Received Feedback

Link to topic: https://codereview.stackexchange.com/questions/289259/beginners-attempt-at-tictactoe

By depperm

First thing that comes to mind is to remove `goto` (see SO question):

- `checkMove` you only need to check a single cell not up to 9 with loops

```
bool checkMove(char grid[][3], int move) { // wouldn't pass move_ok
  bool move_ok;
  if (move >= '1' && move <= '9') {
    // convert from ascii to int, then to position
    int row = (move - 49) / 3;
    int column = (move - 49) % 3;
    if(grid[row][column] == move) {
      move_ok = 1;
    } else {
      move_ok = 0;
    }
  } else {
    move_ok = 0;
  }
  return move_ok;
}
```

- `makeMove` can be very similar to `checkMove` (don't need loop)

```
// convert from ascii to int, then to position
int row = (move - 49) / 3;
int column = (move - 49) % 3;
grid[row][column] = mark
```

- `checkThree` you could easily check cell equality in a single line, and then `return` immediately instead of using `goto`. Also use bool

```
bool checkThree(char grid[][3]) {//don't need three_row
// left to right diagonal
if (grid[0][0] == grid[1][1] && grid[1][1] == grid[2][2]) {
  return true;
}
...
return false; // you don't treat 1 or 2 differently from this function
```

this would require slightly different validating

```
three_row = checkThree(grid);//don't need to pass three_row
if (three_row){
  if(mark == 'O') {
    return 1;
  } else {
    return 2;
  }
}
...
```

- `checkTwo` could be very similar to `checkThree`, just expand the check (I think you can remove `needMove`, `pickCpuMove`, or `resetVars`)

```
move = '0'
// left to right diagonal
if (grid[0][0] == grid[1][1] && grid[1][1] != grid[2][2]) {
  // know 2 are the same, but not the last
  return '9';
} else if (grid[0][0] != grid[1][1] && grid[1][1] == grid[2][2]) {
  // know last are the same, but not the first
  return '1';
} else if (grid[0][0] == grid[2][2]) {
  // know ends are the same, but not the mid
  return '5'
}
...
```

Simplify:

- `if (grid[1][1] != 'X' && grid[1][1] != 'O')` can be `if (grid[1][1] != '5')`
- `} else if (grid[1][1] != 'X' || grid[1][1] == 'O') {` can be `} else {`
- `} else if (mark == 'O') {` can be `} else {`
- don't need to pass so many arguments
    - `getMove` doesn't need `move`, it should return `move` (`cpuMove`, `pickCpuMove` (unless you remove it as noted above) also don't need it, same reason)
    - IMO using function parameters as variables is confusing. Generally I treat function parameters as read-only (there are exceptions to this (see pass by reference), but when I read `myVar = myFunc(myVar)` I don't expect `myVar` to change inside `myFunc`). In `setDifficulty` I would do something like:

```
  if (input == '1') {
    puts("Difficulty set to EASY");
    return '1';
  } else if (input == '2') {
    puts("Difficulty set to HARD");
    return '2';
  } else {
    puts("\nInvalid choice! Difficulty will remain the same");
    return difficulty
  }
```

EDIT Based on https://github.com/Reslashd/tictactoe/blob/main/tictactoe.c

Variable/parameter clean up

- clear names: Reading a variable should let future you/devs know what is stored in it.
- don't pass basic variables as parameters to edit
    - in `main`, `three_row` has several issues. It isn't very clear what it stores. I'd change it `winner` and remove it as a parameter to `gameLoop`. `gameLoop` without the parameter could just put the check in the `if`: `if(checkThree(grid))` and the final `return` would be `return 0` (no winner)
    - `switchMark` is assigning to `mark`, instead it should just `return`. Shorter and clearer code. Don't treat parameters as editable variables, generally should be read only
    - in `gameLoop` just initialize `mark` without passing `mark`: `char mark = startingMark(choice)` (might rename to `getStartingMark`). Inside `startingMark` change the parameter name to be something like `numPlayers` instead of `choice` (clear what is being passed in). The logic of `startingMark` isn't very clear. It initializes `start_mark` to an int later it's a char (technically stored same, but to be clear to developer distinguish the type better).
    - `showMenu` doesn't need `choice`, it returns the player choice, initialize choice like `char choice = getchar();`
- change parameter names to be different than what you pass. This might help separate use/difference
    - `setDifficulty` the parameter could be `currentDifficulty`. Side note, inside this function input could be initialized in on line with correct type `char input = getchar();`
    - `switchMark` the param could be `currentMark`

Simplify Again

- 99% of the time if you see a pattern it can be simplified
    - `checkTwo` and `checkThree` row/col check could be put in a `for` loop. `checkTwo`'s fail safe return should validate the random choice is a valid move like all the rest (as `cpuMove` also has a rand choice, you make a rand choice function, then you won't have to check it later):

```
char randMove;
do{
  randMove = (rand() % 9) + 49;
}while(!checkMove(grid, randMove));
return randMove; // Failsafe return random move.
```

All these together: https://gist.github.com/depperm/b28798d3730c0f8394fd1ed380fa95e5 (almost 100 lines shorter)

By aghast

What happens if you Ctrl+D (or Ctrl+Z on Windows/DOS) while at the menu? For me, it was an endless loop because EOF is not handled.