 출력하여 누구의 차례인지 알려줍니다. 짝수 번호는 'O'의 차례를 나타냅니다.

## 2. 좌표 입력 받기

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

- 입력
  - 사용자로부터 x와 y 좌표를 입력받습니다.
- 결과
  - 사용자에게 "(x, y) 좌표를 입력하세요: "라는 메시지를 출력하고, 사용자가 입력한 x와 y 좌표를 변수 `x`와 `y`에 저장합니다.
- 설명
  - 사용자에게 "(x, y) 좌표를 입력하세요: "라는 메시지를 출력하고, 사용자로부터 x와 y 좌표를 입력받습니다. 이때 입력받은 값을 변수 `x`와 `y`에 저장합니다.

## 3. 입력 받은 좌표 유효성 체크

```
// 3. 입력받은 좌표의 유효성 체크
if (x >= numCell || y >= numCell) { // x혹은 y좌표가 3*3 배열보다 큰 값일 경우 유효성 검사
    cout << x << ", " << y << ": ";
    cout << " x와 y 둘 중 하나가 칸을 벗어납니다." << endl;
    continue;
}
if (board[x][y] != ' ') { // 지정한 x, y 좌표가 ' ' 빈 칸이 아닌 경우 (이미 돌이 차있는 경우) 유효성 검사
    cout << x << ", " << y << ": 이미 돌이 차있습니다.";
    continue;
}
```

- 입력

- **x**: 사용자가 입력한 x 좌표.
- **y**: 사용자가 입력한 y 좌표.
- **numCell**: 가로/세로 칸 개수.
- **board**: 현재 게임 보드 상태를 나타내는 배열.
- 결과
  - 입력받은 좌표의 유효성을 체크하고, 유효하지 않을 경우 해당 이유를 출력합니다.
- 설명
  - 먼저, 입력된 x 또는 y 좌표가 게임 보드의 범위를 벗어났는지 검사합니다. 벗어났다면 해당 이유를 출력합니다.
  - 그 다음, 입력된 좌표에 이미 돌이 놓여있는지 확인합니다. 이미 돌이 있는 경우 해당 이유를 출력합니다.

#### 4. 좌표에 O / X 놓기

```
while (!gameWon && !boardFull) { // 빙고가 완성되거나, 모든 칸이 차지 않는 이상 계속 게임 코드 반복
    // 1. 누구 차례인지 출력
    switch (k % 2) {
    case 0:
        cout << k % 2 + 1 << "번 유저(X)의 차례입니다 -> "; // 홀수번 user = X
        currentUser = 'X';
        break;
    case 1:
        cout << k % 2 + 1 << "번 유저(O)의 차례입니다 -> "; // 짝수번 user = O
        currentUser = 'O';
        break;
    }

    // 4. 입력받은 좌표에 현재 유저의 돌 놓기
    board[x][y] = currentUser; // X 혹은 O 돌 놓기(위의 switch 문에 따라 결정됨)
}
```

- 입력
  - **x**: 사용자가 입력한 x 좌표.
  - **y**: 사용자가 입력한 y 좌표.
  - **currentUser**: 현재 차례의 유저 (X 또는 O).
- 결과
  - 입력받은 좌표에 현재 유저의 돌(X 또는 O)을 놓습니다.
- 설명
  - 현재 유저의 돌을 입력받은 좌표에 놓습니다. 이때, 현재 유저는 'X' 또는 'O'로 결정됩니다.

#### 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**: 가로/세로 칸 개수.
- **board**: 현재 게임 보드 상태를 나타내는 배열.

- 결과

- 현재의 게임 보드 상태를 출력합니다.

- 설명

- 현재 게임 보드의 상태를 가로, 세로, 대각선 방향으로 출력하여 사용자에게 보여줍니다.

## 6. 빙고 시 승자 출력 후 종료

```
bool gameWon = false; // 빙고 여부 체크 (초기상태는 false 로 선언)

// 6. 빙고 확인 (가로, 세로, 대각선)
// 가로 빙고 확인
for (int i = 0; i < numCell; i++) {
    if (board[i][0] == currentUser && board[i][1] == currentUser &&
        board[i][2] == currentUser) {
        gameWon = true;
        break;
    }
}

// 세로 빙고 확인
for (int i = 0; i < numCell; i++) {
    if (board[0][i] == currentUser && board[1][i] == currentUser &&
        board[2][i] == currentUser) {
        gameWon = true;
        break;
    }
}

// 대각선 빙고 확인
if ((board[0][0] == currentUser && board[1][1] == currentUser &&
    board[2][2] == currentUser) ||
    (board[0][2] == currentUser && board[1][1] == currentUser &&
    board[2][0] == currentUser)) {
    gameWon = true;
}
```

```
// 8. 빙고 시 승자 출력 후 종료 (가로, 세로, 대각선)
if (gameWon) {
    cout << "빙고! " << currentUser << " 플레이어가 승리하였습니다." << endl;
} else {
    cout << "무승부! 모든 칸이 찼습니다." << endl;
}
```

- 입력

- `numCell`: 가로/세로 칸 개수.
- `board`: 현재 게임 보드 상태를 나타내는 배열.
- `currentUser`: 현재 차례의 유저 (X 또는 O).

- 결과

- 빙고가 되었는지 확인하고, 빙고가 되었을 경우 해당 유저가 승리했다는 메시지를 출력합니다.

- 설명

- 가로, 세로, 대각선 방향으로 빙고가 완성되었는지를 확인합니다.
- 가로 방향으로 3개의 돌이 동일한 유저의 돌로 채워졌는지 확인합니다.
- 세로 방향으로 3개의 돌이 동일한 유저의 돌로 채워졌는지 확인합니다.
- 대각선 방향으로 3개의 돌이 동일한 유저의 돌로 채워졌는지 확인합니다.
- 빙고가 완성되었다면 해당 유저가 승리했다는 메시지를 출력합니다.
- 그렇지 않으면 모든 칸이 찼다는 무승부 메시지를 출력합니다.

## 7. 모든 칸이 찼으면 종료

```
bool boardFull = false; // 모든 칸이 찼는지 여부 체크 (초기상태는 false 로 선언)

// 7. 모든 칸이 찼는지 확인
boardFull = true;
for (int i = 0; i < numCell; i++) {
    for (int j = 0; j < numCell; j++) {
        if (board[i][j] == ' ') {
            boardFull = false;
            break;
        }
    }
}
if (!boardFull)
    break;
}
```

- 입력

- `numCell`: 가로/세로 칸 개수.
- `board`: 현재 게임 보드 상태를 나타내는 배열.

- 결과

- 모든 칸이 찻는지 여부를 판단하고, 해당 정보를 변수 `boardFull` 에 저장합니다.
- 설명
  - 현재 게임 보드 상태에서 모든 칸이 찻는지 여부를 확인합니다.
  - 모든 칸이 차있으면 `boardFull` 을 `true` 로 설정합니다. 그렇지 않으면 `boardFull` 을 `false` 로 설정합니다.

## 4. 테스트

### 1. 기능 별 테스트 결과

#### 1. 누구의 차례인지 출력

```

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

```

#### 2. 좌표 입력 받기

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

```

---|---|---
X | 0 | X
---|---|---
X | 0 | X
---|---|---
0 |   | 
---|---|---

```

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

```

---|---|---
X | 0 | X
---|---|---
X | 0 | X
---|---|---
0 |   | 0
---|---|---

```

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

1, 1: 이미 둘이 차 있습니다. 1번 유저 (X)의 차례입니다 -> (x, y) 좌표를 입력하세요 : 2 1

```

---|---|---
X | 0 | X
---|---|---
X | 0 | X
---|---|---
0 | X | 0
---|---|---

```

### 3. 입력 받은 좌표 유효성 체크

#### a. 지정된 좌표를 넘어가는 경우

```

X | 0 | 
---|---|---
   | X | 
---|---|---
   |   | 
---|---|---

```

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

```

---|---|---
X | 0 | 
---|---|---
   | X | 
---|---|---
0 |   | 
---|---|---

```

1번 유저 (X)의 차례입니다 -> (x, y) 좌표를 입력하세요 : 3 1

3, 1: x와 y 둘 중 하나가 칸을 벗어납니다

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

```

---|---|---
X | 0 | 
---|---|---
   | X | 
---|---|---
0 |   | X
---|---|---

```

빙고! X 플레이어가 승리하였습니다.

#### b. 이미 둘이 차있는 경우

```

1번 유저 (X)의 차례입니다 -> (x, y) 좌표를 입력하세요 : 1 2
---|---|---
X | 0 | X
---|---|---
X | 0 | X
---|---|---
0 |   | 
---|---|---
2번 유저 (0)의 차례입니다 -> (x, y) 좌표를 입력하세요 : 2 2
---|---|---
X | 0 | X
---|---|---
X | 0 | X
---|---|---
0 |   | 0
---|---|---
1번 유저 (X)의 차례입니다 -> (x, y) 좌표를 입력하세요 : 1 1
1, 1: 이미 둘이 차있습니다 .1번 유저 (X)의 차례입니다 -> (x, y) 좌표를 입력하세요 : 2 1
---|---|---
X | 0 | X
---|---|---
X | 0 | X
---|---|---
0 | X | 0
---|---|---

```

#### 4. 좌표에 O / X 놓기

```

1번 유저 (X)의 차례입니다 -> (x, y) 좌표를 입력하세요 : 1 2
---|---|---
X | 0 | X
---|---|---
X | 0 | X ←
---|---|---
0 |   | 
---|---|---
2번 유저 (0)의 차례입니다 -> (x, y) 좌표를 입력하세요 : 2 2
---|---|---
X | 0 | X
---|---|---
X | 0 | X
---|---|---
0 |   | 0 ←
---|---|---
1번 유저 (X)의 차례입니다 -> (x, y) 좌표를 입력하세요 : 1 1
1, 1: 이미 둘이 차있습니다 .1번 유저 (X)의 차례입니다 -> (x, y) 좌표를 입력하세요 : 2 1
---|---|---
X | 0 | X
---|---|---
X | 0 | X
---|---|---
0 | X | 0
---|---|---

```

#### 5. 현재 보드판 출력



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

X	0	X
X	0	X
0		

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

X	0	X
X	0	X
0		0

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

1, 1: 이미 돌이 차 있습니다. 1번 유저 (X)의 차례입니다 -> (x, y) 좌표를 입력하세요 : 2 1

X	0	X
X	0	X
0	X	0

## 6. 빙고 시 승자 출력 후 종료

X	0	
	X	

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

X	0	
	X	
0		

1번 유저 (X)의 차례입니다 -> (x, y) 좌표를 입력하세요 : 3 1

3, 1: x와 y 둘 중 하나가 칸을 벗어납니다.

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

X	0	
	X	
0		X

빙고! X 플레이어가 승리하였습니다.

## 7. 모든 칸이 찼으면 종료

```

X | 0 | X
---|---|---
X | 0 | X
---|---|---
0 |   | 
---|---|---
2번 유저 (0)의 차례입니다 -> (x, y) 좌표를 입력하세요 : 2 2
---|---|---
X | 0 | X
---|---|---
X | 0 | X
---|---|---
0 |   | 0
---|---|---
1번 유저 (X)의 차례입니다 -> (x, y) 좌표를 입력하세요 : 1 1
1, 1: 이미 돌이 차 있습니다 .1번 유저 (X)의 차례입니다 -> (x, y) 좌표를 입력하세요 : 2 1
---|---|---
X | 0 | X
---|---|---
X | 0 | X
---|---|---
0 | X | 0
---|---|---
무승부! 모든 칸이 찼습니다.

```

## 2. 최종 테스트 스크린샷

```

1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     const int numCell = 3;
6     char board[numCell][numCell]; // numCell*numCell 만큼의 보드 이차원 배열 선언
7     int x, y; // 사용자에게 입력받는 x, y 좌표를 저장할 변수
8
9     // 보드란 배열을 생성 (3, 3)
10    for (x = 0; x < numCell; x++) {
11        for (y = 0; y < numCell; y++) {
12            board[x][y] = ' ';
13        }
14    }
15
16    // 게임하는 코드
17    int k = 0; // 누구 차례인지 체크하기
18    char currentUser = 'X'; // 현재 유저의 돌을 저장하기 위한 문자 변수
19    bool gameWon = false; // 빙고 여부 체크 (초기상태는 false 로 선언)
20    bool boardFull = false; // 모든 칸이 찼는지 여부 체크 (초기상태는 false 로 선언)
21
22    while (!gameWon && !boardFull) { // 빙고가 완성되거나, 모든 칸이 차지 않는 이상 계속 게임 코드 반복
23        // 1. 차례인지 출력
24        switch (k % 2) {
25            case 0:
26                cout << k % 2 + 1 << "번 유저(X)의 차례입니다 -> "; // 홀수번 user = X
27                currentUser = 'X';
28                break;
29            case 1:
30                cout << k % 2 + 1 << "번 유저(0)의 차례입니다 -> "; // 짝수번 user = 0
31                currentUser = '0';
32                break;
33        }
34
35        // 2. 좌표 입력 받기
36        cout << "(x, y) 좌표를 입력하세요: ";
37        cin >> x >> y;
38
39        // 3. 입력받은 좌표의 유효성 체크
40        if (x >= numCell ||

```

```

> sh -c make -s
> ./main
1번 유저 (X)의 차례입니다 -> (x, y) 좌표를 입력하세요 : 0 0
X |  | 
---|---|---
|  |  | 
---|---|---
|  |  | 
---|---|---
2번 유저 (0)의 차례입니다 -> (x, y) 좌표를 입력하세요 : 0 1
X | 0 | 
---|---|---
|  |  | 
---|---|---
1번 유저 (X)의 차례입니다 -> (x, y) 좌표를 입력하세요 : 1 1
X | 0 | 
---|---|---
X |  | 
---|---|---
2번 유저 (0)의 차례입니다 -> (x, y) 좌표를 입력하세요 : 2 0
X | 0 | 
---|---|---
X |  | 
---|---|---
0 |  | 
---|---|---
1번 유저 (X)의 차례입니다 -> (x, y) 좌표를 입력하세요 : 3 1
3, 1: x와 y 둘 중 하나가 칸을 벗어났습니다.
1번 유저 (X)의 차례입니다 -> (x, y) 좌표를 입력하세요 : 2 2
X | 0 | 
---|---|---
0 |  | 
---|---|---
X |  | 
---|---|---
0 |  | 
---|---|---
빙고! X 플레이어 가 승리하였습니다.
>

```

- 기존 VSCode를 사용중인데, 빈 배열 초기화하는 문법이 C++ 11 버전 이후만 가능한 반면 VSCode에서 실행시 오류를 발생시키는 이유로 Replit으로 코드를 작성하고 실행하였습니다.

## 5. 결과 및 결론

### 1. 프로젝트 결과

- Tic Tac Toe 게임을 요구사항에 따라 만들었고, 성공적으로 실행하였음

## 2. 느낀 점

- a. 입력 받은 좌표 유효성 체크
  - b. 빙고 시 승자 출력 후 종료
  - c. 모든 칸이 찼으면 종료
- 위의 세 가지 내용이 특히 구현하는 데 어려움을 느꼈음
  - 최초의 '상태'를 지정해주는 것을 조금 더 머릿속에 정형화시킬 필요가 있다고 느낌
    - 예)

```
// 빙고 여부 체크 (초기상태는 false 로 선언)
bool gameWon = false;

// 모든 칸이 찼는지 여부 체크 (초기상태는 false 로 선언)
bool boardFull = false;
```

- 시험이 두렵다...