

Tic_Tac_Toe 프로젝트 보고서

182659 김성준

1. 서론

프로젝트 목적 및 배경: 4주차 까지 배운 내용에 대한 실습 진행

- 자료형, 상수 & 변수, 입출력 함수, 조건문, 반복문, 배열 & 2차원 배열

목표: Tic Tac Toe 게임 구현

2. 요구사항

사용자 요구사항: 두 명의 사용자가 번갈아가며 O와 X를 놓기

기능 요구사항:

- 누구의 차례인지 출력
- 좌표 입력 받기
- 입력 받은 좌표 유효성 체크
- 좌표에 O/X 놓기
- 현재 보드판 출력
- 빙고 시 승자 출력 후 종료
- 모든 칸이 찼으면 종료

제약사항: 보드판은 2차원 배열로 구현

3. 설계 및 구현

기능별 구현 사항:

- 누구의 차례인지 출력

```
const int numCell = 3; // 보드판 한 행이나 열의 칸의 개수
char board[numCell][numCell]{}; //3*3의 보드판 2차원 배열
int x, y; //사용자에게 입력받는 x,y좌표를 저장할 변수 선언

//보드판 초기화
//반복문을 (x,y)값을 ' ' 으로 초기화 하여 보드판을 초기화 한다.
for (x = 0; x < numCell; x++) {
    for (y = 0; y < numCell; y++) {
        board[x][y] = ' ';
    }
}

//게임하는 코드
int k = 0; //차례를 체크하기 위한 변수
char currentUser = 'x'; //현재 유저의 돌을 저장하기 위한 문자변수. 1번 유저차례가 먼저이므로 'x' 초기화// switch문과 나눗셈 연산을 이용하여 차례를 계산
switch (k % 2) {
    case 0:
        cout << k % 2 + 1 << "번 유저(X)의 차례입니다 ->"; // 첫 번째 유저의 차례인 경우 출력
        currentUser = 'x';
        break;
    case 1:
        cout << k % 2 + 1 << "번 유저(O)의 차례입니다 ->"; // 두 번째 유저의 차례인 경우 출력
        currentUser = 'o';
        break;
}
```

- 입력
 - k: 차례를 체크하기 위한 변수
 - currentUser: 현재 차례 유저의 돌을 나타내는 변수

- 결과
 - 현재 차례 유저의 돌 정보가 currentUser 변수에 저장됩니다.
- 설명
 - currentUser의 초기 값은 'x'돌을 사용하는 1번 유저의 차례입니다.
 - 나머지 연산을 수행하여 switch의 조건문을 이용해 k는 0~1 사이의 값을 출력합니다.
 - k가 0이면 1번유저('X'), k가 1이면 2번유저('O')의 차례임을 나타내고, 유저의 차례를 출력합니다.
- 좌표 입력 받기

```
//2. 좌표 입력 받기
cout << " (x,y)좌표를 입력하세요: ";
cin >> x >> y; // 공백으로 구분되어 입력받은 좌표(x,y)값을 각각의 변수에 저장한다.
```

- 입력
 - x: 돌이 놓일 자리의 x좌표
 - y: 돌이 놓일 자리의 y 좌표
- 결과
 - 사용자 입력을 요구하는 문구를 출력하고, 사용자의 입력값(x,y)을 저장합니다.
- 설명
 - x,y의 값은 돌을 놓는 위치의 좌표를 나타냅니다.
- 입력 받은 좌표 유효성 체크

```
//3. 입력받은 좌표의 유효성 체크

//3-1 .입력 받은 좌표가 칸을 벗어나는것은 아닌지 체크
if (x >= numCell || y >= numCell) {
    cout << x << ", " << y << ": ";
    cout << " x와 y 둘 중 하나가 칸을 벗어납니다." << endl; // 오류 메시지 출력
    continue; // (x,y)좌표 다시 입력 받기
}
//3-2. 입력 받은 좌표에 돌이 차있는지 체크
if (board[x][y] != ' ') {
    cout << x << ", " << y << ": 이미 돌이 차있습니다." << endl; // 오류 메시지 출력
    continue; //(x,y)좌표 다시 입력 받기
}
```

- 입력
 - numCell: 보드판의 가로/세로 칸 개수
 - board: 보드판을 구현하는 2차원 배열
- 결과
 - 입력 받은 좌표의 유효성을 체크하고, 그 결과를 출력하여 사용자에게 나타냅니다.
- 설명
 - 입력 받은 좌표를 조건문을 이용하여 가로/세로 칸을 벗어나는지 체크합니다.
 - 입력 받은 좌표에 돌이 놓여있는지 체크합니다.
 - 체크 결과에 따라 좌표가 유효하지 않으면 좌표입력 단계로 돌아갑니다(continue)
- 좌표에 O/X 놓기

```
//4. 입력받은 좌표에 현재 유저의 돌 놓기(유효성 검사 통과)
board[x][y] = currentUser; // 보드판에 현재 유저의 착수를 'x' 또는 'o'로 표기
```

- 입력
 - board[x][y] : 입력 받은 좌표

- currentUser: 현재 차례 유저의 돌 모양 (X/O)
 - 결과
 - 입력 받은 좌표(board[x][y])에 값에 현재 차례 유저의(X/O) 돌을 놓습니다.
 - 설명
 - 현재 보드판의 상태를 나타내는 2차원배열 board[x][y]의 값에 현재 차례 유저의 돌 모양을 입력하여 업데이트 합니다.
- 현재 보드판 출력

```
//5. 현재 보드 판 출력
//보드판 업데이트가 완료되면 결과를 출력한다.
// 2중 반복문을 이용하여 보드판의 테두리를 적절히 표기한다.
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
```

- 입력
 - numCell: 보드판의 가로/세로 크기
- 결과
 - 최신화된 보드판의 상태를 출력합니다.
- 설명
 - 2중 중첩 반복문을 이용하여 보드판의 테두리를 포함하여 현재 상태를 출력합니다.
 -
- 빙고 시 승자 출력 후 종료

```
// 7. 빙고 승자 출력 후 종료(가로, 세로, 대각선)
//반복문을 통해 보드를 탐색하고, 만약 각 승리조건을 만족하는 경우 승자 출력후 게임종료
bool isWin = false; // 경기 결과를 저장하는 변수. 승자가 정해진경우 true, 승자가 없는경우 false(default)

// 7.1. 가로/세로들 체크하기
for (int i = 0; i < numCell; i++) {
    if (board[i][0] == currentUser && board[i][1] == currentUser && board[i][2] == currentUser) {
        cout << " 가로에 모두 돌이 놓였습니다! ";
        isWin = true; // 승자 정하기 (가로)
    }
    if (board[0][i] == currentUser && board[1][i] == currentUser && board[3][i] == currentUser) {
        cout << " 세로에 모두 돌이 놓였습니다! ";
        isWin = true; // 승자 정하기 (세로)
    }
}

// 7.2 대각선들 체크하기
if (board[0][0] == currentUser && board[1][1] == currentUser && board[2][2] == currentUser) {
    cout << " 왼쪽 위에서 오른쪽 아래 대각선으로 모두 돌이 놓였습니다! ";
    isWin = true; //승자정하기 (대각선)
}
if (board[0][2] == currentUser && board[1][1] == currentUser && board[2][0] == currentUser) {
    cout << " 오른쪽 위에서 왼쪽 아래 대각선으로 모두 돌이 놓였습니다! ";
    isWin = true; //승자정하기 (대각선)
}

//승자를 출력하기
if (isWin == true) {
    cout << k%2+1 << " 번 유저("& << currentUser << ")의 승리입니다!" << endl;
    cout << "종료합니다." << endl;
    break; // 더이상 사용자의 입력을 받지 않음
}
}
```

```

k++;

}
return 0;

```

- 입력
 - isWin: 승부 여부를 나타내는 변수.
 - board[x][y] : 보드판의 상태를 나타내는 2차원 배열
 - k: 차례를 나타내는 변수
- 결과
 - 승부를 판단하는 각 조건을 비교하여 결과를 출력하여 더 이상 입력 받지 않거나, 다음 반복을 진행한다.
 - 업데이트가 완료되면 k++를 통해 차례를 넘깁니다.
- 설명
 - isWin 변수를 이용하여 승패 유무를 확인한다. (true: 경기 종료, false: 경기 재개)
 - if 와 &&연산을 이용하여 가로/세로/대각선의 승패 유무를 확인한다.
 - 승패를 매 착수(돌을 놓는 행위)시 마다 확인하기 때문에 경기 승자는 currentUser이다.
 - isWin 이 true일 경우 경기 결과 출력과 함께 경기를 종료하고, false일 경우 경기를 재개한다.
- 모든 칸이 찼으면 종료

```

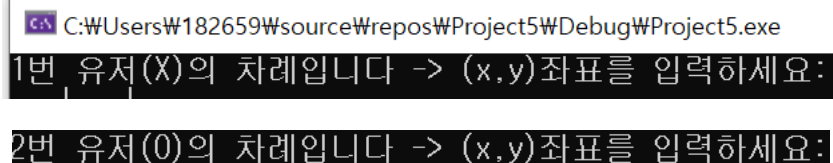
//6. 모든 칸이 다 찼는지 체크
int checked = 0; // 보드판이 다 찼는지 여부를 체크하는 변수
for (int i = 0; i < numCell; i++) {
    for (int j = 0; j < numCell; j++) {
        if (board[i][j] == ' ') { //빈칸이 있는경우 체크변수+1
            checked++;
        }
    }
}
if (checked == 0) { // 빈칸이 없는경우 게임 종료
    cout << "모든 칸이 다 찼습니다. 종료합니다. " << endl;
    break; // 더이상 사용자의 입력을 받지 않음
}

```

- 입력
 - checked: 보드판이 다 찼는지 여부를 체크하는 변수
- 결과
 - 보드판이 다 찬 경우 경기를 종료하고 더이상 사용자의 입력을 받지 않는다. 아닌경우 checked++연산을 해주고 반복을 재개한다.
- 설명
 - 매 반복(경기 좌표 입력)시 마다 checked는 0으로 초기화된다.
 - 2중 반복문을 통해 빈칸이 있는경우 checked++연산을 한다.
 - checked 변수를 이용해 빈칸이 있는경우 checked를 계속 증가시킨다.
 - checked ==0인경우 빈칸이 없기 때문에 결과를 출력하고 break를 통해 경기를 종료시킨다.

4. 테스트

- 누구의 차례인지 출력



```

C:\Users\W182659\source\repos\Project5\Debug\Project5.exe
1번 유저(X)의 차례입니다 -> (x,y)좌표를 입력하세요:
2번 유저(O)의 차례입니다 -> (x,y)좌표를 입력하세요:

```

- 좌표 입력 받기

```
1번 유저(X)의 차례입니다 -> (x,y)좌표를 입력하세요: 1 1
|_|_|
|_|_|
|_|_|
|x_|
|_|_|
|_|_|

2번 유저(O)의 차례입니다 -> (x,y)좌표를 입력하세요: _
```

- 입력 받은 좌표 유효성 체크

```
선택 C:\Users\W182659\source\repos\Project5\Debug\Project5.exe
1번 유저(X)의 차례입니다 -> (x,y)좌표를 입력하세요: 1 1
|_|_|
|_|_|
|_|_|
|x_|
|_|_|
|_|_|

2번 유저(O)의 차례입니다 -> (x,y)좌표를 입력하세요: 3 4
3, 4: x와 y 둘 중 하나가 칸을 벗어납니다.
2번 유저(O)의 차례입니다 -> (x,y)좌표를 입력하세요: 1 1
1, 1: 이미 돌이 차있습니다.
2번 유저(O)의 차례입니다 -> (x,y)좌표를 입력하세요:
|_|_|
|_|_|
|_|_|
|x_|
|_|_|
|_|_|
```

- 좌표에 O/X 놓기

- 좌표의 유효성 검사를 체크를 통과 했는지 확인하기 위해 여러 좌표를 넣었습니다.

```

C:\Users\W182659\source\repos\Project5\Debug\Project5.exe
---|---|---
2번 유저 (0)의 차례입니다 -> (x,y)좌표를 입력하세요: 3 4
3. 4: x와 y 둘 중 하나가 칸을 벗어납니다.
2번 유저 (0)의 차례입니다 -> (x,y)좌표를 입력하세요: 1 1
1. 1: 이미 들어 차있습니다.
2번 유저 (0)의 차례입니다 -> (x,y)좌표를 입력하세요: 0 2
---|---|---
  0
---|---|---
 x
---|---|---
---|---|---
1번 유저 (X)의 차례입니다 -> (x,y)좌표를 입력하세요: 1 1
1. 1: 이미 들어 차있습니다.
1번 유저 (X)의 차례입니다 -> (x,y)좌표를 입력하세요: 2 3
2. 3: x와 y 둘 중 하나가 칸을 벗어납니다.
1번 유저 (X)의 차례입니다 -> (x,y)좌표를 입력하세요: 2 2
---|---|---
  0
---|---|---
 x
---|---|---
  x
---|---|---
2번 유저 (0)의 차례입니다 -> (x,y)좌표를 입력하세요:

```

- 현재 보드판 출력

```

---|---|---
  0
---|---|---
 x
---|---|---

```

- 보드판은 다음과 같이 테두리를 출력하였고, 그 사이 칸에 O.X.로 표기 했습니다

- 빙고 시 승자 출력 후 종료

```

---|---|---
x
---|---|---
x  0
---|---|---
2번 유저 (0)의 차례입니다 -> (x,y)좌표를 입력하세요: 0 2
---|---|---
  0
---|---|---
x
---|---|---
x  0
---|---|---
1번 유저 (X)의 차례입니다 -> (x,y)좌표를 입력하세요: 0 0
---|---|---
x  0
---|---|---
x
---|---|---
x  0
---|---|---
세로에 모두 들어 놓았습니다!: 1 번 유저(x)의 승리입니다!
종료합니다.
C:\Users\W182659\source\repos\Project5\Debug\Project5.exe(프로세스 19272개)이(가) 종료되었습니다(코드: 0개).
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] -> [디버깅] > [디버깅이 중지되면 자동으로 콘솔 닫기]를 사용
하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요...

```

- 모든 칸이 찼으면 종료

```
---|---|---
X | 0 | ---
---|---|---
0 | X | X
---|---|---
X | 0 | ---
2번 유저 (0)의 차례입니다 -> (x,y)좌표를 입력하세요: 0 2
---|---|---
X | 0 | 0
---|---|---
0 | X | X
---|---|---
X | 0 | ---
1번 유저 (X)의 차례입니다 -> (x,y)좌표를 입력하세요: 2 2
---|---|---
X | 0 | 0
---|---|---
0 | X | X
---|---|---
X | 0 | X
모든 칸이 다 찹습니다. 종료합니다.
C:\Users\182659\source\repos\Project5\Debug\Project5.exe(프로세스 2148개)이(가) 종료되었습니다(코드: 0개).
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] -> [디버깅] > [디버깅이 중지되면 자동으로 콘솔 닫기]를 사용
하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요...
```

5. 결과 및 결론

평소에는 이렇게 짜여진 구조 아래에서 코드를 짜본 적이 없었습니다..대충 코드만 구성해서 제출하려다 보니 어디서 틀린 지도 모르는 스파게티 코드를 만드는 게 일상 이었는데 새로운 경험 이였습니다..교수님이 주신 틀 그대로 코딩을 치기 시작했는데 , 각 주석 아래의 기능만 구현하면 된다고 생각하니 마음이 한결 편했다. 하지만 타이핑 속도가 느려서 그런지, 주석을 달면서 코드를 너무 자세히 작성했는지, 문제를 고민하려는 찰나에 정답이 공개되었습니다. 타이핑 칠 양이 많아서인지 스스로 코드에 대해 생각해볼 시간이 부족했던 것 같습니다. 제출에 급급해서 코드를 친 탓인지 오류가 많았습니다. 오류가 있는줄도 모르고 코드를 제출한 후 보고서 작성하면서 기능별로 테스트를 하던 중 몇몇 오류를 발견하였습니다. 비슷한 프로젝트 과목을 수행할 때 힘들게 기능을 구현하고도 실행 중 오류가 발견되어 점수가 깎인적이 많았었는데, 이번 경험을 계기로 여러 방법으로 테스트를 해보는게 중요하다고 느꼈습니다. 또한 기능별로 테스트를 하다보니 오류가 있는 부분을 빨리 찾게 되어서 다행이라는 생각을 했습니다. 다른 선배의 조언으로 notion이라는 툴을 처음 이용해봤는데 매우 편리한것 같습니다. 여러모로 귀중한 경험을 했기에 다음 프로젝트 실습이 기다려지는 것 같습니다.