CPP 프로젝트

214771 정철원

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

① 누구의 차례인지 출력
② 좌표 입력 받기
③ 입력 받은 좌표 유효성 체크
④ 좌표에 O / X 놓기
⑤ 현재 보드판 출력
⑥ 빙고 시 승자 출력 후 종료
⑦ 모든 칸이 찼으면 종료

3. 설계 및 구현

```
// 1. 누구의 차례인지 출력
// 2. 좌표 입력 받기
cout << "(" << currentUser << ") 차례입니다. 좌표를 입력하세요 (x y): ";
cin >> x >> y;
```

입력

• x= 좌표 x 값
• y= 좌표 y 값
• currentUser = 현재 유저값을 저장함

결과

• 누구 차례인지 출력하고 좌표를 입력할 수 있음

설명

• cout으로 차례를 출력하고 cin으로 x y 좌표값을 입력받음

```
// 3. 입력 받은 좌표 유효성 체크
if (x < 0 || x >= numCell || y < 0 || y >= numCell || board[x][y] != ' ') {
    cout << "잘못된 입력입니다. 다시 시도하세요." << endl;
    continue;
}
```

입력

- x= 좌표 x 값
- y= 좌표 y 값
- numCell = 가로/세로 칸 개수
- board[x][y] = 보드판

결과

- 잘못된 값이 입력될 때, 다시 시도하라고 출력해줌
- 출력 후 while문 초반으로 이동

설명

- if로 사용자가 입력한 좌표가 게임 판을 벗어나는지 체크

```
// 4. 보드에 돌을 놓기
board[x][y] = currentUser;
```

입력

- currentUser = 현재 유저값을 저장함
- board[x][y] = 보드판

결과

- 현재 유저값이 보드에 저장됨, 추후 보드에 돌이 어디에 놓아지는지 출력됨

설명

- 대입 연산자로 현재 유저값을 보드에 저장함

```
// 5. 현재 보드판 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) cout << " |";
    }
    cout << endl;
}
cout << "---|---|---" << endl;
```

입력

- numCell = 가로/세로 보드판 개수
- board[x][y] = 보드판

결과

• 현재 보드판이 출력됨

설명

• 이중 반복문을 통하여 ---|---|---로 틀을 만들고 조건을 추가하여 현재 보드판을 출력해줌 endl을 추가하여3*3의 보드판의 형태를 만들고 for 문을 나가고 cout을 추가하여 틀을 완성함

```
// 승리 조건 확인
bool a = false;

// 가로 확인
for (int i = 0; i < numCell; i++) {
    if (board[x][i] != currentUser) {
        break;
    }
    if (i == numCell - 1) {
        a = true;
    }
}
```

입력

• bool a = 승리 여부를 저장하는 변수

• board[x][y] = 보드판

• numCell = 가로/세로 보드판 개수

• currentUser = 현재 유저값을 저장함

• x = 현재 플레이어가 돌을 놓은 행

결과

• 현재 플레이어가 가로로 3개를 연속으로 놓았는지 확인함

설명

• for 루프는 해당 행의 모든 칸(board[x][i])을 순서대로 확인함

• 만약 하나라도 현재 플레이어의 돌이 아닌 것이 있다면 board[x][i] != currentUser, break로 루프를 종료함

• 행의 모든 칸이 currentUser의 돌이면 마지막에 a= true로 설정되어 승리로 간주됨

```
// 세로 확인
for (int i = 0; i < numCell; i++) {
    if (board[i][y] != currentUser) {
        break;
    }
    if (i == numCell - 1) {
        a = true;
    }
}
```

입력

• a = 승리 여부를 저장하는 변수

• board[x][y] = 보드판

• numCell = 가로/세로 보드판 개수

• currentUser = 현재 유저값을 저장함

• y = 현재 플레이어가 돌을 놓은 열

결과

• 현재 플레이어가 세로로 3개를 연속으로 놓았는지 확인함

설명

• for 루프는 해당 열의 모든 칸(board[i][y])을 순서대로 확인함

• 만약 하나라도 현재 플레이어의 돌이 아닌 것이 있다면 board[i][y] != currentUser, break로 루프를 종료함

• 열의 모든 칸이 currentUser의 돌이면 마지막에 a= true로 설정되어 승리로 간주됨

```
// 대각선 확인 (왼쪽 위에서 오른쪽 아래로)
if (x == y) {
    for (int i = 0; i < numCell; i++) {
        if (board[i][i] != currentUser) {
            break;
        }
        if (i == numCell - 1) {
            a = true;
        }
    }
}
```

입력

• a = 승리 여부를 저장하는 변수
• board[x][y] = 보드판
• numCell = 가로/세로 보드판 개수
• currentUser = 현재 유저값을 저장함
• x = 현재 플레이어가 돌을 놓은 행
• y = 현재 플레이어가 돌을 놓은 열

결과

• 대각선 중에서 왼쪽 위에서 오른쪽 아래로 이어지는 대각선에 현재 플레이어의 돌이 놓였는지를 확인함

설명

• 이 대각선은 (0,0), (1,1), (2,2)의 좌표로 이루어지며 x와 y가 같은 위치일 때만 대각선에 해당함 (x == y)
• for 루프는 board[i][i]의 값을 확인하고, 모든 칸에 현재 플레이어의 돌이 놓였으면 a = true로 승리 조건을 만족함

```
// 대각선 확인 (오른쪽 위에서 왼쪽 아래로)
if (x + y == numCell - 1) {
    for (int i = 0; i < numCell; i++) {
        if (board[i][(numCell - 1) - i] != currentUser) {
            break;
        }
        if (i == numCell - 1) {
            a = true;
        }
    }
}
```

입력

• a = 승리 여부를 저장하는 변수
• board[x][y] = 보드판
• numCell = 가로/세로 보드판 개수
• currentUser = 현재 유저값을 저장함
• x = 현재 플레이어가 돌을 놓은 행
• y = 현재 플레이어가 돌을 놓은 열

결과

• 대각선 중에서 오른쪽 위에서 왼쪽 아래로 이어지는 대각선에 현재 플레이어의 돌이 놓였는지를 확인함

설명

• 이 대각선은 (0,2), (1,1), (2,0)의 좌표로 이루어지며 x + y == numCell - 1일 때 대각선에 해당함
• for 루프는 board[i][(numCell - 1) - i]의 값을 확인하고, 모든 칸에 현재 플레이어의 돌이 놓였으면 a = true로 설정됨

```
// 6. 빙고 시 승자 출력 후 종료
if (a) {
    for (int i = 0; i < numCell; i++) {
        cout << "---|---|---" << endl;
        for (int j = 0; j < numCell; j++) {
            cout << " " << board[i][j];
            if (j < numCell - 1) cout << " |";
        }
        cout << endl;
    }
    cout << "---|---|---" << endl;

    cout << currentUser << "가 승리했습니다!" << endl;
    break;
}
```

입력
• a = 승리 여부를 저장하는 변수
• board[x][y] = 보드판
• numCell = 가로/세로 보드판 개수
• currentUser = 현재 유저값을 저장함
결과
• 현재 플레이어가 가로,세로,대각선 중 하나에서 이겼을 때 승리자를 출력함
설명
• a가 true인 경우, 현재 보드 상태를 출력함
• currentUser의 승리 메시지를 출력함
• break로 while 루프를 빠져나와 게임을 종료함

```
// 7. 모든 칸이 찼으면 종료
bool b = true;
for (int i = 0; i < numCell; i++) {
    for (int j = 0; j < numCell; j++) {
        if (board[i][j] == ' ') {
            b = false;
            break;
        }
    }
    if (!b) {
        break;
    }
}
```

입력

- b = 보드가 가득 찼는지 여부를 저장하는 변수
- board[x][y] = 보드판
- numCell = 가로/세로 보드판 개수

결과

- 빈 칸이 있는지 확인하면서 보드의 모든 칸을 확인함

설명

- 이중 for문을 사용하여 보드의 모든 칸을 확인함
- board[i][j] == ‘ ’ 조건을 확인하며 빈칸이 있는지 확인함
- 빈칸이 있으면 b = false이고 break로 루프를 빠져나옴
- 두 번째 if (!b)는 빈 칸이 발견되었을 때, 더 이상의 확인이 필요 없으므로 바깥쪽 for 루프도 종료하는 역할을 함

```
if (b) {
    for (int i = 0; i < numCell; i++) {
        cout << "---|---|---" << endl;
        for (int j = 0; j < numCell; j++) {
            cout << " " << board[i][j];
            if (j < numCell - 1) cout << " |";
        }
        cout << endl;
    }
    cout << "---|---|---" << endl;

    cout << "무승부입니다!" << endl;
    break;
}
```

입력

- b = 보드가 가득 찼는지 여부를 저장하는 변수
- board[x][y] = 보드판
- numCell = 가로/세로 보드판 개수

결과

- 모든 칸이 꽉 찼을 때, 현재 보드 상태를 다시 한 번 출력하고 무승부임을 출력함

설명

- if(b)는 모든 칸이 꽉 찼을 때 실행됨
- 무승부입니다!를 출력한 후, break로 while 루프를 빠져나옴

```
// 플레이어 교체
    currentUser = (currentUser == 'X') ? 'O' : 'X';
```

입력

- currentUser = 현재 유저값을 저장함

결과

- 게임이 계속 진행되면서 플레이어를 계속 바꿔 줌

설명

- 삼항 연산자를 사용하여 currentUser가 X라면 O으로 바꾸고 O이라면 X로 바꿈

4. 테스트

1 기능 별 테스트 결과

① 누구의 차례인지 출력

② 좌표 입력 받기

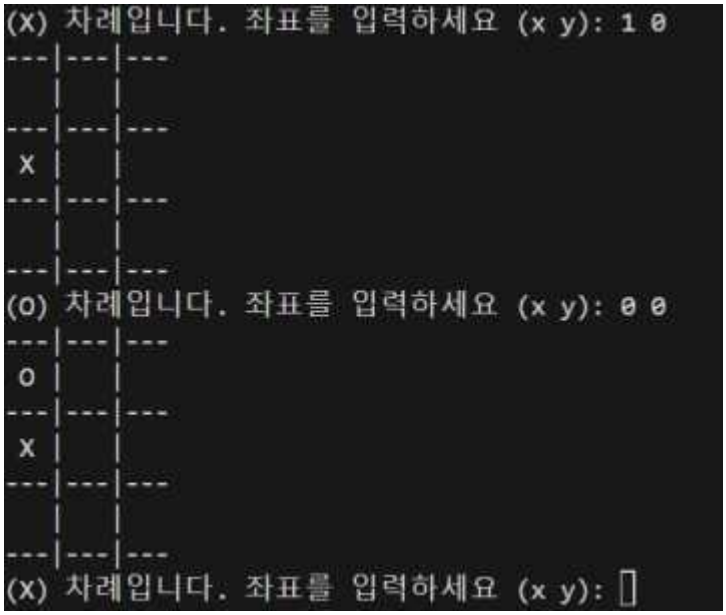
```
(X) 차례입니다. 좌표를 입력하세요 (x y): 0 0
```

③ 입력 받은 좌표 유효성 체크

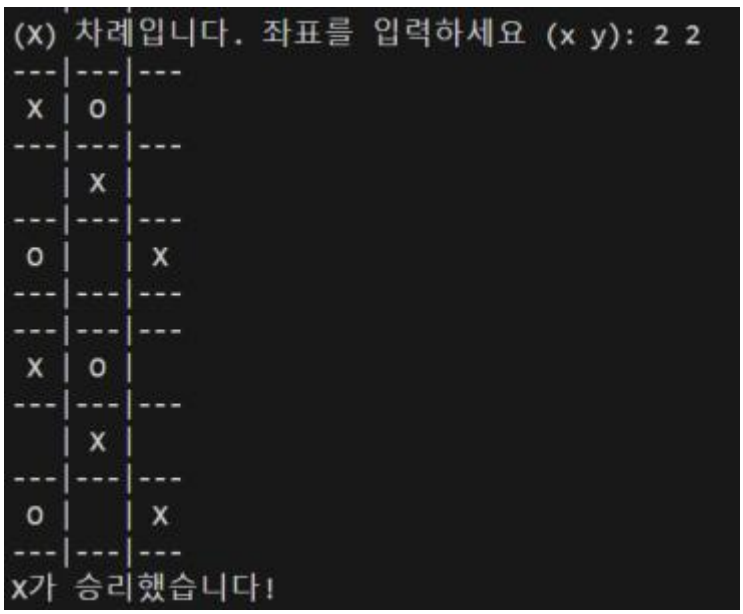
```
(X) 차례입니다. 좌표를 입력하세요 (x y): 5 7
잘못된 입력입니다. 다시 시도하세요.
```

④ 좌표에 O / X 놓기

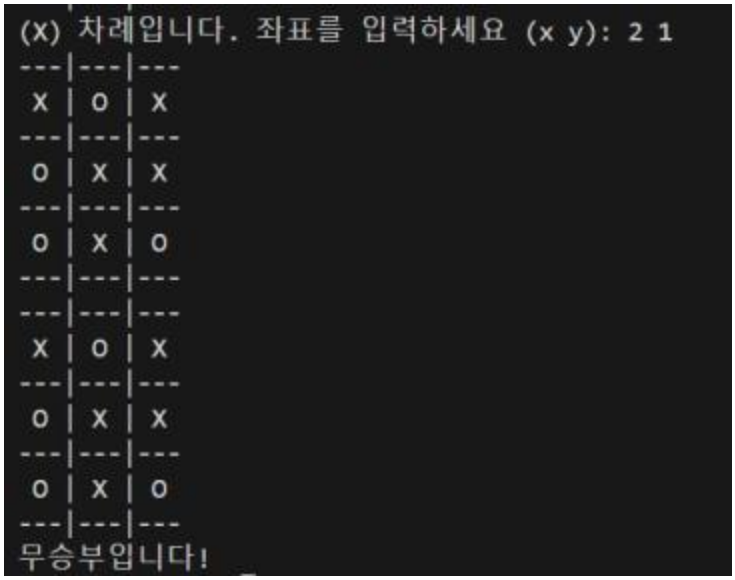
⑤ 현재 보드판 출력



⑥ 빙고 시 승자 출력 후 종료



⑦ 모든 칸이 찼으면 종료



2 최종 테스트 스크린샷

5. 결과 및 결론

1 프로젝트 결과

- Tic Tac Toe 게임을 만들었음

2 느낀 점

- week5까지 배웠던 모든 개념들을 사용하여 Tic Tac Toe 게임을 프로그램을 직접 만들어보면서 개념들을 환기시킬 수 있었음 또한, 프로그래밍 언어를 배우기 시작한 이래로 처음으로 프로젝트를 구성하면서 경험의 폭을 넓힐수 있었음