MANUAL

0810

TIC TAC TOE

Lalesca Gonçalves

Prog02

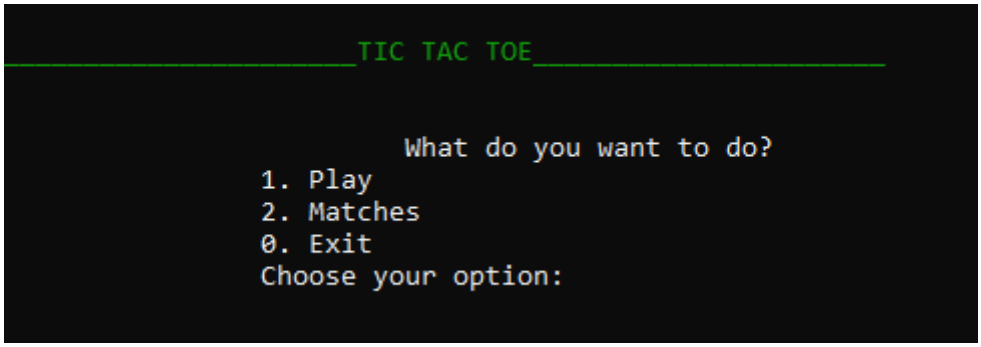
Professor:

Ricardo Farinha

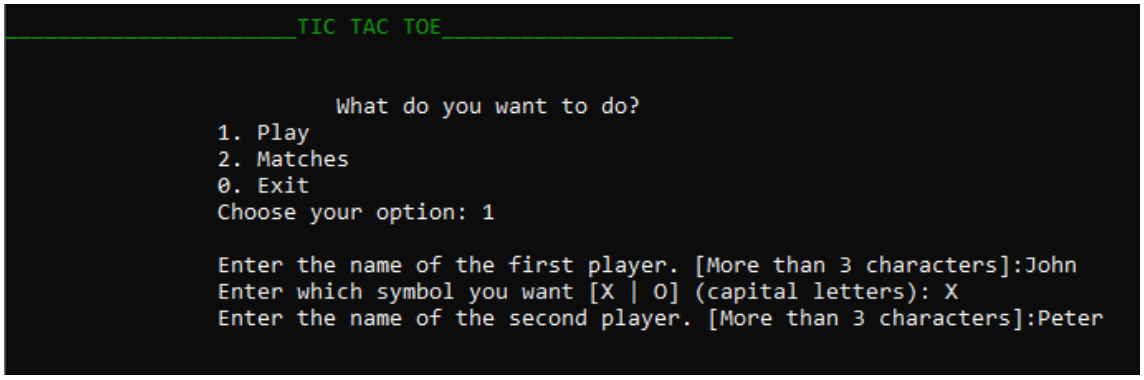
08-2020

IEFP

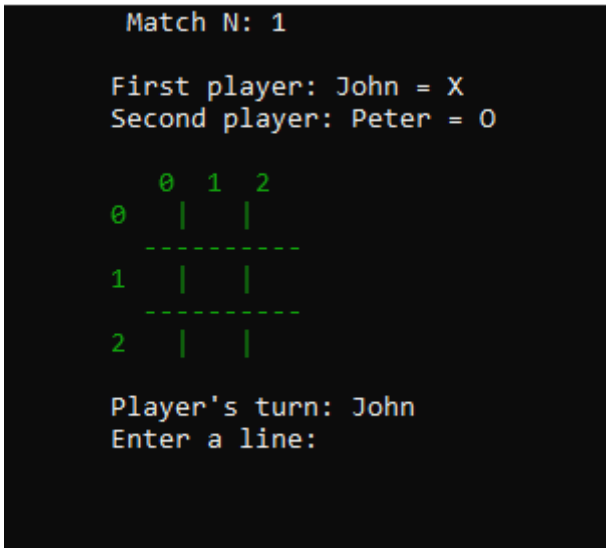
Options menu



When clicking on play for the first time, you must enter the name of each player and the name must be longer than three characters. The first player entered has the right to choose the symbol that he wants to play. When choosing 'X', the other player will automatically been assigned with the symbol 'O', and vice versa.



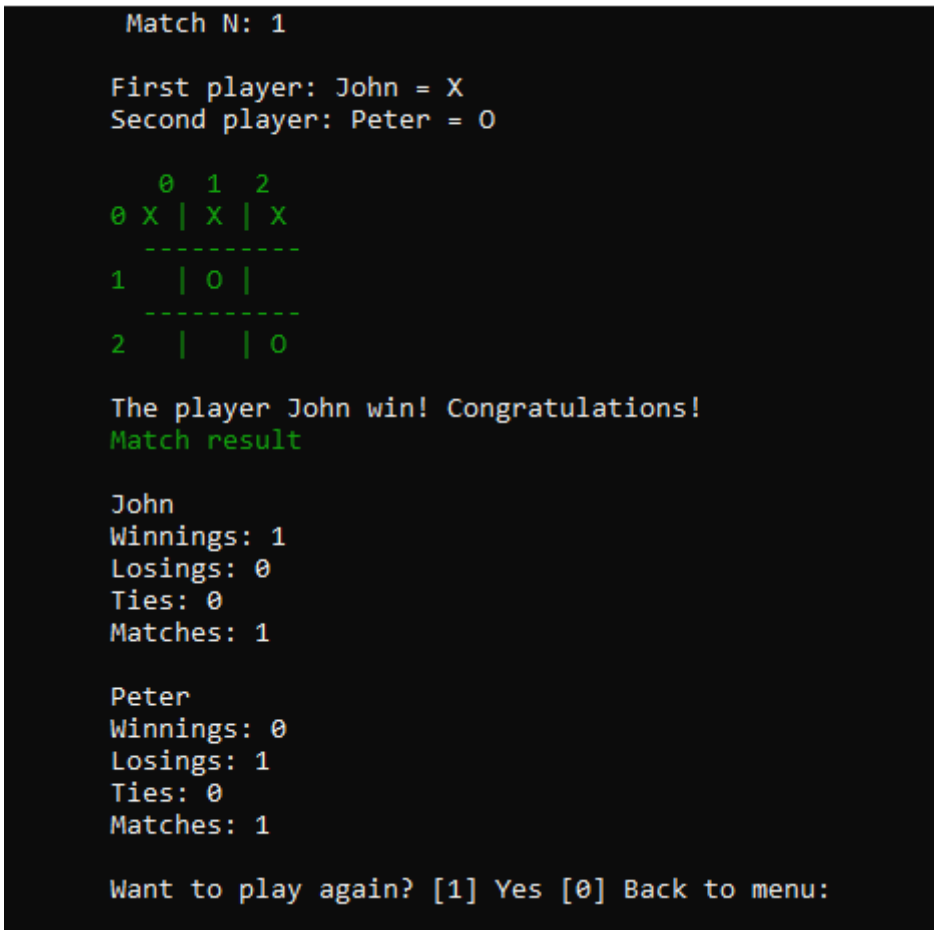
After entering the names of the players and the symbol the game starts.



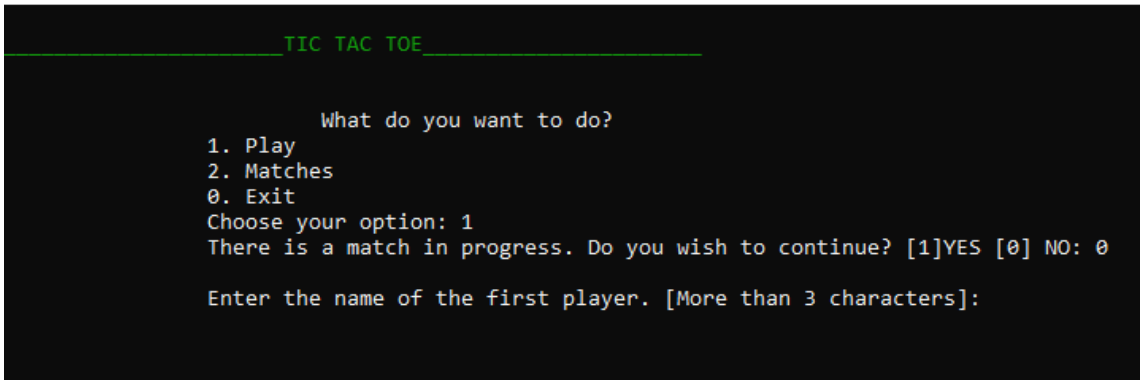
The first player inserted starts the game and must choose the row and column he wants. Being this 0, 1 or 2.

After correctly inserting the values, the board automatically fills in the empty spaces and passes the turn to the next player.

When a player wins, the name of the winner and the score of the game are shown.



The player has the option to play as many times as he wants. With each move the values are updated accordingly. You also can return to the Menu again, where you can save games if you wish or play again. If the player clicks on play again and still has a game in progress, he will be asked if he wants to continue the previous game (thus returning to the board) or not to do so (thus starting a game from scratch, where the names will be asked again of the players).



Tic-tac-toe rules and objectives

The rules of the game are quite simple. In short, two players choose two symbols they want to play with. The program accepts only the letters X and O. The player must try to create a line in sequence with the character he is playing, either in line, column or diagonally. Whoever completes wins.

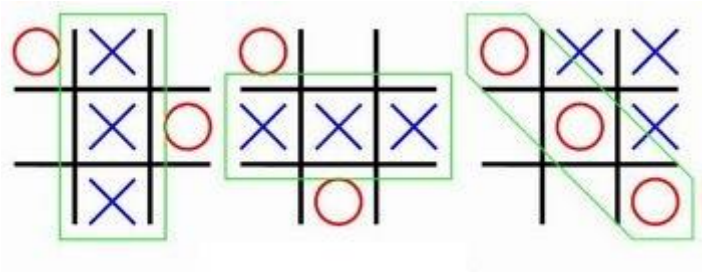
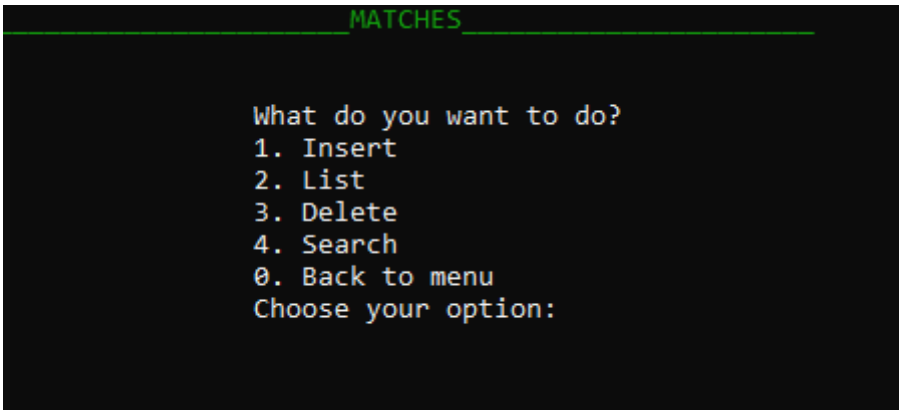


Image 1: Wins use case - Fonte: http://www.ppgia.pucpr.br/~radtke/jogos/velha/projeto-jogo_da_velha.pdf (Access: 27/07/2020).

MATCHES

In this menu option the player finds what he wants to accomplish with the game played, or view previous games that have been saved.



If the player wishes to insert the game to be saved, he can only do so if he has already played at least once. The name of each player and their score will be automatically saved.

For the option to delete and search, the name of any previously added player that you want to search will be asked, it must have more than three characters.

Good game!!

MANUAL

Developer

TIC-TAC-TOE

Lalesca Gonçalves

Prog02

Professor:

Ricardo Farinha

08-2020

IEFP

The tic-tac-toe game was developed using the C ++ language, with the DevC ++ IDE.

The program was started by creating a struct named *player*.

```
using namespace std;
struct player
{
    string name;
    string symbol;
    int winnings;
    int losings;
    int ties;
    int matches;
    bool playerTurn = false;
};
```

These are the functions used, as well as the global variables. The *hConsole* variable was created so that we can add color inside the console. And two variables were created from the struct as mentioned above *player1*, *player2*.

```
HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE); //change colour of the letter
player player1, pl
string gameBoard[3][3];
int count=0;
int column, line, match =0;
fstream file;

string readPlayerName();
void startGame();
void menu(int op);
void selectMatchMenu (int opcao);
void showFileMenu();
void showGameBoard();
void insertGameBoard();
void checkWinner();
void winnerX();
void winnerO();
void startGameBoard();
void clearGame();
void setSymbol();
int checkField();
void startMatch();
void checkKeepPlaying();
void showResult();
void insertFile();
void listFile();
void searchFile();
void deleteFile();
void checkGame();
```

WINBASEAPI HANDLE WINAPI GetStdHandle (DWORD nStdHandle)

In the ***main*** method is called the startGameBoard function, so that it is possible to check further ahead if the space is empty.

```
main()
{
    startGameBoard();
    startGame();
}
void startGameBoard(){//start game board for check in function insertGameBoard
    for (int i=0;i<3;i++)
    {
        for (int k=0; k<3;k++){
            gameBoard[i][k] = "";
        }
    }
}
void clearGame(){
    player1.name = "\0";
    player1.symbol ="\0";
    player1.winnings =0;
    player1.losings =0;
    player1.ties=0;
    player1.matches=0;
    player1.playerTurn=false;

    player2.name = "\0";
    player2.symbol ="\0";
    player2.winnings =0;
    player2.losings =0;
    player2.ties=0;
    player2.matches=0;
    player2.playerTurn=false;

    match =0;
    startGameBoard();
    count =0;//clear count
    startMatch();
}
```

Functions

The **checkGame** function checks whether a match is in progress using the global variable *match*.

```
void checkGame(){//ckeck if there is a game in progress
    int op;
    if(match > 0){
        cout<<"\t\tThere is a match in progress. Do you wish to continue? [1]YES [0] NO: ";
        cin >> op;
        while(cin.fail() || op != 0  && op != 1)
       {
        cin.clear();
        cin.ignore();
        cout << "\t\tInvalid field. Type again: ";
        cin >> op;
        }
        if(op==1){
            startMatch();
        }else{
            clearGame();
        }
    }else{
        clearGame();
    }
}
```

The function **ReadPlayersName** is intended to check the fields in which it will be necessary to read the players' names, it will also be used in the delete and search in file functions.

```
string readPlayerName(){
    string name;
    cin.ignore();
    getline( cin, name);
    while(name.length() < 3){
        cout << "\t\tRetype name: ";
        getline( cin,name );
    }
    return name;
}
void setSymbol(){
```

Function to build the board (showGameBoard). This function is called up each time by a player and inserts the respective symbol on the board

```
void showGameBoard(){
    system ("CLS");
    cout<<"\t\t Match N: "<<match<< "\n";
    cout << "\n\t\tFirst player: "<<player1.name << " = "<<player1.symbol<<endl;
    cout << "\t\tSecond player: "<<player2.name << " = "<<player2.symbol<<"\n"<<endl;
    int l,c;
    SetConsoleTextAttribute(hConsole, 2);
    cout << "\t\t   0  1  2\n";
    for (l=0;l<3;l++)
    {
        cout << "\t\t" << l;
        for (c=0; c<3;c++){
            if (c ==0 ){
                if(gameBoard[l][c].length() == 0){
                    cout << "   ";
                }else{
                    cout << " "<<gameBoard[l][c]<<" ";
                }
            }else{
                if(gameBoard[l][c].length() == 0){
                    cout << "|   ";
                }else{
                    cout << "| "<<gameBoard[l][c]<<" ";
                }

            }
        }
        cout << "\n";
        if(l!=2){
            cout << "\t\t  ----------";
            cout << "\n";
        }
    }
    SetConsoleTextAttribute(hConsole, 15);
    checkWinner();
    insertGameBoard();
}
```

checkField is called to validate data from the first option menu and to validate rows and columns.

```
int checkField(){
    int n;
    cin >> n;
    while(cin.fail()|| n > 2 || n <0 )
       {
        cin.clear();
        cin.ignore();
        cout << "\t\tInvalid field. Type again: ";
        cin >> n;
       }
    return n;
}
```

The delete function in the files creates an auxiliary file where all the contents of matches.txt will be copied, except what should be deleted. After the matches.txt file is deleted, the auxiliary file is renamed to matches.txt.

The auxiliary file is only created if the file can be opened and the name entered is actually found within matches.txt.

```
void deleteFile(){
    bool flag = false;
    string name = "";
    string line;
    fstream ficheiroAux;
    file.open("matches.txt",ios::in);
    cout<< "\t\tWhich player do you want to delete? Enter the name of the player [Attention: Please note this affects all matching names.More than 3 characters]\n"<<endl;
    cout<< "\t\tName: ";
    name = readPlayerName();
    if (file.is_open()){//open file
        while(getline(file,line)){
            if(line.find(name) != string::npos){
                flag=true;
            }
        }
        if (flag==true){
            ficheiroAux.open("aux.txt",ios::app);//just create aux.txt if the file was opened and found the name
            file.clear();
            file.seekp(0, ios::beg);
            while(getline(file,line)){//get line of the file
                if(line.find(name) == string::npos){//if the name was diferent then write in the file aux.txt, else do nothing
                    ficheiroAux << line << endl;
                }
            }
            cout<< "\t\tMatch deleted successfully!";
                file.close();
                ficheiroAux.close();
                remove("matches.txt");//delete previous file
                rename("aux.txt", "matches.txt");//rename the file aux.txt to the original name
        }else{
            cout<< "\t\tName not found. Make sure you typed correctly!" << endl;
            getch();
        }
    }else{
        cout<< "\t\tThere is no match added";
    }
    file.close();
    ficheiroAux.close();
    getch();
}
```