C++ 프로그래밍 및 실습

01. Tic Tac Toe 게임

[10월 11일]

200791 박지성

■ 서론

1. 프로젝트 목적 및 배경 : 4주차까지 배운 내용에 대하여 이해를 심화하기 위한 실습
2. 목표 : Tic Tac Toe 게임 구현

■ 사용자 요구사항

1. 두 명의 사용자가 정해진 게임 보드 안에 O와 X를 번갈아가며 놓기

■ 기능 요구사항

1. 누구의 차례인지 출력
2. 각각의 사용자에게 게임 보드 좌표를 입력받기
3. 좌표에 대한 유효성을 체크
4. 좌표에 O,X를 놓기
5. 현재 보드판을 출력하기
6. 어떤 사용자가 정해진 보드 크기 내에 한 줄 (가로, 세로, 대각선)이 완성되면 게임의 승리
7. 승자가 없이, 정의된 게임 보드가 모두 채워졌다면 승자 없이 종료

■ 설계 및 구현

1-1. 코드 스크린샷

```
// 1. 누구 차례인지 출력
// 플레이어의 차례를 교대로 할수 있도록, k를 2로 나눈 나머지값을 이용
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;
}
```

1-2. 입력

· k = 게임 횟수를 나타내는 변수

1-3. 결과

· 어떤 사용자의 차례인지 출력

· 사용자를 지정

· switch문 탈출

1-4. 설명

1) k를 2로 나눈 나머지값에 따라 차례를 바꾸기 위한 switch문 사용

2) case 0 : k % 2 가 0일 때 (k가 짝수일 때), 실행됨.

3) case 1 : k % 2 가 1일 때 (k가 홀수일 때), 실행됨

4) 따라서, 턴이 넘어갈 때마다 k가 증가하므로 유저가 순서대로 차례가 바뀜.

2-1. 코드 스크린샷

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

2-2. 입력

· x : x의 좌표값

· y : y의 좌표값

2-3. 결과

· 입력 받은 좌표값을 x, y 변수에 저장

2-4. 설명

1) 입력 받는 문구 출력

2) 입력 받은 좌표를 x,y 변수에 각각 저장

3-1. 코드 스크린샷

```
//3. 입력받은 좌표와 유효성 체크
// 만약에 x 나 y, 둘 중 하나가 정의된 게임 보드 크기를 넘어가면 에러 메세지 출력
// 또는, 놓으려고 하는 위치가 공백이 아니라면 돌이 차있는 경우이므로, 에러 메세지 출력
if (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;
}
```

3-2. 입력

- x = x의 좌표값
- y = y의 좌표값
- numCell = 가로/세로 칸 개수

3-3. 결과

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

3-4. 설명

1) 사용자가 입력한 좌표가 게임 판을 벗어나는지 if로 체크
2) 사용자가 입력한 좌표에 돌이 이미 있는지 if로 체크

4-1. 코드 스크린샷

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

4-2. 입력

- board[x][y] : 게임 보드 안에 입력 받은 좌표 위치
- currentUser : 현재 사용자의 돌 모양

4-3. 결과

- 현재 사용자 변수를 좌표에 대입

4-4. 설명

- 현재 사용자의 돌을 입력 받은 좌표 위치에 표시

5-1. 코드 스크린샷

```
// 5. 현재 보드 판 출력
for (int i = 0; i < numCell; i++) { // 게임 보드의 행을 나타냄 (세로)
    cout << "---|---|---" << endl; // 각 행마다 가로 선 그리기
    for (int j = 0; j < numCell; j++) {
        cout << board[i][j]; //  이중 반복 내에서 board를 출력해 놓여 있는 돌을 표시
        if (j == numCell - 1) { // 현재 열이 마지막 열이라면 ( j = 2 라면)
            break;  // 반복 멈추기
        }
        cout << "  |"; //아니면 각 열 사이에 세로 선 그리기
    }
    cout << endl;  // 행의 출력이 끝나면 줄 바꿈
}
cout << "---|---|---" << endl; // 4 번째 가로줄 (마지막 줄 추가)
```

5-2. 입력

- numCell = 가로/세로 칸 개수
- borad[i][j] = 게임 보드 내 좌표

5-3. 결과

- 각 행 마다 가로 선 그리기
- 각 열 사이에 가로선 그리기

5-4. 설명

- 현재 사용자의 돌을 입력 받은 좌표 위치에 표시
- 현재 열이 마지막 열이라면 이중 반복문 멈추기
- 행의 출력이 끝나면 줄 바꿈
- 마지막 가로줄은 반복문 밖에서 추가

6-1. 코드 스크린샷

```
// 6. 모든 칸 다 찼는지 체크
int checked = 0;  // 빈 칸을 체크해줄 변수
for (int i = 0; i < numCell; i++) {
    for (int j = 0; j < numCell; j++) { // 이중 반복으로 빈 칸을 게임 보드 크기만큼 측정
        if (board[i][j] == ' ') {  // 공백이 있는 만큼 checked 변수를 증가
            checked++;
        }
    }
}
if (checked == 0) { // 그러다 빈 칸 변수가 0이 되면 공백이 없는 것
    cout << "모든 칸이 다 찼습니다. 종료합니다.";
    break;
}
```

6-2. 입력

· int checked = 빈 칸을 체크해줄 변수

6-3. 결과

· 공백이 있는 지, 없는 지 checked 변수를 통해서 파악

· 모든 칸이 다 찼으면, 종료 문구 출력 후 break

6-4. 설명

· 이중 반복으로 게임 보드 크기만큼 반복

· if문으로 공백이 있는 지 없는 지 체크해서 checked 변수를 증가

· 만약 checked 변수가 0이면 공백이 없으므로, break

7-1. 코드 스크린샷

```
// 7. 승자 체크하기
bool win = false;


for (int i = 0; i < numCell; i++) {
    // 가로가 모두 찬 경우
    if (board[i][0] == currentUser && currentUser == board[i][1] && currentUser == board[i][2]) {
        cout << "가로에 모든 돌이 놓였습니다!: ";
        win = true;

    }
    // 세로가 모두 찬 경우
    if (board[0][i] == currentUser && currentUser == board[1][i] && currentUser == board[2][i]) {
        cout << "세로에 모두 돌이 놓였습니다!: ";
        win = true;
    }

}
// 좌상단 , 우하단 대각선으로 찬 경우
if (board[0][0] == currentUser &&
    currentUser == board[1][1] && currentUser == board[2][2]) {
    cout << "왼쪽 위에서 오른쪽 아래 대각선으로 돌이 모두 놓였습니다!: ";
    win = true;
}

// 좌하단, 우상단 대각선으로 찬 경우
if (board[0][2] == currentUser &&
    currentUser == board[1][1] && currentUser == board[2][0]) {
    cout << "오른쪽 위에서 왼쪽 아래 대각선으로 돌이 모두 놓였습니다!: ";
    win = true;
}

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

k++; // 게임 횟수 변수 증가
```

7-2. 입력

- win = 게임이 승리되면 끝내기 위한 부울 변수
- currentUser = 현재 사용자를 나타내는 변수
- board[][] : 보드 좌표

7-3. 결과

- 특정 사용자의 돌이 가로, 세로, 대각선으로 모두 찼으면 승리
- 종료 문구 출력

7-4. 설명

· 각 승리의 경우의 수를 if의 조건으로 두고, 조건에 부합하면 부울 변수 win을 true로 초기화

· 부울 변수 win이 true면 승리 문구 출력, 종료

■ **테스트**

1. 누구의 차례인지 출력

```
1번 유저(X)의 차례입니다 ->
2번 유저(O)의 차례입니다 ->
```

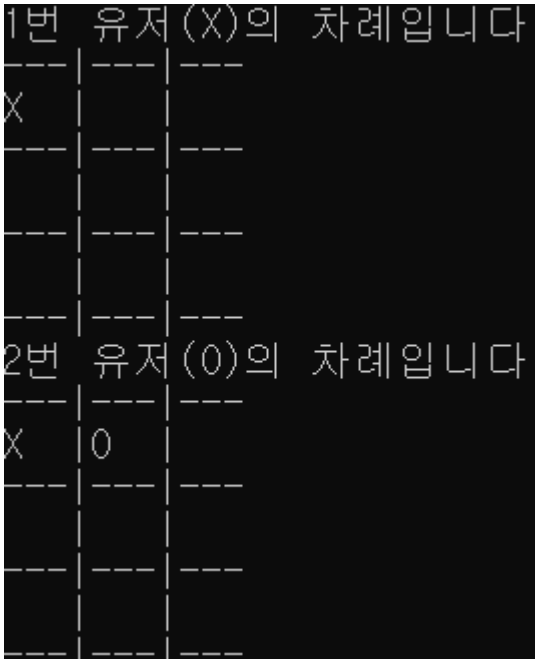
2. 각각의 사용자에게 게임 보드 좌표를 입력받기

```
> (x, y) 좌표를 입력하세요: 0 0



> (x, y) 좌표를 입력하세요: 0 1
```

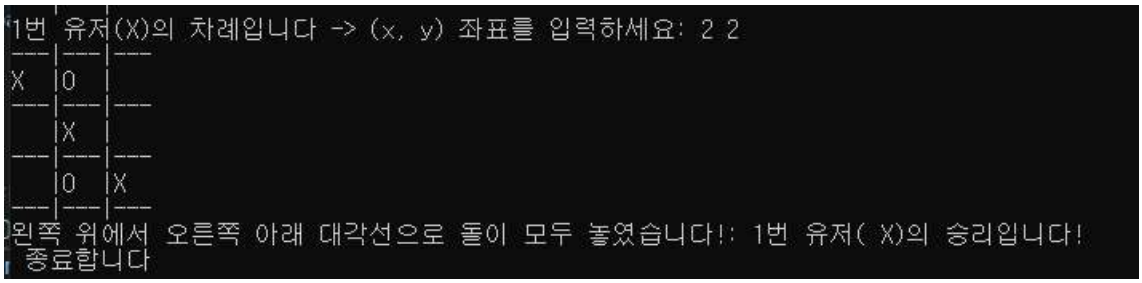
3. 좌표에 대한 유효성을 체크

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

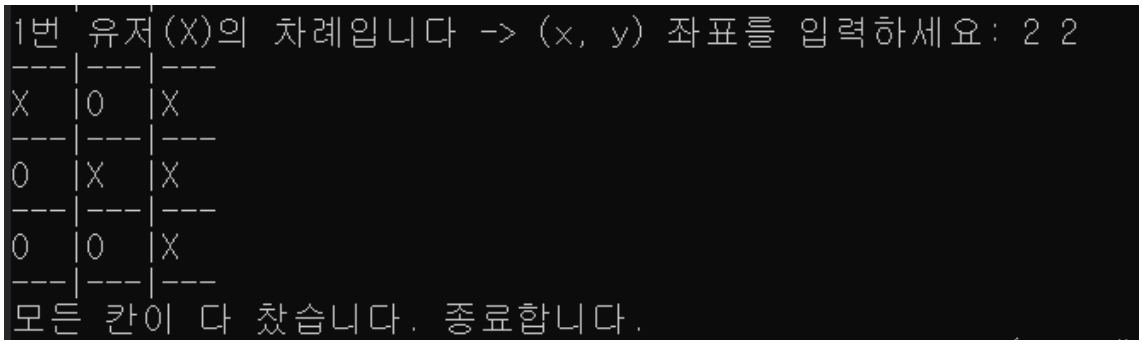
4. 좌표에 O,X를 놓기 및 현재 보드판을 출력하기



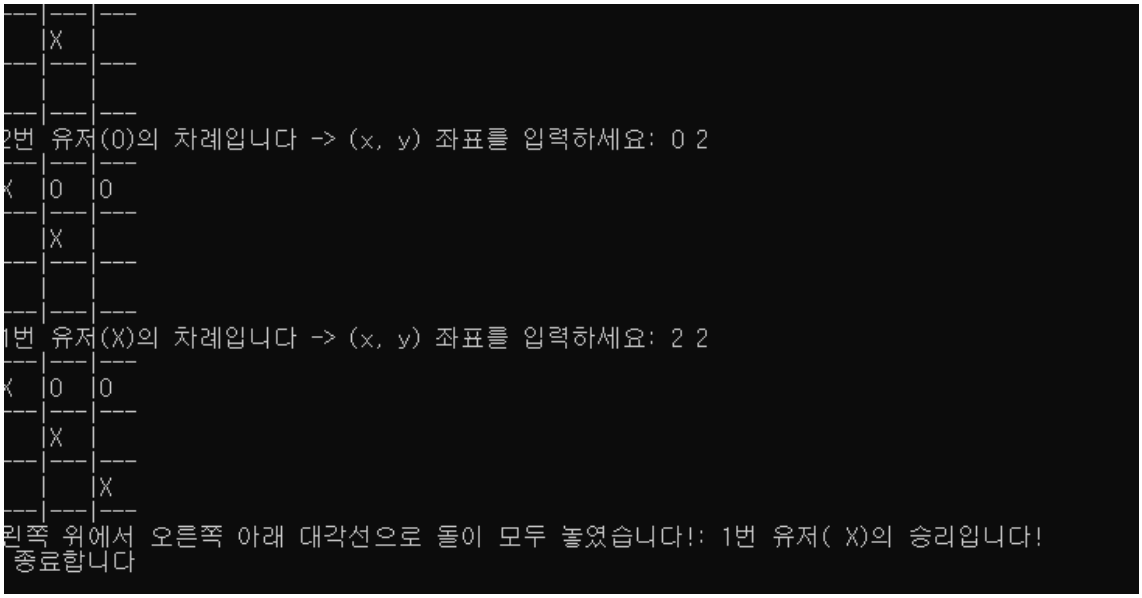
5. 어떤 사용자가 정해진 보드 크기 내에 한 줄 (가로, 세로, 대각선)이 완성되면 게임의 승리



6. 승자가 없이, 정의된 게임 보드가 모두 채워졌다면 승자 없이 종료



7. 최종 테스트



■ 결과 및 결론

4주차까지 진행했던 수업의 이해 증진을 돕기 위한 실습으로 Tic_Tac_Toe 게임을 만들었다.

주로, 2차원 배열에 대한 이해와 이중 반복의 이해도가 있어야 제작 가능한 게임이었는데, 2차원 배열의 행과 열이 각각 이중 반복문에서 어떻게 돌아가는 지를 이해해야만 이 게임이 어떤 원리로 작동하는 지를 이해 할 수 있었다.

플레이어의 턴을 차례로 교대해야 할 때, switch를 이용하여 게임 횟수를 2로 나눈 값으로 case를 정하면 간편하게 코드를 작성할 수 있는 방법도 배웠고, 특히 모든 칸이 전부 채워졌는 지를 체크하는 변수를 만들어 이중반복으로 증가시키고, 계속 반복문을 돌면서 checked 변수가 0이 되는 조건으로 모든 칸이 채워졌는 지를 알 수 있는 코드 블럭이 새로웠다.

처음에 접근할 때는, main 함수 바깥에 따로 win에 대한 함수를 만들어 호출하는 방식을 이용해보려 했는데 생각대로 잘 되지 않아서 결국 모범답을 이용하게 되었다.

아직 코딩에 익숙하지 않아, 설계가 제대로 되지 않는데 특히, currentUser가 k횟수마다 교대로 바뀌는데, 그 부분을 처음에 잘 이해하지 못해 currentUser == 'X' 같은 표현으로 if 문으로 사용해서 승리 조건을 맞추려고 한게 실패 원인 같다.