International Journal of Applied Engineering Research
ISSN 0973-4562 Volume 10, Number 11 (2015) pp. 28779-28795

http://www.ripublication.com

Platform Independent Generic Compiler Design Using Reverse Engineering

Dr. Amit Verma
Professor, CSE, Chandigarh University
Mohali, India, amit.verma@cumail.in
Er.Nikita Bakshi
Research Scholar, CSE, Chandigarh University
Mohali, India, bakshinikita24@gmail.com

Abstract

Computer has its influence in every sphere of human life whether it is education, medical science, social networking and so on. All computers perform their action on the basis of a program, which is a set of instruction in a high level language. This high level language is converted to machine language using compiler. Compiler is computer program which converts the high level programming language into machine language which is understandable by computer. The compilers till now are not generic they are language dependent. In this paper we intend to investigate compiler's current state and further suggest future advancement of compiler. We further propose algorithms for designing language independent compiler.

Keywords: Reverse Engineering, Compiler, Software, Programming Languages, Program, Legacy System.

Introduction

A computer is a general purpose device that can be programmed to perform a set of arithmetic or logical operations automatically. Sequence of operations can be easily changed in computer so it can solve more than one kind of problem like logical operations etc. Computer began appearing in the first century and later used in medieval era for astronomical calculations. In World War II computers were used for specialized medical applications. During this time first digital computer was developed .Initially the size of computer was equal to the room and it consumed power as much as several hundred modern personal computers[1] Modern computers are based on integrated circuits and they are more capable than the early machines. Modern computers are small enough to fit into mobile devices like laptops [2]. Modern computer have influenced various fields such as travel & tourism, social networking, education and research, whether forecasting, e-commerce and many

more. The feature of modern computer which distinguished them from all other machines is that they can be programmed to perform various operations/ tasks. Program is a kind of instruction given to the computer and then instructions are processed. Modern computer are based on von Neumann architecture have machine code in the form of an imperative programming language. Programming language is a group of grammatical rules which is used to instruct a computer to perform specified tasks. Programming Languages are further divided into two languages Low-Level programming language and High-Level programming language. Low-level programming language is a programming language that provides little or no abstraction of computer instructions. Generally this refers to either machine code or assembly language. High-level programming language is a programming language with strong abstraction. High-level languages are usually compiled into machine language (or sometimes into assembly language and then into machine language) with the help of another computer program called compiler.

Compiler is a set of instruction that translate the source code into binary format usually known as object code. In the 1940's ,programming language came into existence, they were machine dependent because assembly language was used to write the programs and that programs can only be run on a machine. Then, in 1952, Grace Hopper wrote the first compiler which was completed before 1957 and introduced at IBM by John Backers [28]. In 1960, COBOL became an early high-level programming to be compiled on different architectures. In 1962, the first self-hosting compiler was assigned for Lisp by Tim Hart and Mike Levis at MIT.

A. Software

Software is a computer program which provides set of step by step instructions that directs the computer what to do and how to do. Software is often divided into two types:

a.) Application Software: Application Software is that computer software which is designed to perform singular or multiple related tasks. They act as guide to the hardware to perform specific tasks and an example of such application software includes compiler, office suites, media players, accounting and graphics software.

b.) System Software: System software is an operating system that provides an effective program to control the computer resources. It is made to control the hardware of the computer and to provide platform for running application software.

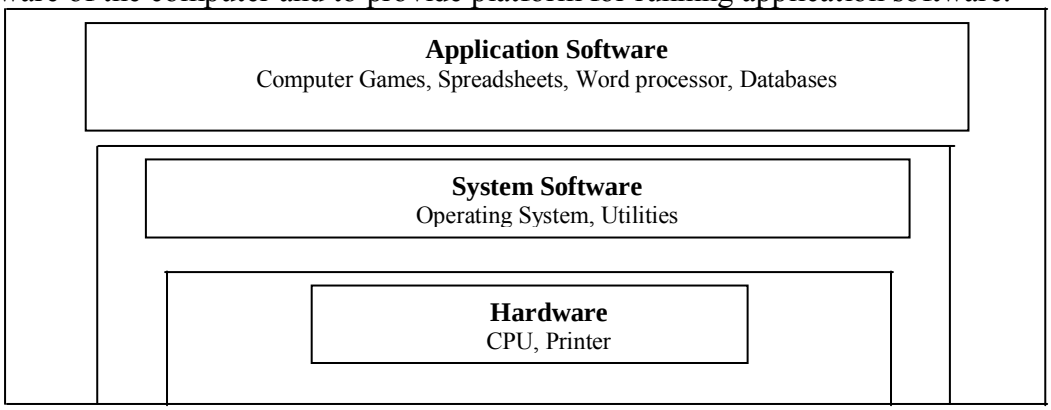


Figure 1: How Software Works [29]

B. Software Engineering

Software Engineering is the application and study of engineering to development, design and maintenance of the software. More and more, society and individual rely on software systems we need to produce truth worthy and reliable system quickly and economically. Software engineering is further divided into two categories they are forward engineering and reverse engineering.

1. *Forward Engineering:* Forward engineering is the process of building high-level model by using lower level details. In forward engineering we move step by step and try to achieve the goal. Forward engineering plays a crucial role in IT sector because it represents the normal software development process.
2. *Reverse Engineering:* Reverse Engineering is the process of analyzing the final product of a design to recreate it. Reverse Engineering is common in both software and hardware. Reverse Engineering is used for discovering the technological principle of object, device of system through analysis of operation, function and structure. It often analyze its components, disassembling something (electronic components, a mechanical matter, computer program, and organic matter, chemical or biological) and check working in detail, just to re create it.

Reverse Engineering was firstly used for the analysis of military advantages and hardware for commercial use. The main purpose of doing reverse engineering is to reduce the design decision of the end product with no additional knowledge or little knowledge about the process involved in the original production.

The same technique is being researched later for industry or legacy software system to protect the end application. Reverse Engineering is used for many purposes like as a teaching tool in a new way, to make compatible products or interoperate more effectively. It is also used to bride data between different operating system and databases; and to reveal the undocumented features of commercial products.

Reverse Engineering is a technique to analyze the system at higher level of abstraction. It can also be seen as going backwards process of development cycle. Other use of reverse engineering include customization of embedded systems, consumer electronics, enabling of extra features on low cost, in house retrofits or repairs, even more relish to curiosity or security auditing.

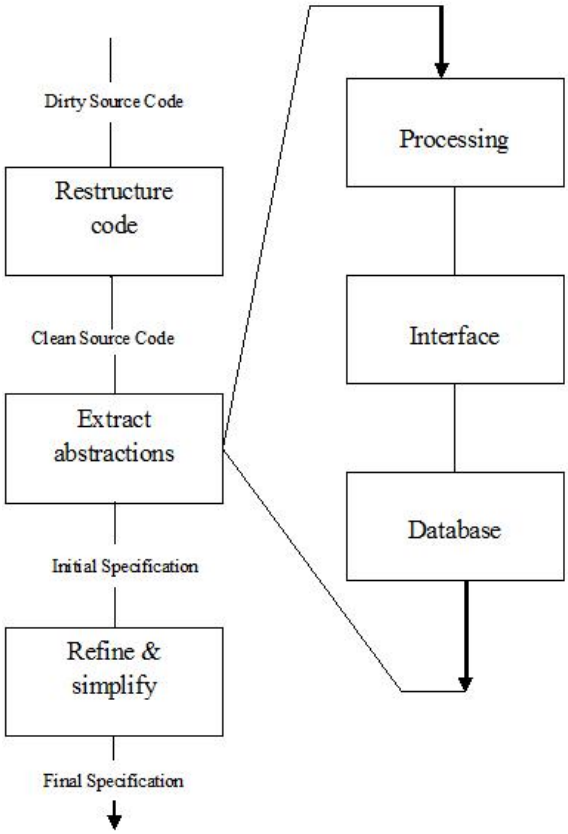


Figure 2: Process of Reverse Engineering [3]

Reverse engineering started with dirty source code. Dirty source code is restructure and clean source code is provided. The clean source provides the basis for all subsequent engineering activities and makes the source code easier to read. The core of reverse engineering activity is extract abstraction. The engineers check the previous program from the source code and fetch meaningful information, after that user interface is applied; database or program data structure is used.

a.) Reverse Engineering in Context to Software: Software Reverse Engineering is a program to retrieve the source code because the source code is lost, to refine the performance of the system, to fix errors, to identify virus or to modify a program written for work with one microprocessor or other.

A famous example of reverse engineering is in 1980 San Jose based Phoenix Technologies Limited. He wanted to develop BIOS for PC's that would be well matched with IBM PC's. To save against the illegal copies of IBM's BIOS he reverse engineered it which is known as clean room or Chinese wall approach. IBM uses two teams for this first team read the code of IBM BIOS and start doing their work without referring or using the original code. The second team has no information about the IBM BIOS and the code, the first team provides the functional requirements they work on it and wrote new BIOS that work as specified.

b.) *Reverse Engineering in Context to Software Engineering*: In 1990 Chikosky and Cross in their paper describe reverse engineering as the process of analyzing a subject to check the system components and relationship between the components. Generate presentation of the system in another form or at higher level of abstraction. [4]

3. Application of Reverse Engineering: Reverse engineering has so many applications as listed below:

a.) Reverse Engineering of Military Aircraft Parts: The process of reverse engineering takes a structure or physical device, compute it and import all its parts and measure it into similar building software or into CAD software. Tool name 3D Laser Scanner take the photo of the device, CAD check the photo and do the reverse engineering. After making change in the product, new program is created. Sometime parts of the device are removed and shell of the device is scanned. Individual pieces are then scanned in the same way the entire device was. Best practices in large parts of helicopters including military aircrafts. New CAD patterns and prints are used by the Govt. suppliers for recreating every piece.

b.) Window Reverse Engineering: In window reverse engineering we do reverse engineering by creating blue prints of windows that are not easily available. It will also provide benefits for hackers to create application coding and help developers to find new ways to protect the system. If we take the example of the USB screen drivers blue prints of the USB screen drivers are not easily available, by using software Soft ICE we are able to create the blue prints of window USB drivers.

c.) Reverse Engineering of Database in Oracle:
In this we do reverse engineering by converting the relational model in logical model by transforming the data from tables to class diagrams.

d.) Reverse Engineering in Education:
It plays a crucial role in education. It helps in self motivation and easy learning of the students. Applying reverse engineering process we get a very effective environment in which students can work. Some universities have already tried to integrate standard computer science courses with reverse engineering technique; an example of university is Missouri-Rolla. The result was inspiring 77% of the students indicates rein formant concept taught during the lectures. Moreover, 82% wanted it to be used in future courses, especially those students who wants to deal with design.

This paper is organized as follows: Section II identifies the compiler and reverse engineering challenges. In Section III we briefly discussed the proposed work. Section IV provides work done to make generic compiler. Finally, Section V is the conclusion of this paper.

Literature Survey

Gerardo Canfora et al. [4] briefly provided an overview of existing work done in the field of software reverse engineering, discusses success stories, main achievement and gives a way for future development in the coming trends in software. Hannani Aman et al. [5] briefly presented the framework of reverse engineering from XML to UML

for creating software requirement specification. They also describe the related work of reverse engineering. This paper provides a method of research to expand the transformation of XML to UML. This type of transformation is called re engineering data. In present software development practices data re engineering is very important. Shivani Budkhar et al. [6] find need for realistic tools to provide support for reverse engineering activities. The capacities of reverse engineering tools are revised to produce class diagram from java source code. In this paper author provide four different reverse engineering tools that are used to generate class diagram from given source code. The purpose behind selecting these tools was to check different types of tools such as commercial, non-commercial, open source tools to support reverse engineering of java code. Tools chosen are Agro UML, Reverse, Enterprise Architecture, Rose.

Amit Kumar Gautam et al. [7] presented an approach to recover design pattern that are used to achieve greater accuracy, better performance, characteristics representation such as behavioral, structure design pattern etc. All work is done by using the matrix and the weight concept to reduce anomalies and false negative rate. They also follow the taxonomy model for reverse engineering and after that apply matrix algorithm for calculating hollow and efficient storage. In it, they also apply the matrix algorithm in designing binary matrix and binary matrix pattern to produce the sources code. In the end they make the comparison with the pattern detection tools and other standard tools for efficiency and performance. Shikon Zhou et al. [8] provided their important contribution in the hierarchical systematic research to improve software metrics for reverse engineering. The fundamental component of the work is to develop a metrics for reverse engineering. Reverse engineering measures are organized into five categories, which included economic measures abstraction and reuse measure, measure of complexity, measure of orintedness. This paper describes the problems of developing software metrics for reverse engineering. A systematic approach is introduced to develop software metrics reverse engineering based on this approach. Yunsik Son et al. [9] provided a brief overview on symbol table. Symbol table is an essential module in compiler construction. It includes phases like lexical analysis, syntax analysis, semantic analysis, code generation. In this paper they deal with reverse technique for the verification of the symbol table in objective C compiler. They also discuss the design and implementation of a reverse translator that verifies and analyses the symbol table designed during the development stage of objective C compiler. Furthermore, based on the symbol table verification, a correct code can be generated by examining the use of identifiers and attribute in the code generation step.

I. Budiselic et al. [10] discussed their experiences with the programming language instruction over the last three years, tool based assignment used before this and their quantitative differences in results. They also discuss the compiler design courses and provided an overview on compiler project at seven different computer science universities in Europe and US. He also provided an overview on programming language translation courses, organization of PTL courses and describes the evolution of programming from its initial stage to its current design. Two important classifications of compiler design project and courses are explained. They roughly

divide compiler design into two courses front end heavy and back end heavy. Jean-Luc Hainaut [11] provided a briefly description about database reverse engineering. Comparison of software reverse engineering and database reverse engineering was done. In this paper he defines that historically, we identify three periods in DBRE they are deepening, discovery and widening. He also discusses the main objectives of the second period for obtaining technique and recovering implicit construction to develop more flexible methods. The scopes of database reverse engineering and the supporting techniques in the present time. After that the future of database reverses engineering, challenges of database reverse engineering and provides contribution in their solving. In fact, the future of software engineering is to be dependent on these solutions. Keith Gallagher et al. [12] describe whether reverse engineering is legal or illegal, what law says about reverse engineering? Few years ago, courts refuse to do reverse engineering but from 2003 courts provide the permission for reverse engineering process.

Hankjin Lee et al. [13] provided a well established algorithm more over a methodology that is used for detection of design patter. In this paper reclassification of GoF pattern takes place. Gang of Four (GoF) is known to be very useful for the detection of projects with reverse engineering methods. He also proposed GoF pattern detection technique. After that evaluation of new technique is done and paper is concluded with the pros and cons of new approach, what other work is to be done in terms of future research. Ivan Keimek et al. [14] provided a brief description that reverse engineering is used in many fields of IT every day like binary code patching, legacy compatibility, network protocol analysis, malware analysis, rapid prototyping or in debugging. Despite its widespread use, reverse engineering is not actively taught as part of computer courses. This paper attempts to provide an overview of real life scenario of reverse engineering. Analysis of skills, ways of thinking that can be developed by reverse engineering and provides example that you can teach reverse engineering by resolution of practical problems. They also focus on the importance of reverse engineering as a tool to turn the self motivation in students and systematically build your logical thinking skills and analytical skills. Chengyong Wu et al. [15] presented an overview of the design of the main components of ORC, especially new features in the code generator. The Open Research Compiler (ORC), jointly developed by the Intel Microprocessor Laboratory of Technology and the Institute of Computer Technical

Academy of Science of China. It has become the leading open source compiler in processor family Itanium TM. ORC development methodology is important for achieving the objectives. Performance comparison with other IPF compiler and a brief summary of research based on ORC are also presented.

John S. Mallozzi et al. [16] talked about one semester course in compiler design presents difficulties to an instructor who want to assign a project in which object oriented techniques are used. This paper describe a method that uses the tool developed by the author to generate a parser that encourages an object-oriented approach, clearly related code written by the student which automatically generate code with intended students to increase understanding. Miodrag Djukic et al. [17] described a technique in which significance of controllability and speed is placed

upward. Many simulation instructional approaches places the retarget ability and cycle precision as the key functions to facilitate the exploration and performance of architecture and also estimate early in the development phase of hardware. The retarget ability and cycle-accuracy to provide a better platform for software development. The main idea of this work is to associate the simulator effort compiled with the development of C and build target language compiler for the processor, using knowledge related to compiler and reusing some common software elements. Mirko Viroli [18] provided a brief description about EGO compiler (Extract Generic On-Demand). EGO is result of a project developed in partnership with Sun Microsystems in order to evaluate a smooth support for generic time function, which does not require changes in the JVM or any other component of the Java Runtime Environment. In this paper they develop solution for sophisticated translation based on the type style step also known as reification. The main aspects of development are presented, from the basic design to implementation and deployment issues.

Johghee M. Youn et al. [19] presented a new coding scheme and instructions are based on the dynamic implied addressing mode (DIAM) to solve the limited space coding and side effects by trimming. They also suggest a generation of code algorithm to fully utilize DIAM. In their work architecture with DIAM exhibition show code size reduction up to 8% and 18% on average, speed compared with the basic architecture without DIAM. Mathieu Acher et al. [20] presented a comprehensive process, compatible architecture FMs tool for reverse engineering. They developed automated technique to extract and combine different descriptions of variability in architecture. Then, alignment techniques and reasoning are applied to integrate knowledge and strengthen architect FM. Cristina Cifuentes et al. [26] presented different type of reverse engineering based on level of code abstraction, which was used to reengineer assembly code, CASE code, machine code and source code. In this paper they elaborate various type of reverse engineering and protection for copyright software. Common uses of reverse engineering were explained. Comparative overview of the legal standing reverse engineering are provided. They also propose the existing and future challenges of the global electronic community for the protection of digital works. Alexandru Telea et al. [22] presented an open architecture that allows easy prototyping exploration and visualization of data reverse engineering scenarios for a wide range of models. They pay special attention to visual and interactive requirements of reverse engineering process. They also compare tool box with an existing visual tools and describe the differences.

From the literature survey we concluded that Reverse Engineering is done for Program Analysis, Design Recovery, and Software Visualization in limited fashion. We also analyzed that the role of reverse engineering would be to extract information needed to enable features such as automatic discovery and compositions of design pattern if we use neural network or automation to make reverse engineering concept completely platform independent. Moreover, software transformations are necessary to make existing systems self healing and automatically reconfigurable. Compiler is a set of program that transforms source code into target code, often having a binary form known as object code. Compiler design proposed so far has two limitations. First, they generate target code for specific languages; hence they are not generic and

most of the compiler generates the target code which is not platform independent. With the advancement of technology, different languages are needed to generate desired results and hence different compilers. These compilers are generate either machine independent on the dependent code depending on the requirements and application i.e. whether you need your application for a specific system or for distributed system. The purpose of our work, we are making generic compiler for object oriented languages that has not been developed yet. Therefore, we propose to develop a platform independent generic compiler design using reverse engineering.

Workdone

Forward engineering is the traditional process of moving from high-level abstractions and logical designs to the physical implementation of a system whereas the process of duplicating an existing component, subassembly, or product, without the aid of drawings, documentation, or computer model is known as reverse engineering. The reverse compiler that we have proposed here will work at the implementation phase of any project. Any software which will be written in C++ /any object oriented language have to follow forward engineering steps which include requirement specification step, design phase , implementation phase, testing phase and finally maintaining phase. In our case, the input to reverse compiler will be provided after implementation phase which will be processed and generates its corresponding java code after extracting its design from the code. Then the generated java code will be passed to testing phase and then to maintenance phase and follow the normal software testing phase.

A. Proposed Work

Algorithm1- Generalized Pseudo Code for converting C++ Source Code to Java Code

```
Begin
Read C++ Source Code Maintain Symbol table ST
L ⟵ Generates tokens from source program
Convert C++ tokens into corresponding java tokens
Err ⟵ Check syntax for generated tokens
if (length (Err)> 0) return error
else
B Generate Byte code using javac compiler
Return (B)
End if End
```

The above pseudo-code is the generalized pseudo-code for converting C++ source code into java source code. It simply takes code written in C++ language and generates its token and maintains symbol table for future references. Along with that it checks for the syntax of the language and if any error exits it will generate the error report, else it will produce corresponding java code and it's Bytecode.

Algorithm 2 - Pseudo Code for generating java code for pointers in C

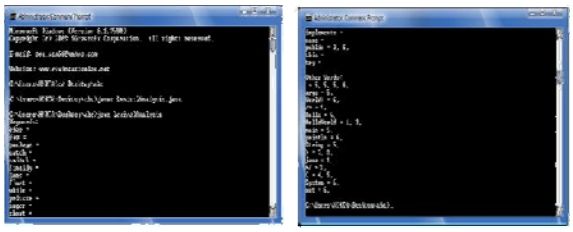
Begin
Read Specified C++ source code (containing pointer)
L ⟵ Breaks source code into tokens For k=1: length (L)
If (L_k is detected as pointer) update and convert detected
tokens into simple
variable of specified data type
else
no change end if
end for k
Err ⟵ Check syntax for updated L if (length(Err)>0) // error exit
return Err
else
B ⟵ Generates byte code for updated token using javac compiler
Return B
End if End

The above pseudo code generates the java code for provided C++ code as mentioned in the generalized pseudo code. It takes the same C++ code containing pointer in it and generates its java code. Again, initially it read the C++ code and breaks it into tokens which will be used for generating java code. First it identifies the pointer and its data type and after successful completion of syntax phase of compiler it simply converts the pointer to variable of same data type. Later on it generates its corresponding java code as done by the earlier mentioned pseudo-code.

Results and Discussions

Based on the proposed work, experiment is conducted for platform independent generic compiler. We use notepad++ editor; we also use net beans IDE. Results are discussed along with its descriptions. In the end over all description of the results are provided.

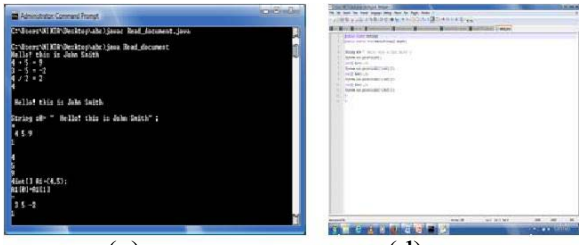
We start our first experiment by checking the keywords. All the keywords of C++ and java are checked. In second experiment we check whether the input data is string or operator and in third experiment code for pointers are generated for java.



(a) (b)

Figure 4 (a): Tokenization of keywords **(b):** Tokenization of Keywords with Line no.

In this part of the compiler design, Lexical phase of it has been implemented which takes input as C++ language and tokenize the whole programs after which it matches the keywords listed in Keyword.txt file. This part of program also maintains the table for generating any error report if exists. It maintains the line number corresponding to the every token and hence able to generate the report of error with line numbers.



(c) (d)

Figure 4 (c): Result of String & Operator in cmd prompt, **(d):** File generate after compiling Read_document

In this part, compiler takes input as a file (output of another program as a text file) and passes the file to Read_document.java . It then process that file and separates the line based on the type whether it is string or an arithmetic operation. Depending on whether the processed line is string or mathematical operation it generates the corresponding code in java language for that line.

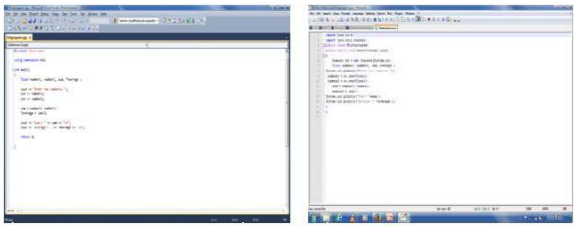


Figure 4 (e): Pointer program in c++, **(f):** Pointer Code generated in java after compiling

In this we passed program written in C++ language containing pointers (FirstProgram.cpp) to ReadAndWriteCode.java, which generates the java code having main class with the name of the file which had been passed to the program. It read the.cpp file and tokenizes the keyword and variable used in the program, if the pointer is found in the program it convert that pointer variable to normal variable according to java syntax and generates the code corresponding to the file input to the java program.

Implementation of compiler which converts independent java/C++ language code into platform independent byte code which runs on JVM has been implemented. It takes java/C++ code as an input and generates its corresponding byte code. Inside the program it first tokenize the code into “integer”, “identifier”, “variables”, etc. and checks it for syntax through syntax analyzer. If it passes the phase of syntax analyzer

then it passes the code to semantic analyzer and then to code generator. Finally if all phases passes that code successfully it will generate the optimized byte-code which runs only on JVM and hence becoming platform independent as well as language independent code.

Experiment Result and Comparison

By thorough analysis of compiler design and reverse engineering we came into conclusion. The comparison results are shown in the below given tables.

Table 6.1: C++ Compiler

S. NO.	Keywords used in C++	Keywords used in Java	Errors
1	I_{nt}	I_{nt}	Error Free
2	F_{or}	F_{or}	Error Free
3	F_{inally}	F_{inally}	Error Free
4	If	If	Error Free
5	W_{hile}	W_{hile}	Error Free
6	N_{app}	S_{tatic}	Error
7	N_{app}	A_{rgs}	Error
8	N_{app}	O_{ut}	Error
9	N_{app}	P_{rintln}	Error

C++ as premeditated for application and system programming enlarge C programming language. C++ compiler added standard library to execute program efficiently. Few examples of keywords used by C++ compiler are int, for, while etc. If we add keywords like static, args, out, println c++ compiler give error. These keywords are not in the standard libraries of c++.

Table 6.2: Java Compiler

S. No.	Keywords used in java	Keywords used in C++	Errors
1	System	Napp	Error
2	Out	Napp	Error
3	Println	Napp	Error
4	In	Napp	Error
5	Args	Napp	Error
6	For	For	Error Free
7	If	If	Error Free
8	While	While	Error Free
9	Napp	Cout	Error
10	Napp	Cin	Error

Java is a general purpose, class based, concurrent and object oriented programming language that is designed to have accomplishment dependences as possible. Java hinges on JVM to be extremely portable and secure. Java also provide standard library to execute program accurately. Few examples of keywords used in java libraries are for, if, while, println etc. If we add keywords like cout, cin while running java program they provide errors.

Table 6.3: C++ and Java Keyword in Generic Compiler

S. No.	Keywords of C++	Keywords of Java	Error
1	Int	Int	Error Free
2	For	For	Error Free
3	Finally	Finally	Error Free
4	Napp	Static	Error Free
5	Cout	Napp	Error Free
6	If	If	Error Free
7	While	While	Error Free
8	Napp	Args	Error Free
9	Napp	Println	Error Free
10	Cin	Napp	Error Free

The reason of generic compiler is to offer a language independent method of representing a complete program. In this we construct a standard library in which both C++ and java keywords are specified. Compiler starts toning the keywords with libraries and provides error free results.

Table 6.4: Pointer in C++

S. No.	Pointers in C++	Error
1	P_{ointer}	Error Free

A pointer is a variable whose importance is the address of a new variable. Pointer provides direct address to the memory position. Like any constant or variable, must declare a pointer before you can apply it to store any variable address. If we employ pointer in C++ it does not show any error.

Table 6.5: Pointer in Java

S. No.	Pointers in Java	Error
1	P_{ointer}	Error

Java array and classes are reference type. It provides reference to array and object as C++ pointer in C++. Unlike C++ pointer though, references in java are completely obscure. There is no technique to change a reference to primitive type and a reference cannot be decrement or increment. They are no size of operator or address of operators like *.

Table 6.6: Pointer in Generic Compiler

S. No.	Pointer in C++	Pointer in Java	Error
1	$A_{pplicable}$	$A_{pplicable}$	Error Free

Generic compiler read tokenizes keyword and variable used in the program, if the pointer is found in the program it convert that pointer variable to normal variable according to java syntax and generates the code corresponding to the file input to the java program. In this way we also apply pointer concept in our work to make our generic compiler independently work on every platform.

Since, the renovation has undergo an exhaustive research, extending compiler design(language independent to platform independent) will be beneficial to project development activity in modern methodology. So, we try to provide a generic compiler in which all object oriented programming languages can work in a particular environment. The reason of generic compiler is simply to give a language-independent technique of presenting a complete function. Generics compiler was instigate to the Java programming language to give tighter checks at compile time and to succor generic programming. This platform independent generic compiler is used to make a bridge between different object oriented languages i.e C++, java to give extended generic types.

Conclusion & Future Scope

Compiler is the set of instruction which translates the source code in to the target code. In this paper, the source code is C++ language and target code is the byte-code which is generated by java code acting as the semi-target for our compiler. Here we had focused on converting the code written in one language to the target code generated by other language. The source code that had been taken comes from the implementation phase of the process of software development cycle then revert back to design then again to implementation phase which will be in java language and hence doing the reverse engineering. Since, the transformation has undergo an intensive research, extending compiler design(language independent to platform independent) will be beneficial to project development activity in modern methodology. For future work this task can be taken to distributed system where such compiler are needed since C++ language are dedicated to only machine and cannot be used for distributed systems whereas java can be used for such problems.

References

[1] Kamthane, Ashok N., and Raj Kamal, “Computer Programming and IT: For RTU”, Pearson Education India, 2011.

[2] Carl, Daniel H., “How To Use Computer Safely: Data Secuarity and Optimization”, Explore Technology, 2015.

[3] Pressman, Roger S., “Software engineering: a practitioner's approach”, Palgrave Macmillan, 2005.

[4] Gerardo Canfora and Massimiliano Di Penta, “New Frontiers of Reverse Engineering”, RCOSI University of Sannio, Benevento, Italy, IEEE, 2007.

[5] Hannauni Aman, Rosziati Ibrahim, “Reverse Engineering: From Xml to Uml for Generation of Software Requirement Specification” 8th International Conference on Information Technology in Asia (CITA), 2013.

[6] Shivani Budhkar, Dr. Arpit Gopal, “Reverse Engineering Java Code to Class Diagram: An Experimental Report International Journal of Computer Application, vol-29, pp. 36-43, September 2011.

[7] Amit Kumar Gautam, Saurabh Diwaker, “Automatic Detection of Software Design Pattern from Reverse Engineering” Special Issue of International Journal of Computer Applicationson Issues and Challenges in Networking, Intelligence and Computing Technologies, pp.17-22, November 2012.

[8] Shikun Zhou, Hongji Yang and Paul Luker, Xudong He, “A Useful Approach to Developing Reverse Engineering Metrics”, IEEE 1999.

[9] Son, Yunsik, Seman Oh, and Yangsun Lee, "A Reversing Technique for Symbol Table Verification on Compiler Constructions" 2014.

[10] Budiselic, I., D. Skvorc, and S. Srbljic, "Designing the programming assignment for a university compiler design course", 37th International

Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), IEEE, 2014.

[11] Jean-Luc Hainaut, "Legacy and Future of Data Reverse Engineering", 16th Working Conference on Reverse Engineering, pp.4, IEEE2009

[12] Keith Gallagher, Cem Kaner JD & Jenifer Deign, "The Law and Reverse Engineering" 19th Working Conference on Reverse Engineering, pp.3-6, 2012.

[13] Lee, Hakjin, Hyunsang Youn, and Seunghwa Lee , "Automatic detection of design pattern for reverse engineering", 5th ACIS International Conference on Software Engineering Research, Management & Applications, IEEE, 2007.

[14] Ivan Klimek, Marián Keltika and František Jakab, "Reverse Engineering as an Education Tool in Computer Science" 9th IEEE International Conference on Emerging eLearning Technologies and Applications, pp. 123-126, IEEE2007.

[15] Wu, Chengyong, et al. "An overview of the open research compiler", Languages and Compilers for High Performance Computing Springer Berlin Heidelberg, pp. 17-31, 2005.

[16] Mallozzi, John S., "Thoughts on and tools for teaching compiler design" Journal of Computing Sciences in Colleges, vol-21.2, pp. 177-184, Springer 2005

[17] Djukic, Miodrag, et al. "An Approach to Instruction Set Compiled Simulator Development Based on a Target Processor C Compiler Back-End Design", First IEEE Eastern European Conference, IEEE, 2009.

[18] Mirko Viroli, "Effective and efficient compilation of run-time generics in Java" Electronic Notes in Theoretical Computer Science, vol-2, pp. 95-116, 2005.

[19] Youn, Jonghee M., et al. "Two versions of architectures for dynamic implied addressing mode" Journal of Systems Architecture, vol-8, pp. 368-383, 2010.

[20] Acher, Mathieu, et al. "Reverse engineering architectural feature models", Software Architecture. Springer Berlin Heidelberg, pp. 220-235, Springer, 2011.

[21] Cifuentes, Cristina, and Anne Fitzgerald, "The legal status of reverse engineering of computer software" vol-2, pp. 337-351,Springer, 2000.

[22] Aarne Ranta, "Implementing Programming Languages: An introduction to Compiler and Interpreters", College Publications, May9, 2012.

[23] Stromia, et al. "Reverse Engineering Legacy Interfaces: An Interaction Driven Approach" 6th Working Conference on Reverse Engineering, IEEE1999.

[24] Di Lucca G. A., Fasolino A.R., De Carlini U., Pace F.,Tramontana P., "WARE: a tool for the Reverse Engineering of Web Applications", 6th European Conference on Software Maintenance and Reengineering, IEEE,2002

[25] G. A. Di Lucca, M. Di Penta, G. Antoniol, G. Casazza, "An approach for reverse engineering of web-based applications", 8th Working Conference on Reverse Engineering, IEEE Computer Society Press, Los Alamitos, CA, 2001.

[26] Leupers, Rainer, "Compiler design issues for embedded processors", Design & Test of Computers, vol-4, pp 51-58, IEEE, 2002

[27] Margaret Rouse, "Computer Definition" Retrival 4/15/2015, http://www.techtarget.com/contributor/Margaret-Rouse.

[28] Terry, Rhodes, "Compilers and Compiler Generators: an introduction with C++." (2014).

[29] Chetan Birla, Mohit Singh, Bhupendra Yadav, Parth Nagar, Bhurhan 2013 "System & Application Software, compiler, interpreter, assembler" Retrival 4/18/2015, http://www.slideshare.net/chetanbirla/it-chetan

[30] Nikita Bakshi, Shruti Gujral, "Reverse Engineering And Its Realistic Applications", ISSN 0975-5462, vol-6, pp. 370-372, International Journal of Engineering Science and Technology, June 2014.