Diversitas Journal
ISSN 2525-5215
Volume 11, Issue 1 (Jan./Mar. 2026) p. 0202 – 0213
https://diversitasjournal.com.br/diversitas_journal

Programming Language Converter

OPELLA, Joe Marlou [1] SARENO, Julius

[1] 0000-0003-2998-7091; Cavite State University. City of Carmona, Cavite, Philippines. joemarlou.opella@cvsu.edu.ph.
[2] 0009-0009-7592-4527; Technological University of the Philippines., Manila, Philippines. julius_sareno@tup.edu.ph.

ABSTRACT

Education sector has drastically changed their way to deliver and share information with the use of the advance technology nowadays. The developed Programming Language Converter serves as an important tool that assist students, educators and to anyone who wants to learn different programming languages since it allows various programming constructs to be coded, and converted in just one click. This standalone programming platform also allows user to create, edit, compile, and execute three different programming languages at a time. It can convert Java to C++ programming source code and C++ to C programming source code and vice versa. For easy and fast construction of program source code an auto help function that lists down correct suggested syntax is designed. The menu bar fully equipped with helpful list of basic concept in programming is included for the user to easily create, open and save file. One of the most important tool is the navigation menu that guides the users on how to use the programming platform. The software can be run on Microsoft Win10 operating system and was developed using Python 64 bit 3.4.1.1 as the main programming language. The programming environment was tested for its ability to handle Java, C++, and C code conversions, and the results showed that it performed well in meeting the objectives of the study. The results of the evaluation indicated that the tool was well-received, with positive feedback on its ease of use and functionality.

ARTICLE INFORMATION

Article process:
Submitted: 07/08/2025
Approved: 02/26/2026
Published: 03/08/2026

***Keywords*:** **Programming language, converter, C++, C,Java**

RESUMO

O setor educacional mudou drasticamente a forma como transmite e compartilha informações com o uso da tecnologia avançada atual. O Conversor de Linguagens de Programação desenvolvido serve como uma ferramenta importante para auxiliar estudantes, educadores e qualquer pessoa que queira aprender diferentes linguagens de programação, pois permite que várias estruturas de programação sejam codificadas e convertidas com apenas um clique. Esta plataforma de programação independente também permite ao usuário criar, editar, compilar e executar três linguagens de programação diferentes simultaneamente. Ela pode converter código-fonte de Java para C++ e de C++ para C, e vice-versa. Para facilitar e agilizar a construção do código-fonte do programa, uma função de ajuda automática lista a sintaxe correta sugerida. A barra de menus, totalmente equipada com uma lista útil de conceitos básicos de programação, está incluída para que o usuário possa criar, abrir e salvar arquivos facilmente. Uma das ferramentas mais importantes é o menu de navegação, que orienta os usuários sobre como usar a plataforma de programação. O software pode ser executado no sistema operacional Microsoft Windows 10 e foi desenvolvido usando Python 64 bits 3.4.1.1 como linguagem de programação principal. O ambiente de programação foi testado quanto à sua capacidade de lidar com conversões de código Java, C++ e C, e os resultados mostraram que ele teve um bom desempenho no atendimento aos objetivos do estudo. Os resultados da avaliação indicaram que a ferramenta foi bem recebida, com feedback positivo sobre sua facilidade de uso e funcionalidade.

***Palavras-chave*:** Linguagem de programação, conversor, C++, C, Java

DOI: 10.48017/dj.v11i1.3523

Introduction

Information has been so valuable in the history of mankind. It is being defined by Encarta as the collected facts and data about a specific subject. There were many sources of information. Some of which were from school, community, internet, and many others. Whether where or when the information came from the understanding varies on how it was being presented.

Few decades ago, the advancement of information technology has emerged the increasing amount of digital information in the society accompanied with the proliferation of modern computer hardware and embedded systems. It has become so significant to the daily life of every individual since early 90's. Development of more advance and versatile applications, and devices has parallel demand for continuous development of the different facets of innovation in the field of information technology. According to Moore's law, the amount of information increases tenfold every five years. In the same way, development of more advanced and fully equipped programming languages continuously rises in order to sustain the craving public for information and intellectual development (NIST, 2012). As Edward Felten, a computer scientist at Princeton University reckons that the improvements in the algorithms driving computer applications have played as important part as Moore's law for decades (The Economist, 2010).

Information technology advancement and modern computer hardware and innovations must leverage in producing tools that can help the coming generations of academicians and educators to impart or disseminate knowledge and information. There is no need to change the ultimate role of academicians and educators but the way it should be obtained and presented must have a drastic change as the proliferation of modern computers and gadgets continuously rising. Likewise, innovations in information technology and modern devices should also be paralleled with the innovations on how information should be delivered. All these have also change how people think, act, do daily chores, study, understand and react to the people in the outside world. In the same manner discipline and classroom management in today's generation is already changing and is one of the most challenging issues in the field of education. As Al Zieni (2008), and Isuku (2018), stated that classroom management problems and strategies are different for every learning environment. Therefore, the legacy form of classroom management might not be the most effective technique to inject knowledge and inculcate good moral character in enhancing more than the capacity of the students to surpass the least minimum in their field of studies.

For the researchers, it has been an experience to use different programming languages for one subject. This is for the benefit of the students to be exposed and be familiar with more than one language. To lessen if not totally eliminate all the aforementioned issues, this study intends to converts syntax of one programming language to another.

This study maximizes the use of available programming languages in the market to be taught simultaneously. Specifically for the students, the system can be a helpful tool in gaining more understanding, and in comparing the different features and capabilities of different programming languages. This can further help the users decide as to what programming language is easier to code or use. The tool can also serve as a virtual teacher by providing suggestions to errors incurred upon construction of a source code on a certain topic. For the teachers and lecturers, this is a very helpful tool in the tedious creation of sample programs for every topic in data structure. Moreover, built-in sample programs will lead the users in understanding more about the topic. In general, this study will contribute to the continuous development of information technology.

The programming platform can accept Java program source code and convert it to C++ and C program source code or accepts C++ program source code to be converted into Java and C program source code or accepts C program source code to be converted to Java and C++ program source. C++ and C program source code will be based on ANSI (American National Standard Institute). It is also provided with auto help function that lists down correct and suggested syntax where the user can choose from. To instill good moral values and honesty while doing the laboratory exercise a device that will keep track the movements of the student in each individual workstation and provide oral warnings. Python 64 bit version 3.4.1.1 was used for the development of the entire software.

As for the limitation, the software is intended only for three different programming languages implemented in structured programming. Sample programs for data structures were also made available except on Stack, Queue, LinkedList and Interface. Class design and object-oriented implementation were not covered by programming platform source code converter. It can only function well in Windows based operating system not older than Windows 10.

Literature

Programming Language

A programming language is a set of commands, instructions, and other syntax used to create software program or applications. According to Techopedia.com, programming language is a computer language engineered to create a standard form of commands. These commands can be interpreted into a code understood by a machine. Programs are created through programming languages to control the behavior and output of a machine through accurate algorithms, similar to the human communication process (Techopedia, n.d.). A programming language is also known as a programming system, computer language or computer system. The programmers use these languages to write code that conveys the specification of the project or application’s requirement. As of the present generation programming languages are classified as "high-level languages" where in the syntax or codes

has nearly the same meaning to an English dictionary (Coursera, 2024). This codes or syntax will be compiled into a "low-level language," which can be recognized directly by the computer hardware.

High-level languages are designed to be easy to read and understand. This allows programmers to write source code in a natural fashion, using logical words and symbols common to human understanding. Many high-level languages are similar enough that programmers can easily understand source code written in multiple languages (TutorialsPoint, 2023).

Low-level languages include assembly and machine languages. An assembly language contains a list of basic instructions and is much more difficult to read than a high-level language. In rare cases, a programmer may decide to code a basic program in an assembly language to ensure it operates as efficiently as possible. An assembler can be used to translate the assembly code into machine code. The machine code, or machine language, contains a series of binary codes that are understood directly by a computer's central processing unit (CPU). Needless to say, machine language is not designed to be human readable.

Programming Source Code Converter

A programming source code converter, also known as a transpiler or source-to-source compiler, is a software tool that reads the source code written in one programming language and outputs equivalent code in another language. These tools are critical in software modernization, legacy code migration, cross-platform development, and educational applications where comparative learning of languages is required (Lachaux et al., 2020).

Traditional source code converters are rule-based systems that rely on syntax and grammar rules of both the source and target programming languages. These tools often use parsers to generate Abstract Syntax Trees (AST), which are then traversed and translated into the corresponding syntactic constructs of the target language (Gupta et al., 2022). However, challenges arise when translating complex language features like memory management, object-oriented structures, or language-specific libraries, which require more than direct syntactic mapping.

To address these limitations, modern converters increasingly use semantic-preserving techniques and intermediate representations to maintain program logic and flow across languages. Research by McIntosh et al. (2021) emphasizes the importance of maintaining semantic equivalence, which ensures that the translated code performs the same operations as the original.

In the educational context, source code converters are particularly useful for helping students compare programming paradigms across languages (e.g., Java vs. C++). By automating the translation process, learners can focus on understanding language-specific

syntax and best practices, rather than rewriting entire programs manually (Kumar & Singh, 2023).

With the advancement of artificial intelligence, some tools now integrate machine learning-based code translation, particularly in multilingual code environments such as GitHub Copilot or OpenAI Codex. However, these systems are often black-box models and not suitable for precise or critical code conversions where full transparency and control are necessary (Roziere et al., 2020).

Despite ongoing improvements, limitations still exist. Most converters support a narrow set of language pairs (e.g., Java to C++, or Python to JavaScript), and many struggle with converting object-oriented or modular programs—especially those involving multiple files or advanced class hierarchies. Furthermore, converting between structurally different languages, such as Java (object-oriented) and C (procedural), requires additional abstraction handling and custom rule definitions (Lano et al., 2024).

The development of programming source code converters—also known as transpilers or source-to-source compilers—has become increasingly relevant in the fields of software engineering and computer science. These tools serve the critical function of converting source code written in one programming language into semantically equivalent code in another language. Their applications range from software modernization and legacy system migration to cross-platform development and pedagogy in multi-language programming environments.

Early implementations of source code converters primarily relied on rule-based translation methods, using formal grammar rules to map constructs from one language to another via Abstract Syntax Trees (ASTs). Although efficient for syntactic transformations, these systems often lacked the flexibility to preserve semantics in more complex, object-oriented or modular programming contexts (Gupta et al., 2022). Recognizing these limitations, researchers have increasingly turned toward semantic-preserving strategies and intermediate representations to maintain logic and functionality across languages (McIntosh & Rahman, 2021).

As the software development landscape evolved, machine learning-based approaches emerged as powerful tools in code translation. Lachaux et al. (2020) introduced an unsupervised neural translation model capable of converting code between Java, C++, and Python without the need for parallel datasets. Similarly, Roziere et al. (2022) explored the use of automated unit tests to guide unsupervised translation, enhancing reliability by aligning generated code with predefined behavioral outputs.

Several deep learning architectures have also been proposed. Models like code2seq (Alon et al., 2019a) and code2vec (Alon et al., 2019b) learn distributed representations of code using structural path-based encoding, enabling tasks like summarization and translation with improved context sensitivity. More recently, CodeT5 was introduced as a unified encoder-

decoder pretraining model that excels in both code generation and comprehension tasks (Wang et al., 2021).

In the context of educational and training environments, source code converters offer significant value. Kumar and Singh (2023) emphasized the pedagogical benefits of allowing students to view equivalent implementations of the same algorithm across multiple languages. Such comparative learning enhances understanding of language-specific syntax, semantics, and design paradigms.

Further contributions in the field have emphasized syntax and domain-aware translation models. Liu et al. (2023) proposed a framework that integrates domain semantics with syntax trees to improve the contextual accuracy of program translation. Likewise, Rajathi et al. (2022) developed "Origin-The Transcoder," focusing on bidirectional translation across Java, Python, and C++, facilitating legacy code migration and educational experimentation.

The field has also benefited from literature on supporting technologies such as code summarization, clone detection, and semantic similarity measurement, all of which contribute to the robustness and accuracy of code translation systems (Zhu & Pan, 2019; Zakeri-Nasrabadi et al., 2023; Sharma et al., 2021). These tools help developers and researchers ensure that the converted code not only functions correctly but is also maintainable and aligned with coding best practices.

Despite the progress, current converters still face challenges in idiomatic translation. Translating code in a way that conforms to the idioms and conventions of the target language and in handling complex language features such as inheritance, interfaces, and memory management. Additionally, many systems are still limited to a narrow set of language pairs and lack full support for object-oriented or modular designs (Lano et al., 2024).

Methodology

The Programming Language Converter is a programming platform that has a capability of accepting Java source code then converts it to equivalent C++ program source code and C source after successfully compiling it without any errors encountered. It can also compile, run and execute existing files written in Java, C++ and C. Figure 11 shows the process flow of the Programming Language Converter.

The first step is that the compiler platform must be opened. The user can either open an existing file or create a program source code from the scratch. During the compilation process a series of conversion was been done. The algorithm of the project after compiling the program source code is the following:

If the program found a valid Java program source code then, it will generate its equivalent C++ and C program source codes. Otherwise, it will check if the Java code has error or if the code is written in C++.

- If the program has Java code error, the conversion will be terminated.

- If the program is written in C++, it will generate C code.

Figure 1.

The Flow of the Program Conversion

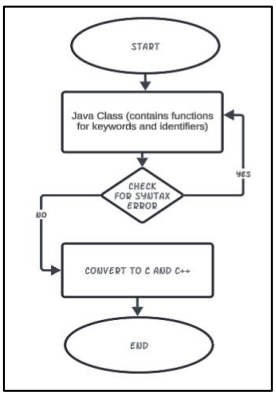


Fig. 1 shows the conversion of the Java program source code to its equivalent C++ and C program source code. The compiled Java program source code: keywords and identifiers are checked by the program and compare to the equivalent program source code: keywords and identifiers of C++ and C programming language. Each programming language is provided with a class and functions that contains the keywords for each programming language. These equivalent keywords and identifiers are directed on the new tabs for the conversions.

Results and Discussions

The Programming Language Converter is an environment used to code, compile and execute Java, C++ and C program source codes. It allows a user to experience the coding and executions of the three programming languages in one programming platform. It has the capability to convert syntax from Java program source code to its equivalent C++ program source code and from C++ program source to its equivalent C program source code. The platform can only convert Java program source code implemented in class design that was saved in one source file into its C++ equivalent program source code. C++ program source code implemented in class design cannot be converted to its C program source code because C does not have that program source code implementation. However, due to the complexity of the program structures for every language the platform can only run but cannot convert built-in libraries of Java like LinkedList, Queue, Stack and etc.

An auto help function that list down correct and suggested syntax if at least two characters are being typed correctly. The user can only select one from the list of syntax suggested by the platform during program source code development. The system runs on Microsoft Window operating systems not older than Windows 7. Python programming language is used to develop the software.

For easy and hassle-free navigation of the platform, menus and tool bars are also provided. A menu to create a new file, open and edit an existing program source code, save file & program source code, compile and execute program source codes are designed. The tool

bar New is used to open a new blank tab for program source code, Save for saving file, Compile for compiling program source code, Run for executing successfully compiled program source code, Stop for terminating all running/currently executed program and Convert for converting program source code from Java to C++ equivalent program source code and from C++ to C equivalent program source code. A help tool is also included for the user to navigate the work-around of the programming platform without much effort.

Figure 2.

The Program Development Environment

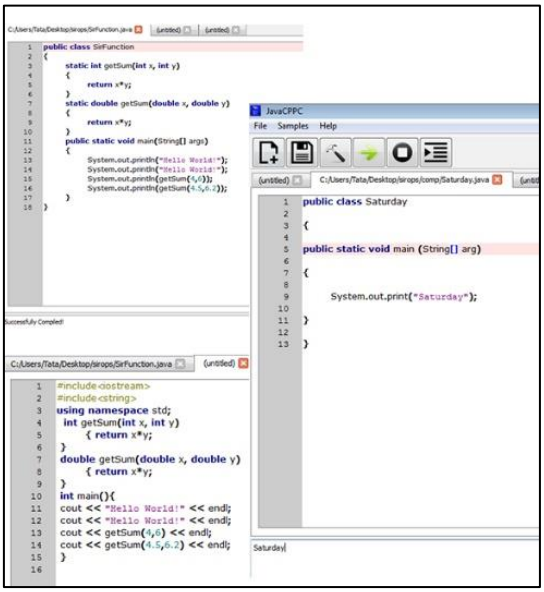
Figure 2 shows the main platform of the Programming Language Converter. A blank tab is provided in a newly opened programming platform. It served as an editor window. It is also provided with helpful menus and tool bars for quick and easy navigations of different transactions.

The Programming Language Converter is designed with several key features aimed at achieving its primary objectives. The platform is user-friendly, ensuring smooth and effortless navigation of its functionalities. It is integrated with three different programming language compilers—specifically for Java, C++, and C—to allow versatile code testing and execution. One of its core functions is the ability to convert Java source code into equivalent C++ and C source code, facilitating cross-language translation. Additionally, the software includes an auto-help function that provides users with correct and suggested syntax to assist during coding. To further enhance its utility, the converter also includes sample programs focused on data structures to aid learning and reference.

However, the software has certain limitations. It does not support reverse conversion, meaning it cannot convert C source code to C++ or C++ to Java. The program is also limited in compatibility, as it can only run on Windows operating systems, particularly those not older than Windows 7. Lastly, it does not support the compilation of Java source codes that are implemented in a class design structure distributed across multiple files, which restricts its usability for more advanced or modular Java programs.

Figure 3.

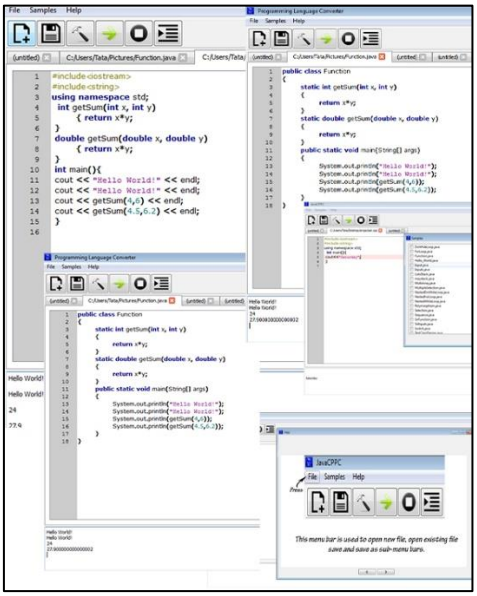
The Program Source Code Conversion



To determine the accuracy of the converter, the output of the converted file is compared to the output of the hardcoded source code as shown in fig. 3. The Java source code was successfully converted into a C ++ and C source code. The conversion will automatically be executed after a successful compile the base source code in which in this case a Java source code. The programming platform can also generate and display error message like other platform such as NetBeans.

Figure 4.

The Sample Executed and Converted Program Source Code and Features



To make the platform user friendly, fig. 4 shows different examples of converted program source code. It is also equipped with recently save Java source file floating on the side of the platform. Another feature that have made the platform advantageous is that it is equipped with program samples for basic program constructs readily available from the menu.

Summary of Findings, Conclusions, and Recommendations

The Programming Language Converter is a platform used to code, compile and execute Java, C++, and C program source codes. It allows users to interact three programming languages at a time. It has the capability to convert syntax from Java source code to C++ source and C source code. It has an auto help function that lists down correct and suggested syntax if at least two characters are typed correctly. The system runs on Microsoft Window operating systems not older than Windows 10. Python programming language was used by the system developed. To make it user friendly it is equipped with menus for different transactions such as creating new file, open and edit existing source code, save file and source code, compile and execute program source codes. A help tool is also included for the user to understand the flow of activities to be performed in the platform without much effort.

In view of the summary of findings and in cognizance with the general objective of the study, which is to develop a programming language converter, the following conclusions were drawn. A compiler platform was successfully designed with several defining characteristics. It can accept Java source code and converting it into C++ source code, which is then further converted into C source code. Additionally, the software features an auto-help function that provides users with correct and suggested syntax, enhancing usability and learning. It also includes sample programs on data structures to serve as practical references for users. The software was developed using Python 64-bit, version 3.4.1.1, as the programming platform.

REFERENCES

Alon, U., Zilberstein, M., Levy, O., & Yahav, E. (2019b). code2vec: Learning distributed representations of code.

Alon, U., Zilberstein, M., Levy, O., & Yahav, E. (2019a). code2seq: Generating sequences from structured representations of code.

Coursera Staff. (2024, October 9). *Low-Level vs. High-Level Programming Languages*. Coursera.

Gupta, A., Banerjee, A., & Singh, R. (2022). *Source-to-Source Translation Frameworks: A Survey of Techniques and Tools*. ACM Computing Surveys, 54(3), 45–67.

Isuku, E. J. (2018). *Classroom Management and Problems Associated with It*. In O. Kolawole & B. Lawal (Eds.), *A Handbook of Teaching Practice*. Faculty of Education, University of Ibadan.

Kumar, D., & Singh, A. (2023). *Improving Programming Language Pedagogy through Source Code Conversion Tools*. Journal of Computer Science Education, 18(2), 111–124.

Lachaux, M.-A., Roziere, B., Chanussot, L., & Lample, G. (2020). *Unsupervised Translation of Programming Languages*.

Lano, K., Kolahdouz-Rahimi, S. (2024). *Using Model-Driven Engineering to Automate Software Language Translation*. Automated Software Engineering, 31(1), 1–28.

Liu, J., Wang, Y., Zhang, L., & Luo, X. (2023). *A Syntax and Domain-Aware Model for Unsupervised Program Translation*. *Technologies*, 12(12), 244.

McIntosh, T., & Rahman, S. (2021). *Semantic Preservation in Programming Language Translation: A Comparative Review*. Journal of Software Engineering Research and Development, 9(1), 22–38.

NIST. (2012). *NIST Cryptographic Standards and Guidelines Development Process*.

Rajathi, P., Vijayalakshmi, K., & Sundari, P. (2022). *Origin-The Transcoder: A Translation Engine for Cross-Language Code Conversion*. *Technologies*, 12(12), 244.

Roziere, B., Lachaux, M.-A., Chanussot, L., & Lample, G. (2020). *CodeXGLUE: A Machine Learning Benchmark Dataset for Code Translation*.

Sharma, A., Thakur, S., & Jindal, R. (2021). *Machine Learning in Source Code Analysis: A Survey*

The Economist. (2010, February 27). *Data, Data Everywhere*.

TutorialsPoint. (2023). *Difference Between High-Level and Low-Level Language*.

Wang, Y., Xia, X., Zhang, L., & Lo, D. (2021). *CodeT5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation*.

Yang, W., Tang, C., & Zhang, H. (2023). *Deep Learning-Based Code Generation: A Survey*.

Zakeri-Nasrabadi, M., Tabrizi, N. M., & Tabrizi, M. H. (2023). *Source Code Similarity Measurement: A Systematic Literature Review*.

Zhu, Y., & Pan, L. (2019). *A Systematic Literature Review on Automatic Code Summarization*. *arXiv preprint arXiv:1909.04352*.