OPERATOR PRECEDENCE TABLE

18CSC304J – Compiler design Mini Project Report

TEAM MEMBERS :

ASHWIN KUMAR K - RA2011003010586

SYED ABSAR QADRI – RA2011003010585

INTRODUCTION :

Operator precedence table, also known as operator precedence chart or operator precedence hierarchy, is a tool used in computer programming and mathematics to determine the order in which operators are evaluated in an expression. Operators are symbols or keywords used in programming languages to perform operations on operands, which are the values or variables that the operators act upon.

An operator precedence table typically lists the different operators used in a programming language along with their associated precedence levels, which indicate the order in which they are evaluated. Operators with higher precedence are evaluated before operators with lower precedence. When an expression contains multiple operators, the one with the higher precedence is evaluated first, followed by operators with lower precedence, unless overridden by parentheses or other rules.

PROBLEM STATEMENT :

- The problem statement regarding operator precedence table typically includes issues related to the incorrect evaluation of expressions due to ambiguous or undefined precedence rules.

- The problem may arise when different operators with different precedence levels are used in the same expression, leading to unexpected results.
- Operator precedence table problems can also involve errors in the order of operations, leading to incorrect results or unintended behavior in the program.
- The problem may also involve inconsistencies or conflicts in the precedence rules defined by programming languages or compilers, resulting in confusion or incorrect evaluation of expressions.
- Another issue related to operator precedence table is the lack of standardization across programming languages, leading to differences in behavior and results when expressions with multiple operators are used.

PRECEDENCE AND ASSOCIATIVITY :

Operator precedence and associativity are concepts used in computer programming to determine the order in which operators are evaluated in expressions. Here's a brief explanation of each:

Operator Precedence:

- Operator precedence defines the priority of operators in an expression, determining which operators are evaluated first.
- Operators with higher precedence are evaluated before operators with lower precedence.
- For example, in the expression "2 + 3 * 4", the multiplication (*) has higher precedence than addition (+), so it is evaluated first, resulting in 2 + (3 * 4) = 14.
- Associativity:

- Associativity defines the order in which operators with the same precedence are evaluated.
- Operators can be left-associative, meaning they are evaluated from left to right, or right-associative, meaning they are evaluated from right to left.
- For example, in the expression "5 - 3 - 1", the subtraction (-) operator is left-associative, so it is evaluated from left to right, resulting in (5 - 3) - 1 = 1.

OPERATOR PRECEDENCE TABLE

An operator precedence table is a reference table that lists the operators in a programming language along with their respective precedence levels. It helps determine the order in which operators are evaluated when multiple operators are used in the same expression. Here is an example of an operator precedence table

```diff
Operator       | Precedence Level
-------------- | ----------------
()             | Highest
++, --         |
*, /, %        |
+, -           |
<, <=, >, >=   |
==, !=         |
&&             |
||             |
=              | Lowest
```

CONSTRUCTION OF OPERATOR PRECEDENCE TABLE :

Constructing an operator precedence table involves identifying the operators used in a programming language and determining their precedence levels. Here are the general steps to construct an operator precedence table:

1. Identify the operators: Make a list of all the operators used in the programming language for which you want to create an operator precedence table. This may include arithmetic operators (e.g., +, -, *, /), relational operators (e.g., <, <=, >, >=), equality operators (e.g., ==, !=), logical operators (e.g., &&, ||), assignment operators (e.g., =), and any other operators specific to the language.
2. Determine precedence levels: Assign precedence levels to the operators based on their relative priority. Operators with higher priority should have higher precedence levels, while operators with lower priority should have lower precedence levels. For example, multiplication (*) and division (/) usually have higher precedence than addition (+) and subtraction (-), so they would have higher precedence levels.
3. Consider associativity: Determine the associativity of the operators, which specifies the order of evaluation for operators with the same precedence level. Operators can be leftassociative, meaning they are evaluated from left to right, or right-associative, meaning they are evaluated from right to left. For example, in some languages, assignment operators (=) are right-associative, while most arithmetic operators are leftassociative.
4. Create the table: Based on the precedence levels and associativity of the operators, construct a table that organizes the operators in rows and columns. The rows represent the operators' precedence levels, and the columns represent the operators themselves. Fill in the table with the operators and

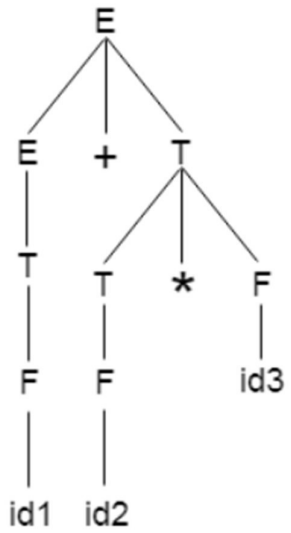
their respective precedence levels, taking into account the associativity rules.

5. Verify and test: Once the operator precedence table is constructed, verify it against the language's documentation and test it with different expressions to ensure that the table accurately reflects the language's precedence and associativity rules. Make adjustments as necessary based on the language's specifications or any observed discrepancies.

Grammer :

1. E → E+T/T
2. T → T*F/F
3. F → id

Given string: w = id + id * id



	E	T	F	id	+	*	$
E	X	X	X	X	≐	X	⋗
T	X	X	X	X	⋗	≐	⋗
F	X	X	X	X	⋗	⋗	⋗
id	X	X	X	X	⋗	⋗	⋗
+	X	≐	⋖	⋖	X	X	X
*	X	X	≐	⋖	X	X	X
$	⋖	⋖	⋖	⋖	X	X	X

Now let us process the string with the help of the above precedence table:

$ ⋖ id1 ⋗ + id2 * id3 $

$ ⋖ F ⋗ + id2 * id3 $

$ ⋖ T ⋗ + id2 * id3 $

$ ⋖ E ≐ + ⋖ id2 ⋗ * id3 $

$ ⋖ E ≐ + ⋖ F ⋗ * id3 $

$ ⋖ E ≐ + ⋖ T ≐ * ⋖ id3 ⋗ $

$ ⋖ E ≐ + ⋖ T ≐ * ≐ F ⋗ $

$ ⋖ E ≐ + ≐ T ⋗ $

$ ⋖ E ≐ + ≐ T ⋗ $

$ ⋖ E ⋗ $

CONCLUSION :

the operator precedence table is a critical component of an operator precedence parser, which is used to parse expressions involving operators with different levels of precedence. It determines the order in which operators are evaluated during parsing, based on their precedence levels. Operators with higher precedence are evaluated before operators with lower precedence. An operator precedence table typically assigns a precedence level to each operator, which is used by the parser to properly handle the expression. Operator precedence

parsing is a widely used parsing technique in computer programming for parsing arithmetic expressions and other expressions involving operators, and it plays a crucial role in ensuring correct parsing and evaluation of expressions.

REFERENCE :

1. https://www.programiz.com/pythonprogramming/precedence-associativity

2.https://www.dremendo.com/python-programming-tutorial/python-operator-precedence