

C Quiz Game Project

Console Based Quiz Game with Score Tracking and Replay Functionality

Author

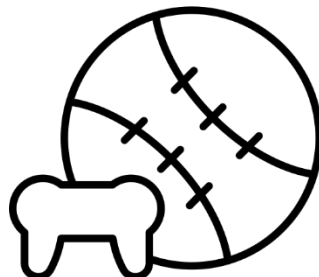
Deepayan Das

<https://github.com/Deepayan80/Quiz-Game>

Project Version: - 1.0

Date of Completion: - May 2025

Description: - This document presents the planning, development process, and technical overview of a Quiz Game built using the C programming language. It includes feature breakdowns, code structure, development workflow, and lessons learned.



Acknowledgement

I would like to take this opportunity to express my sincere gratitude for the guidance, support, and motivation that helped me complete this project successfully.

This quiz game project was a valuable learning experience, allowing me to apply and strengthen my knowledge of the C programming language. I am thankful to the online learning resources, open-source community, and developer forums that provided valuable insights during the development process.

I would also like to thank my peers and well-wishers who encouraged me to document this project in the form of a thesis for future reference and learning.

Lastly, I acknowledge my own curiosity and dedication, which drove me to complete this project with enthusiasm and focus.

- Deepayan Das

Abstract

This project is a console-based quiz game developed in the C programming language, designed to test users' knowledge through a series of multiple-choice questions. The program allows users to start a quiz, answer questions, receive immediate feedback, and view their final score. It features input validation, score tracking, and a simple user interface for an engaging experience. Built entirely from scratch, the project emphasizes fundamental C concepts such as arrays, functions, conditionals, loops, and structured programming. It serves as both a learning exercise and a portfolio piece, showcasing the practical application of core programming skills.

What is the Quiz Game?

The Quiz Game is a simple, interactive console-based program developed in the C programming language. It presents users with a set of multiple-choice questions, one at a time, and allows them to select their answers by entering a corresponding option. The game evaluates each response, keeps track of the score, and displays the result at the end. It includes features like user-friendly prompts, input validation, and an option to replay or exit the game. This project is designed to reinforce basic C programming skills such as functions, arrays, loops, and conditionals while offering an engaging user experience.

Why I Built it?

I built this quiz game project to strengthen my understanding of the C programming language through a hands-on, goal-oriented application. Rather than only focusing on theoretical concepts, I wanted to challenge myself with a real project that would involve logic building, user interaction, and basic data handling. Creating this game allowed me to apply core programming principles like functions, arrays, and control structures in a practical context. Additionally, I aimed to develop a project that could be added to my portfolio, demonstrating both my learning progress and problem-solving abilities. This project also helped me build discipline in planning, structuring, and completing a self-driven coding task.

Key Features in Brief

- ❑ **Interactive Menu:** The game starts with a clear menu allowing users to start the quiz or exit the program.
- ❑ **Multiple-Choice Questions:** Each question offers multiple options, and the user selects their answer by entering a choice.

- ❑ **Score Tracking:** The program automatically tracks the user's score based on correct answers.
- ❑ **Instant Feedback:** Users receive immediate confirmation on whether their answer was right or wrong.
- ❑ **Structured Code:** The program is modular, with separate functions for each major task, improving readability and maintainability.
- ❑ **Replay Option:** After completing a quiz, users have the option to play again without restarting the program.
- ❑ **Input Validation:** Handles invalid input gracefully to ensure a smooth user experience.

Table of Content

| | |
|---|-----------|
| 1. Introduction..... | 6 |
| 1.1 About Quiz Games..... | 6 |
| 1.2 Importance for Building Small Games for Learning | |
| C Programming..... | 6 |
| 1.3 What I Aim to Achieve with this Project..... | 7 |
| 2. Objective of the Project..... | 8 |
| 3. System Requirements..... | 8 |
| 4. Project Description..... | 10 |
| 4.1. Working Flow..... | 10 |
| 4.2 Modules/Function..... | 11 |
| 5. Implementation Details..... | 13 |
| 6. Sample Code Snippet..... | 14 |
| 7. Testing and Output Screenshot..... | 15 |
| 8. Challenges Faced..... | 20 |
| 9. Future Improvements..... | 21 |
| 10. Conclusion..... | 22 |
| 11. References..... | 23 |

Introduction

Programming is best learned through building, and small projects often serve as effective tools for reinforcing core concepts. This thesis presents the design and development of a console-based quiz game using the C programming language. The project was created as a personal initiative to apply fundamental programming skills in a practical setting. The game is designed to engage users with a series of multiple-choice questions, test their responses, and provide feedback along with a final score. Through this project, concepts such as input handling, control structures, functions, and arrays were explored and implemented. The quiz game not only helped in enhancing technical skills but also introduced the importance of structured thinking and code organization. This document outlines the step-by-step development process, key features, and the learning outcomes derived from the project.

1.1 About Quiz Games

Quiz games are interactive programs or activities designed to test a user's knowledge across various topics through a series of questions. Typically structured as multiple-choice or true/false formats, these games challenge users to think quickly and accurately. They are widely used in education, entertainment, and training to make learning more engaging and to assess knowledge in a fun and competitive way. With the rise of digital platforms, quiz games have become increasingly popular in both online and offline formats. They encourage participation, reinforce learning, and can be adapted to suit audiences of all ages. Whether for casual play or formal assessment, quiz games remain a simple yet powerful tool for knowledge evaluation.

1.2 Importance for Building Small Games for Learning C Programming

Building small games is an effective and practical approach to learning C programming. Games often require the use of fundamental concepts such as variables, loops, conditionals, functions, arrays, and user input — all of which are essential to mastering the language. Creating a game project like a quiz not only helps reinforce these topics but also improves problem-solving and logical thinking skills. It transforms theoretical learning into hands-on experience, making it easier to understand how different parts of a program interact. Additionally, small games provide instant visual feedback and a sense of achievement, which keeps the learning process engaging and motivating. For beginners, such projects lay a strong foundation for tackling more advanced programming challenges in the future.

1.3 What I Aim to Achieve with this Project

With this project, my primary goal is to strengthen my foundational knowledge of the C programming language by applying it to a real-world task. I aim to gain hands-on experience in designing, structuring, and implementing a functional program from scratch. Through building the quiz game, I wanted to improve my understanding of core programming concepts such as input/output handling, control flow, functions, arrays, and user interaction. Additionally, I aimed to develop a complete, presentable project that could be included in my portfolio to showcase my skills and dedication to learning. This project also serves as a personal milestone in my journey toward becoming a more confident and capable programmer.

Objectives of the Project

The primary objectives of this quiz game project are as follows:

1. **Apply Core Concepts of C Programming:** To implement fundamental C programming constructs such as loops, conditional statements, functions, arrays, and user-defined data structures.
2. **Develop a Functional Console-Based Game:** To design and build an interactive quiz game that can take user input, display questions, and track scores.
3. **Improve Logical Thinking and Problem-Solving:** To enhance programming logic by solving real-time challenges during development, such as input validation and control flow management.
4. **Practice Modular Programming:** To structure the code in a clean, readable, and modular way using functions for different tasks.
5. **Create a Portfolio-Worthy Project:** To produce a complete and presentable software project that demonstrates programming ability and can be included in a personal resume or portfolio.
6. **Understand User Interaction in CLI Applications:** To gain experience in designing user-friendly command-line interfaces and handling user interactions efficiently.

System Requirements

The quiz game is a lightweight, console-based application developed in C and requires minimal system resources. Below are the basic requirements needed to develop, compile, and run the project:

Hardware Requirements:

- **Processor:** Intel/AMD CPU (1 GHz or higher)
- **RAM:** Minimum 512 MB (1 GB or more recommended)
- **Storage:** Less than 10 MB of free space
- **Display:** Standard terminal or command-line interface

Software Requirements:

- **Operating System:** Windows, Linux, or macOS
- **Compiler:** GCC (for Linux/macOS) or MinGW/Turbo C/Borland C (for Windows)
- **Text Editor or IDE:** Code::Blocks, Visual Studio Code, Dev-C++, or any C-compatible IDE
- **Command-Line Tool:** Terminal (Linux/macOS) or Command Prompt (Windows)

The game is fully compatible with any standard C compiler and runs smoothly on most modern systems without additional dependencies.

Project Description

This project involves the design and development of a console-based quiz game using the C programming language. The game presents a series of multiple-choice questions to the user, records their answers, provides instant feedback, and calculates the final score. It is designed to run entirely within a terminal or command-line environment and emphasizes simplicity, interaction, and functionality.

The program features a user-friendly menu, modular code structure, score tracking, and replay functionality. It was built as a personal learning project to apply key programming concepts such as arrays, loops, conditionals, and functions. Through this project, the goal is not only to build a functional application but also to improve coding discipline, logical thinking, and software development practices.

The project follows a step-by-step development process, from planning and coding to testing and documentation, and is intended to serve as both a learning tool and a showcase of technical capability.

4.1 Working Flow

The quiz game follows a linear and interactive flow, allowing the user to engage with the game step by step. Below is the general working flow of the program:

1. **Program Launch:** When the program starts, a welcome message and main menu are displayed.
2. **User Selection:** The user is prompted to choose between starting the quiz or exiting the program.
3. **Question Display:** If the quiz starts, a series of multiple-choice questions are presented one by one.
4. **User Input:** The user selects their answer by entering the corresponding option number.
5. **Answer Evaluation:** The program checks the user's input against the correct answer and updates the score accordingly.
6. **Feedback Display:** Immediate feedback is given after each question to inform the user whether the selected answer is correct or incorrect.
7. **Score Summary:** After all questions are answered, the final score is displayed along with an optional remark or message.

8. **Replay Option:** The user is asked if they want to play again or exit the program.
9. **Exit:** If the user chooses to exit, a thank-you message is displayed and the program terminates.

This workflow ensures an organized and user-friendly experience while demonstrating proper use of basic C programming principles.

4.2 Modules/Functions

To maintain clarity, reusability, and structure in the code, the quiz game is divided into several functional modules. Each function is designed to handle a specific task, promoting modular programming and easier debugging. The main modules/functions used in the project are:

1. **main()**
The entry point of the program. It displays the main menu, takes the user's choice, and calls the appropriate functions based on the input.
2. **startQuiz()**
This function manages the entire quiz flow. It presents questions to the user, takes answers, calls the score-checking function, and displays the final score at the end.
3. **displayQuestion()**
Displays a single question along with its multiple-choice options.
4. **getUserAnswer()**
Captures and validates the user's input for each question, ensuring only valid choices are accepted.
5. **checkAnswer()**
Compares the user's answer with the correct one and returns whether it is right or wrong.
6. **showScore()**
Calculates and displays the final score along with a personalized message based on performance.
7. **playAgain()**
Prompts the user to decide whether to replay the quiz or exit the program.

These functions collectively enable a smooth and organized game experience. They also demonstrate the effective use of modular programming in C.

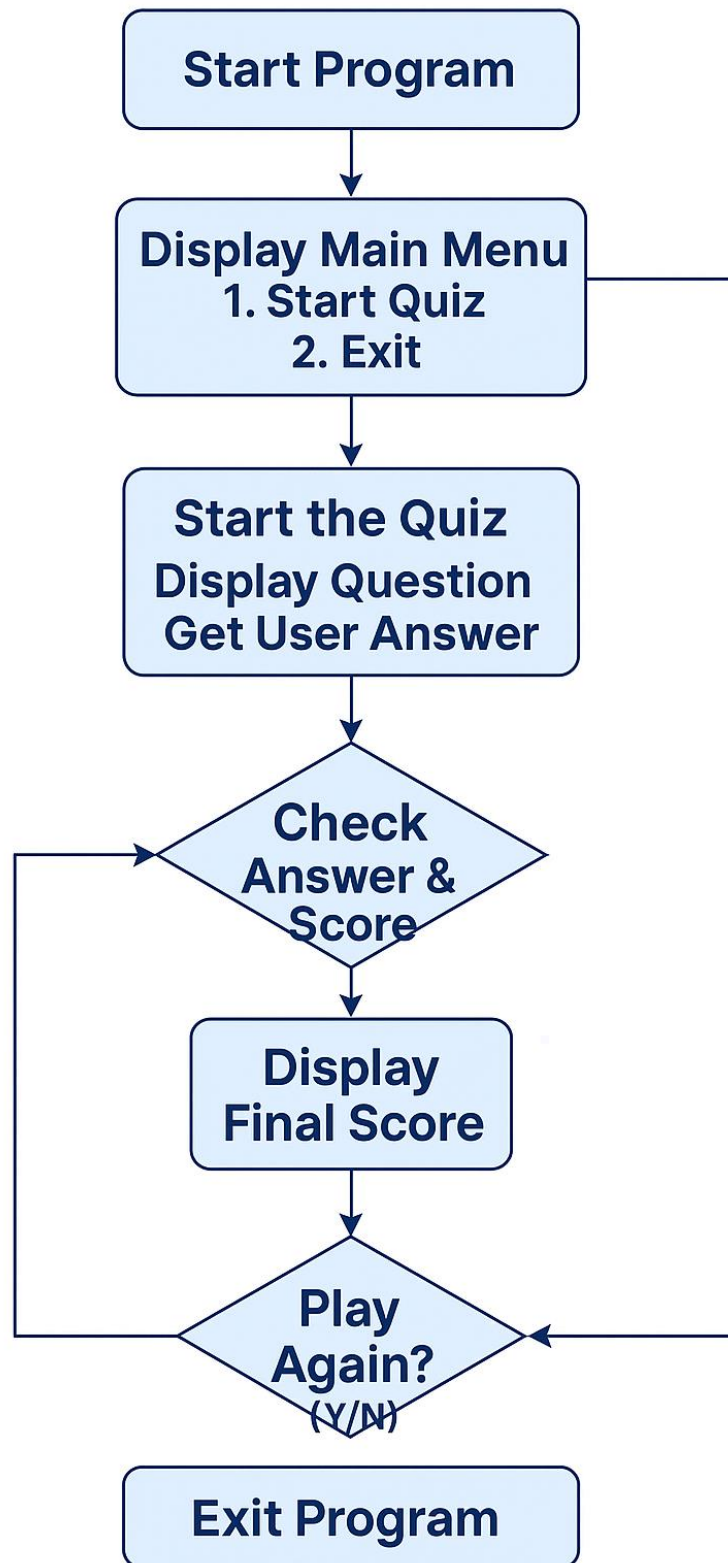


Fig4.1 Flowchart

Implementation Details

The quiz game project is implemented using the C programming language, structured into multiple functional modules to enhance clarity, maintainability, and reusability. The implementation is console-based, using standard input and output for interaction. Below are the main aspects of the implementation:

Programming The quiz game project is implemented using the C programming language, structured into multiple functional modules to enhance clarity, maintainability, and reusability. The implementation is console-based, using standard input and output for interaction. Below are the main aspects of the implementation:

1. **Programming Language:**
The game is developed in C using standard libraries like `stdio.h` and `stdlib.h` for input/output operations and control flow.
2. **Data Storage:**
Questions, options, and correct answers are stored using arrays of structures. Each structure holds a question string, an array of four options, and the index of the correct option.
3. **User Interaction:**
The user is prompted via the terminal to select options from a menu. For each quiz question, the user enters a choice, which is then validated.
4. **Answer Checking and Scoring:**
The user's response is compared against the stored correct answer. A score counter is incremented for every correct answer.
5. **Replay and Exit Options:**
After the final score is displayed, the user is asked whether they want to attempt the quiz again. The program continues or exits based on the user's input.
6. **Code Modularity:**
Functions such as `startQuiz()`, `displayQuestion()`, `checkAnswer()`, and `showScore()` handle specific tasks. This modularity promotes better readability and maintenance.
7. **Error Handling:**
Basic input validation ensures that invalid menu selections or answers are handled gracefully.
8. **Logo Integration (Optional):**
The final executable can have a custom icon embedded using resource editing tools, providing a polished, professional look.

The program demonstrates effective use of control structures, user-defined functions, and structured data types, making it a valuable learning experience in procedural programming.

Sample Code Snippet

Below is a representative code snippet that demonstrates the core logic of the quiz game. This part of the code shows how a question is displayed, how user input is collected, and how the answer is evaluated:

```
#include <stdio.h>

#include <string.h>


// Structure to hold a question
struct Question {
    char question[200];
    char options[4][100];
    int correctOption;
};


// Sample question
struct Question q1 = {
    "What is the capital of India?",
    {"Mumbai", "New Delhi", "Chennai", "Kolkata"},
    2 // Indexing starts from 1
};


void askQuestion(struct Question q) {
    int answer;

    printf("%s\n", q.question);
    for (int i = 0; i < 4; i++) {
        printf("%d. %s\n", i + 1, q.options[i]);
```

```

    }

    printf("Enter your answer (1-4): ");
    scanf("%d", &answer);

    if (answer == q.correctOption) {
        printf("Correct!\n\n");
    } else {
        printf("Wrong! The correct answer is: %s\n\n", q.options[q.correctOption - 1]);
    }
}

int main() {
    askQuestion(q1);
    return 0;
}

```

This snippet illustrates how the game handles a single question using a structure and basic control flow. In the actual implementation, multiple questions are stored in an array of structures, and the same function is called in a loop to conduct the quiz.

Testing and Output Screen

Testing Approach:

To ensure the quiz game functions correctly and handles various inputs smoothly, manual testing was conducted. Each component and feature was tested under different scenarios to identify bugs and improve user experience. The following types of tests were performed:

1. Functional Testing

- Verified that the main menu displays correctly.
- Checked if quiz questions are presented sequentially.
- Validated the correct answer evaluation and scoring mechanism.

2. Input Validation Testing

- Tested with valid and invalid menu choices.
- Entered answers outside the valid option range (e.g., 0 or 5) to ensure the program handles such cases gracefully.

3. Replay Functionality

- Checked if the quiz restarts correctly when the user selects “play again.”

4. Exit Condition Testing











- Ensured the program exits properly when the user chooses to quit.

5. Boundary Testing

- Tested the system with the maximum and minimum number of questions to observe program behavior.

Output Screenshot:

Below is a sample screenshot of the output during a typical quiz session:

| | | | |
|---|-------------------|---------------------|----------|
|  .gitattributes | 4/27/2025 3:30 PM | Text Document | 1 KB |
|  a | 5/1/2025 4:35 PM | Application | 136 KB |
|  C Quiz Game Project | 5/12/2025 7:26 PM | Microsoft Word D... | 1,278 KB |
|  Lisence.yiml | 5/2/2025 7:28 PM | YIML File | 1 KB |
|  quiz | 5/2/2025 7:27 PM | C Source File | 4 KB |
|  quiz | 5/2/2025 7:36 PM | Application | 140 KB |
|  quiz | 5/2/2025 7:31 PM | Icon | 5 KB |
|  README | 5/2/2025 7:27 PM | Markdown Source ... | 1 KB |
|  resource.o | 5/2/2025 7:35 PM | O File | 5 KB |
|  resource.rc | 5/2/2025 7:34 PM | RC File | 1 KB |

Date created: 5/2/2025 7:36 PM
Size: 139 KB

Fig 11.1 Executable File

```

E:\Vs Code\Quiz-Game\quiz.exe
=====
Welcome to the Quiz Game
=====
1. Start Quiz
2. Exit
Enter your choice:

```

Fig 11.2 Code Running Output

```
E:\Vs Code\Quiz-Game\quiz.exe
=====
Welcome to the Quiz Game
=====
1. Start Quiz
2. Exit
Enter your choice: 1

Starting the Quiz

Q1: What is the capital of France?
A. Berlin
B. Madrid
C. Paris
D. Rome
Your answer(A/B/C/D): C_
```

Fig 11.3 Demo Quiz Question 1

```
E:\Vs Code\Quiz-Game\quiz.exe
=====
Welcome to the Quiz Game
=====
1. Start Quiz
2. Exit
Enter your choice: 1

Starting the Quiz

Q1: What is the capital of France?
A. Berlin
B. Madrid
C. Paris
D. Rome
Your answer(A/B/C/D): C
Correct!

Q2: Which planet is known as the red planet?
A. Earth
B. Mars
C. Jupiter
D. Venus
Your answer(A/B/C/D): B_
```

Fig 11.4 Demo Quiz Question 2

```
E:\Vs Code\Quiz-Game\quiz.exe
Welcome to the Quiz Game
=====
1. Start Quiz
2. Exit
Enter your choice: 1

Starting the Quiz

Q1: What is the capital of France?
A. Berlin
B. Madrid
C. Paris
D. Rome
Your answer(A/B/C/D): C
[Éà Correct!

Q2: Which planet is known as the red planet?
A. Earth
B. Mars
C. Jupiter
D. Venus
Your answer(A/B/C/D): B
[Éà Correct!

Q3: Who wrote the play 'Romeo and Juliet'?
A. William Shakespear
B. Charles Dicken
C. Mark Twain
D. Jain Austien
Your answer(A/B/C/D): A
```

Fig 11.5 Demo Question 3

```
E:\Vs Code\Quiz-Game\quiz.exe
Q1: What is the capital of France?
A. Berlin
B. Madrid
C. Paris
D. Rome
Your answer(A/B/C/D): C
[Éà Correct!

Q2: Which planet is known as the red planet?
A. Earth
B. Mars
C. Jupiter
D. Venus
Your answer(A/B/C/D): B
[Éà Correct!

Q3: Who wrote the play 'Romeo and Juliet'?
A. William Shakespear
B. Charles Dicken
C. Mark Twain
D. Jain Austien
Your answer(A/B/C/D): A
[Éà Correct!

=====
Quiz Result
=====
You scored 3 out of 3
=====
Do you want to play again...! Enter Your Choice (Y/N):
```

Fig 11.6 Prompting User if they want to play again or just Quit the Game

Challenges Faced

During the development of the Quiz Game project in C, several challenges were encountered that tested both technical and logical problem-solving skills. These challenges are summarized below:

1. Structuring Questions Efficiently:

Initially, storing questions, options, and correct answers in a clean and manageable way was a challenge. This was resolved by using structures and arrays, which made data handling more organized.

2. Validating User Input:

Ensuring that the user entered only valid choices (between 1 and 4) required additional logic. It was important to prevent the program from crashing due to invalid or unexpected input.

3. Score Calculation Logic:

Designing a system to correctly track and display scores after each question and at the end required careful implementation, especially when scaling up to multiple questions.

4. Reusability of Functions:

To keep the code clean, modular, and maintainable, functions had to be designed thoughtfully. Initially, all logic was in `main()`, but breaking it into reusable components improved readability and flow.

5. User-Friendly Interface:

Since the game is console-based, displaying questions and feedback in a clear and appealing format was a challenge. Efforts were made to align text and give feedback in a way that enhances user experience.

6. Debugging and Logical Errors:

Some logical errors went unnoticed during the first few tests. Consistent debugging, testing different user flows, and using `print` statements for tracking variables helped resolve these issues.

Future Improvement

While the current version of the quiz game meets its basic objectives, there is significant scope for enhancing its functionality and user experience. Some of the future improvements considered for this project are:

1. **GUI Integration** Migrating the console-based game to a graphical user interface (GUI) using tools like GTK or Windows API can make the game visually engaging and easier to use for beginners.
2. **Category-Based Questions** Implementing a system where users can select specific topics or categories (e.g., science, history, programming) will provide a more personalized experience.
3. **Timer Functionality** Adding a countdown timer for each question can make the game more challenging and simulate real-time quiz environments.
4. **Question Randomization** Currently, questions are displayed in a fixed order. Adding randomization can make repeated attempts more interesting and prevent predictability.
5. **Database or File Integration** Instead of hardcoding questions, future versions can load them from external text files or a database, allowing for easier updates and scalability.
6. **User Score Tracking** Implementing user profiles and storing scores for performance tracking across multiple sessions can increase user engagement.
7. **Multiplayer Mode** Expanding the game to allow two or more players to compete in real-time would add an interactive and competitive dimension.
8. **Difficulty Levels** Introducing different difficulty settings (easy, medium, hard) would make the game more flexible and cater to various levels of users.

These enhancements would make the quiz game not only more feature-rich but also more suitable for educational and entertainment purposes on a larger scale.

Conclusion

The development of the Quiz Game using the C programming language has been a valuable learning experience, reinforcing core programming concepts such as data structures, functions, conditional logic, and user input handling. This project demonstrates how even simple games can be effective tools for both learning and entertainment.

Throughout the process, the focus remained on creating a functional, interactive, and user-friendly application. Challenges such as structuring data, validating input, and maintaining modularity provided real-world insights into the software development lifecycle. These challenges were overcome through consistent debugging, thoughtful design, and iterative testing.

The project serves as a foundational exercise that not only strengthens programming skills but also opens doors for future enhancements, such as graphical interfaces, real-time multiplayer capabilities, and data-driven content. It highlights the importance of building small yet meaningful projects to gain confidence and mastery in programming.

In conclusion, this quiz game project is more than just a game—it is a stepping stone toward more complex and impactful software development endeavors.

References

- **Balagurusamy, E.** (2019). *Programming in ANSI C* (8th ed.). McGraw-Hill Education.
 - A comprehensive textbook that helped in understanding the core concepts of C programming.
- **Kernighan, B. W., & Ritchie, D. M.** (1988). *The C Programming Language* (2nd ed.). Prentice Hall.
 - Often referred to as the "K&R book", it provided deep insights into the structure and syntax of C.
- **GeeksforGeeks.** (n.d.). <https://www.geeksforgeeks.org>
 - Used as a reference for examples on struct, arrays, loops, and input/output handling in C.
- **TutorialsPoint.** (n.d.). <https://www.tutorialspoint.com/cprogramming/>
 - Served as a quick guide for understanding standard C libraries and functions.
- **Stack Overflow.** (n.d.). <https://stackoverflow.com>
 - Community-driven help for debugging and best practices during development.
- **Programiz.** (n.d.). *Learn C Programming*. <https://www.programiz.com/c-programming>
 - Helped in reviewing concepts such as modular programming and input validation.