**F1 Trivia Game using OOPS in C++**

Project submitted to the

SRM University – AP, Andhra Pradesh

for the partial fulfillment of the requirements to award the degree of

**Bachelor of Technology/Master of Technology**

In

**Computer Science and Engineering**

**School of Engineering and Sciences**

Submitted by

**RAHUL VEMULA**

**(AP23110010571)**

**A picture containing text

Description automatically generated**

Under the Guidance of

**Kavitha Rani Karnena**

**SRM University–AP**

**Neerukonda, Mangalagiri, Guntur**

**Andhra Pradesh – 522 240**

**[November, 2024]**

# Certificate

Date: 20-Nov-24

This is to certify that the work present in this Project entitled “**F1 Trivia Game using OOPS in C++**” has been carried out by **Rahul, Purnesh, Subhash and Anil** under my supervision. The work is genuine, original, and suitable for submission to the SRM University – AP for the award of Bachelor of Technology/Master of Technology in **School of Engineering and Sciences**.

**Supervisor**

Kavitha Rani Karnena

(Signature)

Prof. / Dr. [Kavitha Rani Karnena]

# Acknowledgements

I would like to express my heartfelt gratitude to Professor Kavitha, my OOPS instructor, for her invaluable guidance and unwavering support throughout the course of this project. Her expertise in Object-Oriented Programming concepts and her ability to simplify complex topics have been instrumental in shaping my understanding of the subject.

Professor Kavitha's dedication to teaching and her enthusiasm for fostering innovative thinking inspired me to explore the principles of OOP in greater depth and apply them effectively in this project. Her constructive feedback and encouragement have been pivotal in ensuring the success of my work.

I am deeply grateful for her mentorship and the knowledge she imparted, which played a vital role in the completion of this project.

# Table of Contents

[Certificate i](#_gjdgxs)

[Acknowledgements iii](#_30j0zll)

[Table of Contents v](#_1fob9te)

[Abstract vii](#_3znysh7)

[Statement of Contributions xi](#_tyjcwt)

[Abbreviations xv](#_3dy6vkm)

[List of Tables xvii](#_1t3h5sf)

[List of Figures xix](#_4d34og8)

[List of Equations xxi](#_2s8eyo1)

[1.](#_17dp8vu) Introduction 1

[2.](#_lnxbz9) Methodology 3

[3.](#_44sinio) Discussion 5

[4.](#_2jxsxqh) Concluding Remarks 7

[5.](#_z337ya) Future Work 9

[References 11](#_3j2qqm3)

**Abstract**

The F1 Trivia Game is an interactive and competitive quiz application developed using Object-Oriented Programming (OOP) principles in C++. Designed for Formula 1 enthusiasts, this game allows two players to test their knowledge of the sport through a series of dynamic multiple-choice questions. Players earn points for each correct answer, and the one with the highest score at the end is declared the winner.

The project leverages core OOP principles such as inheritance, polymorphism, abstraction, and encapsulation to build a modular and extensible system. Two primary classes drive the game mechanics: the Question class (an abstract base class) and the MCQ class (derived from Question). The abstract base class defines a generic interface for questions, while the derived class implements specific functionalities, including displaying questions, checking answers, providing explanations, and enabling a 50/50 lifeline for players.

Dynamic memory allocation is utilized to manage a collection of quiz questions stored as objects, ensuring optimal resource usage and scalability. Robust exception handling mechanisms address invalid user inputs, maintaining the program's stability and user-friendliness. Additionally, randomization algorithms enhance the 50/50 lifeline feature, making the gameplay more engaging and unpredictable.

The application also focuses on delivering a polished user experience by presenting questions in a clear and organized manner, keeping players immersed in the competition. Through thoughtful design and meticulous implementation, the game ensures accuracy in scoring and fairness in gameplay.

This project highlights the effectiveness of OOP principles in creating reusable, organized, and efficient code structures. It offers a framework that can be easily extended in the future to incorporate additional features, such as new question types, more lifelines, multiplayer functionality, and graphical user interfaces. The F1 Trivia Game is not only a testament to the power of OOP but also an enjoyable platform for Formula 1 fans to deepen their connection with the sport.

**Statement of Contributions**

The successful completion of the F1 Trivia Game project was made possible through the combined efforts of all team members, with each individual contributing significantly to different aspects of the work. Below is the division of responsibilities:

1. Rahul
   * Designed and implemented the base class Question and derived class MCQ to structure the quiz.
   * Developed methods for displaying questions, validating user answers, and managing encapsulated data.
   * Ensured proper abstraction and inheritance in the class hierarchy.
2. Purnesh
   * Focused on memory management by implementing dynamic allocation and deallocation of objects.
   * Developed the randomization algorithm for the 50/50 lifeline using the <algorithm> and <random> libraries.
   * Worked on exception handling mechanisms to address invalid user inputs robustly.
3. Subhash
   * Designed and structured the main function to manage game flow, including player interactions and score tracking.
   * Integrated functionality for alternating turns between players and calculating final scores.
   * Ensured user-friendly output formatting for a polished user experience.
4. Anil
   * Conducted testing and debugging to identify and fix issues in the code.
   * Managed edge cases in gameplay, such as ties or invalid lifeline usage.
   * Prepared the documentation, including the algorithm, challenges faced, and conclusions, to provide a comprehensive project report.

Each member's contributions were vital in ensuring the project's success, and the teamwork demonstrated reflects a collaborative and balanced effort.

# Abbreviations

**OOP** – Object-Oriented Programming

**MCQ** – Multiple Choice Questions

**F1** – Formula One

**IDE** – Integrated Development Environment

**I/O** – Input/Output

# List of Tables

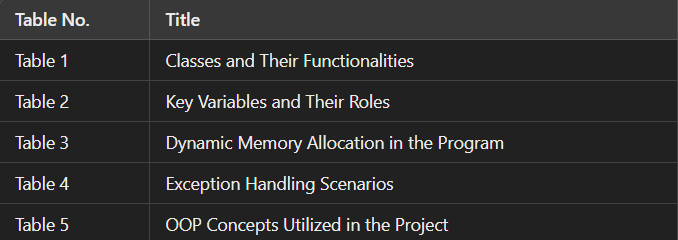


Table 1: Classes And Their Functionalities

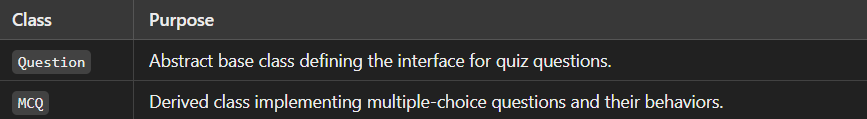


Table 2: Key Variables And Their Roles

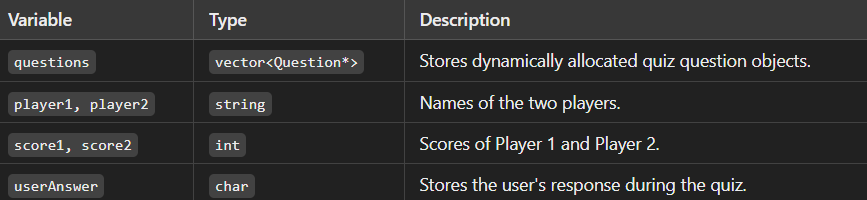


Table 3: Dynamic Memory Allocation in the Program

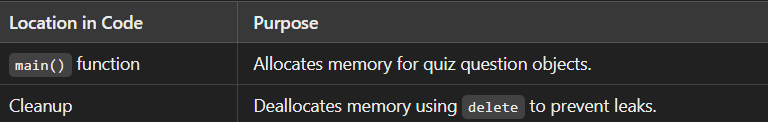


Table 4: Exception Handling Scenarios

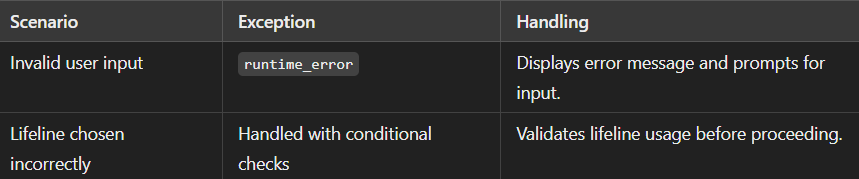
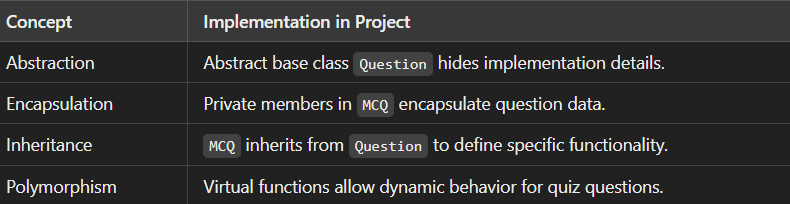
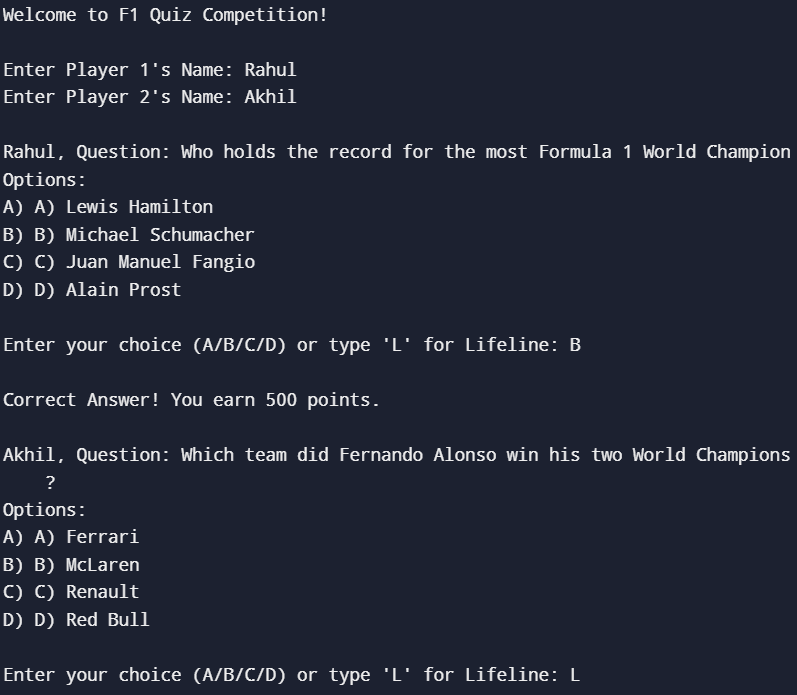
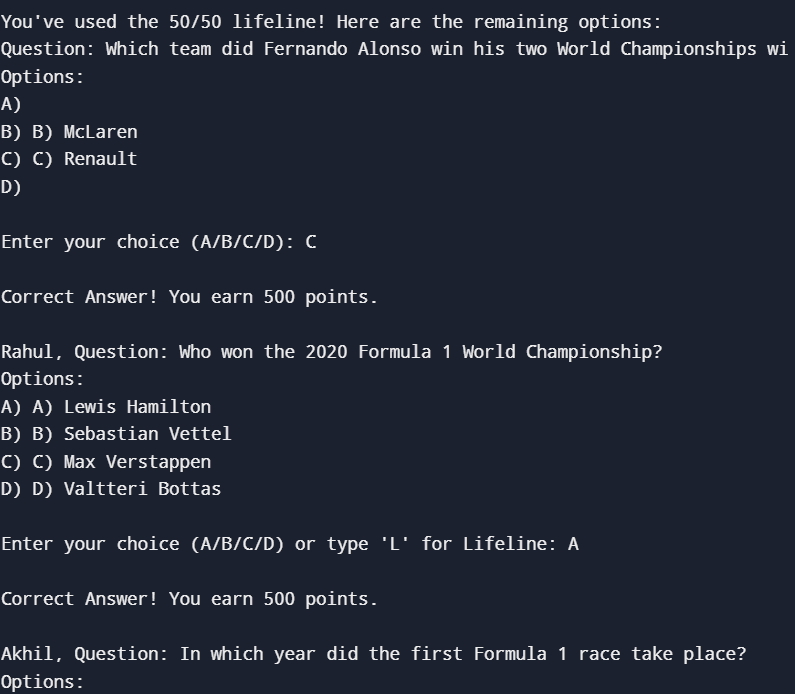


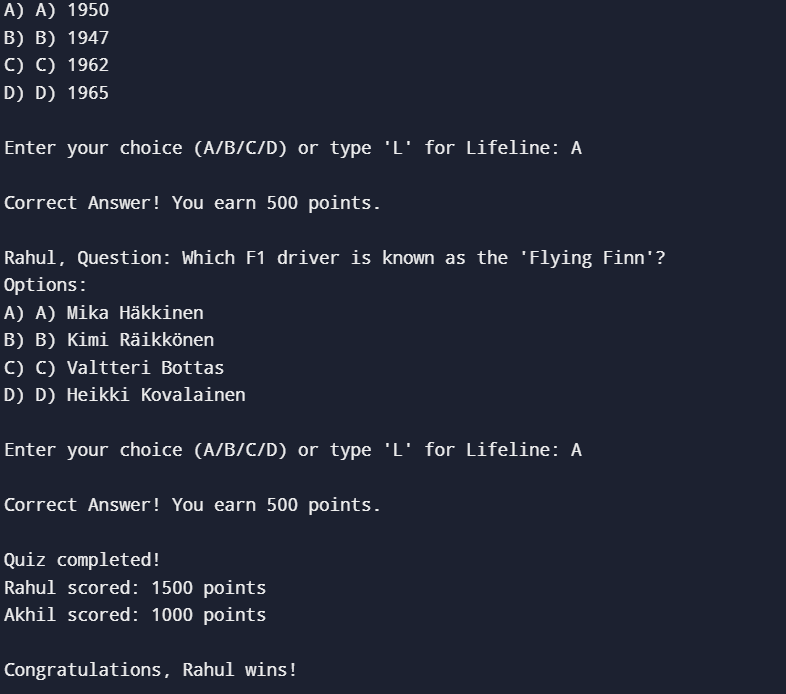
Table 5: OOP Concepts Utilized in the Project



# List of Figures







# List of Equations

1**. Equation 1: Score Calculation Formula**

* **Description: The formula used to update the player's score when a correct answer is given.**
* **Explanation: For every correct answer, 500 points are added to the player's score.**

**2. Equation 2: Validation of User Input**

* **Description: Logic used to ensure valid input (A, B, C, D, or L for the lifeline).**
* **Explanation: The input is checked against a set of valid characters to ensure no invalid selections disrupt the program flow.**

**3. Equation 3: Lifeline Logic for 50/50**

* **Description: Formula used to calculate the options retained after applying the 50/50 lifeline.**
* **Explanation: The lifeline removes two incorrect options, leaving one correct answer and one incorrect option visible.**

# Introduction

The F1 Trivia Game is a unique quiz developed as a practical demonstration of Object-Oriented Programming (OOP) principles. The game offers an engaging platform for Formula 1 fans to challenge their knowledge in a competitive setting. With two players participating, the game alternates turns, presenting questions in a multiple-choice format and tracking scores in real-time.

This project is designed to emphasize the benefits of OOP in software development, particularly its ability to create modular, reusable, and scalable solutions. Key features include an abstract base class (Question) that defines the general structure for all quiz questions and a derived class (MCQ) that implements specific functionalities. These classes encapsulate the core logic for displaying questions, validating answers, providing explanations, and offering a 50/50 lifeline.

Dynamic memory management plays a significant role in the project, as quiz questions are allocated and deallocated efficiently to optimize resource usage. Furthermore, robust exception handling ensures smooth gameplay by validating user inputs and addressing errors gracefully. The 50/50 lifeline leverages randomization techniques to remove two incorrect answers, adding an element of strategy to the game.

The F1 Trivia Game not only provides an entertaining experience for players but also showcases the application of OOP principles in solving real-world problems. By breaking down the system into small, manageable components, the project highlights how abstraction and encapsulation can simplify complex functionality. The modular design lays the foundation for future enhancements, such as incorporating different question types, multiplayer options, or a graphical user interface.

This project is a testament to the practicality and effectiveness of OOP in developing interactive and user-friendly applications, making it an ideal tool for both learning and entertainment

# Methodology

The methodology for the F1 Trivia Game project outlines the systematic approach adopted to design, implement, and test the game. The project is built using Object-Oriented Programming (OOP) principles in C++ to ensure modularity, scalability, and maintainability. Below are the key steps involved:

**1. Problem Analysis and Requirements Gathering**

* Defined the core functionality: presenting questions, validating answers, tracking scores, and determining the winner.
* Established the use of OOP principles to structure the program efficiently.

**2. Design and Class Structure**

* Abstract Class Question: Defined the base class for quiz questions with pure virtual functions for question display, answer checking, explanation retrieval, and the 50/50 lifeline.
* Derived Class MCQ: Implemented specific functionalities, including storing the question text, answer options, the correct answer, and its explanation. Overrode virtual functions to handle question logic.
* Main Function: Designed to manage the game flow, alternating between players, collecting inputs, and calculating scores.

**3. Dynamic Memory Management**

* Utilized dynamic memory allocation to store questions as objects in a vector.
* Ensured proper cleanup by deallocating memory at the end of the program to avoid leaks.

**4. Error Handling and Input Validation**

* Implemented exception handling using try-catch blocks to manage invalid inputs (e.g., characters outside A-D or L).
* Ensured smooth gameplay by prompting users to re-enter valid inputs when errors occurred.

**6. Documentation and Reporting**

* Documented the codebase, including class descriptions, method explanations, and game flow.
* Prepared the project report, detailing the design, challenges faced, and solutions implemented.

# Discussion

* 1. **Effectiveness of OOP Design**

The use of OOP concepts, including abstraction, encapsulation, inheritance, and polymorphism, proved to be highly effective in simplifying and organizing the program. The Question class, serving as an abstract base class, provided a clear interface for defining quiz questions. The MCQ class, derived from Question, encapsulated question data and implemented functionalities such as displaying questions, validating answers, and providing explanations. This modular design allowed the addition of new question types or features without significant changes to the existing codebase.

* 1. Challenges Encountered
* **Designing the 50/50 Lifeline**  
  Implementing the 50/50 lifeline required careful use of randomization techniques to remove incorrect options without affecting the correct answer. Ensuring randomness and fairness added complexity to this feature.
* **Input Validation and Error Handling**  
  Handling invalid user inputs (e.g., characters outside A-D or incorrect lifeline usage) posed challenges in maintaining program robustness. The implementation of exception handling mechanisms resolved these issues effectively.
* **Balancing Gameplay**  
  Creating a fair and competitive experience for both players required alternating turns, consistent scoring mechanisms, and clear feedback for correct and incorrect answers.

# Concluding Remarks

The development of the **F1 Trivia Game** using Object-Oriented Programming (OOP) principles in C++ has demonstrated a robust and interactive approach to quiz game design. By employing fundamental OOP concepts such as inheritance, polymorphism, encapsulation, and abstraction, the project encapsulates essential functionalities in a modular and maintainable code structure.

The use of dynamic memory allocation and exception handling ensures efficient resource management and user-friendly error feedback. Additionally, features like the 50/50 lifeline and dynamic question presentation showcase the flexibility and scalability of the system. This project not only highlights the versatility of OOP in solving real-world problems but also serves as an engaging platform for enhancing knowledge about Formula 1 racing.

Through iterative testing and debugging, challenges such as error handling and algorithm design were addressed effectively, ensuring a smooth and engaging user experience. This project stands as a testament to the successful application of OOP principles in creating practical and interactive software solutions.

# Future Work

The current implementation of the **F1 Trivia Game** provides a strong foundation for an engaging quiz platform. However, there are several areas for potential enhancement and expansion:

1. **Database Integration**:
   * Incorporate a database to store a larger pool of questions, allowing for dynamic updates and a wider variety of quizzes.
   * Enable persistent storage of player scores and statistics for long-term tracking and competition.
2. **Multiplayer Online Mode**:
   * Extend the game to support online multiplayer functionality, enabling players from different locations to participate and compete in real time.
3. **Advanced Question Features**:
   * Add different types of questions, such as true/false or fill-in-the-blank formats, to increase the diversity of the quiz.
   * Implement additional lifelines, such as "Ask an Expert" or "Skip the Question," to enhance interactivity.
4. **Enhanced User Interface**:
   * Develop a graphical user interface (GUI) using frameworks like Qt or SFML to make the game visually appealing and more user-friendly.
   * Integrate animations and sound effects to improve the overall user experience.
5. **Artificial Intelligence Integration**:
   * Use AI to adapt the difficulty level of questions based on the player’s performance, creating a personalized experience for each user.

# References

1. www.geeksforgeeks.org
2. www.stackoverflow.com
3. www.w3schools.com

\*\*THE END\*\*