A PROJECT

ON

“QUIZ OF QUSTIONS”

Submitted by :

Nutan Surwase

B.Tech (computer science Engineering)

First Year

Academic Year: 2025-26

INTRODUCTION

The project I have developed is a **Quiz Application** using the **C programming language**. The purpose of this project is to create a simple, interactive console-based program where users can participate in a multiple-choice quiz and immediately receive feedback about their performance. This project not only helps the user test their knowledge but also demonstrates my understanding of core programming concepts such as **arrays, loops, conditional statements, functions, and file handling**.

In today's digital world, quizzes and interactive learning platforms have become an essential tool for education and skill assessment. This project simulates a small portion of such platforms in a console environment, providing a clear demonstration of structured programming. The program is designed to be **user-friendly**, allowing participants to answer questions, receive immediate feedback, and see their overall performance in terms of scores and percentages. Additionally, the program evaluates whether the user has passed or failed based on their performance and records the results in a text file for future reference.

The program uses **arrays** to organize data efficiently. All questions, options, and correct answers are stored in arrays, which makes it easy to manage and retrieve data during the quiz. A loop iterates through each question and displays it along with its multiple-choice options. The user inputs their answer, which is then compared against the correct answer stored in the array. This approach demonstrates the use of structured data and efficient programming logic. Using loops not only reduces code repetition but also ensures that the program can handle multiple questions in a scalable manner.

To enhance the program's functionality, I have implemented **conditional statements** to check the correctness of each answer. For each question, the program evaluates whether the user's input matches the correct answer. If the answer is correct, the user receives points; if not, the program informs the user that the answer is incorrect. This real-time evaluation keeps the user engaged and provides immediate feedback, which is an important feature of any quiz application. The use of **functions** helps organize the program into logical sections. A separate function is responsible for starting the quiz, calculating scores, determining the result, and saving the data to a file. This modular approach makes the code easier to read, debug, and maintain.

An important aspect of this project is **file handling**. The program saves the user's score, percentage, and pass/fail status into a text file named quiz.txt. File handling demonstrates the ability to **store data persistently**, allowing results to be reviewed at any time. Using append mode ensures that multiple attempts by different users can be recorded without overwriting previous results. This not only makes the program more practical but also introduces an important programming concept that is widely used in software development.

In addition to technical aspects, this project also demonstrates good **user interaction**. The menu provides clear options for starting the quiz or exiting the program. At the end of the quiz, users are asked whether they want to play again, giving them control over the program's flow.

The score, percentage, and pass/fail evaluation are displayed in a clear and organized manner, making the application easy to use even for beginners. This attention to detail shows an understanding of how programming and user experience intersect.

The Quiz Application also serves an educational purpose. It encourages **logical thinking, problem-solving, and attention to detail**. While developing this project, I learned how to organize data using arrays, write efficient loops, implement conditional logic, and handle files. This experience has strengthened my programming skills and prepared me to tackle larger and more complex projects in the future.

Overall, this project is a simple yet complete demonstration of my ability to develop structured and functional programs in C. It combines multiple core concepts such as **data storage, input validation, control flow, modular programming, and file handling**. By completing this project, I have not only created a working application but also reinforced my understanding of essential programming practices. It is a practical example of how theoretical knowledge gained during study can be applied to create functional software that is both useful and interactive.

OBJECTIVES

The main objectives of this Quiz Application project are as follows:

To develop an interactive quiz platform:

The primary objective is to create a program that allows users to participate in a quiz and answer multiple-choice questions. By providing real-time feedback on each answer, the program aims to engage users and make the learning process interactive and enjoyable.

To implement core programming concepts effectively:

This project provides an opportunity to apply fundamental programming concepts such as arrays, loops, conditional statements, and functions. By using these concepts in a practical scenario, I aim to strengthen my understanding of structured programming in C.

To utilize file handling for data persistence:

Another important objective is to store user results in a file named quiz.txt. This allows for persistent storage of quiz data, including score, percentage, and pass/fail status. Users can later refer to this file to track their performance over multiple attempts.

To enhance problem-solving and logical thinking:

Developing the quiz application requires careful planning of program logic, decision-making, and handling user input correctly. This helps improve my analytical skills and ability to think logically while designing efficient solutions.

To practice modular programming and function usage:

The program is divided into multiple functions such as quiz start, score calculation, result display, and file handling. This modular approach ensures that the code is organized, readable, and maintainable, which is an essential skill for larger software projects.

To improve user experience and interaction:

A clear and simple menu system allows users to navigate the program easily. Options like Start Quiz, Exit, and Replay Quiz are implemented to make the program intuitive. Prompting users clearly and providing immediate feedback ensures a positive user experience.

To strengthen data management skills using arrays:

The program uses arrays to store questions, options, and correct answers, which demonstrates how data can be organized efficiently. By iterating through arrays using loops, the program dynamically handles multiple questions and provides a scalable solution.

To understand and apply percentage calculation and result evaluation:

The project calculates the score and percentage of correct answers, and determines whether the user has passed or failed. This objective demonstrates how programming can be applied to evaluate performance systematically and accurately.

To gain experience in project development and testing:

By developing, testing, and refining this project, I learn how to plan, implement, and troubleshoot software effectively. Testing includes checking correct answers, validating user input, and ensuring the results are stored properly in a file.

To prepare for future programming and software development tasks:

This project serves as a foundation for more complex applications. By combining multiple programming concepts in one project, I build confidence and practical skills necessary for advanced projects and professional software development.

MAIN FEATURES

The Quiz Application developed in C has the following main features:

Menu-Driven Interface:

The program starts with a simple and clear menu that allows the user to either start the quiz or exit the application.

This makes the program easy to navigate and user-friendly, even for beginners.

Multiple-Choice Questions:

The quiz consists of multiple-choice questions where each question has four options.

Users select the correct option by entering the corresponding character (A, B, C, or D).

Dynamic Question Display Using Arrays:

Questions, options, and correct answers are stored in arrays.

Using loops, the program dynamically displays each question and its options in a structured format.

Real-Time Answer Evaluation:

As the user enters their answers, the program immediately checks whether the answer is correct or incorrect.

Feedback is provided instantly, which keeps the user engaged throughout the quiz.

Score Calculation:

The program calculates the total number of correct answers to determine the user’s score.

Scores are displayed immediately after the quiz is completed.

Percentage Calculation:

In addition to the score, the program calculates the user’s percentage based on the number of correct answers.

The percentage provides a more detailed measure of performance.

Pass/Fail Evaluation:

The program determines whether the user has passed or failed based on their score or percentage.

This feature provides a clear assessment of the user’s performance.

Replay Option:

After completing the quiz, the program asks the user whether they want to play again.

If the user chooses yes, the quiz restarts; otherwise, the program returns to the main menu.

File Handling to Store Results:

All quiz results, including score, percentage, and pass/fail status, are stored in a text file named quiz.txt.

This ensures that the user’s performance is recorded for future reference and multiple attempts are maintained.

Error Handling and Input Validation:

The program validates user input to ensure that only valid choices (A, B, C, D) are accepted.

Invalid inputs are handled gracefully, prompting the user to enter a valid option.

Modular Design Using Functions:

The program is organized into multiple functions such as startQuiz, score calculation, result display, and file handling.

This makes the code easier to read, debug, and maintain.

Educational Purpose:

This program serves as a learning tool for users to test their knowledge.

It also provides practical experience in programming concepts like arrays, loops, conditional statements, and file handling.

TECHNOLOGIES USED

The Quiz Application is developed using C programming language, which is a widely used language for system programming and developing console-based applications. The technologies and tools used in this project include:

C Programming Language:

The entire project is implemented in C, which provides a strong foundation in structured programming.

Core concepts of C such as loops, conditional statements, arrays, functions, and file handling are extensively used in this project.

Code Editor / IDE:

The program can be developed using any C compiler or Integrated Development Environment (IDE) such as Code::Blocks, Dev-C++, or Turbo C.

These IDEs provide features like syntax highlighting, debugging, and easy compilation to simplify the development process.

File Handling:

The project uses text files (quiz.txt) to store the results of the quiz.

Functions like fopen, fprintf, and fclose are used to create, write, and close files.

File handling allows persistent storage of user performance, which is a key feature in real-world applications.

Arrays:

Arrays are used to store questions, options, and correct answers.

This approach makes data management easier and allows dynamic iteration through all quiz questions using loops.

Loops and Conditional Statements:

For loops are used to iterate over questions and options efficiently.

If-else statements are used to check the correctness of answers and determine pass/fail status.

Functions:

Functions are used to organize the program into logical sections such as startQuiz, calculateScore, and displayResults.

Using functions improves readability, maintainability, and modularity of the program.

Console Input and Output:

The project uses printf and scanf for displaying questions, options, and results, and for accepting user input.

Console I/O is suitable for simple projects like this and helps beginners understand basic programming interaction.

Basic Data Types:

The project uses data types such as int for score counting, float for percentage calculation, and char for storing user responses and options.

Correct choice of data types ensures accurate calculation and efficient memory usage.

Cross-Platform Compatibility:

Since the program is written in standard C, it can be compiled and run on multiple platforms including Windows, Linux, and macOS.

This enhances the usability and reach of the project

CONCEPTS APPLIED

In this Quiz Application project, several fundamental programming concepts and techniques have been applied to ensure the program works efficiently and correctly. The main concepts used are as follows:

Arrays:

Arrays are used to store the questions, multiple-choice options, and correct answers.

This allows for organized data storage and easy retrieval during the quiz.

Using arrays ensures the program can handle multiple questions without repeating code unnecessarily.

Loops:

For loops are used to iterate through all the questions and options.

Do-while loops are used to allow users to replay the quiz or return to the menu until they choose to exit.

Loops help in reducing repetitive code and make the program scalable.

Conditional Statements:

If-else statements are used to check whether the user's answer is correct or incorrect.

Conditional statements are also used to determine the pass/fail status based on the user's percentage.

Functions:

The program is divided into multiple functions for modularity and better organization.

Key functions include:

startQuiz() – Handles quiz execution

File Handling:

File handling is applied to store the results of each quiz attempt in a text file (quiz.txt).

Functions like fopen, fprintf, and fclose are used to write data persistently so that results can be accessed later.

Input Validation:

The program validates user input to ensure only valid options (A, B, C, D) are accepted.

Invalid inputs are handled gracefully, prompting the user to enter a correct value.

Type Casting and Data Types:

The program uses float to calculate percentage accurately.

int is used for counting scores, and char is used for storing answers.

Type casting ensures that arithmetic operations provide precise results.

Modular Programming:

The program follows modular design principles, separating tasks into functions to improve readability, maintainability, and reusability.

User Interaction:

The program applies user interaction concepts through menu-driven input and instant feedback.

Users are asked if they want to replay the quiz, enhancing engagement and control over the program.

Problem-Solving Logic:

Logical thinking is applied to determine correct answers, calculate scores, evaluate pass/fail, and store results in a structured manner.

This ensures the program works reliably under different scenarios.

PROGRAM STRUCTURE

The Quiz Application is designed using a modular programming approach, which means the program is divided into separate sections or functions to improve readability, maintainability, and efficiency. The structure of the program is as follows:

Main Function (main)

The main function serves as the entry point of the program.

It displays a menu to the user with two primary options:

Start Quiz

Exit Program

Using a do-while loop, the menu keeps appearing until the user chooses to exit.

The main function handles the user's choice using a switch-case statement, directing the program to either start the quiz or terminate.

Start Quiz Function (startQuiz)

This function contains the core logic of the quiz.

It displays all questions and their corresponding options using arrays.

A for loop is used to iterate through all the questions and display them sequentially.

Users input their answers, which are validated and compared to the correct answer array.

Each correct answer increases the score.

Score and Result Calculation

Within the quiz function, the program calculates the total score by counting correct answers.

The percentage is calculated by dividing the score by the total number of questions.

A pass/fail status is determined based on the percentage.

Replay Option

After displaying the results, the program asks the user whether they want to play again.

If the user chooses yes, the quiz restarts; otherwise, the program returns to the main menu.

File Handling Function

The user's score, percentage, and pass/fail status are saved in a text file named quiz.txt.

File handling ensures that all quiz attempts are recorded, providing persistent storage for future reference.

Validation and Error Handling

The program ensures that only valid input is accepted (A, B, C, D).

Invalid inputs prompt the user to enter the answer again.

Data Storage Using Arrays

Question Array: Stores all quiz questions.

Option Array: Stores four multiple-choice options for each question.

Answer Array: Stores the correct answer for each question.

Arrays make the program scalable and easy to manage.

OUTPUT DESCRIPTION

Menu Display

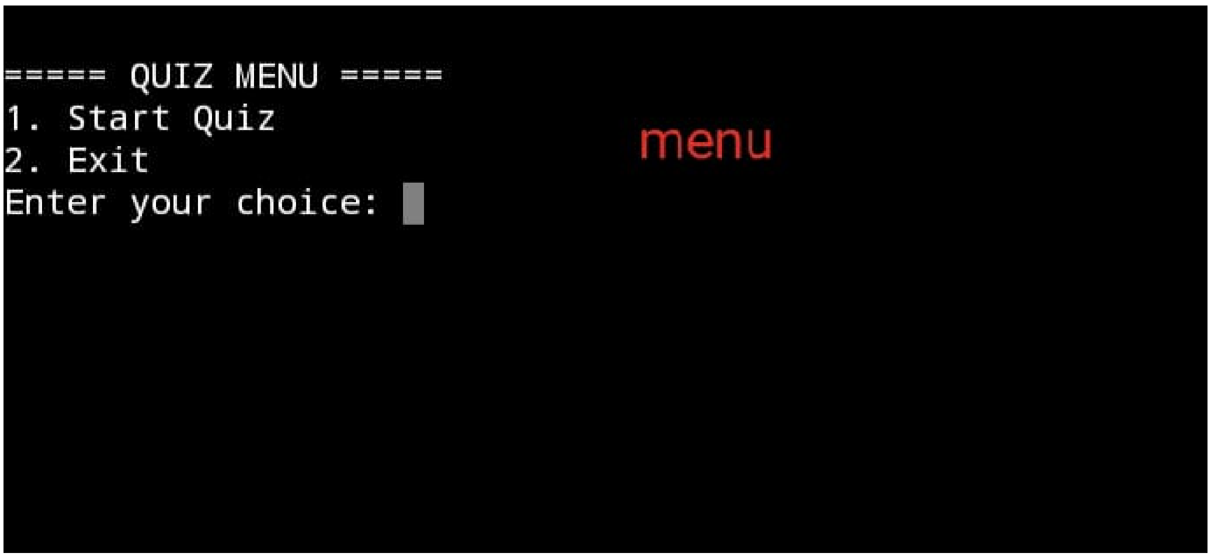
When the program starts, a menu appears on the screen.

The menu has two options:

Start Quiz

Exit

The user can select the option by entering the corresponding number or character.



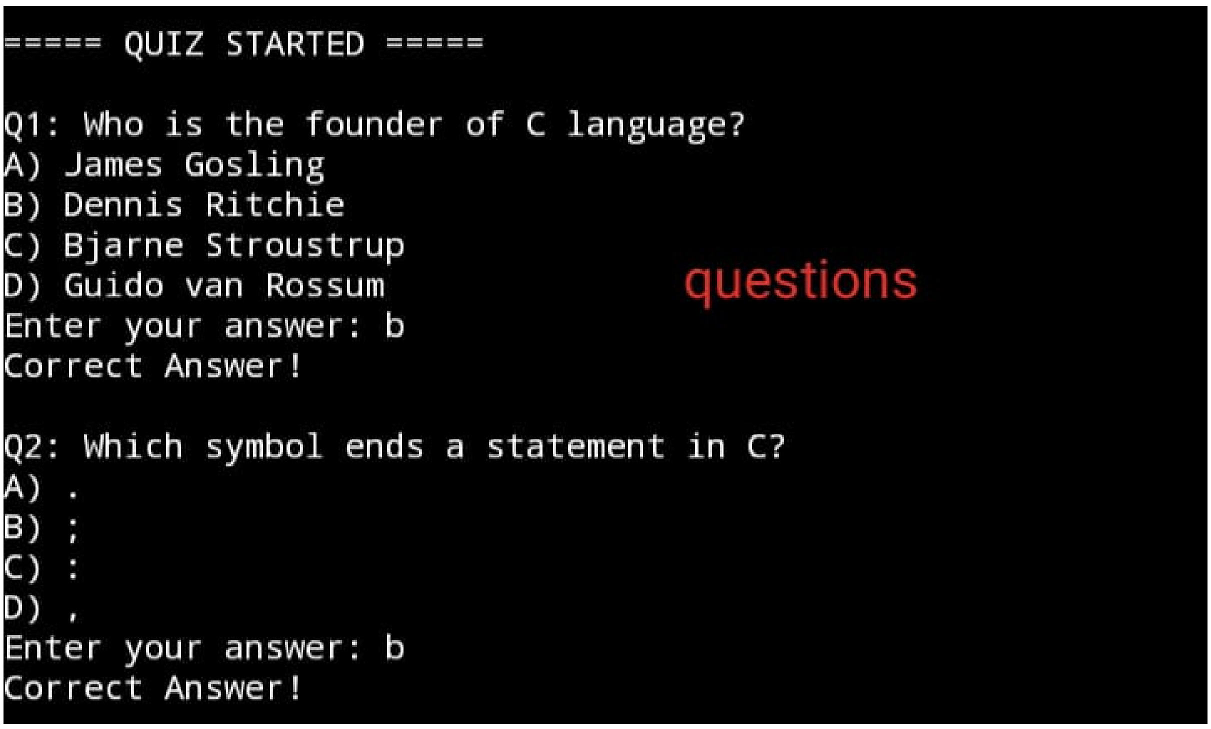
```
===== QUIZ MENU =====
1. Start Quiz
2. Exit
Enter your choice:
```

Question Display

If the user chooses Start Quiz, the program displays each question one by one.

For every question, all four possible options are shown clearly.

Questions and options are stored in arrays and displayed using a loop.



```
===== QUIZ STARTED =====

Q1: Who is the founder of C language?
A) James Gosling
B) Dennis Ritchie
C) Bjarne Stroustrup
D) Guido van Rossum
Enter your answer: b
Correct Answer!

Q2: Which symbol ends a statement in C?
A) .
B) ;
C) :
D) ,
Enter your answer: b
Correct Answer!
```

Answer Input

The user inputs their answer for each question.

The program accepts the input and compares it with the correct answer stored in the correct answer array.

If the answer is correct, the score is increased by 1, and a message “Answer correct” is displayed.

If the answer is wrong, the program simply moves to the next question.

Scoreboard

After all questions are answered, the program calculates the total score.

The program also calculates the percentage using the formula:

Percentage

Total Questions

Score

The program then displays:

Total Score: Number of correct answers out of total questions

Percentage: Percentage of correct answers

Status: Pass or Fail (Pass if percentage ⩾ 50%, Fail otherwise)

File Output

The program opens a file named quiz.txt.

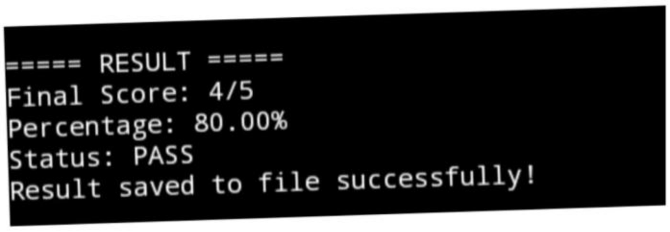
All results are stored in the file, including:

Score

Percentage

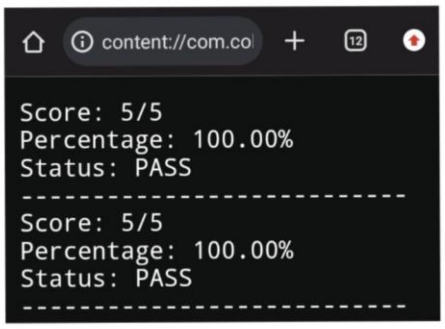
Status (Pass/Fail)

This ensures that the quiz results are saved permanently and can be viewed later.



```
===== RESULT =====
Final Score: 4/5
Percentage: 80.00%
Status: PASS
Result saved to file successfully!
```

print on screen



```
Score: 5/5
Percentage: 100.00%
Status: PASS
---------------------------
Score: 5/5
Percentage: 100.00%
Status: PASS
---------------------------
```

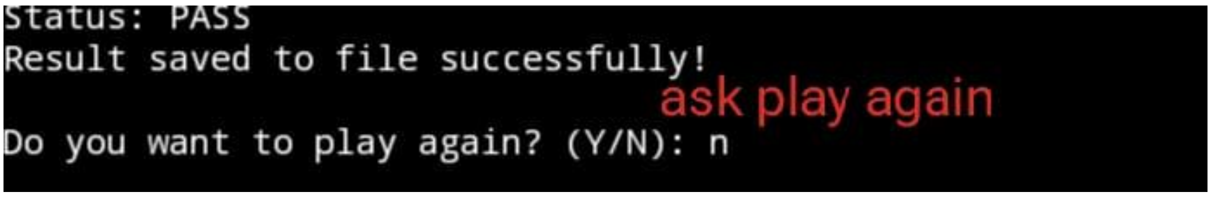
print in file

Replay Option

After showing the scoreboard and saving the result in the file, the program asks the user if they want to play the quiz again.

If yes, the quiz restarts from the menu.

If no, the program exits.



```
Status: PASS
Result saved to file successfully!

Do you want to play again? (Y/N): n
```

CONCLUSION

In this project, I successfully developed a Quiz program using the C programming language, which allows users to answer multiple-choice questions, calculates their score and percentage, and determines their pass or fail status. The program demonstrates the use of arrays for storing questions, options, and correct answers, as well as file handling to save results permanently. By implementing loops, conditional statements, and input/output functions, the program provides an interactive and user-friendly interface. This project not only strengthens programming logic and problem-solving skills but also gives practical experience in creating functional software that can store and manage data efficiently.