Tic Tac Toe 보고서_192195 박경준

1. 서론

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

2. 요구사항

1. 사용자 요구사항 : 두 명의 사용자가 번갈아가며 O와 X를 놓기
2. 기능 요구사항
    a. 어떤 플레이어의 차례인지 출력
    b. 좌표 입력 받기
    c. 입력 받은 좌표 유효성 체크
    d. 좌표에 O / X 놓기(자리 선정)
    e. 현재 보드판 출력
    f. 모든 칸이 찼으면 종료
    g. 빙고 시 승자 출력 후 종료

3. 설계 및 구현

- 기능 별 구현 사항

```
#include <iostream>
using namespace std;

int main() {
  const int numCell = 3;

  // {}를 사용하여 board 배열 모든 칸을 0이나 null로 초기화시킴
  char board[numCell][numCell]{};
  int x, y; // 사용자에게 입력받는 x,y좌표를 저장할 변수

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

  // 게임하는 코드
  int k = 0; // 차례를 체크하기 위한 변수
  char currentUser = 'X'; // 현재 유저의 돌을 저장하기 위한 문자 변수
  while (true) {
    // 1. 누구의 차례인지 출력하는 코드
    // 2명이서 게임을 진행하기 때문에 홀수, 짝수 순번을 표현하기 위해 k % 2로 설정
    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(0)의 차례입니다 -> ";
      currentUser = 'O';
      break;
```

1. 어떤 플레이어의 차례인지 출력
2. 입력
    a. x, y는 사용자에게 입력 받는 좌표를 저장할 변수
    b. k는 차례를 체크하는 변수
    c. currentUser는 현재 유저의 돌을 저장하는 변수
3. 결과
    a. 게임 보드판을 초기화
    b. 유저 차례를 번갈아가며 출력
4. 설명
    a. 코드 설명의 이해를 높이기 위해 차례를 출력하는 코드 윗 부분도 추가
    b. 보드판을 3x3 형식 (numCell = 3)으로 구성
    c. k % 2 연산을 통해 두 명의 플레이어가 각각 X, O로 차례를 구분한다. k 값이 짝수이면 X, 홀수이면 O의 차례
    d. currentUser는 차례마다의 X 또는 O로 설정
    e. switch문을 사용하여 연산 값에 따른 차례를 출력

2. 좌표 입력 받기

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

1. 좌표 입력 받기
2. 입력
    a. x, y는 좌표를 입력 받기 위한 변수
3. 결과
    a. 입력한 x, y 좌표가 각각의 변수에 저장된다.
4. 설명
    a. x, y 좌표는 보드판 어느 부분에 위치시킬 것인가의 좌표를 설정하기 위한 과정

```
// 3. 입력받은 좌표의 유효성 체크
    // 첫번째 조건은 입력받은 x 또는 y좌표가 설정한 칸에서 벗어난 경우
    if (x >= numCell or y >= numCell) {
      cout << x << "; " << y << ": ";
      cout << "x와 y 둘 중 하나가 칸을 벗어납니다." << endl;
      continue;
    }
// 아래 조건은 빈 공간이 아니라는 의미로 돌이 있다면 아래 문장 출력
    if (board[x][y] != ' ') {
      cout << x << ", " << y << ": 이미 돌이 차있습니다." << endl;
      continue;
    }
```

1. 입력 받은 좌표 유효성 체크
2. 입력
    a. x, y는 각각 x, y 좌표에 해당
    b. numCell은 보드판의 가로 세로 칸의 개수에 해당
3. 결과
    a. 유효성을 체크해서 칸을 놓지 못하는 이유 출력
    b. 만약 조건에 해당하면 출력 후 while문 초반으로 이동
4. 설명
    a. if문을 사용하여 위에서 설정한 x 좌표 또는 y 좌표가 벗어난다면 설정한 문장 출력하고 continue를 진행
    b. if문을 사용하여 이미 채워져 있는 공간에 다시 값을 입력할 경우에도 설정한 문장 출력하고 continue 진행

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

1. 좌표에 O / X놓기(자리선정)
2. 입력
    a. x, y는 각각의 좌표 값에 해당
3. 결과
    a. 사용자로부터 입력받은 좌표에 현재 유저의 돌로 설정
4. 설명
    a. switch문을 통해 설정된 currentUser(현재 유저)가 보드 상의 해당 좌표에돌을 배치하는 역할에 해당하는 코드

```
// 5. 현재 보드 판 출력
    for (int i = 0; i < numCell; i++) {
      // 가로 구분선 출력
      cout << "---|---|---" << endl;
      for (int j = 0; j < numCell; j++) {
        cout << board[i][j];
        // 열 사이의 구분 출력
        if (j == numCell - 1) {
          break;
        }
        cout << "  |";
      }
      cout << endl;
    }
    // 행의 끝에 가로 구분선 출력
    cout << "---|---|---" << endl;
    // 게임 턴의 수를 증가시켜 다음 플레이어 차례로 이동
    k++;
```

1. 현재 보드판 출력
2. 입력
3. 결과
    a. 이중 반복문을 사용하여 보드판의 가로선과 세로선이 화면에 출력
    b. k++를 통해서 다음 플레이어의 차례로 이동
4. 설명

a. 바깥 for문을 사용하여 먼저 보드판의 가로 구분선을 출력

b. 내부 for문을 사용하여 현재 행의 각 열에 있는 돌 출력

c. 내부 for문을 사용하여 열 사이에 열 구분선 ' | ' 출력

d. 이중 for문이 끝난 뒤 행의 끝에 가로 구분선을 출력하여 보드판 시각화 마무리

e. swtich문에서 설정한 연산을 통해 다음 플레이어로 차례를 넘기기 위해 k의 값을 증가

```
// 6. 모든 칸이 찼으면 종료 및 체크
    bool draw = true; // 무승부일 경우의 불리안변수를 draw로 설정
    // 모든 칸을 확인했을 때 빈 칸이 존재하면 종료하지 않음
    for (int i = 0; i < numCell; i++) {
      for (int j = 0; j < numCell; j++) {
        if (board[i][j] == ' ') {
          draw = false;
        }
      }
    }
    // draw가 true라면 모든 칸이 다찼다는 의미으로 무승부이자 게임이 종료된다
    if (draw == true) {
      cout << "모든 칸이 다 찼습니다. 종료합니다" << endl;
      break;
    }
```

1. 모든 칸이 찼으면 종료
2. 입력
    a. draw는 무승부를 판별하기 위한 불리안 변수 설정
3. 결과
    a. 보드판에 있는 모든 칸을 확인해서 빈 칸이 존재하면 if문을 통해 draw가 false로 설정되어 종료하지 않음
    b. 빈 칸이 존재하지 않으면 draw가 true로 유지되어 설정한 문장 출력
4. 설명
    a. 모든 칸이 차서 종료가 된다는 말은 곧 무승부라는 결과를 얻었다는 것을 통해 draw라는 불리안 변수를 설정해서 true로 초기 설정
    b. 모든 칸을 확인했을 때 빈 칸이 존재하면 게임이 끝나지 않은 경우이므로 draw를 false로 설정한다. 만약에 draw가 true가 된다면 모든 칸이 찬 경우이므로 설정한 문장을 출력하고 break를 하여 종료

```
// 7. 빙고시 승자 출력 (가로,세로, 대각선)
    bool isWin = false; // 게임 종료를 위해 불리안 변수 설정
    char winner = ' '; // 승자를 확인하기 위한 변수
    // 승리할 경우(가로 및 세로) 확인
    for (int i = 0; i < numCell; i++) {
      // 가로로 승리할 경우
      if (board[i][0] == currentUser and board[i][1] == currentUser and board[i][2]) {
        winner = currentUser;
        cout << "가로에 모두 돌이 놓였습니다!: " << k % 2 << "번 유저(" << currentUser << ")의 승리입니다!" << endl;
        isWin = true;
      }
      // 세로로 승리할경우
      if (board[0][i] == currentUser and board[1][i] == currentUser and board[2][i] == currentUser) {
        winner = currentUser;
        cout << "세로에 모두 돌이 놓였습니다!: " << k % 2 << "번 유저(" << currentUser << ")의 승리입니다!" << endl;
        isWin = true;
      }
    }
    // 대각선 경우로 승리할 경우 확인
    // 오른쪽으로 하강하는 대각선 경우
    if (board[0][0] == currentUser and board[1][1] == currentUser and board[2][2] == currentUser) {
      winner = currentUser;
      cout << "오른쪽 위에서 왼쪽 아래 대각선으로 모두 돌이 놓였습니다!: " << k % 2 << "번 유저(" << currentUser << ")의 승리입니다!" << endl;
```

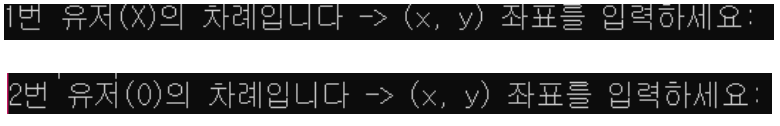
```
      isWin = true;
    }
    // 왼쪽으로 하강하는 대각선 경우
    if (board[0][2] == currentUser and board[1][1] == currentUser and board[2][0] == currentUser) {
      winner = currentUser;
      cout << "왼쪽 위에서 오른쪽 아래 대각선으로 모두 돌이 놓였습니다!: " << k % 2 << "번 유저(" << currentUser << ")의 승리입니다!" << endl;
      isWin = true;
    }

    // 위에서 설정한 종료에 관한 불리안 변수를 사용하여 승자 표시와 게임 종료
    if (isWin == true) {
      cout << (k % 2 == 0 ? 2 : 1) << "번 유저(" << currentUser << ")의 승리입니다!" << endl;
      cout << "종료합니다" << endl;
      break;
    }
```

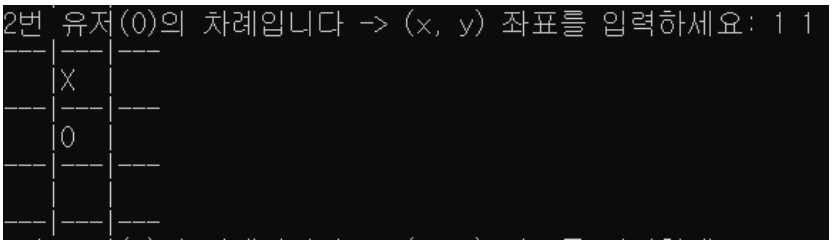
1. 빙고 시 승자 출력 후 종료
2. 입력
   a. 승자가 발생하여 종료하기 위한 불리안 변수 isWin 설정
   b. 승자를 출력하기 위한 문자 변수 winner 설정
3. 결과
   a. 반복문을 통해 보드판을 체크
   b. 조건문을 사용하여 가로, 세로 및 대각선로 빙고일 경우 체크
   c. 조건문을 통해 만약 한 유저가 승리할 경우 isWin 변수를 true로 설정하여 종료하기 위한 문장 및 break를 통한 종료 진행
4. 설명
   a. for 반복문과 if 조건문, and 연산자(모두 만족해야하는 경우이므로 사용)를 사용하여 가로 및 세로 방향 승리 여부를 확인
      i. 가로 : 같은 유저의 돌이 가로 줄에 3개 연속으로 놓인 경우이므로 반복문에서 [i][0~2]로 설정
      ii. 세로 : 같은 유저의 돌이 세로 줄에 3개 연속으로 놓인 경우이므로 반복문에서 [0~2][i]로 설정
   b. if조건문과 and 연산자(모두 만족해야하는 경우이므로 사용)를 통하여 대각선 방향 승리 여부를 확인
      i. 오른쪽으로 하강하는 경우 : 반복문을 통해 체크할 수 없으므로 해당하는 좌표 값을 설정
      ii. 왼쪽으로 하강하는 경우 : 반복문을 통해 체크할 수 없으므로 해당하는 좌표 값을 설정
   c. 승자가 발생하면 isWin이 true로 설정하는 코드를 각각의 승리 조건 마지막에 작성하여 활성화시켜 해당 유저의 승리를 출력하고 게임을 종료

4. 테스트

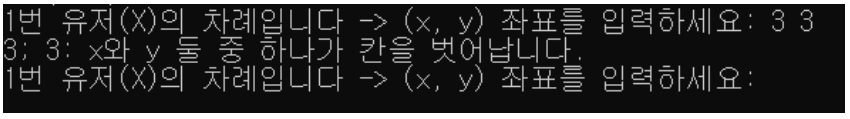
- 기능 별 테스트 결과 : (요구사항 별 스크린샷)
  - 어떤 플레이어의 차례인지 출력

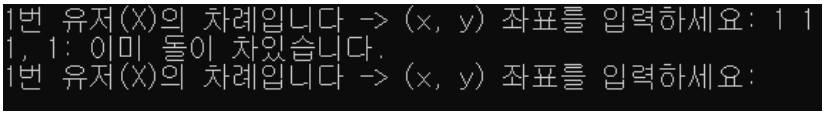


  - 좌표 입력 받기 & 현재 보드판 출력 & 좌표에 O / X 놓기

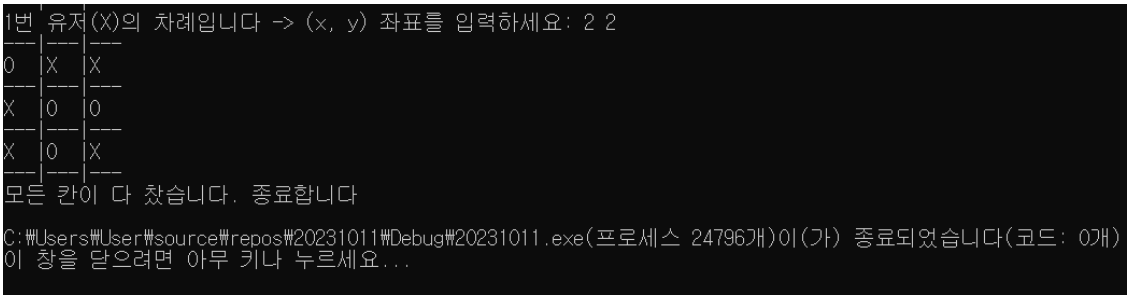


- 입력 받은 좌표 유효성 체크

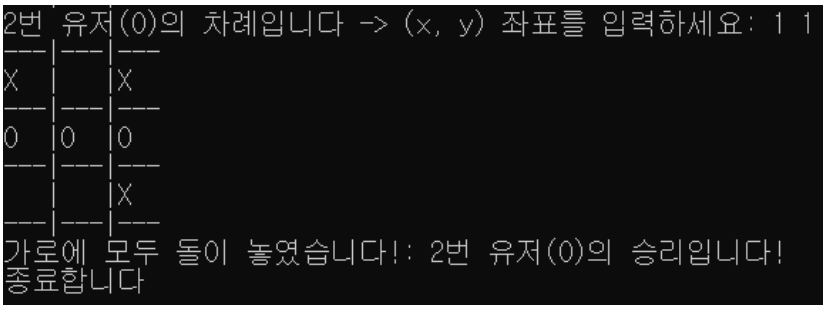




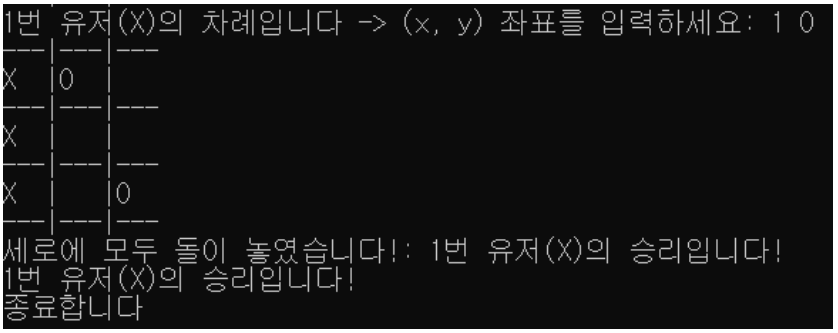
- 모든 칸이 찼으면 종료



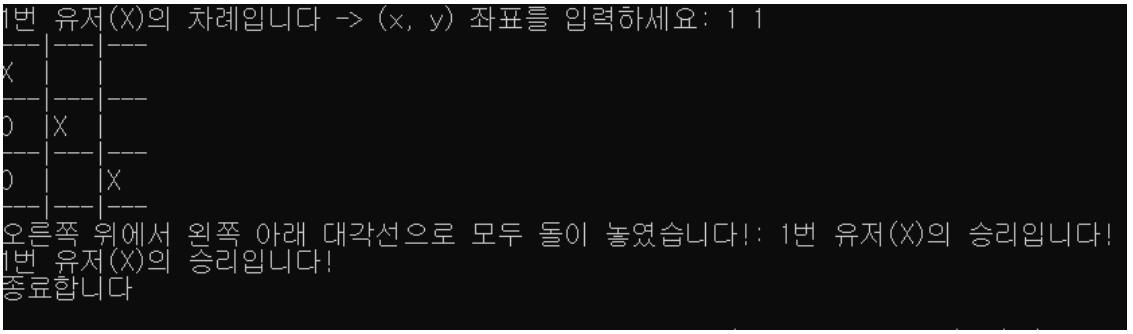
- 빙고 시 승자 출력 후 종료
  - 가로



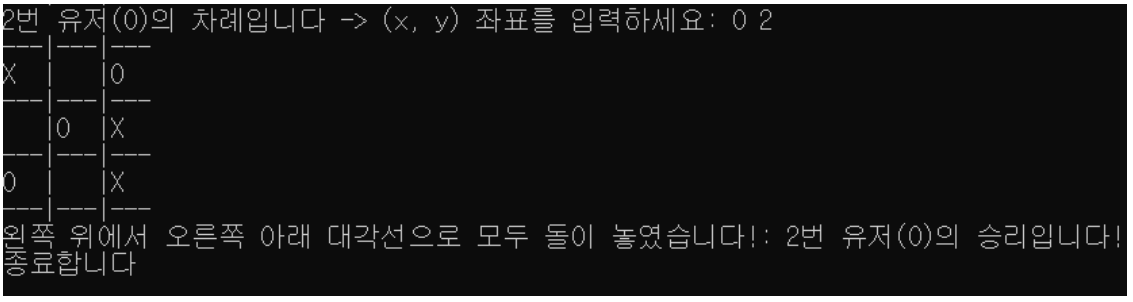
- 세로



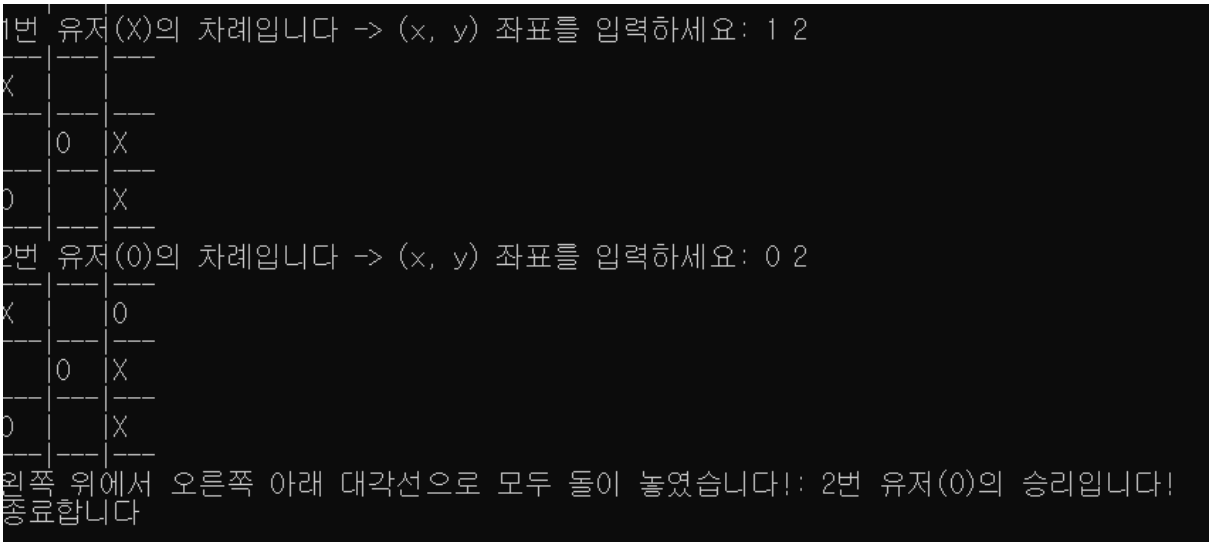
- 대각선_1



- 대각선_2



- 최종 테스트 스크린샷 : (프로그램 전체 동작 스크린샷)



5. 결과 및 결론

1. 프로젝트 결과 : C++ 언어를 통해 Tic Tac Toe 게임을 구현
2. 느낀 점

단순히 개념을 공부한 것에 벗어나 프로젝트를 직접 진행해보니 코드를 논리적으로 작성하는 것이 중요한 것을 느꼈다. 언어에 대한 문법만 가지고는 프로젝트를 구현하는 것에는 한계가 있었다. 특히 게임에서의 승자를 파악하는 조건과 승자가 결정되지 않고 무승부로 결정이 날 경우 두 가지 경우를 고려하는 것이 생각보다 쉽지 않았다. 또한 코드로 구현하는데 있어서 이해가 어려웠던 부분은 보드판을 시각화하는 코드였다. 이중 반복문을 사용하는 것 까지는 이해하였지만 이후 부분에 대한 이해에 시간을 투자하였다. 논리적인 요소의 중요성을 느낀 프로젝트 실습이었다.