Tic Tac Toe 게임 만들기

소속: 소프트웨어공학과

학번, 이름: 203342, 조영진

1. 서론

A. 프로젝트 목적 및 배경

i. CPP 언어에 대해 배운 내용을 복습하고 실습하기 위함

B. 목표

i. Tic Tac Toe 게임 구현

2. 요구사항

A. 사용자 요구사항

i. 두 명의 사용자가 번갈아 가며 O와 X를 놓기

B. 기능 요구사항

i. 누구의 차례인지 출력

ii. 좌표 입력받기

iii. 입력 받은 좌표의 유효성 체크

iv. 좌표에 O / X 놓기

v. 현재 보드판 출력

vi. 빙고 시 승자 출력 후 종료

vii. 모든 칸이 찼으면 종료

3. 설계 및 구현

A. 기능 별 구현 사항

i. 누구의 차례인지 출력

```
// 턴 개념 구현
if (!turn) {
    cout << "첫번째 플레이어(X)의 차례입니다 -> ";
    currentPlayerStone = 'X';
} else {
    cout << "두번째 플레이어(O)의 차례입니다 -> ";
    currentPlayerStone = 'O';
}
```

turn이라는 Boolean 변수를 선언하여 두 플레이어의 턴을 출력하도록 구현. (turn = false 일 때 X의 턴) 또한 chat 변수 currentPlayerStone을 선언하여 유저의 돌을 놓을 수 있도록 함.

ii. 좌표 입력받기

```
// 좌표 입력 받기
cout << "(x, y) 좌표를 space를 기준으로 분할하여 입력하세요(시작: 0, 0) : ";
cin >> x >> y;
```

누구의 턴이든 좌표 x, y를 space를 기준으로 분할하여 입력 받도록 구현

iii. 입력 받은 좌표의 유효성 체크

```
// 좌표 유효성 체크
// 1. board의 범위 체크
if (x >= SIZEOFBOARD || y >= SIZEOFBOARD || x < 0 || y < 0) {
    cout << x << " " << y << ": ";
    cout << "x와 y중 보드의 범위를 벗어났습니다." << endl;
    continue;
}
// 2. 중복 검사
if (board[x][y] != ' ') {
    cout << x << ", " << y <<" 좌표는 이미 돌이 차있습니다." << endl;
    continue;
}
```

유효성 검사는 크게 2가지로 나뉜다.

1. Board의 범위에 벗어나지 않는지
2. 입력받은 좌표에 돌이 존재 하는지

이를 검사하기 위해 x, y의 입력값을 board의 총 크기, 0 과 비교하여 범위를 체크함. 또한 이미 좌표에 돌이 차 있는지 확인하기 위해 입력받은 위치의 값이 ' '(공백)이 아닌지 확인한다.

유효성 검사에서 유효하지 않을 경우 continue를 이용해 턴을 유지한 채 다시 입력 받도록 한다.

iv. 좌표에 O / X 놓기

```
// 유저의 돌 놓기
board[x][y] = currentPlayerStone;
```

유효성 검사가 끝나면 돌을 놓아도 되기에 board[x][y]에 현재 유저의 돌을 놓는다.

v. 현재 보드판 출력

```
// 보드판 출력
for (int i=0; i < SIZEOFBOARD; i++) {
    cout << "|---|---|---|" << endl;
    cout << "|";
    for (int j=0; j<SIZEOFBOARD; j++) {
        cout << " " << board[i][j] << " |";
    }
    cout << endl;
}
cout << "|---|---|---|" << endl;
```

유저의 돌을 놓은 후 보드판을 출력하기 위해 2차원 배열을 순회하는 이중 for 구문을 작성하였음. 또한 판의 시인성을 향상시키기 위해 |---|---|---| 처럼 가장자리에 | 을 추가하였음

vi. 빙고 시 승자 출력 후 종료

```
// 빙고 체크
// 1. 가로 체크
for (int i=0; i<SIZEOFBOARD;i++) {
    check = 0;
    for (char stone : board[i]) {
        if (stone == currentPlayerStone)
            check++;
    }
    // 가로로 빙고인 경우
    if (check == 3) {
        if (currentPlayerStone == 'X') {
            cout << "첫번째 플레이어(X)가 " << i+1 << "번째 가로 선에 모두 돌을 놓았습니다. 승리했습니다.";
            return 0;
        } else {
            cout << "두번째 플레이어(O)가 " << i+1 << "번째 가로 선에 모두 돌을 놓았습니다. 승리했습니다.";
            return 0;
        }
    }
}
```

```
// 2. 세로 체크
for (int i=0; i<SIZEOFBOARD; i++) {
    check = 0;
    for (int j=0; j<SIZEOFBOARD; j++) {
        if (board[j][i] == currentPlayerStone)
            check++;
    }
    // 세로로 빙고인 경우
    if (check == 3) {
        if (currentPlayerStone == 'X') {
            cout << "첫번째 플레이어(X)가 " << i+1 << "번째 세로 선에 모두 돌을 놓았습니다. 승리했습니다.";
            return 0;
        } else {
            cout << "두번째 플레이어(O)가 " << i+1 << "번째 세로 선에 모두 돌을 놓았습니다. 승리했습니다.";
            return 0;
        }
    }
}
check = 0;
```

```
// 3. 대각선 체크
for (int i=0; i<SIZEOFBOARD; i++) {
    // check는 좌상단 -> 우하단 빙고, check2는 우상단 -> 좌하단 빙고
    if (board[i][i] == currentPlayerStone)
        check++;
    if (board[i][SIZEOFBOARD-1-i] == currentPlayerStone)
        check2++;
}
if (check == 3) {
    if (currentPlayerStone == 'X') {
        cout << "첫번째 플레이어(X)가 좌상단 -> 우하단 선에 모두 돌을 놓았습니다. 승리했습니다.";
        return 0;
    } else {
        cout << "두번째 플레이어(O)가 좌상단 -> 우하단 선에 모두 돌을 놓았습니다. 승리했습니다.";
        return 0;
    }
}
if (check2 == 3) {
    if (currentPlayerStone == 'X') {
        cout << "첫번째 플레이어(X)가 우상단 -> 좌하단 선에 모두 돌을 놓았습니다. 승리했습니다.";
        return 0;
    } else {
        cout << "두번째 플레이어(O)가 우상단 -> 좌하단 선에 모두 돌을 놓았습니다. 승리했습니다.";
        return 0;
    }
}
```

vii. 모든 칸이 찼으면 종료

4. 테스트

A.

5. 결과 및 결론