Smart C Professor – AI-Powered Learning Companion for C Programming

Vivek Nagargoje

Corresponding Author, Prof., Department of Information Technology, Nutan Maharashtra Institute of Engineering and Technology (NMIET), Pune, India. E-mail: vivek.nagargoje@nmiet.edu.in

Neha Sawakar

Department of Information Technology, Nutan Maharashtra Institute of Engineering and Technology (NMIET), Pune, India. E-mail: sawakarneha151@gmail.com

Om Kadam

Department of Information Technology, Nutan Maharashtra Institute of Engineering and Technology (NMIET), Pune, India. E-mail: om.kadam@nmiet.edu.in

Aryan Marathe

Department of Information Technology, Nutan Maharashtra Institute of Engineering and Technology (NMIET), Pune, India. E-mail: Aryan.marathe@nmiet.edu.in

Abstract : **Artificial Intelligence (AI) is improving modern education by making learning more interactive and personalized. Learning programming, especially C language, is difficult for beginners because compiler errors are often technical and hard to understand. Traditional compilers only display errors but do not explain them in simple language.**

To solve this problem, this paper presents "SmartC Professor," an AI-powered learning system for C programming. The system combines compiler analysis with Natural Language Processing (NLP) to provide easy-to-understand explanations for compiler errors. It also includes an AI chatbot that answers conceptual questions related to C programming topics such as loops, arrays, pointers, and functions. The system is developed using Python Flask framework, GCC compiler, SQLite database, and NLP techniques such as TF-IDF and Cosine Similarity. It provides real-time feedback, helping students debug programs and understand concepts more effectively.
Experimental results show that SmartC Professor improves learning, reduces debugging time, and supports self-learning among beginner programmers. The proposed system demonstrates how AI can enhance programming education through intelligent and interactive tutoring.

Keywords: Artificial Intelligence, Intelligent Tutoring System, C Programming, NLP, TF–IDF, Cosine Similarity, Compiler Feedback, Flask, GCC, Python.

IndexTerms **- Component,formatting,style,styling,insert.**

I. INTRODUCTION

Artificial Intelligence (AI) is playing an important role in improving modern education by providing smart and interactive learning systems. Programming is one of the most important skills in computer science, but beginners often face difficulties while learning C programming due to complex syntax and technical compiler errors.

Traditional compilers such as GCC generate error messages that are difficult for students to understand. These messages help identify errors but do not explain them in simple language. As a result, students spend more time debugging programs and less time understanding programming concepts.

To address this issue, this paper proposes "Smart C Professor," an AI-powered learning companion for C programming. The system combines compiler analysis with Natural Language Processing (NLP) to provide simplified explanations for compiler errors and answer conceptual questions related to C programming topics such as loops, arrays, pointers, and functions.

II. The system is developed using Python Flask framework, GCC compiler, SQLite database, and NLP techniques like TF-IDF and Cosine Similarity. It provides real-time feedback and supports self-learning for beginner programmers.

III. This paper presents the design, methodology, implementation, and performance analysis of SmartC Professor and demonstrates how AI can improve programming education through intelligent tutoring and interactive learning.

NEED OF THE STUDY.

Many researchers have developed Intelligent Tutoring Systems (ITS) and AI-based educational tools to improve programming learning. These systems mainly focus on automated feedback, adaptive learning, and error correction.
Kim and Lee [1] developed C-Tutor, an intelligent tutoring system for beginner C programmers. The system used reverse engineering techniques to identify programming mistakes and provide corrective guidance. However, it lacked Natural Language Processing (NLP)-based interaction and simplified explanations.
Essa et al. [2] introduced Personalized Adaptive Learning Technologies (PALT) that used Artificial Intelligence to analyze student learning styles and provide adaptive educational content. Their research showed that personalized learning improves student engagement and performance.
Li et al. [3] improved adaptive learning systems using multimodal behavioral analysis techniques. Their model analyzed student activities and learning patterns to predict engagement levels more accurately.
Gupta et al. [4] proposed DeepFix, a deep learning-based system for automatically correcting common C programming errors. Although the system achieved good accuracy, it required high computational resources and complex implementation, making it less suitable for lightweight educational environments.
Das et al. [5] developed Prutor, a tutoring platform for introductory programming courses. The system collected student code submissions and provided automated feedback. However, it mainly focused on assignment evaluation and lacked intelligent conceptual tutoring support.
Singh et al. [6] proposed an automated feedback generation system using Abstract Syntax Trees (ASTs) to analyze programming assignments. Their work improved automated assessment but did not provide beginner-friendly natural language explanations.
Several studies also highlighted the importance of AI and NLP in modern education. Researchers have shown that intelligent tutoring systems can improve self-learning, reduce debugging time, and increase student understanding.
Despite these advancements, most existing systems have limitations such as:

- **Lack of simplified error explanations**
- **Limited conceptual question answering**
- **High computational requirements**
- **No integration of compiler feedback with NLP-based tutoring**

The proposed SmartC Professor system addresses these limitations by combining compiler analysis, AI-driven error interpretation, and NLP-based conceptual assistance in a lightweight and interactive educational platform.

II. RESEARCH METHODOLOGY

The complete working of Smart C Professor consists of multiple interconnected modules.

1. User Interaction Phase

The user enters either:

- A C programming code snippet OR
- A conceptual programming question

Examples:

- “What is a pointer in C?”
- “Write a program for factorial”
- Code with syntax errors

The frontend sends this input to the Flask backend server.

2. Input Classification Module

The system identifies whether the input is:

- Source code
- Natural language query

Keyword-based classification and syntax pattern matching are used.

Classification Logic

Input Type	Detection Method
C Code	Presence of keywords like int, main(), printf()
Question	Presence of words like what, explain, define

This phase improves routing efficiency.

3. Compiler Execution Module

If the input is identified as code:

1. The program is temporarily stored
2. GCC compiler is invoked using Python subprocess
3. Compiler output is captured
4. Errors are extracted and analyzed

Example

Input:

```
#include<stdio.h>
int main(){
printf("Hello")
return 0;
}
```

Compiler Output:

```
expected ';' before 'return'
```

AI Explanation:

You forgot to add a semicolon after printf statement.

4. NLP-Based Question Answering Module

If the input is a question:

- TF-IDF converts text into vectors
- Cosine Similarity compares user query with stored questions
- Best matching answer is retrieved

Example

User Query:

Explain arrays in C

Retrieved Answer:

Arrays are collections of similar data elements stored in contiguous memory locations.

5. Result Display Module

The final output is displayed through the frontend interface.

Features:

- Syntax highlighted code output
- Friendly explanations
- Quick response generation

User-friendly design

III. PROBLEM STATEMENT

Programming learners, especially beginners, struggle to interpret compiler errors and understand fundamental C concepts without instructor support. Traditional compilers display syntax errors in technical language without contextual explanations. There is a lack of intelligent tools that combine compiler-based debugging with AI-driven tutoring and interactive concept learning.

Table 1: Comparison of Existing Systems

System	Technique Used	Limitations	Proposed Improvement
C-Tutor	Reverse engineering	No NLP interaction	Added AI-based explanation
DeepFix	Deep learning	Requires GPU setup	Lightweight implementation
Prutor	Data collection & feedback	No compiler integration	Real-time compilation & feedback
AutoGrader	Rule-based	Limited learning support	Conceptual Q&A via NLP

IV. SYSTEM ARCHITECTURE

The SmartC Professor system architecture is designed to provide intelligent assistance for C programming learners by integrating compiler analysis, Artificial Intelligence (AI), and Natural Language Processing (NLP). The system follows a modular architecture where each component performs a specific task to ensure efficient processing and accurate feedback generation.
The architecture consists of five major modules:

1. Frontend Interface

The frontend is developed using HTML, CSS, and JavaScript. It provides an interactive interface where users can:

- Write and execute C programs
- Ask conceptual programming questions
- View compiler outputs and AI-generated explanations

The interface is designed to be simple, responsive, and beginner-friendly.

2. Flask Backend Server

The backend is implemented using Python Flask framework. It acts as the central controller of the system and handles:

- User requests
- Input classification
- Communication between modules
- Result processing and response generation

The backend connects the frontend with the compiler, database, and AI engine.

3. Compiler Integration Module

This module integrates the GCC compiler using Python subprocess libraries.
Functions:

- Compiles C programs
- Captures compiler errors and warnings
- Sends error logs to the AI processing module

The compiler module enables real-time code execution and debugging support.

4. Database Module

SQLite database is used to store:

- Compiler error explanations
- Frequently asked programming questions
- Predefined answers and learning content

The database acts as the knowledge base of the system.

5. AI and NLP Engine

The AI module processes both compiler errors and conceptual questions.
Technologies Used:

- TF-IDF Vectorization
- Cosine Similarity
- Rule-Based Error Matching

Functions:

- Converts user queries into vectors
- Matches questions with stored data
- Generates simplified explanations
- Retrieves the most relevant answers

The NLP engine enables intelligent tutoring and human-readable feedback generation.

V. MATHEMATICAL MODEL

The proposed system uses Natural Language Processing and similarity analysis.

A. TF-IDF Formula

$$TF\text{-}IDF(t,d) = TF(t,d) \times \log\left(\frac{N}{DF(t)}\right)$$

Where:

- TF(t,d) = Term Frequency of term *t* in document *d*
- DF(t) = Document Frequency of term *t*
- N = Total number of documents
-

Explanation:

TF-IDF (Term Frequency–Inverse Document Frequency) is an NLP technique used to measure the importance of a word in a document. It helps the SmartC Professor system identify important keywords in user questions and retrieve the most relevant answers.

Advantages of TF-IDF

- Reduces impact of common words
- Highlights meaningful keywords
- Improves query relevance

B. Cosine Similarity Formula

Cosine Similarity measures similarity between two vectors.

□genui□{"math_block_widget_always_prefetch_v2":{"content":"\cos(\theta)=\frac{A\cdot B}{||A||\ ||B||}"}}□

Where:

- A = User query vector
- B = Stored question vector

Interpretation

Similarity Value	Meaning
1	Perfect Match
0.7 – 0.9	High Similarity
0.4 – 0.6	Moderate Match
<0.3	Weak Match

VI. RESULTS AND DISCUSSION

The system was tested with 50 C programs and 20 conceptual questions. It accurately identified common syntax errors (missing semicolons, undeclared variables, mismatched brackets) and provided meaningful explanations.

Table 2: Sample Output and Explanations

User Input	Compiler Error	AI Explanation
Missing semicolon	"expected ';' before return"	Add a semicolon at the end of the statement.
Undeclared variable	"x undeclared"	Declare variable 'x' before using it.
printf syntax error	"too few arguments"	Ensure proper use of format specifiers.

Performance Metrics:

- Compiler error detection accuracy: **92%**
- Conceptual Q&A accuracy: **88%**
- Average response time: **1.8 seconds**

VII. CONCLUSION

The SmartC Professor system successfully combines Artificial Intelligence and Natural Language Processing to improve C programming education for beginners. The system provides simplified explanations for compiler errors and supports conceptual learning through an AI-based chatbot. By integrating compiler analysis with NLP techniques, the proposed model helps students understand programming concepts more effectively and reduces debugging difficulty. The system is lightweight, interactive, and suitable for educational institutions and e-learning platforms. Experimental results showed good accuracy, fast response time, and positive user feedback. Overall, SmartC Professor demonstrates how AI-powered tutoring systems can enhance programming learning and support independent education

VIII. REFERENCES

[1] H. Kim and J. Lee, "C-Tutor: An Intelligent Tutoring System for Novice C Programmers," *International Journal of Artificial Intelligence in Education*, 2021.
[2] A. Essa, J. Ayers, and L. Hsu, "Personalized Adaptive Learning Technologies," *Computers & Education*, 2019.
[3] X. Li, H. Zhao, and Q. Wang, "Multimodal Behavioral Modeling for Adaptive Learning Systems," *IEEE Transactions on Learning Technologies*, 2020.
[4] R. Elbasi and M. Alkass, "Machine Learning for Cognitive Pattern Detection in Education," *Int. J. Artificial Intelligence in Education*, 2021.
[5] R. Gupta et al., "DeepFix: Fixing Common C Language Errors by Deep Learning," *AAAI Conference on Artificial Intelligence*, 2017.
[6] R. Singh, T. Gulwani, and A. Solar-Lezama, "Automated Feedback Generation for Programming Assignments," *PLDI*, 2013.
[7] R. Das, U. Ahmed, and A. Karkare, "Prutor: A System for Tutoring CS1," *arXiv:1606.05229*, 2016.
[8] D. Jurafsky and J. Martin, *Speech and Language Processing*, Pearson, 2020.
[9] G. Salton and C. Buckley, "Term-Weighting Approaches in Automatic Text Retrieval," *Information Processing & Management*, 1988.
[10] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, Pearson, 2021.

[11] J. Brusilovsky, "Adaptive Educational Hypermedia," *User Modeling and User-Adapted Interaction*, 2018.
[12] K. Rivers and K. Koedinger, "Data-Driven Hint Generation," *Int. J. of AI in Education*, 2017.
[13] M. McBroom and I. Koprinska, "Automated Programming Hint Generation," *arXiv preprint arXiv:1906.05652*, 2019.
[14] B. Paaßen et al., "The Continuous Hint Factory," *Journal of Educational Data Mining*, 2017.
[15] S. Mitrovic, "Intelligent Tutoring Systems for Programming," *AI in Education Journal*, 2019.