NED University of Engineering
and Technology

Programming Fundamentals
CT-175

Semester Project

TIC-TAC-TOE

A 3 x 3 grid game
Implemented in C language,
Using Structures and file Handling

PREPARED BY

Abdul Rafay Khatri DT-039
Ajiya Anwer DT-006
Hassan Hayat DT-010

PREPARED FOR

Dr. Muhamamad Kamran

PROJECT REPORT
TIC-TAC-TOE

TABLE OF CONTENTS

1. Introduction
2. Background Research
3. Problem Analysis
4. Flowchart
5. Specifications
6. Implementation
7. Conclusion

Introduction

Tic-tac-toe, also known as noughts and cross is played between two players traditionally, the players take alternate turns to fill the boxes in a 3 x 3 grid, the first player to get three consecutive spaces filled in a horizontal, vertical or diagonal manner is declared the winner. This simply means that two agents work against each other with alternative turns. In some cases, when none of the players succeed to fill the boxes in the above mention manner, the game will be considered a draw.

Background Research

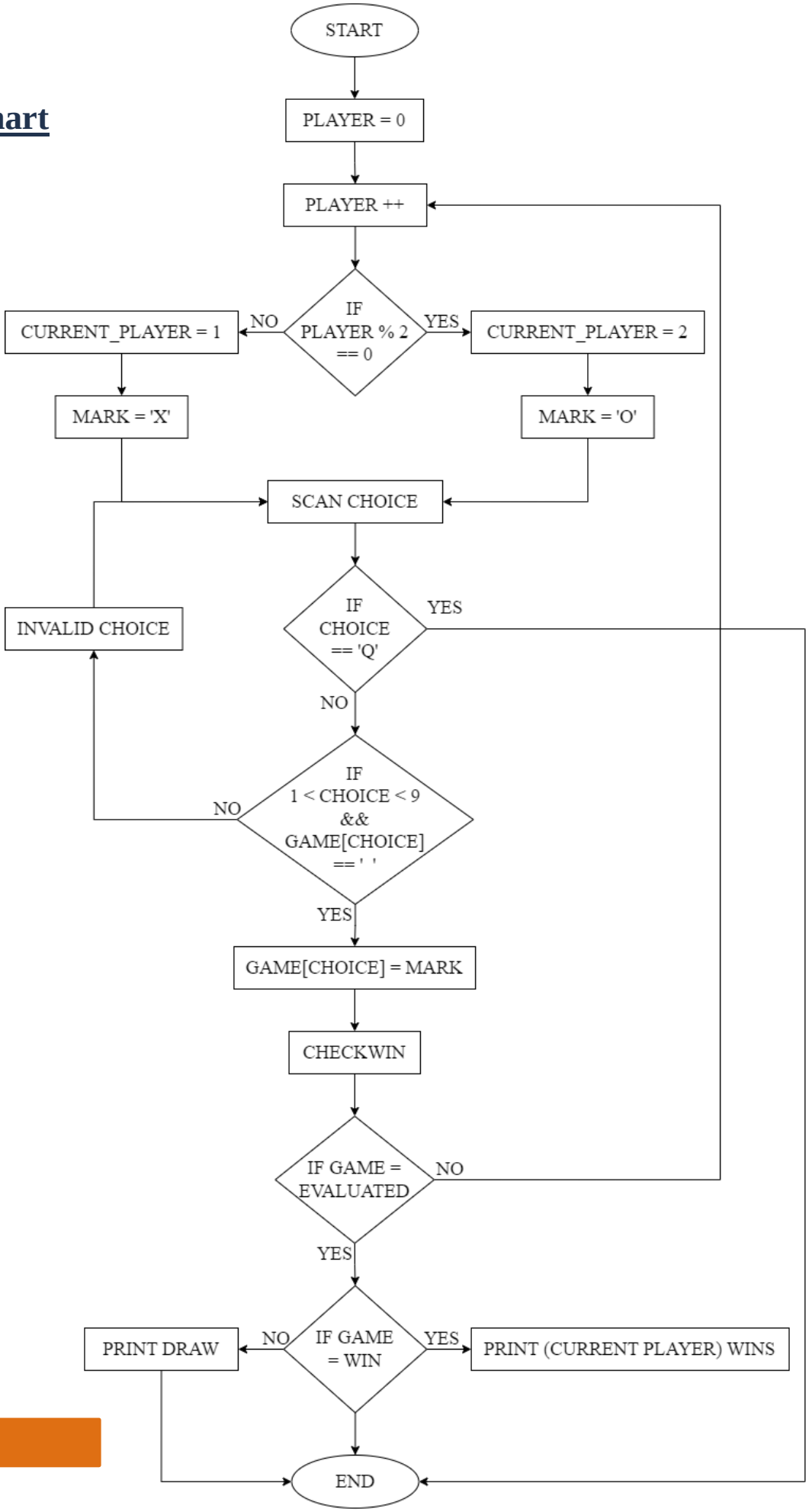
The software that has been coded is specified as a game. It is designed by coding a sample layout of 9 boxes distributed in 3 rows and 3 columns. The significance of this software arose while having a look at the work loaded environment where students and people are eagerly looking for an escape from their hectic schedules. To ease overwhelming stress in a person's personal life and let them spend time with a colleague, the software also gives a chance for multiplayer game environment. The main objective was the creation of an error-free environment so it does not become a source of any sort of trouble for its users.

Problem Analysis

As far as the game is concerned, tic-tac-toe is a simply designed game that should not face any hurdles in its development in your IDE but, considering the specifications we designed in our code there have been a number of things that we have encountered. The code that has been designed contains the player records and take input from both the users in our multiplayer mode. The next step in the code was "how the game works". The software stops after one of the players meet any of the 8 wining conditions of tic-tac-toe and declare him as the winner. The creation of bot has been a task that requires plenty of time to tackle, the bot have been designed with such conditions that it put hurdles in front of its opponents by placing a mark in the most appropriate places.

We were also required to design conditions for draw and implement such conditions that hinders such scenarios where the game might end up in draw. Keeping the strategies and ideas in mind we were required to design a code that fulfils the desired tasks in an error free manner.

Flowchart


START
PLAYER = 0
PLAYER ++
IF PLAYER % 2 == 0
NO
YES
CURRENT_PLAYER = 1
CURRENT_PLAYER = 2
MARK = 'X'
MARK = 'O'
SCAN CHOICE
IF CHOICE == 'Q'
YES
NO
INVALID CHOICE
IF 1 < CHOICE < 9 && GAME[CHOICE] == ' '
NO
YES
GAME[CHOICE] = MARK
CHECKWIN
IF GAME = EVALUATED
NO
YES
IF GAME = WIN
NO
YES
PRINT DRAW
PRINT (CURRENT PLAYER) WINS
END

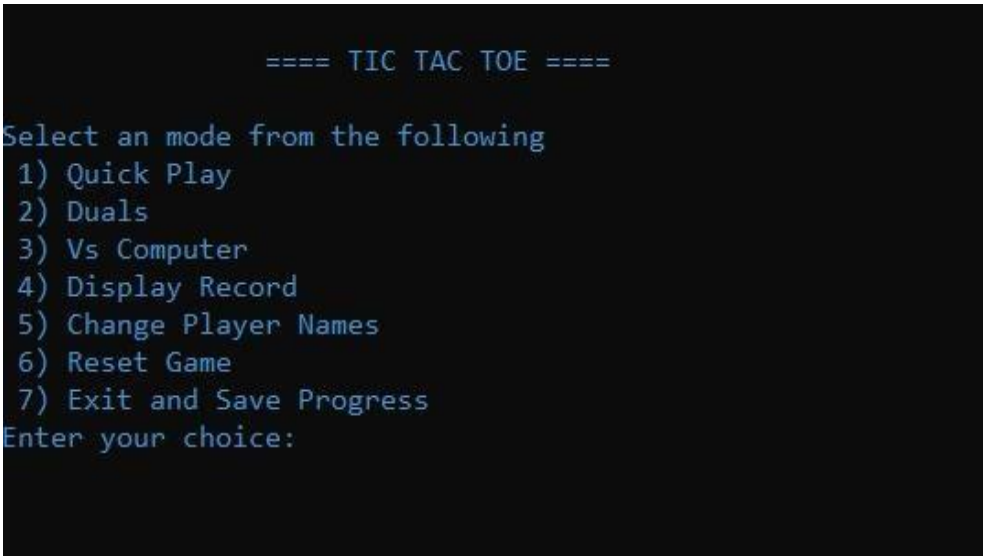
Specifications

The software under consideration have been deployed with numerous features and specifications. The respective features are mentioned as follows:

- **Multi Player (2 users):** In our program, the player perceives the environment and acts through keyboard, the multiplayers program allows two players to get their alternative turns in filling the spaces of the grid.
- **Single Player (against BOT):** The option of bot has also been placed in the program. The bot mode enables the user to input his mark in his choice of box in the grid and the other mark is placed in the grid by the bot. After alternate turns between the bot and the user, if one of them succeeds in filling the grid in the winning manner then they are declared as the winner.
- **Record Maintenance:** One of the features include the ability of the program to store and keep record of the data, it maintains stats of every individual player and keeps track of games played and win counts. We have also included an option for displaying the record of any instance at any desired time, the option is required to be selected by one of the players.

And some additional features to improve UI and smoothness of the game, like we have designed the code in a way that if you are the first one to mark a position in round one, you will automatically be the second one to mark in the next round.

Main Menu



In Game

Implementation

The main c source code for our tic-tac-toe starts with declaration of functions and structures before the main function, we declared functions for the grid, loading screen, data entry and other features that are going to be used in the game, the structures include structure for dual player and bot that contains variables of two different data types (char and int) for the information about a player. The next step was declaring global in variables for the results.

The main function starts with the loading screen and later function calling for the header, the player is presented with two choices on whether he wants to play the game or check the records. If else statements are used inside Switch Conditional statement to check the input by the user and display the desired output. If the input doesn't match any condition the code goes back to the label for choice. After the choices have been made for the getting previous record, the rules and description function gets called to provide their required assistance. A while loop starts with a new grid and choice of mode for the game. The player enters his desired choice and pursue the game. Again switch statements have been used that check the condition and send the input to the respective functions. One of the cases also asks the user whether they want to change the names and turns after a round, save the record or exit the game.

For each of the 7 options in the main menu, we wrote different functions that will be called respectively upon user's choice.

The checkWin function

First Task was to write a function that keeps checking the status of the game, that is to check the 8 winning conditions for tic-tac-toe that are, if there are same marks in horizontal, diagonal or vertical manner the player is declared as winner.

The Quick Play Function

This function just simply starts a multiplayer game and ask users the choice of box from the grid where he wants to put the mark. If the player enters Q it ends the game, while other times the player has to enter a single digit to place his mark, then every player move it checks the status of game by the checkWin function.

The Duals Function

The duals function works almost as similar as the Quick Play but to be more systematic we added a record maintenance feature in our duals game mode.

The Bot Function

Again the same logic as the Quick Play function, but this time, user is able to play against our very own made bot.

How the BOT works

The real challenge in this mode was designing of the bot, we searched for many algorithms but none seemed to be preferable for our requirements so we designed our own. Its implementation was in the way that we gave multiple conditions to the bot to choose his place of mark on the tic-tac board. For giving it the sample conditions we first understood the logics and techniques of winning and stopping our opponent from winning, we analyzed each and every condition and started feeding our bot to do such moves if it detects this arrangements of marks on the board placed by his opponent. And, finally we designed the final conditions. Its first priority was filling the center box and after that second priority was to win a game (if by placing a single marks it can win), if no conditions from the eight (8 ways of winning the game (diagonally, horizontally and vertically)) is met then check for the same condition but for the opponent, and if there is again no such condition of opponent winning in one move. Then our next task was to make the bot enough efficient that it automatically detects which trick user is planning to execute. For that we first researched about the tricks and we found many. If we started putting each and every condition individually, then the code would have become a mess. So in order to prevent such a long code, we analyzed the tricks combined and chose the precedence of bot move decision in such a way that we didn't even had to use each trick and apply its all conditions. So we worked more on it, and gave it the important conditions of the tricks user might use with appropriate precedence and after feeding all the possible ways in which the tricks can be applied we tested it and this time our BOT was actually unbeatable.

Functions for Keeping and Showing Record

- **getRecord Function:** When the user options to start TICTACTOE, the program first asks the user to load his previous data(record). This functions simply open a file named record.bin (in which data is saved by default) and if there is any record available, it loads it into RAM and assign the values to their respective variables and strings. And in the case no record is found, it prompts a message that no record found and then automatically starts a new game.
- **saveRecord Function:** when the user is done playing the game and options to close the game, this function is automatically called and it overwrites the current stats of the game to the default file record.bin.
- **showRecord Function:** This function show the user the current stats of the game. This functions works simply by reading values stored in variables of stats in use, and print it in a format.

Other simple void as return data type functions have been used to display the header, rules, description, loading screen and for printing and resetting the board.

Conclusion

As all the above points have demonstrated, the project source code constitutes a game that bears the name “Tic-Tac-Toe”. The game has been developed under certain conditions that allows the user to have multiple options while playing. We also made use of functions to make the code more readable and also due to usage of similar techniques of C we compressed them into one single function. The goal of the game is one of the opponents make continuous marks in 3 cells of the grid in a winning manner and the other player hinders his tries of winning. A bot has been added to make the game a bit more challenging. Record maintenance has been added to give the players the ability to keep track of their record and past history regarding the game, for this purpose we used file handling. The techniques and algorithms of C language have been used in designing the desired code. However, there have been certain ideas that couldn’t get coded in the period of time that we were allotted. One of the options that we would have added was of the choice of mark to be taken as an input. The user would e at the liberty to decide whether he wants to go with an ‘X’ or ‘0’ or any other mark. We do hope to witness this variation in future by any other group.