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

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 ()
- countspace = 보드판의 채워진 칸 (처음에는 0 칸 채워졌다.)

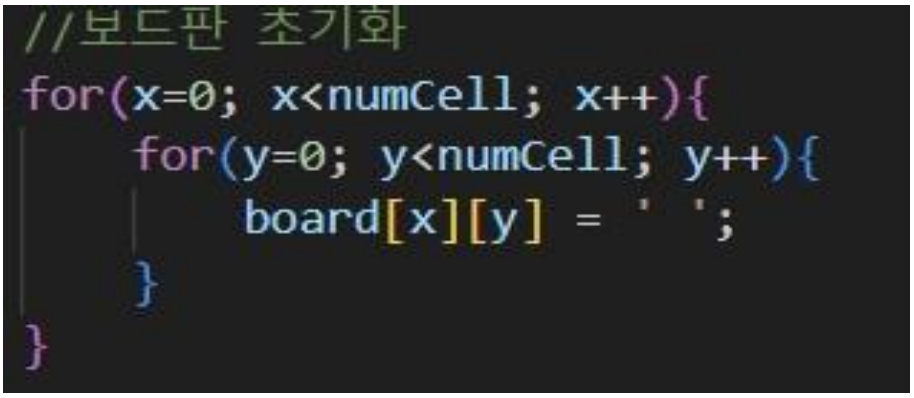
3. 결과

- 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 새는 변수로 추후 보드판이 다 채워졌는지 확인하여 게임의 종료여부를 결정할 때 사용한다.

1. 코드블록



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

```
// 게임하는 코드
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%2){
        case 0:
            cout <<"첫번째 유저(X)의 차례입니다 -> ";
            currentUser = 'X';
            break;
        case 1:
            cout <<"두번째 유저(O)의 차례입니다 -> ";
            currentUser = 'O';
            break;
    }
```

2. 입력

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

3. 결과

- 플레이어 차례를 구분, 설정해준다.

4. 설명

- K 는 플레이어의 순서를 정하는 변수로 틱택토 돌 놓기 1 회를 정상적으로 수행하면 k 가 1 오르고 k 를 플레이어 수만큼 나누어서 플레이어의 차례를 구분한다.

```
switch(k%3){
    case 0:
        cout <<"첫번째 유저(X)의 차례입니다 -> ";
        currentUser = 'X';
        break;
    case 1:
        cout <<"두번째 유저(O)의 차례입니다 -> ";
        currentUser = 'O';
        break;
    case 2:
        cout <<"세번째 유저(V)의 차례입니다 -> ";
        currentUser = 'V';
        break;
}
```

왼쪽은 플레이어가 3 명일 때의 코드로 k 를 플레이어의 수 3 으로 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 입력받음

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
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;
}
```

2. 입력

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

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

1. 코드블록

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

// 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;
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 새는 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.

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)
for(int i =0; i<numCell; i++){
    if(board[i][i] != currentUser){
        break;
    }
    if(i == numCell-1){
        cout <<"대각선 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

3. 결과

- 가로, 세로, 대각선 빙고 시 승자를 출력한다.

4. 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.

- 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.

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 새는 변수
- numCell = 보드판 한 줄의 칸 수(3)

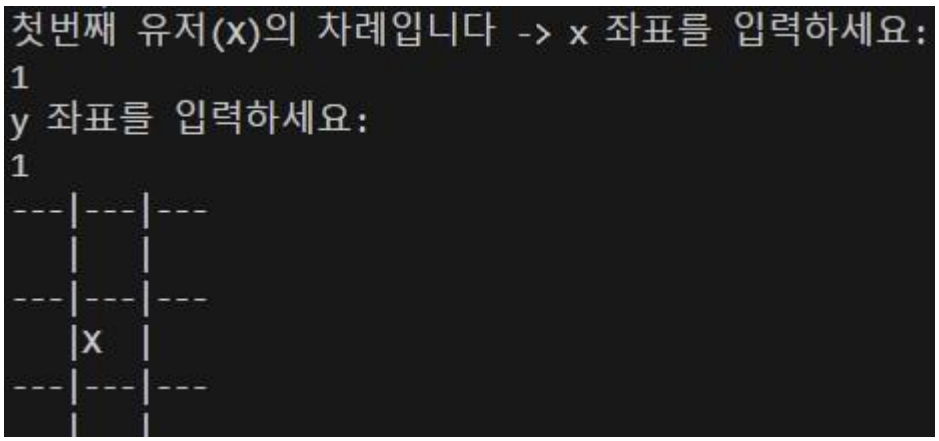
3. 결과

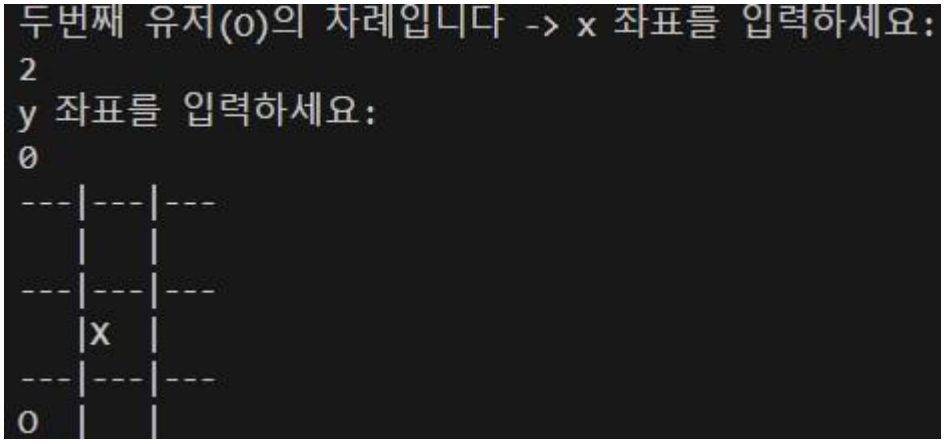
- 게임의 승자가 결정되지 않고, 빙고 칸이 전부 차면 게임을 종료한다.

4. 설명

- 보드칸의 총 개수는 numCell*numCell (3*3)이고, countspace 는 4 번 코드에서 사용자가 돌을 두고 1 씩 올라간다 즉 1 칸 채우면 countspace 도 1 올라간다 이렇게 countspacce 가 올라가 보드칸의 총 개수와 같아질 때 즉 모든 칸에 사용자가 돌을 두고, 승자가 정해지지 않으면 게임은 모든 칸이 차서 종료된다.

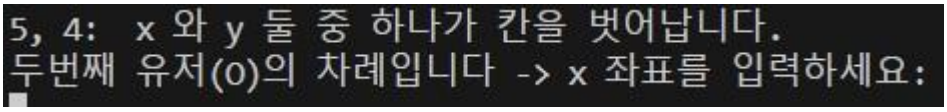
4. 테스트



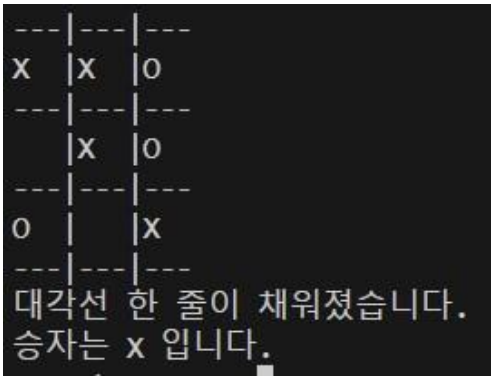


누구의 차례인지 표시되고, x, y 좌표를 입력 받고 게임판에 표시

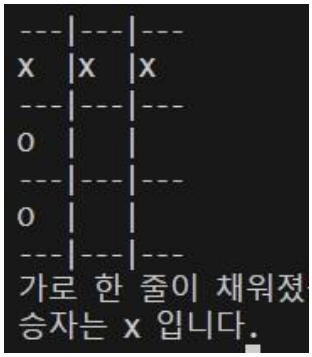
유효성 검사를 통해 돌이 놓여 있는 좌표는 둘 수 없다.



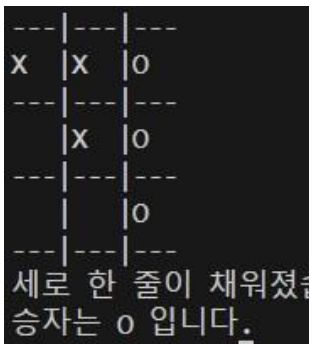
유효성 검사를 통해 게임판의 크기를 벗어나는 좌표는 입력할 수 없다.



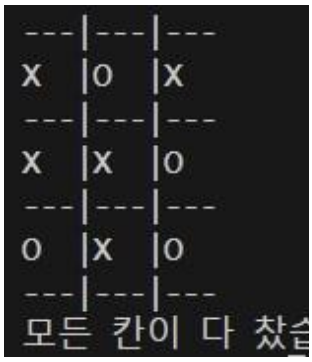
대각선 한 줄이 채워지면 승자를 표시하고 게임 종료



가로선 한 줄이 채워지면 승자를 표시하고 게임 종료

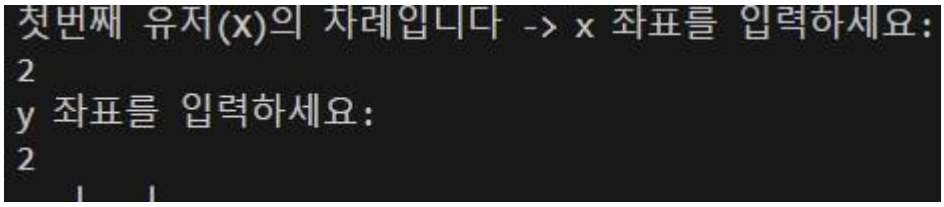


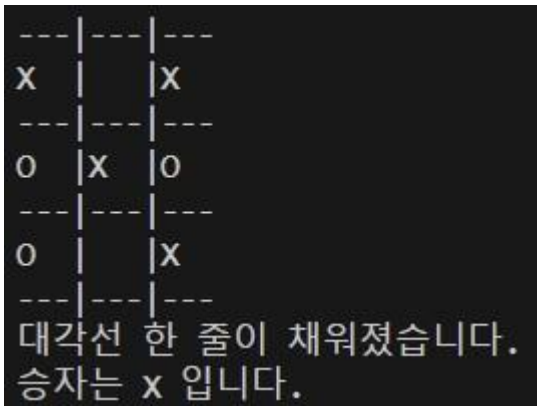
세로선 한 줄이 채워지면 승자를 표시하고 게임 종료



모든 칸이 채워지고, 승자가 정해지지 않으면 게임종료

최종 테스트 결과





결과 및 결론

프로젝트 결과: Tic Tac Toe 게임을 만들었음

2. 느낀 점: 최대한 호환성이 좋도록 만들기 위해 노력했다.