

**Data Structures and   
Object-oriented Programming (EC-204)**

**DE 44 Mechatronics**

**Syndicate – C**

**Project Report**

**“Tic-Tac-Toe Game”**

**Team Members:**

* NC Muhammad Ahmad Kashif

Reg # 415199

* NC Saim Raza

Reg # 412868

**Submitted to:**

Assoc Prof Dr. Tahir Nawaz

LE Hamza Sohail

**Table of Contents**

* **Abstract …...….……...……………………………………. 3**
* **Objectives …….……...……………………………………. 3**
* **Project Description .……………………………………….. 3**
* **Key Features ….…………………………………………… 4**
* **Code Explanation ....…………………………………….... 4**
* **Flowchart Diagram .………………………………………. 10**
* **Results and Observations  ………………………………... 11**
* **Conclusion ..………………………………………………. 12**

**Tic-Tac-Toe Game**

**Abstract:**

The Tic Tac Toe Game Project is an object-oriented programming (OOP) adaptation of the popular Tic Tac Toe game in C++. The project's goal is to deliver a simple and intuitive console-based gaming experience. The architecture stresses code modularity, reusability, and maintainability by leveraging inheritance and polymorphism principles. This paper delves into the development process, significant features, and insights obtained while making this C++ Tic Tac Toe game.

**Objective:**

The objective of the C++ Tic Tac Toe Game Project utilizing OOP ideas is to illustrate modular code design, encapsulation, and polymorphism. The project stresses accessibility, error management, and a clear game logic by developing a console-based interface. The goal is to present a realistic example of OOP principles in the form of a simple, instructive game.

**Project Description:**

The Tic Tac Toe Game Project is a console-based program that aims to recreate the popular Tic Tac Toe game. To structure the codebase, the major programming paradigm used is object-oriented, with an emphasis on encapsulation, inheritance, and polymorphism.

The game is played in a two-player mode, with players taking turns marking cells on a 3x3 grid. Classes are used in the project to model the game board, participants, and the game itself. Inheritance is used to construct specialized classes for various areas of the game, encouraging code reuse and establishing a clear hierarchy.

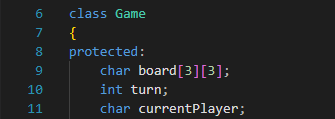
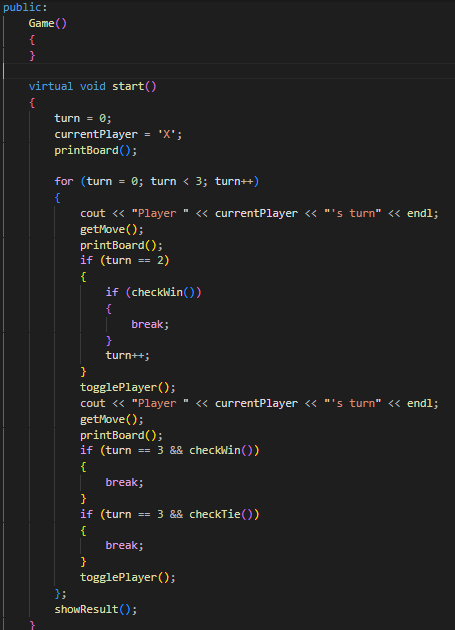
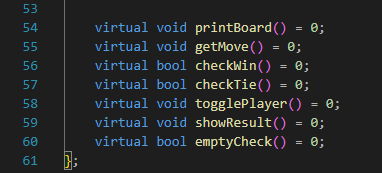
**Key Features:**

* **Modular Design:** The project is divided into classes that represent different elements like the game board, players, and the overall game. This modular approach improves code readability and allows for future changes or extensions.
* **Inheritance:** Class ties are established by inheritance. The base class captures general traits and behaviors, whereas derived classes inherit these aspects and add unique functions, hence encouraging code reuse.
* **Polymorphism:** Polymorphism is used to provide versatility in dealing with various game aspects. This allows the software to treat objects of different classes consistently, which simplifies implementation and improves maintainability.
* **Console Interaction:** The game is totally console-based, with a simple and user-friendly interface. Player input is recorded using standard input/output, assuring platform compatibility.
* **Game Logic:** To handle player actions, check for victory criteria, and control the overall flow of the game, the project includes a powerful game logic implementation. This logic is intended to be clear and efficient.
* **Error Handling:** The program provides error-handling features to resolve erroneous user inputs or unforeseen scenarios during gameplay, increasing the application's overall robustness.

**Code Explanation:**

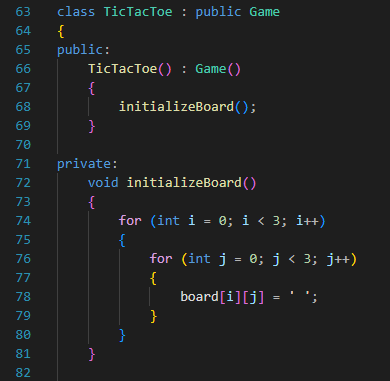
This C++ program implements a console-based Tic Tac Toe game using object-oriented programming (OOP) principles. Here's a breakdown of the code:

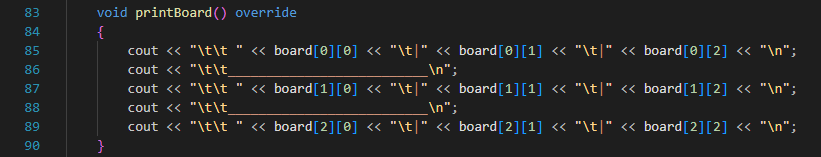
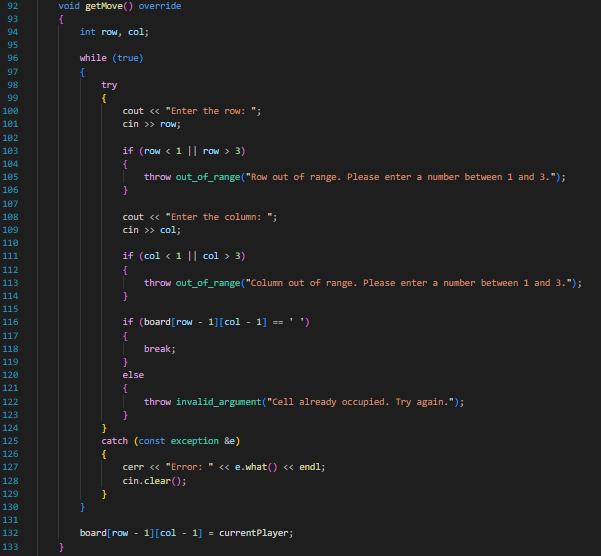
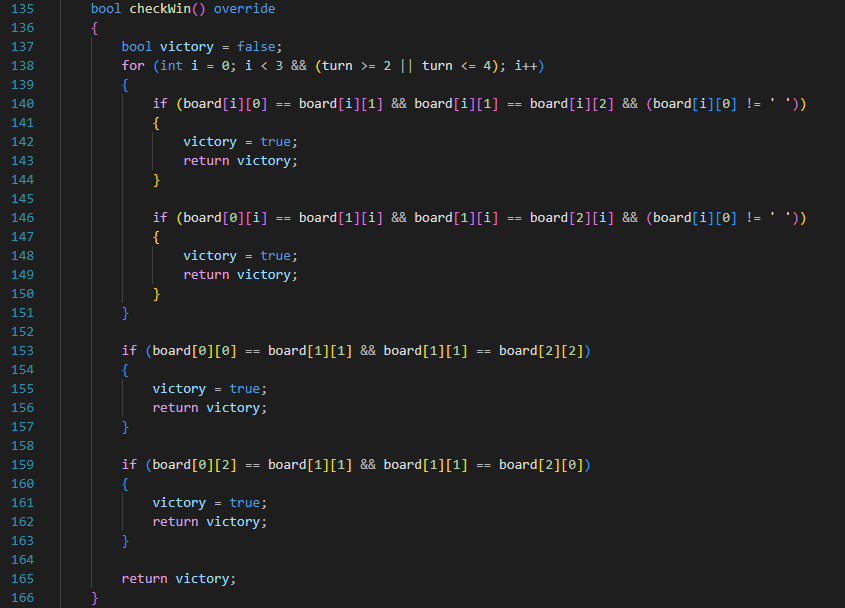
1. **Game Class:**

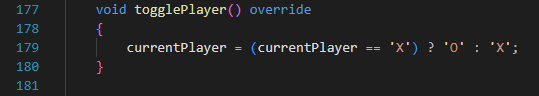
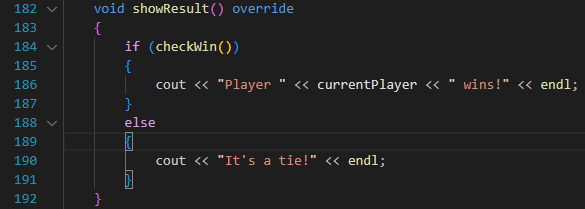
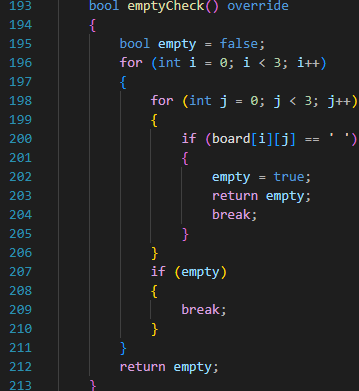
* **Attributes:**
* **board[3][3]:** Represents the Tic Tac Toe board.
* **turn:** Tracks the current turn.
* **currentPlayer:** Stores the symbol ('X' or 'O') of the current player.
* **Methods:**
* **start():** Initiates the game loop, taking turns between players until a win or tie occurs.
* **printBoard():** Pure virtual function to be implemented by derived classes. Displays the current state of the board.
* **getMove():** Pure virtual function to be implemented by derived classes. Collects player input for row and column and updates the board accordingly.
* **checkWin():** Pure virtual function to be implemented by derived classes. Checks if the current player has won.
* **checkTie():** Pure virtual function to be implemented by derived classes. Checks if the game is a tie.
* **togglePlayer():** Pure virtual function to be implemented by derived classes. Switches the current player between 'X' and 'O'.
* **showResult():** Pure virtual function to be implemented by derived classes. Displays the result of the game.

1. **TicTacToe Class (Derived from Game):**

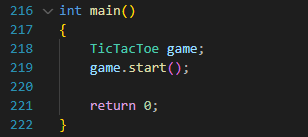
* **Constructor:**

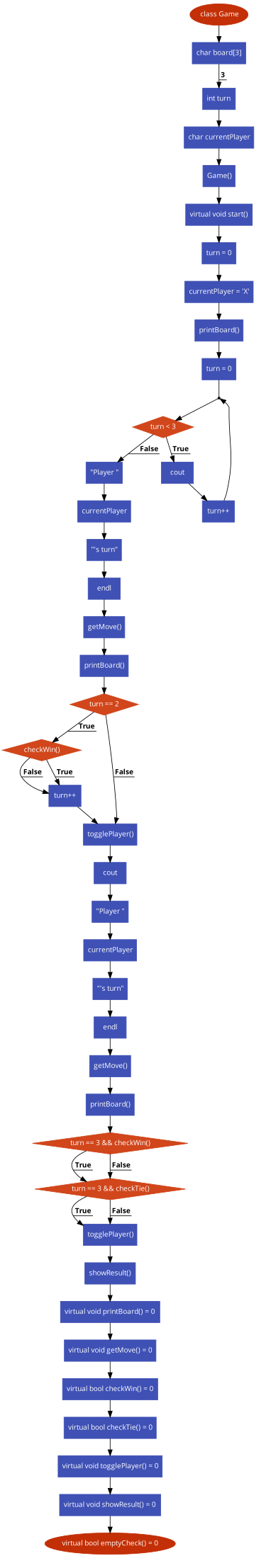
Initializes the Tic Tac Toe board using the **initializeBoard()** function.

* **Overridden Methods:**
* **printBoard():** Displays the Tic Tac Toe board with the current state.
* **getMove():** Collects player input for row and column, handling exceptions for invalid input.
* **checkWin():** Checks for a win by examining rows, columns, and diagonals.
* A computer screen shot of text

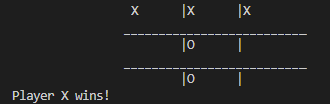
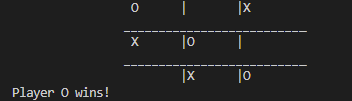
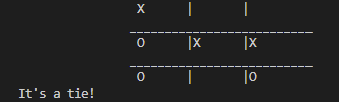
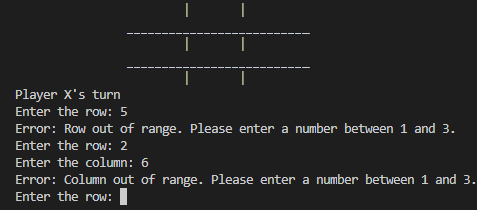
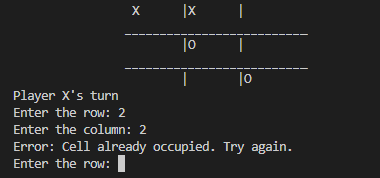
  Description automatically generated**checkTie():** Checks for a tie by verifying if there are no empty cells left.
* **togglePlayer():** Switches the current player between 'X' and 'O'.
* **showResult():** Displays the result of the game.
* **Additional Method:**
* **emptyCheck():** Checks if the board has any empty cells.

1. **Main Function:**

****Creates an instance of the **TicTacToe** class and initiates the game by calling **start()**.

**Flowchart Diagram:**

**Results and Observations:**

1. **Player X wins:**
2. **Player O wins:**
3. **Tied:**
4. **Error: Invalid Entry:**
5. **Error: Occupied Move:**

**Consclusion:**

In conclusion, this program shows how to construct a Tic Tac Toe game utilizing OOP concept such as inheritance and polymorphism. The game is played on a console, with participants taking turns entering their movements until a winner or a tie is established. Error management is included to ensure correct user input.