International Conference on Innovative Research in Engineering, Management and Sciences
International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2019 IJSRCSEIT | Volume 4 | Issue 9 | ISSN : 2456-3307

MoC++ Interpreter for the C++ Language

Moni Krithika S[1], S. Shanmuga Priya[2]
Student[1], Senior Assistant Professor[2]
Department of Computer Science Engineering, New Horizon College of Engineering, Outer Ring Road, Marathalli, Bengaluru, Karnataka, India

ABSTRACT

MoC++ Interpreter is a novel project that directly executes source program/instructions written in C++ language without translating it into a machine code or object code. MoC++ maps input to output statement by statement, where each instruction is thoroughly checked for syntax and semantic errors. MoC++ is an efficient interpreter which has a well-developed error diagnostics system. MoC++ interpreter solves complicated real – world problems by abstracting constructing the problem mathematically. MoC++ interprets a source code that adheres to a particular language specification that is C++, and can interpret possibly thousand lines of code. MoC++ doesn't alter the meaning of the original instruction being interpreted.
Keywords: Interpreter, C++ Language, Design, Developers, Students

I. INTRODUCTION

MoC++ interpreter is a novel project that directly executes a source program/instruction written in C++ language without translating it into machine code or object code. MoC++ maps input to output statement by statement, where each instruction is thoroughly checked for syntax and semantic errors. MoC++ is an efficient interpreter which has a well-developed error diagnostics system.

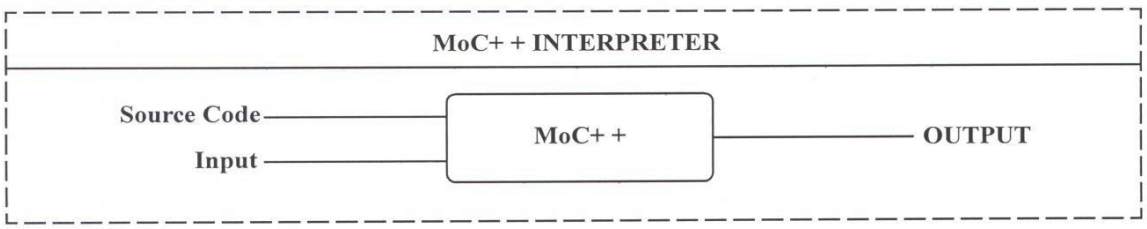

Figure 1: MoC++ Interpreter

MoC++ interpreter solves complicated real-world problems by abstracting constructing the problem mathematically. The mathematical modeling that MoC++ interpreter follows is:-

- Analyses a problem statement.
- Construct mathematical abstraction capturing key characteristics of the problem.
- Find an appropriate mathematical technique to provide a solution.

MoC++ interprets a source code that adheres to a particular language specification that is C++, and can interpret possibly thousand lines of code. MoC++ doesn't alter the meaning of the original instruction being interpreted.

The rest of the paper is organized as follows: Section 2 discusses about analysis and design, Section 3 is about the algorithm/pseudo code, Section 4 gives the conclusion.

II. ANALYSIS AND DESIGN

A. OBJECTIVES OF THE PROJECT

The goal of MoC++ interpreter is to translate a source program in a high-level language (C++) to the output directly without converting it into machine code. MoC++ Interpreter executes a source program statement by statement, outputs the solution in an easy to understand format, and provides an environment for efficient debugging of the program. MoC++ parses the source code and performs its behaviour directly. MoC++ implements self-modifying code hence it forms a base for artificial intelligence and machine learning research. MoC++ acts as an emulator for running computer software written for old languages which don't have a present day compiler to run source programs written in them.

III. ALGORITHM / PSUEDO CODE

A. CONCEPTUAL DESIGN

The conceptual design of MoC++ is a high level view of its software design and architecture. Conceptual design of MoC++ details regarding the primary components that make up the project. It includes the organization of the project and its interaction and interconnection with other components of the Mo C++ system. Conceptual design does not speak about the implementation technique of these components but provides a platform for thorough examination and understanding of the components. MoC++ consists of front end and back end functions. In the initial interpreting stage the front end of MoC++ helps in reading the source code. The primary components of MoC++ front end are the Source class, Scanner class, Token class, Parser class.

The Source class gets the file name from the user using the getfilename() and validates if the command and the file name entered by the user is correct. The Scanner class reads the source file or source program statement by statement using the get_token() and calls the extract_ token() to extract each token are divide the statement into several parts. The Token class returns the current token to the Parser class by using the current token of function.

The Parser class is the master that controls the entire translation process in the front end. It controls the entire process by owning the Token class, indirectly owning the Scanner class hence having indirect access to the Source class. The Parser class deciphers the token type and evaluates its characteristics by using the decipher() function. The result of the decipher()

Function determines whether the Parser class has to evaluate a key word using eval_key() function or evaluate an expression using the eval_exp() function. The Intermediate tier performs the second translation stage where the symbol table and the stack play an important role. This provides an interface for smooth, fast and efficient execution of the program at the Back End. The symbol table is generated using a SymbolTable class where createtable() function generates the initial template of the table. The variables and their values can be stored in the table with the insert() function, and their values updated using the update().

The search() function is used to search for a variable during expression evaluation. The stack is implemented using the Stack class. The stack data structure helps in expression evaluation. The Back end is the most important phase of an instruction execution where the interpreter evaluates, validates the instruction and outputs the result. The interpreter Back End consists of an executor which reads the symbol table and executes the source code.

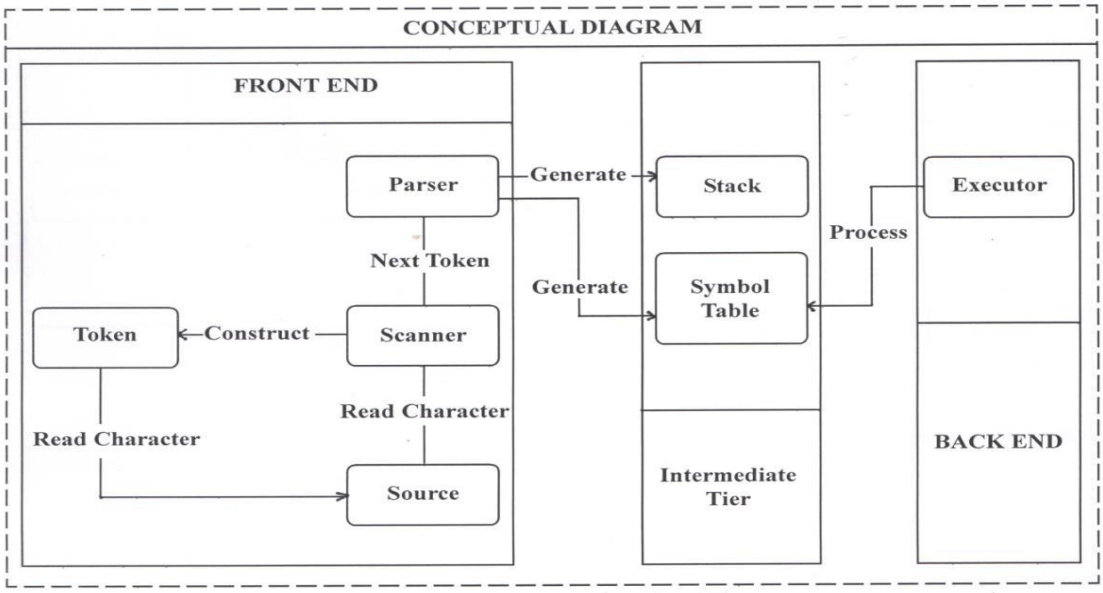

Figure 2: Conceptual Diagram

B. ALGORITHM

Step 1: Get the file name

Step 2: Read the file statement by statement till the end of the file. Step 3: Extract token by token from the read statement

Step 4: Decipher the token extracted and identify the token type and take the appropriate action.

Step 5: If the extracted token is a keyword then evaluate the keyword accordingly

Step 6: If the extracted token is an expression then evaluate the expression using an appropriate function call

Step 7: Repeat the Steps 2 to 6 until the entire file is parsed.

C. CLASS DIAGRAM

Class diagram shows the relationship, interconnection and dependency of one class on another class.

MoC++ has six classes where one class inherits from another class using the Hybrid inheritance technique.

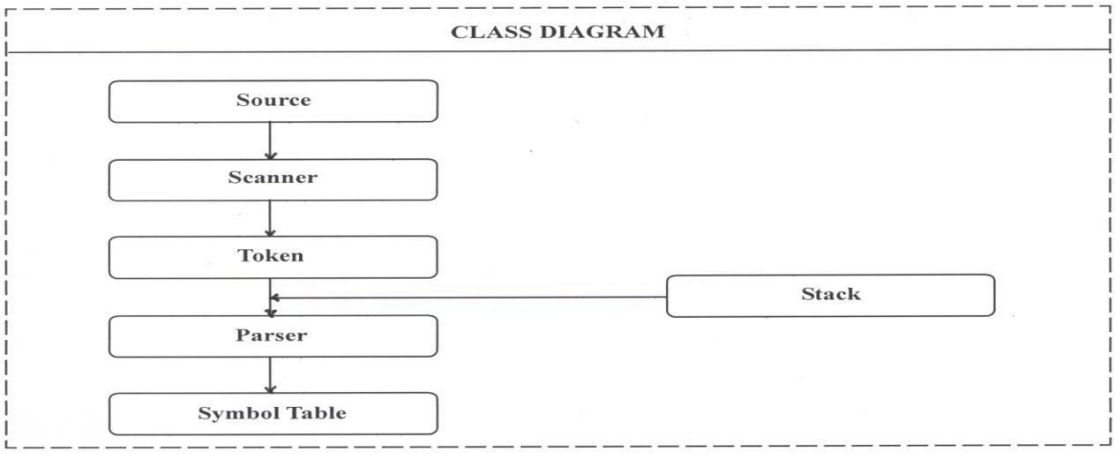

Figure 3: Class Diagram

IV. CONCLUSION

The goal of MoC++ is to translate a source program in a high-level language to the output directly without converting it into machine code. MoC++ provides the power of Python programming and Java programming in C++, where efficiency, garbage value collection, and many other best features of these programming languages are incorporated in it. MoC++ Interpreter executes a source program statement by statement, outputs the solution in an easy to understand format, and provides an environment for efficient debugging of the program. MoC++ parses the source code and performs its behaviour directly. MoC++ implements self-modifying code, hence, it forms a base for artificial intelligence and machine learning research. The abbreviation of MoC++ is derived from the first two letters of the name of the author of this unique interpreter: Moni Krithika. The author authoritatively works on C++, the generally accepted and implemented Computer Language; and hence the name MoC++, is most suitable for this interpreter program. This interpreter is being developed further for its higher accuracy for debugging, superior reading capacity and other valuable and user-friendly features.

II. REFERENCES

[1]. Alfred Aho, S. Monica Lam, Ravi Sethi, and Jeffrey D. Ullman, "Compilers: principles, techniques and tool," 2nd ed., India: Pearson India Education Services Pvt. Ltd, 2014.

[2]. E. Balaguruswamy, "Object oriented programming with c++, 6th ed., India: McGraw Hill Education Pvt. Ltd., 5th Reprint 2015.

[3]. Paul Deitel, and Harvey Deitel, "C++ how to program," 10th ed., India: Pearson India Education Services Pvt. Ltd., 2017.

[4]. Ronald Mak, "Writing compiler and interpreters: a software engineering approach," 3rd Indian ed., Indianapolis US: Wiley Publishing Inc., 2009.

[5]. Herbert Schildt, "The complete reference c++," 4th ed., India: McGraw Hill Education Pvt. Ltd., 2003, 37th Reprint 2016.