

1. 서론

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

2. 요구사항

- 사용자 요구사항: 두 명의 사용자가 번갈아가며 O 와 X 를 놓기
- 기능 요구사항: 1. 빙고 시 승자 출력 후 종료 (가로, 세로, 대각선)

2. 모든 칸이 찼으면 종료

3. 설계 및 구현

1. 코드블록

```
const int numCell = 3; // 보드 한 줄의 칸 수
char board[numCell][numCell]{}; // 보드판 설정
int x, y; // 사용자에게 입력받는 x,y좌표를 저장할 변수
int countspace = 0; // 보드판의 채워진 칸을 확인하는 변수

// 게임을 하는 인원 수 3명 이상이면 switch문의 case를 인원 수 만큼 만들어야한다.
int players = 2;
```

2. 입력

- numCell = 보드판 한 줄의 칸 수(3)
- board[][] = 보드판 가로*세로 설정 (3*3 으로 설정)
- x = x 좌표 (미정)
- y = y 좌표 (미정)
- countspace = 보드판의 채워진 칸 (처음에는 0 칸 채워졌다.)
- players = 게임하는 인원 (추후 확장성을 위해 생성하였다.)

3. 결과

- 게임을 하기위한 기본적인 변수들을 설정하였다.

4. 설명

- board[numCell][numCell]은 3*3 크기의 보드판 설정을 위한 2D 배열이다.
- x, y 는 사용자로부터 x, y 좌표를 입력 받고 저장할 변수이다.
- countspace 는 보드판의 채워진 칸을 세는 변수로 추후 보드판이 다 채워졌는지 확인하여 게임의 종료여부를 결정할 때 사용한다.
- Players 는 플레이어가 추가되었을 때 간편하게 바꾸기 위해 생성했다.

1. 코드블록

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

2. 입력

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

3. 결과

- 게임시작을 위한 보드판 초기화

4. 설명

- 이중 반복문을 통해 이중 반복문을 통해 2D 배열의 줄, 칸에 0 부터 numCell-1 까지 ' '을 입력
- 2D 보드판을 빈칸(' ')으로 초기화하였다.

1. 코드블록

```
int k = 0; // 누구 차례인지 체크하기 위한 변수
char currentUser = 'X'; //현재 유저의 돌을 저장하기 위한 문자 변수
while(true){
    //1. 누구 차례인지 출력
    switch(k%players){
        case 0:
            cout <<"첫번째 유저(X)의 차례입니다 -> ";
            currentUser = 'X';
            break;
        case 1:
            cout <<"두번째 유저(O)의 차례입니다 -> ";
            currentUser = 'O';
            break;
        /* case n: // 플레이어가 늘어나면 추가하시오
            cout <<"n번째 유저(N)의 차례입니다 -> ";
            currentUser = 'N';
            break; */
    }
}
```

2. 입력

- K= 플레이어 차례 확인
- currentUser = 현재 플레이어의 돌('X', 'O')
- players = 게임하는 인원

2. 결과

- 플레이어 차례를 구분, 설정해준다.

3. 설명

- K 는 플레이어의 순서를 정하는 변수로 틱택토 돌 놓기 1 회를 정상적으로 수행하면 k 가 1 오르고 k 를 players 만큼 나누고 남은 수를 통해 플레이어의 차례를 구분한다.
- 주석 처리된 부분은 plays 가 늘어나면 case 를 추가해 여러명이 게임을 플레이할 수 있도록 확장성을 위해 남긴 부분이다.

```

while(true){
    //1. 누구 차례인지 출력
    switch(k%players){
        case 0:
            cout <<"첫번째 유저(x)의 차례입니다 -> ";
            currentUser = 'x';
            break;
        case 1:
            cout <<"두번째 유저(o)의 차례입니다 -> ";
            currentUser = 'o';
            break;
        case 2: // 플레이어가 늘어나면 추가하시오
            cout <<"n번째 유저(v)의 차례입니다 -> ";
            currentUser = 'v';
            break;
    }
}

```

왼쪽은 플레이어가 3 명일 때의 코드로
k 를 players(3)로 나누어서
플레이어 3 명의 차례를 구분하였다.

1. 코드블록

```

// 2. 좌표 입력 받기 (직관성을 위해 x와 y를 따로따로 입력받게 만들었다.)
cout <<"x 좌표를 입력하세요: "<< endl;;
cin >> x ;
cout <<"y 좌표를 입력하세요: "<< endl;;
cin >> y ;

```

2. 입력

- x = x 좌표
- y = y 좌표

3. 결과

- x, y 좌표 입력 받음

4. 설명

- 플레이어로부터 x, y 좌표를 입력 받고 각 좌표를 변수 x, y 에 담는다.
- 사용자의 직관성을 위해 x, y 를 따로 입력하게 코드를 수정하였다.

1. 코드블록

```
// 3.입력받은 좌표의 유효성 체크(음수 확인도 추가하였다.)
if(x < 0 || y < 0 || x >= numCell || y >= numCell){
    cout << x << ", " << y << ": ";
    cout << " x 와 y 둘 중 하나가 칸을 벗어납니다." <<endl;
    continue;
}

if (board[x][y] != ' '){
    cout << x << ", " << y << ": 이미 돌이 차있습니다." << endl;
    continue;
}
```

2. 입력

- x = x 좌표
- y = y 좌표
- numCell = 보드판 한 줄의 칸 수(3)
- board[][] = 보드판 가로*세로 설정 (3*3 으로 설정)

3. 결과

- 칸을 놓을 수 없을 때 놓을 수 없는 이유를 서술
- 놓을 수 없으면 while 문의 초반으로 이동

4. 설명

- if 문을 통해 사용자가 입력한 좌표가 보드판의 칸을 벗어나는지 확인
- if 문을 통해 사용자가 입력한 좌표에 돌이 있는지 확인
- 음수도 검사하도록 조건문을 수정하였다.

1. 코드블록

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

// 5. 현재 보드 판 출력 (보드 칸 수가 늘어도 작동하도록 고침)
for (int i = 0; i < numCell; i++){
    for(int i = 0; i<numCell-1; i++){
        cout << "---|";
    }
    cout << "---" << endl;
    for (int j =0; j < numCell; j++){
        cout << board[i][j];
        if(j == numCell-1){
            break;
        }
        cout << "  |";
    }
    cout << endl;
}
for(int i = 0; i<numCell-1; i++){
    cout << "---|";
}
cout << "---" << endl;
k++;
countspace++; // 표시된 공간을 카운트
```

2. 입력

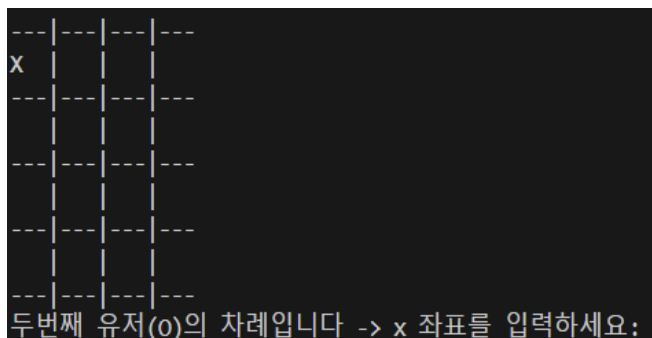
- board[x][y] = x, y 좌표가 가리키는 보드칸
- currentUser = 현재 플레이어의 돌
- numCell = 보드판 한 줄의 칸 수(3)
- k = 플레이어의 차례를 구분하는 변수
- countspace = 보드판에서 돌을 둔 공간을 세는 변수

3. 결과

- 입력받은 x, y 좌표에 현재 유저의 돌을 놓는다.
- 현재 보드판을 출력한다.
- k 값을 올려 다음 플레이어의 차례로 바꾼다.
- countspace 값을 올려 보드판에서 해당 칸이 채워짐을 표시

4. 설명

- 유효성 검사를 통과한 사용자가 입력한 좌표 (x,y)에 현재 유저의 돌을 놓는다.
- 이중 반복문을 통해 현재 게임 보드판의 상황을 시각적으로 나타낸다.
- 차례를 표시하는 switch(k%2)를 통해 k 값이 플레이어 차례를 바꾼다.
- countspace 는 추후 모든 칸이 채워지면 종료하는 코드에 사용된다.



두번째 유저(0)의 차례입니다 -> x 좌표를 입력하세요:

보드 칸 수 numCell 을 4 로
변경했을 때 보드의 이미지

1. 코드블록

```
// 6-1. 빙고 시 승자 출력 후 종료 (가로)
for(int k = 0; k < numCell; k++){
    for(int t = 0; t < numCell; t++){
        if(board[k][t] != currentUser){
            break;
        }
        if(t == numCell-1){
            cout << "가로 한 줄이 채워졌습니다." << endl;
            cout << "승자는 " << currentUser << " 입니다." << endl;
            return 0;
        }
    }
}
```



```
// 6-2. 빙고 시 승자 출력 후 종료 (세로)
for(int k = 0; k < numCell; k++){
    for(int t = 0; t < numCell; t++){
        if(board[t][k] != currentUser){
            break;
        }
        if(t == numCell-1){
            cout << "세로 한 줄이 채워졌습니다." << endl;
            cout << "승자는 " << currentUser << " 입니다." << endl;
            return 0;
        }
    }
}
```

```
// 6-3. 빙고 시 승자 출력 후 종료 (대각선 왼쪽위 -> 오른쪽 아래)
for(int i = 0; i < numCell; i++){
    if(board[i][i] != currentUser){
        break;
    }
    if(i == numCell-1){
        cout << "대각선 왼쪽위 -> 오른쪽 아래 한 줄이 채워졌습니다." << endl;
        cout << "승자는 " << currentUser << " 입니다." << endl;
        return 0;
    }
}

// 6-4. 빙고 시 승자 출력 후 종료 (대각선 오른쪽 위 -> 왼쪽 아래)
for(int i = 0; i < numCell; i++) {
    if(board[i][numCell - 1 - i] != currentUser) {
        break;
    }
    if(i == numCell - 1) {
        cout << "대각선 오른쪽 위 -> 왼쪽 아래 한 줄이 채워졌습니다." << endl;
        cout << "승자는 " << currentUser << " 입니다." << endl;
        return 0;
    }
}
```

2. 입력

- x = x 좌표
- y = y 좌표
- numCell = 보드판 한 줄의 칸 수(3)
- board[x][y] = x, y 좌표가 가리키는 보드칸
- currentUser = 현재 플레이어의 돌

1. 결과

- 가로, 세로, 대각선 빙고 시 승자를 출력한다.

2. 설명

- 틱택토의 승리는 현재 플레이어가 돌을 두는 순간에만 결정된다.

(X 유저가 돌을 둘 때 X 유저의 승리가 결정되지 O 유저의 승리는 결정되지 않는다.)

Ex) X 가 (2,2)에 돌을 둘 때 X 가 빙고를 완성하면 승리, 아니면 계속 플레이가 결정되지, 갑자기 O 의 승리가 발생하지 않는다.

즉 currentUser 의 돌만 빙고인지 확인하면 된다.

- 각 코드는 board[x][y]에 currentUser 의 돌을 둘 때 for 문을 통해 보드판의 가로, 세로, 대각선에 놓인 돌을 확인, if 문으로 각 가로,세로,대각선의 돌 중 하나라도 currentUser 의 돌이 아니면 (빙고줄이 완성되지 X) for 문을 나가고 board[x][y]의 마지막 칸까지 확인했을 때 모두 currentUser 의 돌이 있으면(빙고줄이 완성) 승리를 선언하고 게임을 종료한다.
- 대각선의 경우 (왼쪽 위 -> 오른쪽 아래)의 경우와 (오른쪽 위 -> 왼쪽 아래)의 경우 두가지가 있으므로 두가지 경우로 나누어서 확인하여야 한다.

1. 코드블록

```
// 7. 모든 칸이 다 차면 종료
if(countspace == (numCell)*(numCell)){
    cout <<"모든 칸이 다 찼습니다."<<endl;
    return 0;
}
```

2. 입력

- countspace = 보드판에서 돌을 둔 공간을 세는 변수
- numCell = 보드판 한 줄의 칸 수(3)

3. 결과

- 게임의 승자가 결정되지 않고, 빙고 칸이 전부 차면 게임을 종료한다.

4. 설명

- 보드칸의 총 개수는 numCell*numCell (3*3)이고, countspace 는 4 번 코드에서 사용자가 돌을 성공적으로 둘 때 1 씩 올라간다 즉 1 칸을 채우면 countspace 도 1 올라간다

이렇게 countspace 가 증가하다가 보드칸의 총 개수와 같아질 때, 즉 사용자가 보드의 모든 칸에 돌을 두었고, 승자가 정해지지 않으면 게임은 모든 칸이 차서 종료하게 된다.

4. 테스트

```
첫번째 유저(x)의 차례입니다 -> x 좌표를 입력하세요:
1
y 좌표를 입력하세요:
1
---|---|---
|   |   |
---|---|---
| x |   |
---|---|---
|   |   |
```

```
두번째 유저(o)의 차례입니다 -> x 좌표를 입력하세요:
2
y 좌표를 입력하세요:
0
---|---|---
|   |   |
---|---|---
| x |   |
---|---|---
0 |   |
```

누구의 차례인지 표시되고, x, y 좌표를 입력 받고 게임판에 표시

```
0, 0: 이미 돌이 차있습니다.
두번째 유저(o)의 차례입니다 -> x 좌표를 입력하세요:
```

유효성 검사를 통해 돌이 놓여 있는 좌표는 둘 수 없다.

```
5, 4: x 와 y 둘 중 하나가 칸을 벗어납니다.
두번째 유저(o)의 차례입니다 -> x 좌표를 입력하세요:
■
```

유효성 검사를 통해 게임판의 크기를 벗어나는 좌표는 입력할 수 없다.

```

---|---|---
X  |   |O
---|---|---
   |O  |X
---|---|---
O  |   |X
---|---|---
대각선 오른쪽 위 -> 왼쪽 아래 한 줄이 채워졌습니다.
승자는 O 입니다.

```

오른쪽 위 -> 왼쪽 아래 대각선 우승

```

---|---|---
X  |O  |
---|---|---
   |X  |O
---|---|---
   |   |X
---|---|---
대각선 왼쪽위 -> 오른쪽 아래 한 줄이 채워졌습니다.
승자는 X 입니다.

```

왼쪽 위 -> 오른쪽 아래 대각선 우승

```

---|---|---
X  |O  |X
---|---|---
X  |X  |O
---|---|---
O  |X  |O
---|---|---
모든 칸이 다 찼습니다.

```

모든 칸이 다 찼을 때 게임 종료

최종 테스트 결과

```
두번째 유저(o)의 차례입니다 -> x 좌표를 입력하세요:
1
y 좌표를 입력하세요:
1
---|---|---
x  |o  |x
---|---|---
x  |o  |
---|---|---
   |o  |
---|---|---
세로 한 줄이 채워졌습니다.
승자는 o 입니다.
```

결과 및 결론

프로젝트 결과: Tic Tac Toe 게임을 만들었음

느낀 점: 최대한 확장성이 좋도록 만들기 위해 노력했습니다.

대각선 빙고줄을 만들 때 처음에는 왼쪽 위 -> 오른쪽 아래만 만들었는데 코드를 검사하는 과정에서 왼쪽 아래 -> 오른쪽 위 대각선은 정답처리가 안되는 걸 확인하고 대각선 확인 코드를 2 종류로 만들어 해결했습니다.

처음 코드를 만들 때 확장성에 집중해서 만들어서 3 인 플레이어나 더 나아가 4*4, 5*5, 4*4 보드에서 3 명 플레이 등 다양한 틱택토 환경을 쉽게 바꿀 수 있었습니다.

```
x  |o  |o  |
---|---|---
   |x  |   |
---|---|---
o  |   |x  |
---|---|---
   |x  |o  |x
---|---|---
대각선 왼쪽 위 -> 오른쪽 아래 한 줄이 채워졌습니다.
승자는 x 입니다.
```

4*4 2 인 플레이 대각선 우승

```
---|---|---|---
x  |o  |v  |x
---|---|---|---
o  |o  |o  |v
---|---|---|---
v  |x  |o  |v
---|---|---|---
v  |x  |x  |x
---|---|---|---
모든 칸이 다 찼습니다.
```

4*4 3 인 플레이 무승부