No. 01

# 2024 CPP Project

## Tic-Tac-Toe Game

Student ID: 214930

Name: 범지성

## 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. 제약 조건: 보드판은 2차원 배열 사용

## 3. 설계 및 구현

1) 누구의 차례인지 출력

```
20          // 1. Who's Turn
21          switch(k % 2){
22              case 0:
23                  cout << "첫번째 유저(X)의 차례입니다 -> ";
24                  currentUser = 'X';
25                  break;
26              case 1:
27                  cout << "두번째 유저(O)의 차례입니다 -> ";
28                  currentUser = 'O';
29                  break;
30          }
```

-입력

- k = 현재 유저 구분자

-결과

- 어떤 유저의 입력을 받을지 출력
- 문장 출력 후 좌표 입력으로 넘어감

-설명

- k를 2로 나눈 나머지가 0이면 첫 번째 유저, 1이면 두 번째 유저임을 알림
- 각 조건에 대응하는 유저의 기호를 currentUser에 대입

2) 좌표 입력 받기

```
32          // 2. Get Position
33          cout << "(x, y) 좌표를 입력하세요: ";
34          cin >> x >> y;
```

-입력

- x = 좌표 x 값
- y = 좌표 y 값

-결과

- 좌표 입력하라는 지시문 출력 후 유저 입력 대기

-설명

- cin으로 x와 y에 유저 입력을 받는다.

3) 입력 받은 좌표 유효성 체크

```
36          // 3. Check Validity
37          if(x >= numCell || y >= numCell){
38              cout << x << ", " << y << ": ";
39              cout << " x와 y 둘 중 하나가 칸을 벗어납니다.\n";
40              continue;
41          }
42          if(board[y][x] != ' '){
43              cout << x << ", " << y << ": 이미 돌이 차있습니다.\n";
44              continue;
45          }
```

-입력

- x = 좌표 x 값
- y = 좌표 y 값
- numCell = 가로/세로 칸 개수

-결과

- 칸을 놓을 수 없는 이유를 출력
- 출력 후 while문 초반으로 이동

-설명

- 사용자가 입력한 좌표가 게임 판을 벗어나는지 if로 체크
- 사용자가 입력한 좌표에 이미 돌이 있는지 if로 체크

4) 좌표에 O/X 놓기

```
47          // 4. Set Marker on Position
48          board[y][x] = currentUser;
49          count++; // Increase when Turn ends
```

-입력

- x = 좌표 x 값
- y = 좌표 y 값
- board = 게임판

- currentUser = 현재 유저의 돌 기호
- count = 유저들이 돌을 놓은 총 횟수

-결과

- 게임판의 (x, y)칸에 currentUser의 값이 저장된다

-설명

- 이후 기능을 위해 유저들의 돌 놓은 횟수를 추가

5) 현재 보드판 출력

```
51          // 5. Print Board
52          for(int i = 0; i < numCell; i++){
53              cout << "---|---|---\n";
54              for(int j = 0; j < numCell; j++){
55                  cout << board[i][j];
56                  if(j == numCell - 1) break;
57                  cout << "  |";
58              }
59              cout << endl;
60          }
61          cout << "---|---|---\n";
62          k++;
```

-입력

- numCell = 가로/세로 칸 개수
- board = 게임판
- k = 현재 유저 구분자

-결과

- 현재 게임판에 저장된 정보가 그대로 출력된다

-설명

- 2중 반복문으로 2차원 배열로 저장된 board의 정보를 2차원으로 출력한다.
- 한 차례가 지났기 때문에 구분자를 전환하기 위해 k에 1을 더한다.

6) 빙고 시 승자 출력 후 종료

```
64          // 6-1. Winner (Winner can appear at least count reach 5)
65          if(count >= 5){
66              // For every case check Vertical and Horizontal Lines
67
68              // Check Vertical Line
69              int lineCheck[numCell] = {0, };
70              for(int i = 0; i < numCell; i++){
71                  if(board[i][x] == currentUser)
72                      lineCheck[i] = 1;
73              }
74              if ((lineCheck[0] == lineCheck[1]) && (lineCheck[0] == lineCheck[2]) && lineCheck[0]){
75                  cout << "vert\n";
76                  winnerAppear = 1;
77                  break;
78              }
79              // Check Horizontal Line
80              for(int i = 0; i < numCell; i++) lineCheck[i] = 0; // lineChecker initaltize
81
82              for(int i = 0; i < numCell; i++){
83                  if(board[y][i] == currentUser)
84                      lineCheck[i] = 1;
85              }
86              if ((lineCheck[0] == lineCheck[1]) && (lineCheck[0] == lineCheck[2]) && lineCheck[0]){
87                  cout << "hori\n";
88                  winnerAppear = 1;
89                  break;
90              }
```

```
92              // When mark is on Top to Bottom Tilted Line
93              if(x == y){
94                  for(int i = 0; i < numCell; i++) lineCheck[i] = 0; // lineChecker initaltize
95
96                  for(int i = 0; i < numCell; i++){
97                      if(board[i][i] == currentUser)
98                          lineCheck[i] = 1;
99                  }
100                 if ((lineCheck[0] == lineCheck[1]) && (lineCheck[0] == lineCheck[2]) && lineCheck[0]){
101                     winnerAppear = 1;
102                     cout << "ttb\n";
103                     break;
104                 }
105             }
106
107             // When mark is on Bottom to Top Tilted Line
108             if((x + y) == 2){
109                 for(int i = 0; i < numCell; i++) lineCheck[i] = 0; // lineChecker initaltize
110
111                 for(int i = 0; i < numCell; i++){
112                     if(board[2 - i][i] == currentUser)
113                         lineCheck[i] = 1;
114                 }
115                 if ((lineCheck[0] == lineCheck[1]) && (lineCheck[0] == lineCheck[2]) && lineCheck[0]){
116                     winnerAppear = 1;
117                     cout << "btt\n";
118                     break;
119                 }
120             }
121         }
```

```
128         // If Winner Exist or Not
129         if(winnerAppear == 1)
130             cout << "유저(" << currentUser << ")가 이겼습니다!.\n";
```

-입력

- numCell = 가로/세로 칸 개수
- board = 게임판
- lineCheck = 해당 줄 빙고가 완성됐는지 확인하는 배열
- x = 좌표 x 값

- y = 좌표 y 값
- winnerAppear = 승자 등장 여부 확인 플래그
- count = 돌 놓인 횟수(k와 동일하나 코드 작성 중 가독성을 위해 추가함)

-결과

- 현재 돌을 넣은 유저가 한 줄 빙고를 완성하면 while 탈출
- 반복문을 탈출한 유저가 승리했음을 알리고 프로그램이 종료

-설명

- 승리자가 발생할 수 있는 최소 횟수 5회 진행됐을 때 승리 조건 탐색을 시작한다.
- 최근 돌을 놓은 위치가 승리 시 줄에 포함되니 해당 돌을 기준으로 라인을 탐색한다.
- 탐색하고 싶은 줄의 모든 돌들이 최근 사용한 돌과 같으면 lineCheck의 원소를 1로 바꾼다.
- lineCheck의 모든 원소가 동일하고 그 값이 1이면 winnerAppear를 1로 바꾸고 while문을 탈출한다.
- While문 탈출 후 winnerAppear가 1이면 현재 유저가 승리했음을 알린다.
- 라인 검사는 놓인 돌이 포함된 최소 2개에서 4개의 라인을 검사하고, 라인 검사 시작 조건은 이하와 같다.
- 돌이 어느 위치에 놓여도 해당 돌이 포함된 가로줄, 세로줄을 탐색하니 추가 조건 없이 해당 줄들을 검사한다.
- 돌이 우하향 대각선 위에 놓였는지 여부를 if로 체크하고 (이 경우의 수의 x == y인 경우의 수와 같다) 해당 대각선을 탐색한다.
- 돌이 우상향 대각선 위에 놓였는지 여부를 if로 체크하고 (이 경우의 수는 x + y 값이 2인 경우의 수와 같다) 해당 대각선을 탐색한다.

7) 모든 칸이 찼으면 종료

```
118         // 6-2. No Winner and Every Board is Occupied
119         if(count == 9){
120             cout << "Nobody Won, Game Ends\n";
121             break;
122         }
```

-입력

- count = 돌 놓인 횟수(k와 동일하나 코드 작성 중 가독성을 위해 추가함)

-결과

- 모든 칸에 돌이 놓이면 무승부임을 출력
- 출력 후 while문 탈출

-설명

- 승리 조건 탐색 이후에 진행되며 count 값이 9로 게임이 완료됐는지 if로 체크

## 4. 테스트

1. 기능 별 테스트 결과:

1) 누구의 차례인지 출력

```
첫번째 유저(X)의 차례입니다 ->
```

```
두번째 유저(O)의 차례입니다 ->
```

2) 좌표 입력 받기

```
(x, y) 좌표를 입력하세요: 0 0
```

3) 입력 받은 좌표 유효성 체크

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

4) 좌표에 O/X 놓기 및 현재 보드판 출력 (기능 4, 5 통합 확인)

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

5) 빙고 시 승자 출력 후 종료

```
유저(X)가 이겼습니다!.
```

```
유저(O)가 이겼습니다!.
```

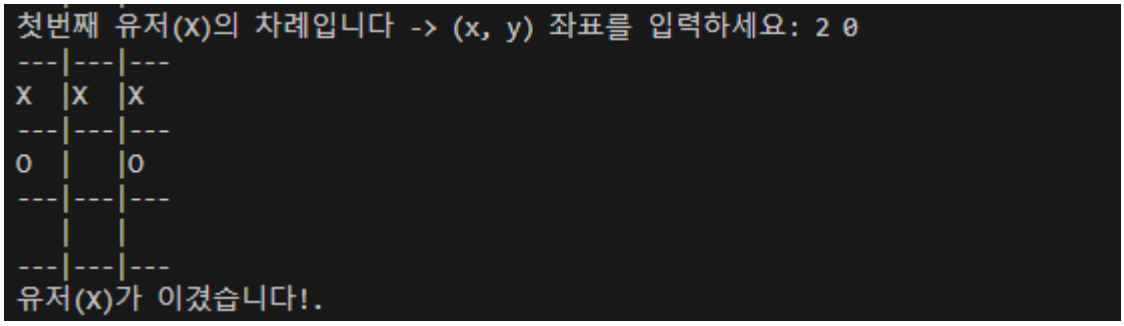
6) 모든 칸이 찼으면 종료

```
Nobody Won, Game Ends
```

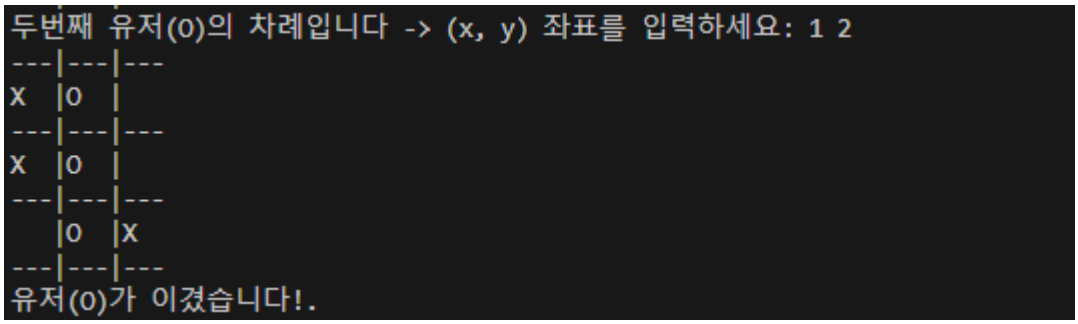
2. 최종 테스트 스크린샷:

-승리 조건 분기 4개와 무승부 확인

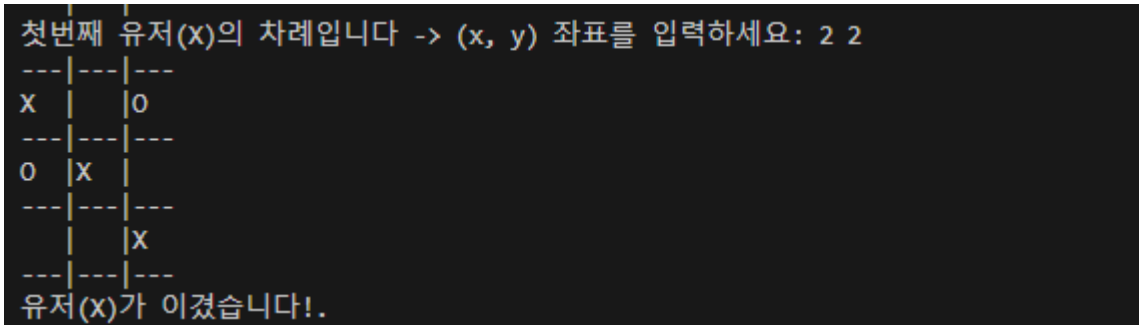
S1) 유저 X가 가로줄 빙고를 완성해 승리



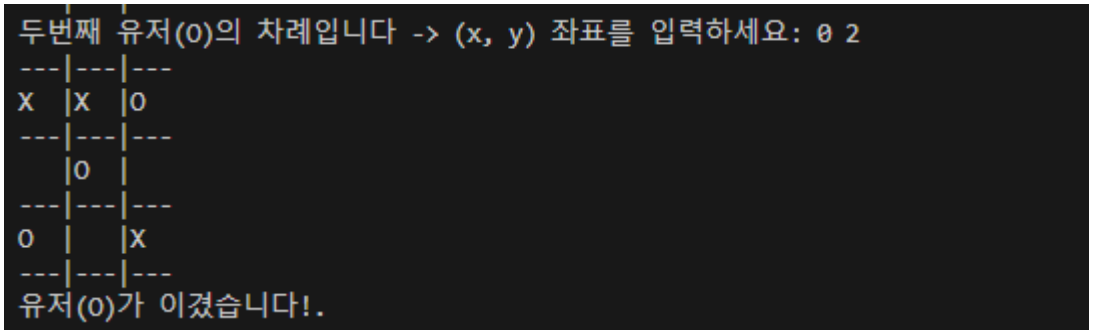

S2) 유저 O가 세로줄 빙고를 완성해 승리




S3) 유저 X가 우하향 대각선 빙고를 완성해 승리




S4) 유저 O가 우상향 대각선 빙고를 완성해 승리




S5) 빙고판이 가득차고 무승부로 종료

```
첫번째 유저(X)의 차례입니다 -> (x, y) 좌표를 입력하세요: 1 2
---|---|---
X  |X  |O
---|---|---
O  |O  |X
---|---|---
X  |X  |O
---|---|---
Nobody Won, Game Ends
```

## 5. 결과 및 결론

1. 프로젝트 결과: Tic Tac Toe 게임을 제작함

2. 느낀 점: 틱택토 게임이 종료되는 조건 17개 (플레이어 승리 경우 각 8회, 무승부 1회)를 코드 실행 시간을 간소화하기 위해 고민해 5개의 분기점(가로, 세로, 대각선 둘 빙고 + 무승부)으로 축소하는 사고과정이 즐거웠음.