iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 3 **Issue:** VI **Month of publication:** June 2015

DOI:

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)

Generic Parser

Ms. Shrunkhala Satish Wankhede[1], Mr. Shubham Itankar[2], Mr. Sumit Borse[3]
Department of CSE, Priyadarshini Bhagwati College Of Engineering, Nagpur-24, Maharashtra, India

Abstract - This paper shows how to implement generic parsers as a family of streams. This allows for very readable, maintainable and flexible parsers. The method is illustrated with a parser for simple arithmetic expressions, but can easily be extended to a parser for a full-fledged programming language. Moreover, the same technique can be applied to the entire process from lexing to execution, since actions can be associated with each sub-parser. The parsing of input is a very important problem appearing in many different parts of software development - parsing user input in the form of command-line options, the parsing of arithmetic expressions in a calculator, parsing values in a user-defined configuration file or compiling some programming language. This makes it important to have different approaches. What we will present here in this project, is a method of parsing inspired by what is done in functional programming (FP). The paper is not about functional programming in C++. Moreover, the presence of too many parentheses makes the code harder to read and is hence also more error prone, rather it is about how to implement a particularly elegant idea from FP in an object oriented context.
Keywords— Generic Parser, Integer Point Parser, Floating Point Parser

I. INTRODUCTION

In this paper, we will concentrate on the parsing step, but in such a way that the remaining steps of the process could be implemented in a similar fashion. In fact, we will see how to make a simple modification to our parser and turn it into an expression calculator. For concreteness, we will consider a particular example, which is simple enough not to introduce unnecessary complications yet complex enough to be non-trivial.

The chosen example is the parsing of arithmetic expressions like 1+ (2-3)*4. We will only consider integers. Furthermore, we will not worry about the precedence levels of the standard arithmetic operators - this could be done by slightly modifying the grammar as shown in for instance. It will be shown later how to accommodate this with very few changes to our framework.

The parsing of input is a very important problem appearing in many different parts of software development - parsing user input in the form of command-line options, the parsing of arithmetic expressions in a calculator, parsing values in a user-defined configuration file or compiling some programming language. The aim of this approach is flexibility: it is easy to omit or modify individual steps, which is important for playing around with different approaches in language design or compiler construction.

II. MODULES

A. Graphical Module

This module contains the basic instruction about the using the parser. In this module we have displayed the name of developers. Some basic information displayed on screen as for the use of Parser.

B. Start Module

User has to enter an expression. If any variable is not defined previously, then parser would automatically consider its value to be zero. Give statement must be syntactically correct expression, otherwise it return an error which may cause garbage value as answer.

C. Evaluation Module

In this module, the result will be displayed in the form of answer of given algebraic expression. By entering a period (.) you can exit from the default parser. For exiting from the program again hit hash button.

III. WORKING OF THE PROJECT

The Generic Parser is basically takes an input of an expression and tokenizes it like every parser does inside the compiler or the other application. It classifies the whole strings characters into the delimiters, variables and numbers. Initially it assigns the value of every variable as zero. To exit from the floating point pointer. And after that you will be inside the Integer point parser. This parser will only be able to proceed with the integer value. Even if you give a floating point value it will assume it as simple integer and

takes the first digit of the number. In the end it displays the execution time of your program.

IV. REQUIREMENTS

A. Hardware requirements

TABLE I: HARDWARE REQUIREMENTS

Hardware	Minimum	Recommended
CPU	1.6 GHz	2.2GHz or Higher
RAM	385MB	512MB or more
Hard disk	60GB	80 GB

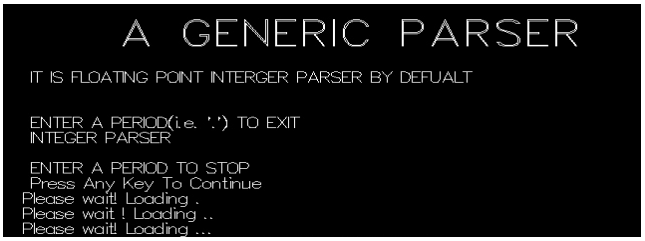
B. Software Requirements

TABLE II: SOFTWARE REQUIREMENTS

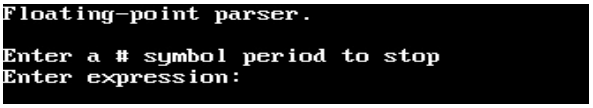
Operating System	Microsoft XP/7/VISTA/8/8.1
Software Platform	Turbo C

V. OUTPUT

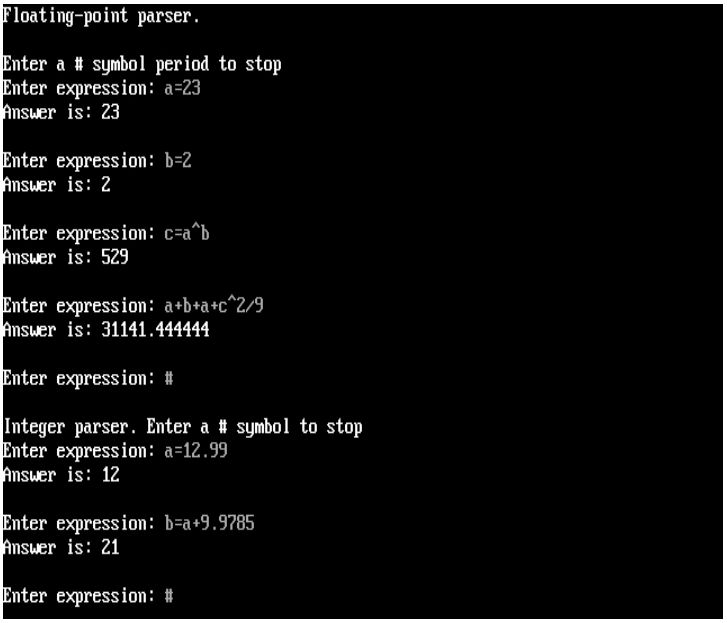
A. Start module



B. Read instruction before parser



C. Test gets started



D. Result

VI. APPLICATION

The parsing of input is a very important problem appearing in many different parts of software development - parsing user input in the form of command-line options, the parsing of arithmetic expressions in a calculator, parsing values in a user-defined configuration file or compiling some programming language. Applications of parsing include everything from simple phrase finding e.g. for proper name recognition to full semantic analysis of text.

VII. CONCLUSION

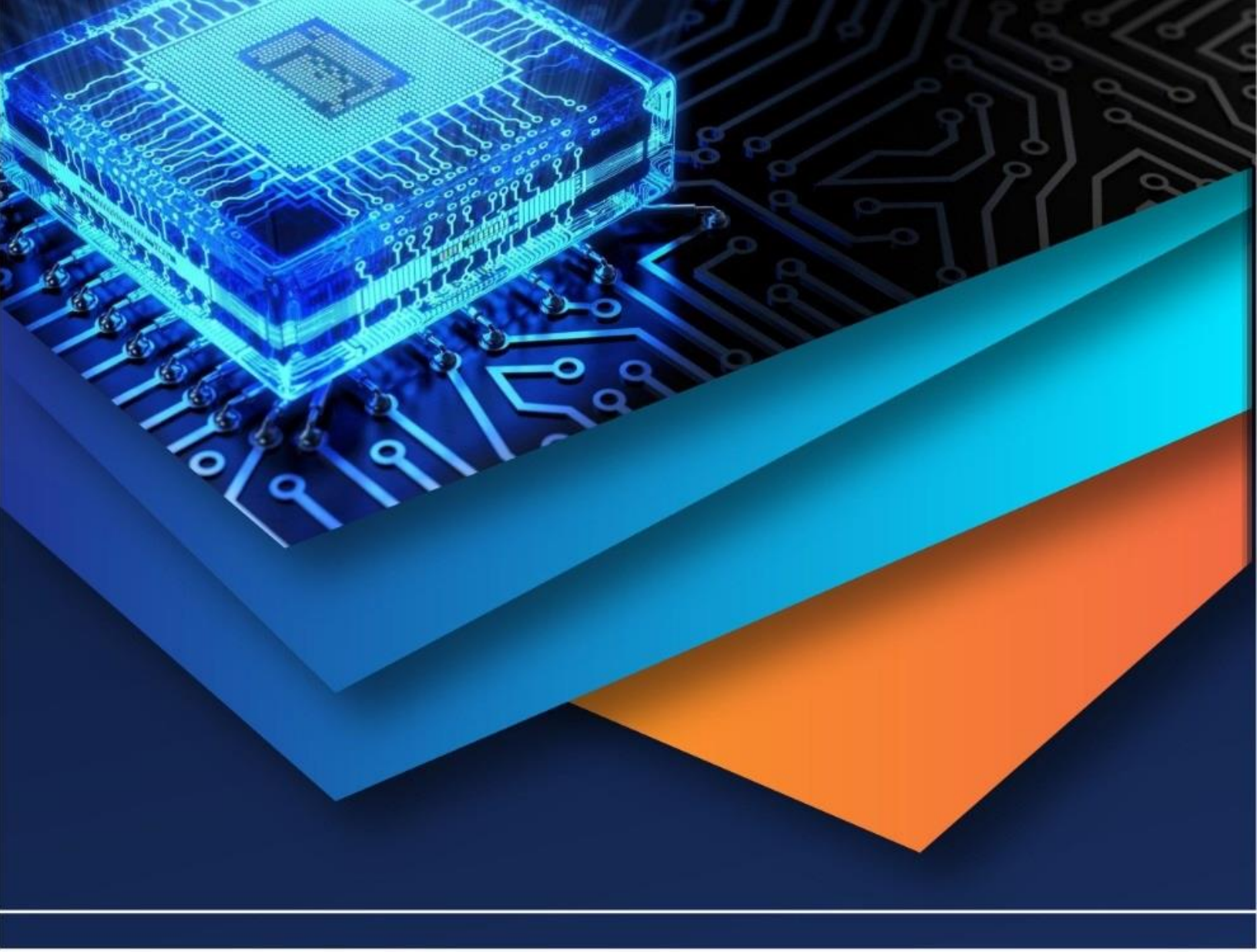
We can make your own grammar rules and implement in this Parser. It can be used for the learning of Compiler/Interpreter construction. It can be customized with different types of Interpreter/Compilers.

VIII. ACKNOWLEDGMENT

We express our deep sense of gratitude to Mr. M. S. Chaudhari, H.O.D. of Computer Science & Engineering department for his continuous encouragement and motivations. We are grateful to our management for providing the excellent infrastructural and technical facilities. We sincerely thank our Principal, Dr. N.K. Choudhari for his keen. We express our admirations for Dr. (Mrs.) A.R. Chaudhari, Academic incharge, for her valuable advice and support throughout this venture. We deeply indebted to numerous Authors whose books and papers we have consulted to clear our perception about various parts of this project.

REFERENCES

[1] B.Smith , “An approach to graphs of linear forms (Unpublished work style),” unpublished, [2005] IEEE conference publications
[2] Research on Parsing System Based on Computer -Yu Jinping; Sun Wenjie[2002]
[3] Design and implementation of Generic parser System based on Compiler-Huiping Wang;Ruowu Zhong[1997]
[4] Zhenghong Gao, Yinfei Pan, Ying Zhang, Kenneth Chiu: A High Performance Schema-Specific XML Parser, Third IEEE International Conference on e-Science and Grid Computing 2007
[5] Zhou Yanming, Qu Mingbin: A Run-time Adaptive and Code-size Efficient XML Parser, Proceedings of the 30th Annual International Computer Software and Applications Conference (COMPSAC'06), 2006 IEEE
[6] Wei Zhang and Robert A. van Engelen: An Adaptive XML Parser for Developing High-Performance Web Services, IEEE 2008
[7] Fernando Farfán, Vagelis Hristidisa, Raju Rangaswami, "2LP: A double-lazy XML parser", Information Systems, Volume 34, Issue 1, pp 145-163, March 2009.
[8] R. Bourret, C. Bornhövd, A. Buchmann: A Generic Load/Extract Utility for Data Transfer Between XML Documents and Relational Databases, IEEE 2003.
[9] Zhang, W. and Van Engelen, R. (2006): A Table-Driven Streaming XML Parsing Methodology for High-Performance Web Services, 2006 IEEE International Conference on Web Services (ICWS): 197-204
[10] Mehdi Haïtami, Ghyslain Abel, Alain C. Houle, Brigitte Jaumard, "ONDE: A Generic XML-Based Development Environment for Optimization of WDM Optical Networks", IEEE CCECE/CCGEI, Ottawa, May 2006
[11] A. Aho, R. Sethi, and J. Ullman. Compilers: Principles, Techniques and Tools. Addison-Wesley Publishing Company, Reading MA, 1985



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY