International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

ADAPTIVE COMPILER

Koushik Kumar S[1], Mukkesh S*[2], Naresh S*[3], Muthumuniyasamy K*[4]*

*[1] Sri Shakthi Institute Of Engineering & Technology, (Anna University Affiliated), Coimbatore, Tamil Nadu, India
*[2] Sri Shakthi Institute Of Engineering & Technology, (Anna University Affiliated), Coimbatore, Tamil Nadu, India
*[3] Sri Shakthi Institute Of Engineering & Technology, (Anna University Affiliated), Coimbatore, Tamil Nadu, India
*[4] Sri Shakthi Institute Of Engineering & Technology, (Anna University Affiliated), Coimbatore, Tamil Nadu, India

ABSTRACT :

In today's fast-paced software development world, developers frequently work across multiple programming languages, each with its own syntax and quirks. This paper introduces a new universal compiler that supports multiple programming languages—specifically C, C++, Java, and Python. The compiler aims to simplify the coding process by providing real-time syntax checking, suggesting corrections instantly, and automatically translating code between languages while keeping the original logic intact. With an intuitive interface built using a split-screen layout, the tool allows users to write code in one language and convert it to another, all within a single environment. The paper discusses the core features of the compiler, its architecture, and potential use cases.

INTRODUCTION

Modern software development often involves switching between different programming languages. Whether you're working on legacy code in C or writing a new feature in Python, the need to navigate various syntax rules can slow down productivity. Most compilers are language-specific and lack the ability to translate code between languages. In this paper, we present a universal compiler that bridges this gap. It supports four popular programming languages—C, C++, Java, and Python—and enables real-time syntax checking and automatic translation between them. This tool is designed to improve productivity by allowing developers to write code in one language and effortlessly switch to another without manually rewriting entire programs. Our approach focuses on making this transition smooth, with an easy-to-use interface and robust backend logic.

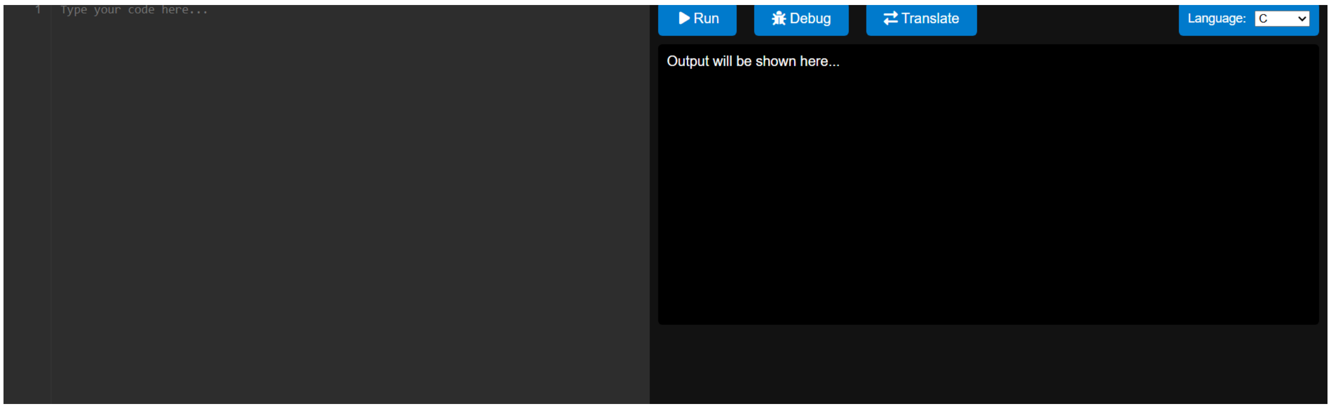
METHODOLOGY

The architecture of the universal compiler consists of three main components:
- **Frontend**: The user interface is built using HTML, CSS, and JavaScript, providing a split-screen design. On the left side, users can input their code, and on the right, they can see the terminal output, which includes error messages, compiled results, and a translation feature. The user can choose to run, debug, or translate the code using the buttons at the top of the screen.
- **Backend**: Built with Flask, the backend handles all the logic of compiling, parsing, and translating code. The main engine uses Tree-sitter, an efficient parsing library, to understand and manipulate code written in different programming languages.
- **Real-Time Syntax Checking**: The compiler checks the syntax of the code as the user types. If there's an error, it's immediately flagged with a ✕ in the terminal, and the system suggests possible fixes.
- **Code Translation**: One of the key features is the ability to translate code from one language to another, such as converting a C program into Python. This is achieved by mapping constructs in one language to their equivalents in another. It's a powerful tool for developers working in a multi-language environment.

MODELING AND ANALYSIS

Most compilers are language-specific and lack the ability to translate code between languages. In this paper, we present a universal compiler that bridges this gap. It supports four popular programming languages—C, C++, Java, and Python—and enables real-time syntax checking and automatic translation between them.

RESULTS AND DISCUSSION SCREENSHOTS:



IMPLEMENTATION

The implementation of the compiler is broken down into several steps:
1. **Input Parsing and Syntax Checking**: As users write code, the system parses it using Tree-sitter. This allows the compiler to identify syntax errors and provide suggestions in real time.
2. **Translation Engine**: When the user presses the 'Translate' button, the system converts the code from one language to another, keeping the original logic intact. This involves identifying constructs like loops, conditionals, and functions, and mapping them to equivalent structures in the target language.
3. **Compilation and Execution**: The 'Run' button compiles and executes the code in the language selected by the user, and the output appears in the terminal on the right.
The system can currently translate between C, C++, Java, and Python, and future plans include adding more languages.

DISCUSSION

The universal compiler provides a valuable tool for developers working with multiple programming languages. By combining real-time error checking, automatic translation, and a user-friendly interface, it simplifies the process of writing, debugging, and converting code. This can significantly reduce the time developers spend switching between different languages and help maintain code across different platforms.
The main challenge moving forward is enhancing the translation accuracy for more complex scenarios. For example, translating multi-threaded code between languages may require handling specific platform features or language-specific behavior that differs between languages.

CONCLUSION

In conclusion, this paper presents a universal compiler that supports multiple languages, provides real-time syntax feedback, and offers automatic code translation. By making the coding process more efficient and less error-prone, this tool has the potential to improve productivity for developers working in multi-language environments. Future work will focus on improving the translation engine and expanding the number of supported languages.

REFERENCES :

1. 'Tree-sitter: An Incremental Parsing System' – GitHub repository and documentation.
2. Flask Documentation' – https://flask.palletsprojects.com/
3. Babel: The JavaScript Compiler' – https://babeljs.io/