# R E P O R T

# [Tic_Tac_Toe 게임 만들기]


| | |
|---|---|
| • 과 목 명 | : C++프로그래밍및실습 |
| • 담 당 교 수 | : 김 미 수 |
| • 제 출 일 | : 2023.10.12.(목) |
| • 학 과 | : 통 계 학 과 |
| • 학 번 | : 2 0 3 1 8 5 |
| • 성 명 | : 김 지 웅 |

# <목차>

## 1. 서론

1. 프로젝트 목적 및 배경

   Tic_tac_toe 게임은 전세계적으로 유명한 보드게임으로 3x3 크기의 판에서 플레이어가 번갈아가며 'X'와 'O'를 표시하는 게임이다. 이 게임의 규칙은 플레이어가 가로, 세로, 대각선 방향에 연속으로 같은 표시를 하면 이기는 것이다. 또한 상대가 표시한 좌표에는 다시 두지 못하며 정해진 판의 밖으로는 좌표를 표시하지 못한다.

   우리는 4주차동안 배웠던 변수 선언, 조건문, 반복문, 배열 등의 기본적인 개념들을 통해 Tic_tac_toe게임을 만들어 봄으로써 프로그래밍의 개념과 활용을 학습하는 데에 효과적이라 보고 이 프로젝트를 진행한다.

2. 목표 : Tic Tac Toe 게임 구현

   궁극적인 목표는 Tic_tac_toe 게임 구현이지만 게임을 구현하는 과정에서 필요한 프로그래밍 개념들을 숙지하고 프로젝트를 완성하는 일련의 과정들을 이해하는 것을 목표로 한다.

## 2. 요구사항

1. 사용자 요구사항

사용자 요구사항은 사용자 관점에서 게임에 대한 기대와 특별한 기능에 대한 요구를 정의한 부분이다. 예를 들어, 사용자는 게임 시작, 플레이어의 순서, 게임 보드에 놓을 위치 입력 등을 통해 게임을 조작하기 위한 요구사항들을 의미한다.

따라서 다음과 같이 사용자 요구사항을 작성할 수 있다.

- 두 명의 사용자가 번갈아가며 "X"와 "O"를 놓기

2. 기능 요구사항

기능 요구사항은 실제로 구현할 기능들에 대해 상세하게 작성한 것으로 이는 게임의 핵심 부분인 플레이어의 움직임 처리, 보드판 관리, 승리 조건 등을 정의한 부분들이 필요하다.

따라서 다음과 같이 기능 요구사항을 작성할 수 있다.

- 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 찰 경우 게임 종료

## 3. 설계 및 구현

1. 기능별 구현사항

위에서 작성한 요구사항들을 어떻게 구현할 것인지에 대한 상세한 설명을 작성한다. 이 과정에 클래스, 제어 구조, 배열에 대한 구체적인 설명이 부가된다.

**- 보드판 초기화** :

2차원 배열로 만든 보드판을 초기화하기 위해서 0이나 null값으로 채워진 판으로 만들기 위해서 배열 선언문의 마지막에 {}을 작성한다. 또한 for문을 사용하여 board[x][y]에 ' '인 빈칸을 집어넣어 최초의 보드판을 만든다.

초기화를 또 하는 이유는 유효성 체크 과정에 이미 돌이 차있는 경우, 공백을 확인하는 과정에서 쓰레기값이 아님을 확인을 하기 위해서이다.

**- 누구의 차례인지 출력 :**

플레이어의 차례를 정하기 위해서는 0과 1로만 표현되어져야 하는데 이는 어떠한 수 k를 사용하여 k를 2로 나눈 나머지인 0 또는 1에 각각 +1를 하여 1번 유저, 2번 유저로 나타낸다. 이를 구분 짓기 위해서 switch문을 사용하여 case 0인 경우에는 'X' 차례이고 case 1인 경우에는 'O'의 차례로 나타낸다.

**- 좌표 입력 받기 :**

좌표를 입력 받기 위해서 cin 을 사용하여 각각 x값과 y값을 받는다.

**- 입력 받은 좌표 유효성 체크 :**

유효성 체크란 게임 진행과정 중 룰을 벗어나는 경우를 체크하는 것이다.

따라서 유효성을 체크해야 되는 상황은 두가지가 있는데 입력 받은 좌표가 보드판의 범위를 넘어가는 경우와 이미 좌표가 들어가 있는 경우이다.

먼저 보드판의 범위를 넘어가는 경우에는 조건문을 사용하여 x값이 numCell보다 크거나 y값이 numCell보다 큰 경우를 조건으로 사용하여 오류가 발생했음을 나타낸다.

두번째로 이미 좌표가 들어가 있는 경우에는 보드판의 배열인 board[x][y]이

' ' 빈칸이 아니라는 의미로 !=를 사용하여 조건문을 작성한다.

**- 좌표에 O/X 놓기 :**

입력 받은 좌표를 현재 유저에 해당하는 돌(X, O)을 놓는 과정으로

좌표를 의미하는 board[x][y]와 현재 유저를 나타내는 currentUser를 사용하여

board[x][y] = currentUser로 작성한다.

**- 현재 보드판 출력 :**

현재 유저 상황에서 좌표를 입력을 받으면 보드판에 출력을 해야 게임이 진행될 수 있다. 이를 나타내기 위해서 반복문 안에 조건문을 넣어 표현할 수 있다.

먼저 반복문을 사용하여 변수 i의 크기가 배열의 크기 범위 내에서 판을 그려준다. 판을 표현하는 방법은 "---|---|---"을 출력한다. 다음으로 변수 j가 배열의 크기 범위 내에서 board[i][j]인 현재 유저의 돌을 표현해준다. 또한 무한루프를 배제하기 위해서 조건문을 사용하여 변수 j가 2가 되는 경우 break를 걸어준다. 이때 보드판의 표시인 "  |"를 출력한다. 마지막으로 보드판의 아랫부분인 "---|---|---"를 출력하면 현재 유저의 돌을 표시한 보드판이 완성된다.

**- 빙고 시 승자 출력 후 종료 :**

빙고가 완성되어 승자가 나온 경우 게임을 종료하기 위해서 가로/세로 빙고 경우와 대각선 빙고 경우로 나누어 코드를 작성한다.

먼저 승자를 확인하기 위한 변수로 논리형 데이터인 bool을 사용하여 isWin 변수를 false로 선언한다.

• 가로/세로 빙고

반복문과 조건문을 사용하여 빙고가 되는 상황을 표현할 수 있다.

For문에 변수 i가 배열의 크기 범위 내에서 해당 조건을 만족할 때 isWin이 true가 되는 코드를 짤 수 있다. 해당 조건은 가로의 경우 행을 가리키는 앞부분 인덱스에 i를 집어넣어 [i][0], [i][1], [i][2]가 현재 유저인 currentUser인지를 확인하는 조건문이다. 세로의 경우는 열을 가리키는 뒷부분 인덱스에 i를 집

어넣어 [0][i], [1][i], [2][i]가 currentUser인지를 확인하는 조건문을 작성하여 가로/세로 빙고가 되었는지를 체크할 수 있다.

• 대각선 빙고

대각선 빙고가 되었는지를 체크하는 방법은 조건문을 활용하여 표현할 수 있다.

먼저 대각선의 좌표를 확인하면

[0][0], [1][1], [2][2](우하단좌상단), [0][2], [1][1], [2][0](좌하단우상단)이다.

해당 좌표가 현재 유저인 currentUser일 경우 isWin이 true값이 되는 조건문을 작성하여 대각선 빙고를 체크할 수 있다.

빙고가 확인되어 게임을 종료하기 위해서 isWin이 true값이 되면 게임을 멈추는 조건문을 작성하여 게임을 종료할 있다.

**- 모든 칸이 찰 경우 게임 종료 :**

모든 칸이 다 차 게임을 종료해야 되는 경우에 코드를 작성하면

checked라는 변수를 선언하여 checked 변수가 해당 조건을 만족하면 모든 칸이 찼다고 판단하고 게임을 종료하는 식으로 나타낼 수 있다.

for문을 활용하여 변수 i와 j를 배열 범위 내에서 선언하고 돌의 위치인 board[i][j]가 " "일 경우 checked가 1씩 커지는 조건문을 통해 기본적인 조건을 만들어 내고 만약 checked가 커지지 않고 0인 상태일 경우는 빈칸이 없다는 것을 의미하므로 모든 칸이 다 찼다고 판단 하에 게임을 종료할 수 있다.

## 4. 테스트

### 1. 기능별 테스트

- 누구의 차례인지 출력

```
18            int k = 0; //순서 확인을 위한 변수 선언
19            char currentUser = 'X';
20            while (true) {
21                    //순서 출력 k값을 2로 나누어 나머지 0과 1로 구분
22                    switch (k % 2) {
23                    case 0:
24                            cout << k % 2 + 1 << "번 유저(X)의 차례입니다 -> ";
25                            currentUser = 'X';
26                            break;
27                    case 1:
28                            cout << k % 2 + 1 << "번 유저(O)의 차례입니다 -> ";
29                            currentUser = 'O';
30                            break;
31                    }
```

- 좌표 입력 받기

```
32                    cout << "(x, y) 좌표를 입력하세요 : ";
33                    cin >> x >> y;
```

- 입력 받은 좌표 유효성 체크

```
35                    // 3. 좌표의 유효성을 체크함.
36                    if (x >= numCell || y >= numCell) {
37                            cout << x << ", " << y << ": ";
38                            cout << "x와 y 둘 중 하나가 칸을 벗어났습니다.";
39                            continue;
40                            //x나 y값이 3보다 큰 값이 나오면 벗어나 오류 발생
41                    }
42                    if (board[x][y] != ' ') {
43                            cout << x << ", " << y << ": 이미 돌이 차있습니다." << endl;
44                            continue;
45                            //값이 겹치게 되면 오류 발생
46                    }
```

- 좌표에 O/X 놓기

```
47                    // 입력받은 좌표에 돌 놓기
48                    board[x][y] = currentUser;
```

- 현재 보드판 출력

```
50                  // 보드판 출력
51                  for (int i = 0; i < numCell; i++) {
52                          cout << "---|---|---" << endl;
53                          for (int j = 0; j < numCell; j++) {
54                                  cout << board[i][j];
55                                  if (j == numCell - 1) {
56                                          break;
57                                  }
58                                  cout << "  |";
59
60                          }
61                          cout << endl;
62                  }
63                  cout << "---|---|---" << endl;
```

- 빙고 시 승자 출력 후 종료

```
78                  // 빙고가 완성된 경우 종류
79                  bool isWin = false;
80                  for (int i = 0; i < numCell; i++) {
81                          if (board[i][0] == currentUser && board[i][1] == currentUser && board[i][2] == currentUser) { //가로줄에 i를 넣어 모
82                                  cout << "가로에 모두 돌이 놓였습니다. : ";
83                                  isWin = true;
84                          }
85                          if (board[0][i] == currentUser && board[1][i] == currentUser && board[2][i] == currentUser) { //세로줄에 i를 넣어 모
86                                  cout << "세로에 모두 돌이 놓였습니다. : ";
87                                  isWin = true;
88                          }
89                  }
90
91                  if (board[0][0] == currentUser && board[1][1] == currentUser && board[2][2] == currentUser) {
92                          cout << "왼쪽 위에서 오른쪽 아래 대각선으로 모두 돌이 놓였습니다. : ";
93                          isWin = true;
94                  }
95                  if (board[0][2] == currentUser && board[1][1] == currentUser && board[2][0] == currentUser) {
96                          cout << "오른쪽 위에서 왼쪽 아래 대각선으로 모두 돌이 놓였습니다. : ";
97                          isWin = true;
98
99                  }
100                 if (isWin == true) {
101                         cout << k % 2 + 1 << "번 유저(" << currentUser << ")의 승리입니다." << endl;
102                         cout << "종료합니다." << endl;
103                         break;
104                 }
105                 k++; //마지막에 k++를 해야 while문에서 번호가 바뀌지 않음.
106         }
107         return 0;
```

- 모든 칸이 찰 경우 게임 종료

```
65                  // 모든 돌이 다 찼는 경우 종료
66                  int checked = 0;
67                  for (int i = 0; i < numCell; i++) {
68                          for (int j = 0; j < numCell; j++) {
69                                  if (board[i][j] == ' ') {
70                                          checked++;
71                                  }
72                          }
73                  }
74                  if (checked == 0) {
75                          cout << "모든 칸이 다 찼습니다. 종료합니다." << endl;
76                          break;
77                  }
```

## 2. 최종 테스트

### 1) 모든 빙고칸이 찬 경우

```
1번 유저(X)의 차례입니다 -> (x, y) 좌표를 입력하세요 : 0 0
---|---|---
X  |X  |O
---|---|---
O  |O  |X
---|---|---
X  |O  |X
---|---|---
모든 칸이 다 찼습니다. 종료합니다.
Program ended with exit code: 0
```

All Output ⌃ Filter

### 2) 가로 빙고로 종료된 경우

```
1번 유저(X)의 차례입니다 -> (x, y) 좌표를 입력하세요 : 0 2
---|---|---
X  |X  |X
---|---|---
   |O  |
---|---|---
   |   |O
---|---|---
가로에 모두 돌이 놓였습니다. : 1번 유저(X)의 승리입니다.
종료합니다.
Program ended with exit code: 0
```

### 3) 세로 빙고로 종료된 경우

```
1번 유저(X)의 차례입니다 -> (x, y) 좌표를 입력하세요 : 1 0
---|---|---
X  |   |
---|---|---
X  |O  |O
---|---|---
X  |   |
---|---|---
세로에 모두 돌이 놓였습니다. : 1번 유저(X)의 승리입니다.
종료합니다.
Program ended with exit code: 0
```

All Output ⌃ Filter

4) 대각선(좌하단 -> 우상단) 빙고로 종료된 경우

```
1번 유저(X)의 차례입니다 -> (x, y) 좌표를 입력하세요 : 2 0
---|---|---
O  |   |X
---|---|---
   |X  |O
---|---|---
X  |   |
---|---|---
오른쪽 위에서 왼쪽 아래 대각선으로 모두 돌이 놓였습니다. : 1번 유저(X)의 승리입니다.
종료합니다.
Program ended with exit code: 0
```

All Output ⌃ Filter

5) 대각선(우하단 -> 좌상단) 빙고로 종료된 경우

```
1번 유저(X)의 차례입니다 -> (x, y) 좌표를 입력하세요 : 2 2
---|---|---
X  |   |
---|---|---
O  |X  |
---|---|---
O  |   |X
---|---|---
왼쪽 위에서 오른쪽 아래 대각선으로 모두 돌이 놓였습니다. : 1번 유저(X)의 승리입니다.
종료합니다.
Program ended with exit code: 0
```

All Output ⌃ Filter

## 5. 결과 및 결론

1) 프로젝트 결과

프로젝트를 통해 c++언어를 사용하여 tic_tac_toe 게임을 만들어냈다.

게임은 두 명의 사용자가 번갈아가며 "X"와 "O" 놓는 기본적인 규칙을 바탕으로 세부적인 승리 조건을 만족하면 게임이 종료되는 룰을 따랐다.

게임을 구현하기 위해 변수 및 상수, 제어구조, 배열 등의 기본적인 c++언어 개념을 활용하여 게임의 상세한 기능들을 표현해냈다.

2) 느낀 점

이 과목을 처음 공부하는 입장에서 간단하지만 벌써 하나의 프로젝트를 진행했다는 것에 큰 의미가 있다. 기본적인 프로젝트의 흐름과 코드를 이해하고 함수들을 어떻게 적절하게 사용하여 표현할 수 있을지에 대해 스스로 고민해봄으로써 c++ 언어 개발에 대한 이해력과 통찰력을 기를 수 있었다.