**NITTE MEENAKSHI INSTITUTE OF TECHNOLOGY**

(AN AUTONOMOUS INSTITUTION, AFFILIATED TO VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELGAUM, APPROVED BY AICTE & GOVT.OF KARNATAKA

Logo

Description automatically generated

**LA-2 REPORT**

on

**OPERATOR PRECEDENCE**

*Submitted in partial fulfilment of the requirement for the LA Component of Compiler Design of 7th Semester*

*Bachelor of Engineering*

*in*

*Computer Science and Engineering*

Submitted by:

V VENKATA SREE HARSHA 1NT18CS181

NAGARIGERI SAICHARAN REDDY 1NT18CS104

|  |  |
| --- | --- |
| POORNAVIKAS A S | 1NT18CS113 |
|  |  |

*Logo

Description automatically generated*

**Department of Computer Science and Engineering**

2021-22

**Text

Description automatically generatedNitte Meenakshi Institute of TechnologyLogo

Description automatically generated**

**(AN AUTONOMOUS INSTITUTION AFFILIATED TO VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELGAUM)**

**PB No. 6429, Yelahanka, Bangalore 560-064, Karnataka**

**Telephone: 080- 22167800, 22167860**

**Fax: 080 – 22167805**

**Department of Computer Science and Engineering**

**CERTIFICATE**

This is to certify that the Course Project titled “Operator Precedence” is an authentic work carried out by V Venkata Sree Harsha(1nt18s181),Nagarigeri Saicharan Reddy(1nt18cs104), Poornavikas A S (1nt18cs113) Bonafide students of **Nitte Meenakshi Institute of Technology**, Bangalore in partial fulfilment for the LA Component of Compiler Design Subject of 7th Semester of ***Bachelor of Engineering*** in Computer Science and Engineering of Visvesvaraya Technological University, Belagavi during the academic year ***2021-2022.***

|  |  |  |
| --- | --- | --- |
| **Ms. Uma R**  **Faculty In-charge** |  | **Dr. Sarojadevi H**  **HOD-CSE** |

**DECLARATION**

I hereby declare that

(i) This Presentation does not contain text, graphics or tables copied and pasted from the Internet, unless specifically acknowledged, and the source being detailed in the report and in the References sections.

(ii) All corrections and suggestions indicated during the internal presentation have been incorporated in the report.

(iii) Content of the report has been checked for the plagiarism requirement.

|  |  |  |
| --- | --- | --- |
| **NAME** | **USN** | **Signature** |
| NAGARIGERI SAI CHARAN REDDY | 1NT18CS104 | A picture containing text, whiteboard  Description automatically generated |
| V VENKATA SREE HARSHA | 1NT18CS181 | Text, letter  Description automatically generated |
| POORNA VIKAS A S | 1NT18CS113 | Text  Description automatically generated with low confidence |

|  |
| --- |
|  |
|  |  |  |

**Date:** 20-01-2022

**ACKNOWLEDGMENT**

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crowned our effort with success. We express our sincere gratitude to our Principal **Dr. H. C. Nagaraj**, Nitte Meenakshi Institute of Technology for providing the facilities.

We thank our HoD**, Dr.Sarojadevi H** for the excellent environment created to further educational growth in our Department.

We also thank all our friends, teaching and non-teaching staff at NMIT, Bangalore, for all the direct and indirect help provided in the completion of the project.

|  |  |  |
| --- | --- | --- |
| **NAME** | **USN** | **Signature** |
| NAGARIGERI SAI CHARAN REDDY | 1NT18CS104 | A picture containing text, whiteboard  Description automatically generated |
| V VENKATA SREE HARSHA | 1NT18CS181 | Text, letter  Description automatically generated |
| POORNA VIKAS A S | 1NT18CS113 | Text  Description automatically generated with low confidence |

Date:20-01-2022

**TABLE OF CONTENTS**

**ABSTRACT**

**1:** INTRODUCTION

**2:** BACKGROUND OF THE WORK

**3:** DESIGN

**4:** IMPLEMENTATION

**5:** RESULT

**6:** CONCLUSION

**7:** REFERENCES

**ABSTRACT**

An operator-precedence parser is a simple [shift-reduce parser](https://en.wikipedia.org/wiki/Shift-reduce_parser) that is capable of parsing a subset of [LR(1)](https://en.wikipedia.org/wiki/LR_parser) grammars. More precisely, the operator-precedence parser can parse all LR(1) grammars where two consecutive [nonterminal](https://en.wikipedia.org/wiki/Nonterminal) and [epsilon](https://en.wikipedia.org/wiki/Empty_string) never appear in the right-hand side of any rule.

Operator-precedence parsers are not used often in practice, however, they do have some properties that make them useful within a larger design. First, they are simple enough to write by hand, which is not generally the case with more sophisticated right shift-reduce parsers. Second, they can be written to consult an operator table at [run time](https://en.wikipedia.org/wiki/Run_time_(program_lifecycle_phase)), which makes them suitable for languages that can add to or change their operators while parsing

**1.OPERATOR-PRECEDENCE-GRAMMAR:**

A grammar that is used to define mathematical operators is called an operator grammar or operator precedence grammar. Such grammars have the restriction that no production has either an empty right-hand side (null productions) or two adjacent non-terminals in its right-hand side

Technically, an operator precedence grammar is a [context-free grammar](https://en.wikipedia.org/wiki/Context-free_grammar) that has the property (among others[]](https://en.wikipedia.org/wiki/Operator-precedence_grammar#cite_note-1)) that no production has either an empty right-hand side or two adjacent nonterminal in its right-hand side. These properties allow precedence [relations](https://en.wikipedia.org/wiki/Relation_(mathematics)) to be defined between the terminals of the grammar. A [parser that exploits these relations](https://en.wikipedia.org/wiki/Operator-precedence_parser) is considerably simpler than more general-purpose parsers such as [LALR parsers](https://en.wikipedia.org/wiki/LALR_parser). Operator-precedence parsers can be constructed for a large class of context-free grammars.

**2. OPERATOR PRECEDENCE PARSER:**

An operator precedence parser is a bottom-up parser that interprets an operator grammar. This parser is only used for operator grammars. Ambiguous grammars are not allowed in any parser except operator precedence parser.

There are two methods for determining what precedence relations should hold between a pair of terminals.

The first method is to use the conventional associativity and precedence of operator.

The second method of selecting operator-precedence relations is first to construct an unambiguous grammar for the language, a grammar that reflects the correct associativity and precedence in its parse trees.

**3. PRECEDENCE RELATIONS:**

In operator-precedence parsing, we define three disjoint precedence relations between certain pairs of terminals

a<b ‘b' has higher precedence than ‘a’

a=b ‘b’ has same precedence as ‘a’

a>b ‘b’ has lower precedence than ‘a’

The determination of correct precedence relations between terminals are based on the traditionals are based on the traditional notions of associativity and precedence of operators

**Using Operator-Precedence Relations:**

E -> e+e | e\*e | e | e/e | e^e | (e) | -e | id | e-e

The partial operator-precedence table for this grammar

Then the i/p string (id+id\*id) with the precedence relations inserted will be

$<id> + <id> \* <id>$

Table

Description automatically generated

**4. OPERATOR-PRECEDENCE PARSING ALGORITHM:**

The input string is w$, the initial stack is $ and a table holds precedence relations between certain terminals

Set p to point to the first symbol of w$:

repeat forever

if ($ is on top of the stack and p points to $) then return

else {

let a be the topmost terminal symbol on the stack and let b be the symbol pointed to by p;

if (a<b or a=b ) then { /\*shift \*?

push b onto the stack

advance p to the next input symbol;

}

else if (a<b) then /\*reduce\*/

repeat pop stack

until (the pop of stack terminal is related by < to the terminal most recently popped);

else error();

}