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';
}
```

- 입력

turn	게임의 턴 개념을 위한 변수(bool) (false일 때 첫번째 플레이어의 차례)
currentPlayerStone	현재 플레이어가 놓는 돌의 모양을 표현하기 위한 변수(char)

- 결과
  - ① 플레이어의 차례를 출력
  - ② 좌표를 입력 받는 블록으로 이동
- 설명
  - ① turn(boolean) 변수를 선언하여 플레이어의 차례를 체크
  - ② 현재 플레이어가 놓을 돌의 모양을 변경

ii. 좌표 입력받기

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

- 입력

x	플레이어가 입력하는 x좌표를 저장하는 변수(int)
y	플레이어가 입력하는 y좌표를 저장하는 변수(int)

- 결과
  - ① x, y를 입력 받고 이를 x, y에 저장
- 설명
  - ① x, y를 int로 선언하고 이를 space 기준으로 분할하여 x, y에 대입.

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;
}
```

- 입력

x	플레이어가 입력한 x좌표
y	플레이어가 입력한 y좌표
SIZEOFBOARD	게임에서 사용되는 board의 크기를 선언하는 변수
board	게임에서 사용되는 게임판 (2-dim char array)

- 결과
  ① 칸을 놓을 수 없는 이유를 출력
  ② 출력 후 continue로 while구문 초반으로 이동
- 설명

유효성 검사의 2가지 유형

① board의 범위를 벗어나지 않는지를 확인하기 위해 SIZEOFBOARD 변수와 입력 값을 비교하는 if 구문 또 0보다 작은 값이 입력될 때 indexError 를 방지하기 위한 조건문

② 플레이어가 입력한 좌표 x, y에 이미 돌이 있는 경우를 확인하기 위한 if 구문

iv. 좌표에 O / X 놓기

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

- 입력

board	게임에서 사용되는 게임판 (2-dim char array)
currentPlayerStone	현재 플레이어가 놓는 돌의 모양을 표현

	하기 위한 변수(char)

- 결과
  - ① board[x][y]에 currentPlayerStone의 값을 대입함
- 설명
  - ① 2-dim char array board에 crrentPlayerStone(char) 변수의 값을 대입

v. 현재 보드판 출력

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

- 입력

SIZEOFBOARD	게임에서 사용되는 board의 크기를 선언하는 변수
board	게임에서 사용되는 게임판 (2-dim char array)

- 결과
  - ① 보드의 모양이 SIZEOFBOARD의 크기에 맞게 터미널에 출력됨
- 설명
  - ① 상단, 하단의 출력을 위해 SIZEOFBOARD 크기만큼 for 구문을 돌려 출력
  - ② 2-dim char array 출력을 위해 이중 for 구문으로 출력

vi. 빙고 시 승자 출력 후 종료

```
// 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 == SIZEOFBOARD) {
        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 == SIZEOFBOARD) {
        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 == SIZEOFBOARD) {
    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 == SIZEOFBOARD) {
    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;
    }
}
```

- 입력

check	가로, 세로, 좌->우 대각선 빙고를 판별하기 위한 변수(int)
check2	우->좌 대각선 빙고를 판별하기 위한 변수(int)
SIZEOFBOARD	게임에서 사용되는 board의 크기를 선언하는 변수
board	게임에서 사용되는 게임판 (2-dim char array)
currentPlayerStone	현재 플레이어가 놓는 돌의 모양을 표현하기 위한 변수(char)

- 결과
  ① 빙고를 완성한 경우 승리 메시지를 출력
  ② 출력 후 return으로 프로그램을 종료
- 설명
  ① 빙고는 3종류로 나눌 수 있다. 가로, 세로, 대각선
  ② 가로, 세로 빙고를 검사하기 위해서는 2-dim array를 모두 순회해야 하지만 대각선은 선형적으로 증가, 감소하기 때문에 2-dim array를 모두 순회하지 않아도 됨.
  ③ 따라서 가로, 세로 빙고는 2중 for구문, 대각선은 1중 for 구문을 사용하여 구현.
  ④ tic-tac-toe 게임 자체의 판이 크지 않기 때문에 반복적인 for 구문을 사용함.

vii. 모든 칸이 찼으면 종료

```
// 모든 칸이 찼는지 확인
check = 0;
for (int i=0; i<SIZEOFBOARD;i++)
    for (int j=0;j<SIZEOFBOARD;j++)
        if (board[i][j] != ' ')
            check++;
if (check == SIZEOFBOARD*SIZEOFBOARD) {
    cout << "모든 칸이 다 찼습니다. 무승부로 게임을 종료합니다.";
    return 0;
} else {
    // 턴 넘기기: bool을 보수화하면서 턴을 변경
    turn = !turn;
}
```

- 입력

check	모든 칸이 다 찼는지 확인하기 위한 변수(int)
turn	턴 개념을 구현하기 위한 변수(bool)
SIZEOFBOARD	게임에서 사용되는 board의 크기를 선언하는 변수

- 결과
  ① 모든 칸이 다 찬 경우, 무승부라는 메시지를 출력 후 프로그램을 종료
  ② 그렇지 않은 경우 다음 턴으로 넘어감.
- 설명
  ① 2-dim array를 순회하며 비워져 있지 않은 경우 check변수에 1씩 더함
  ② 모두 순회 후 check가 한 변의 길이가 SIZEOFBOARD인 정사각형의 넓이와 같을 때 (= SIZEOFBOARD^2) 무승부를 출력 후 종료
  ③ 칸이 다 차지 않은 경우 turn(bool)을 보수화해 다음 턴으로 넘어가도록 실행

4. 테스트

A. 기능 별 테스트 결과

i. 누구의 차례인지 출력

```
첫번째 플레이어(X)의 차례입니다 ->
```

ii. 좌표 입력받기

```
(x, y) 좌표를 space를 기준으로 분할하여 입력하세요(시작: 0, 0) : 0 0
```

iii. 입력 받은 좌표의 유효성 체크

```
두번째 플레이어(O)의 차례입니다 -> (x, y) 좌표를 space를 기준으로 분할하여 입력하세요(시작: 0,
0, 0 좌표는 이미 돌이 차있습니다.
두번째 플레이어(O)의 차례입니다 -> (x, y) 좌표를 space를 기준으로 분할하여 입력하세요(시작: 0,
10 10: x와 y중 보드의 범위를 벗어났습니다.
```

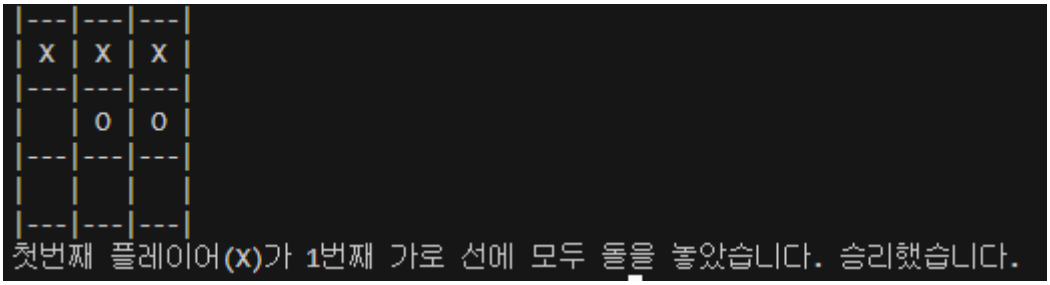
iv. 좌표에 O / X 놓기

```
두번째 플레이어(O)의 차례입니다 -> (x, y) 좌표를 space를 기준으로 분할하여 입력하세요(시작: 0
|---|---|---|
| X |   |   |
|---|---|---|
|   | O |   |
|---|---|---|
|   |   |   |
|---|---|---|
```

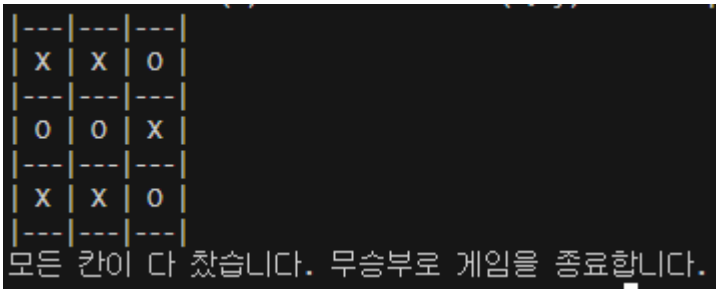
v. 현재 보드판 출력

```
|---|---|---|
| X | X |   |
|---|---|---|
|   | O |   |
|---|---|---|
|   |   |   |
|---|---|---|
```

vi. 빙고 시 승자 출력 후 종료



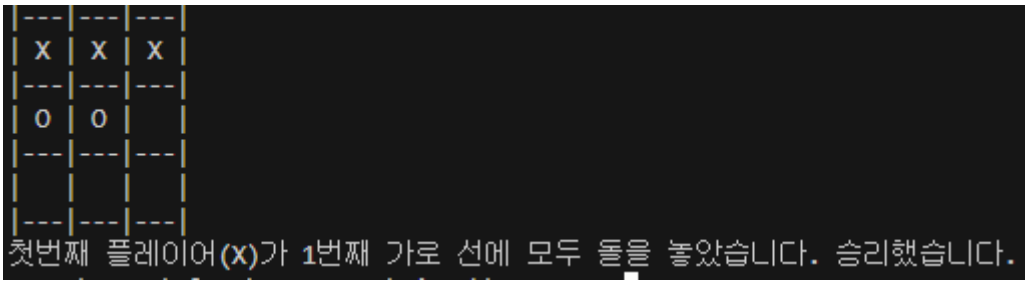
vii. 모든 칸이 찼으면 종료



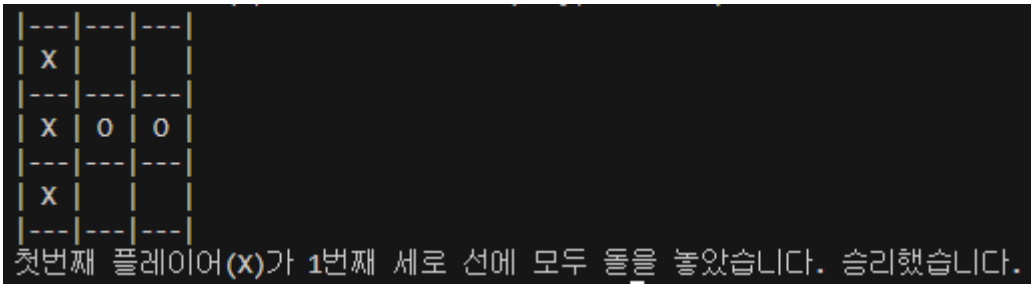
B. 최종 테스트 스크린샷

i. 첫번째 플레이어(X)의 승리

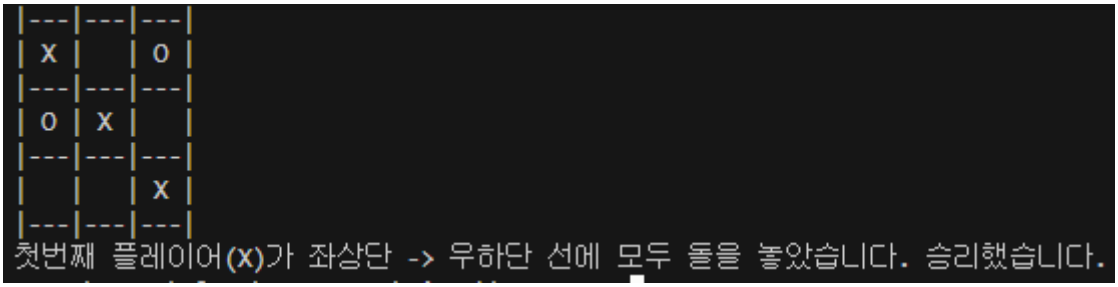
- 가로 빙고 승리



- 세로 빙고 승리

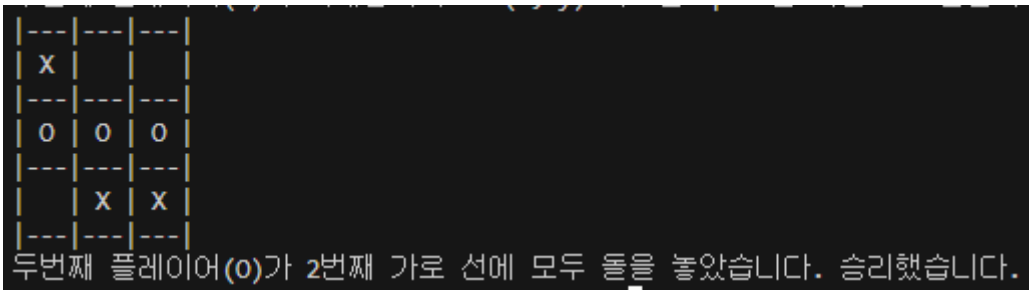


- 대각선 빙고 승리

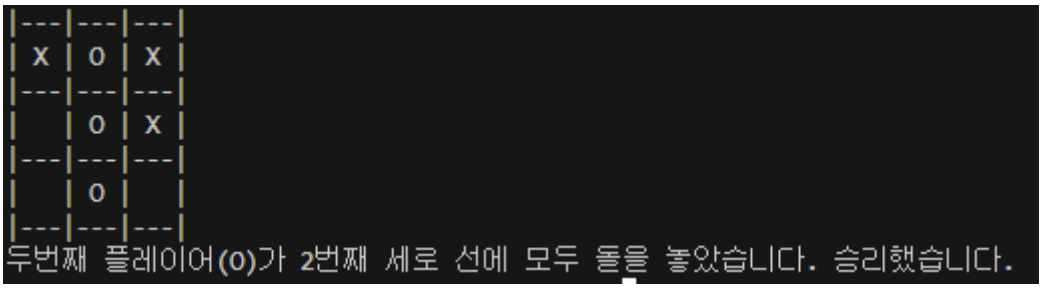


ii. 두번째 플레이어(O)의 승리

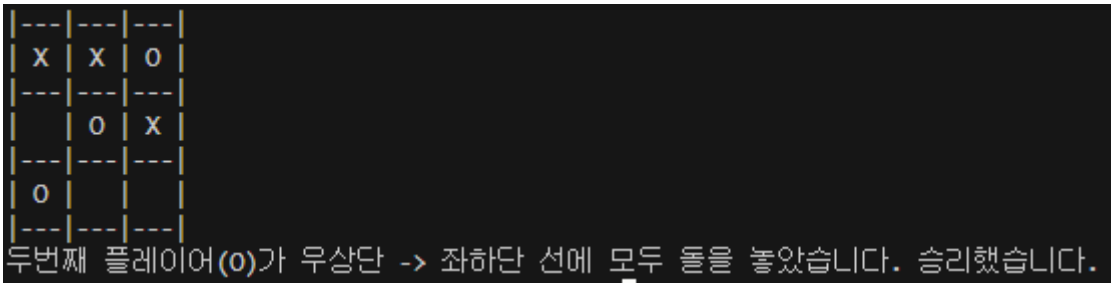
- 가로 빙고 승리



- 세로 빙고 승리

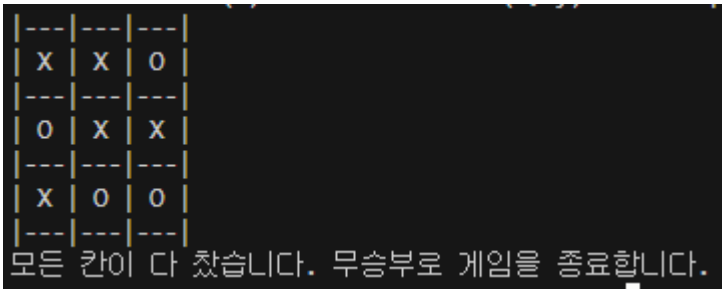


- 대각선 빙고 승리



iii. 무승부

- 무승부



5. 결과 및 결론

A. 프로젝트 결과

i. Tic Tac Toe 게임 제작 완료

ii. 게임판의 확장을 생각하여 SIZEOFBOARD 변수를 적극 활용함

B. 느낀 점

i. 조건 분기 설정이 생각보다 까다로움.

ii. 빙고 조건의 for 구문을 합칠 수 있을 것으로 보임. 추후 리펙터링 예정

iii. 확장성을 고려한 프로그래밍에 대한 고민이 필요해보임.