

# 200244 김민호 C++프로그래밍 언어 보고서(2023.10.11)

## 1. 서론

- A. 프로젝트 목적 및 배경 : 4주차까지 배운 내용에 대한 실습
- B. 목표 : Tic Tac Toe 게임 구현

## 2. 요구사항

- A. 사용자 요구사항
  - 두 명의 플레이어가 돌아가며 3 x 3 보드판에 O와 X를 놓기
  - 한 명의 플레이어가 먼저 가로, 세로, 대각선 중 한 줄을 완성하면 승리 후 종료
  - 9칸이 모두 채워졌을 경우 종료
- B. 기능 요구사항
  - 현재 턴의 플레이어가 누구지 출력, 플레이어 둘 모양 저장
  - 좌표 입력 받기
  - 입력받은 좌표 유효성 검사
  - 좌표에 둘 놓기
  - 보드판 현황 출력
  - 가로, 세로, 대각선 줄 빙고 검사
  - 빙고 시 승리자 이름 출력 후 종료
  - 칸이 모두 채워졌으면 종료

## 3. 설계 및 구현

- A. 기능 별 구현 사항: 요구사항 별 코드

보드판 배열 초기화

```
const int numCell = 3;
char board[numCell][numCell]{}; // 보드 크기 : 3 x 3
int x, y; // x : 입력받은 행 좌표, y : 입력받은 열 좌표 저장 변수
// 보드판 초기화
for (x = 0; x < numCell; x++) {
    for (y = 0; y < numCell; y++) {
        board[x][y] = ' ';
    }
}
```

입력 : x, y = 입력받은 행, 열의 좌표

numCell = 보드판 행, 열의 크기

결과 : 보드판 배열 초기화

설명 : for loop 구문을 사용하여 보드판 배열을 (space)로 초기화

현재 턴의 플레이어가 누구지 출력, 플레이어 둘 모양 저장

```
int k = 0; // 현재 턴 플레이어 구분을 위한 변수
char currentUser = 'X'; // 현재 플레이어 둘 모양을 저장하기 위한 변수
while (true) {
    // 1. 현재 턴 플레이어 출력, 플레이어 둘 모양 저장
    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(O)의 차례입니다 -> ";
            currentUser = 'O';
            break;
    }
}
```

입력 : k = 현재 턴의 회차

currentUser = 현재 턴의 플레이어의  
둘 모양을 저장하기 위한 변수

결과 :

현재 턴의 플레이어가 누구인지 화면에 출력  
현재 턴에 보드판에 놓일 둘 모양을 저장

설명 : 턴이 k회일 때, 두 명의 플레이어  
번갈아 바뀌는 것을 k % 2으로 구현, 두 플  
레이어에 대한 출력과 저장을 switch구문으  
로 구분

## 입력받은 좌표 유효성 검사

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

//3. 입력받은 좌표 유효성 검사
if (x >= numCell || y >= numCell) {
    cout << x << ", " << y << ": ";
    cout << "x와 y 둘 중 하나가 칸을 벗어납니다." << endl;
    continue;
}
if (board[x][y] != ' ') {
    cout << x << ", " << y << ": 이미 돌이 차있습니다." << endl;
    continue;
}
```

## 좌표에 돌 놓기 & 보드판 현황 출력

```
// 4. 입력받은 좌표에 돌 놓기
board[x][y] = currentUser;

// 5. 보드판 현황 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;
```

## 가로, 세로줄 빙고 검사

```
// 6. 가로줄 빙고 검사
bool win = false;
for (int i = 0; i < numCell; i++) {
    for (int j = 0; j < numCell; j++) {
        if (board[i][j] == currentUser) {
            if (j == numCell - 1) {
                cout << "가로에 모두 돌이 놓였습니다!";
                win = true;
                break;
            }
        }
        else
            break;
    }
}

//7. 세로줄 빙고 검사
for (int j = 0; j < numCell; j++) {
    for (int i = 0; i < numCell; i++) {
        if (board[i][j] == currentUser) {
            if (i == numCell - 1) {
                cout << "세로에 모두 돌이 놓였습니다!";
                win = true;
                break;
            }
        }
        else
            break;
    }
}
```

입력 : x, y = 입력받은 행, 열의 좌표

Board[][] = 보드판의 돌 배치

결과 : 입력받은 좌표를 x, y에 저장

칸을 놓을 수 없는 이유를 설명

출력 후 다시 입력하도록 이동

설명 : 입력받은 행, 열의 좌표가 3 x 3 칸을 벗어나는지 if로 체크

해당 좌표가 비어있는지 if로 체크

입력 : currentUser = 현재 플레이어 돌 모양

i, j = 보드판 행, 열 좌표 변수

Board[][] = 보드판의 돌 배치

numCell = 보드판 행, 열 크기

결과 : 보드판과 플레이어 입력에 따른 돌 배치 출력

설명 : board배열을 for 구문으로 출력

3줄이므로 가로줄 출력을 for 구문 3회 반복

마지막 가로줄 따로 출력

입력 : i, j = 보드판 행, 열 좌표 변수

Board[][] = 보드판의 돌 배치

currentUser = 현재 플레이어 돌 모양

numCell = 보드판 행, 열 크기

결과 : 가로나 세로줄이 모두 같은 모양 돌이면 문구 출력

출력 후 win 변수에 true 저장

설명 : 각 행마다 for로 반복하여 각 열의 돌 모양과 현재 턴의 돌 모양 일치 여부 for, if 구문으로 확인, 해당 행의 마지막 열까지 일치하는지 if구문으로 확인 후 문구 출력, 출력 시 win 에 true를 저장하여 승리 상태 저장, 세로줄 빙고 검사도 행, 열의 순서(i, j)를 바꿔 같은 방식으로 진행

#### 대각선 줄 빙고 검사

```
// 8. 대각선1 빙고 검사
for (int i = 0; i < numCell; i++) {
    if (board[i][i] == currentUser) {
        if (i == numCell - 1) {
            cout << "왼쪽 위에서 오른쪽 아래 대각선으로 모두 돌이 놓였습니다!";
            win = true;
            break;
        }
    }
    else
        break;
}

// 9. 대각선2 빙고 검사
for (int i = 0; i < numCell; i++) {
    if (board[i][numCell - i - 1] == currentUser) {
        if (i == numCell - 1) {
            cout << "오른쪽 위에서 왼쪽 아래 대각선으로 모두 돌이 놓였습니다!";
            win = true;
            break;
        }
    }
    else
        break;
}
```

입력 : i = 보드판 행, 열 좌표 변수

Board[][] = 보드판의 돌 배치

currentUser = 현재 플레이어 돌 모양

numCell = 보드판 행, 열 크기

결과 : 대각선에 모두 같은 모양 돌이 있으면 문구 출력

설명 : 왼쪽 위의 돌 모양과 현재 턴의 돌 모양의 일치 여부 if로 체크, 오른쪽 아래로 내려가며 for로 반복하여 검사, 마지막 좌표까지 일치하는지 if로 확인 후 문구 출력. 위 내용을 오른쪽 위에서 왼쪽 아래로도 수행

#### 빙고 시 승리자 이름 출력 후 종료

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

입력 : win = 승리 상태를 저장하는 변수

k = 턴이 진행된 횟수 - 1

currentUser = 현재 턴의 돌 모양

결과 : 종료 이유와 종료 문구 출력

설명 : win에 의해 승리자 여부를 if로 체크, 현재 턴 플레이어, (k % 2 + 1)번 유저의 승리 출력 후 종료

#### 칸이 모두 채워졌으면 종료

```
k++;
// 11. 칸이 모두 찼을 때 종료 출력
if (k >= 9) {
    cout << "모든 칸이 다 찼습니다. 종료합니다";
    break;
}
```

입력 : win = 승리 상태를 저장하는 변수

k = 턴이 진행된 횟수 - 1

결과 : 종료 이유와 종료 문구 출력

설명 : while 구문이 9회 반복된 후 모든 칸이 채워지므로 모든 문장 실행 후 k가 9가 될 때 문구 출력 후 종료

## 4. 테스트

### A. 기능 별 테스트 결과 : 요구사항 별 스크린샷

#### 1. 현재 턴의 플레이어가 누군지 출력, 좌표 입력 받기

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

#### 2. 현재 보드판 현황 출력

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

#### 3. 입력받은 좌표가 칸을 벗어난 경우 경고 문구 출력

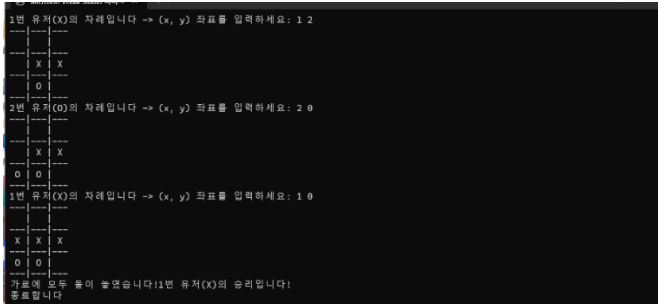
1번 유저(X)의 차례입니다 -> (x, y) 좌표를 입력하세요: 0 3  
0, 3: x와 y 둘 중 하나가 칸을 벗어납니다.  
1번 유저(X)의 차례입니다 -> (x, y) 좌표를 입력하세요: |

#### 4. 입력받은 좌표에 이미 돌이 채워져 있는 경우 경고 문구 출력

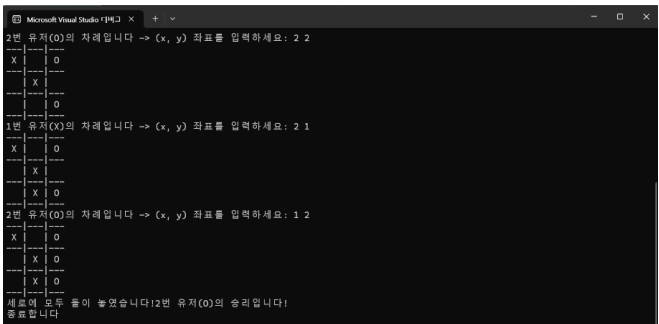
2번 유저(O)의 차례입니다 -> (x, y) 좌표를 입력하세요: 0 0  
0, 0: 이미 돌이 차 있습니다.  
2번 유저(O)의 차례입니다 -> (x, y) 좌표를 입력하세요: |

B. 최종 테스트 스크린샷 : 프로그램 전체 동작 스크린샷

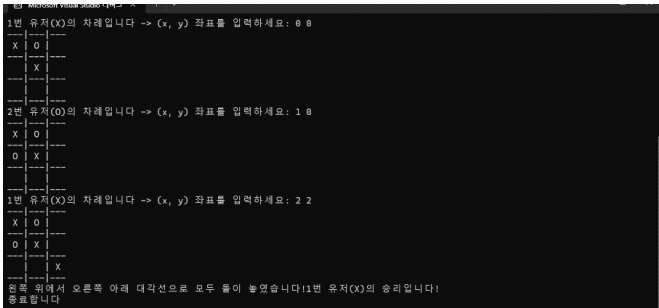
## 1. 가로줄 빙고



## 2. 세로줄 빙고



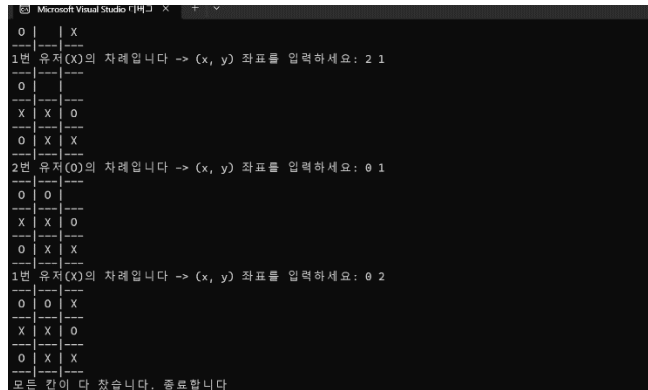
### 3. 대각선1 빙고



#### 4. 대각선2 빙고



## 5. 모든 칸이 찼을 경우



```
Microsoft Visual Studio [144] X
0 | | X
---|---
1번 유저(X)의 차례입니다 -> (x, y) 좌표를 입력하세요: 2 1
0 | | 
---|---
X | X | O
---|---
0 | X | X
2번 유저(O)의 차례입니다 -> (x, y) 좌표를 입력하세요: 0 1
0 | O | 
---|---
X | X | O
---|---
0 | X | X
1번 유저(X)의 차례입니다 -> (x, y) 좌표를 입력하세요: 0 2
0 | O | X
---|---
X | X | O
---|---
0 | X | X
모든 칸이 다 찼습니다. 종료합니다
```

## 5. 결과 및 결론

A. 프로젝트 결과 : Tic Tac Toe 게임 완성

B. 느낀 점 :

실습과 답과 나의 코드가 달라 어떤 코드가 더 나은 코드인지 궁금합니다. 한 교수님께서 가독성과 확장성이 좋은 코드가 좋은 코드라고 하셨는데 실습 답보다 코드 길이가 훨씬 길어져 비효율적인 코드 같아 보입니다. 코드를 어떤 방식으로 작성해야 좋은 코드를 작성할 수 있는지 수업시간에 조금씩 알려주시면 훨씬 유익할 것 같습니다. 또, 교수님께서 실무적인 부분을 많이 알려주셔서 도움이 되었습니다. (git hub, naming conventions, 프로젝트 보고서 등)