# SER: 502 Emerging Languages and Programming Paradigms

# Language :ROCH

Team: 25

# TEAM MEMBERS

- ADITYA KUMAR: 1213250361
- NISHITI SAWANT: 1213233747
- AANCHAL MAHAJAN: 1213094075
- MAHIMA GUPTA: 1213110741

# CONTENTS

# INTRODUCTION

- Language suited for beginners
- Imperative programming language
- JAVA is used to create the compiler and the runtime
- Uses **hashmap** and **stack** as the data structure for SYMBOL TABLE
- Runtime is simple and fast

# TOOLS USED

- Lexical Analysis and Parsing is done using Antlr 4.7
- Intermediate code generation is done using Antlr and JAVA
- Runtime Environment is in Java 1.8

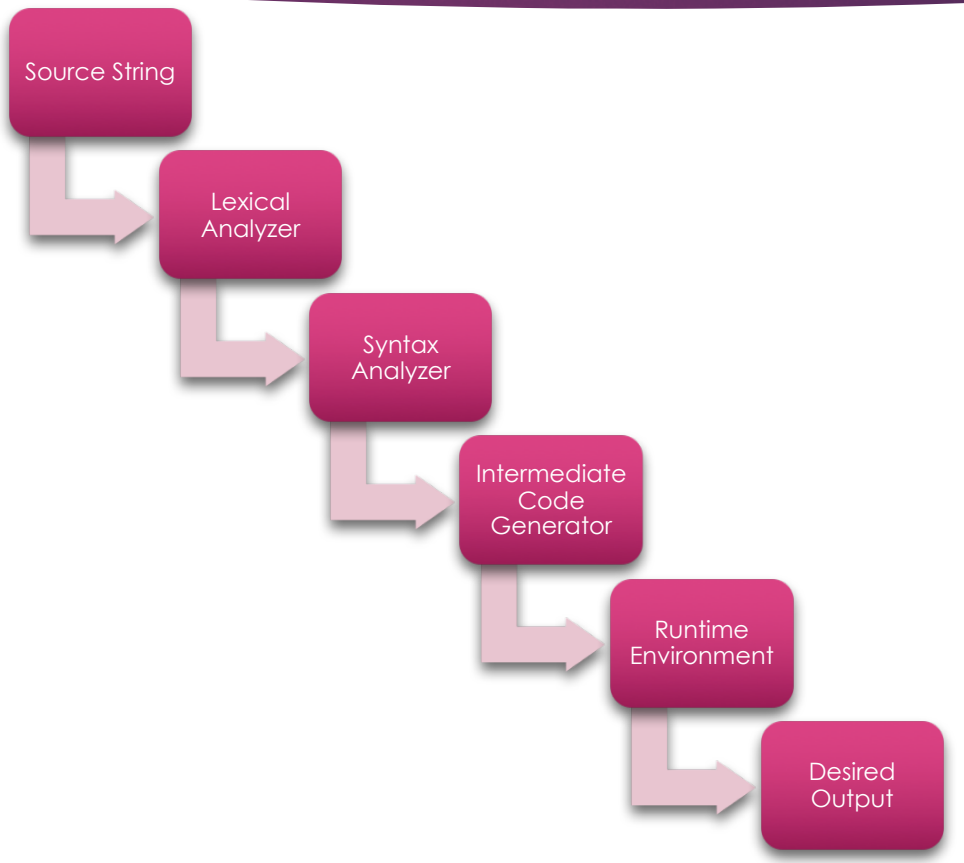
# FEATURES

- Data Types
  - Num – For integer type
  - Bool – For boolean type
- Operators
  - Arithmetic - *,/,+,-
  - Conditional - <,>,<=,>=
- Precedence of the operators
  - Handles precedence of the operators and can calculate complex expressions like a+b*c/e*f according to the rules of the precedence

# FEATURES

- Conditional Constructs
  - Check-otherwise which checks for an expression to be true or false and executes the statements accordingly
  - Supports nested Check
- Iterative Constructs
  - Until construct which runs until the expression passed to it is true

# DESIGN


Source String
Lexical Analyzer
Syntax Analyzer
Intermediate Code Generator
Runtime Environment
Desired Output

# GRAMMAR

- Roch has a Context Free Grammar.

1. grammarstart : block of code
2. block : set of statements
3. statement : variable declaration or assignment or display or checkCondition or until Loop
4. assignment : arithmeticExpression | conditionalExpression
5. dataType: ('num' | 'bool')
6. display: 'display(' (arithmeticExp | conditionalExp) ')';

# Compiler

- Our parser works in the top-down manner.
- ANTLR tool operates on the RochGrammar.g4 and performs scanning and parsing of the file to produce the compiler and the parser file.
- The scanner and the parser classes are used to generate the parse tree.
- Walker class is used subsequently to walk through all the nodes of the parse tree.
- The traversal of the nodes is followed by the generation of the intermediate files.
- Runtime operates on the intermediate file generated to produce the output. Runtime is generated using JAVA.

# INTERMEDIATE CODE GENERATION

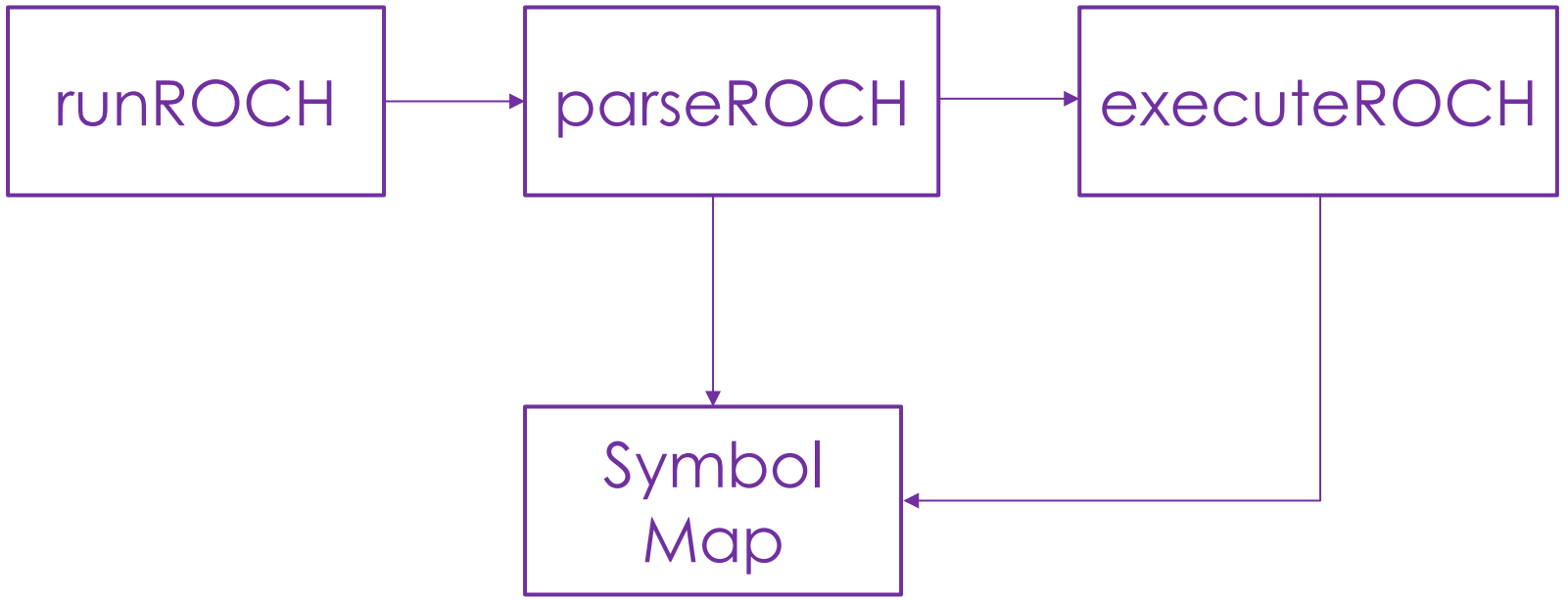
```
start
num a
a = 15
num b
b=10
num c
c = a+b
display(c)
c = a-b
display(c)
c = a*b
display(c)
c = a/b
display(c)
terminate
```

```
NUM a
PUSH a 15
NUM b
PUSH b 10
NUM c
NUM load1
ADD load1 a b
PUSH c load1
PRINT c
NUM load2
SUB load2 a b
PUSH c load2
PRINT c
NUM load3
MUL load3 a b
PUSH c load3
PRINT c
NUM load4
DIV load4 a b
PUSH c load4
PRINT c
TERMINATE
```

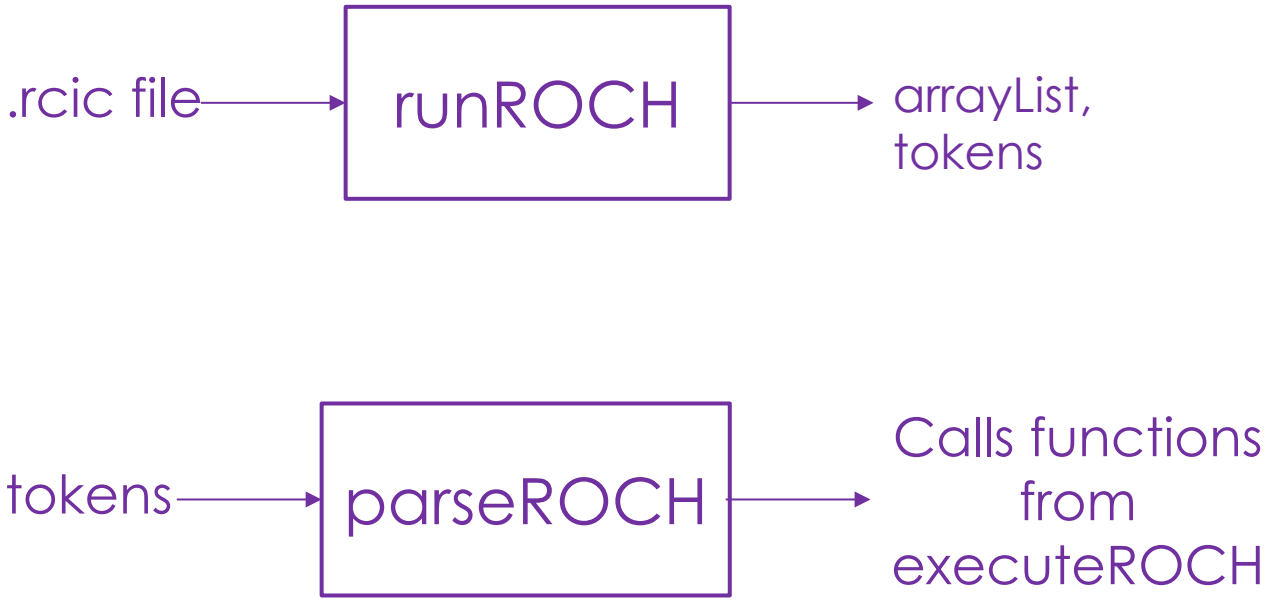
High level language code

Intermediate code

# RUNTIME



# RUNTIME


.rcic file
runROCH
arrayList,
tokens
tokens
parseROCH
Calls functions
from
executeROCH

# RUNTIME


tokens
parseROCH
Calls functions from executeROCH


Symbol Table
executeROCH
Output

# FUTURE ADVANCEMENTS

- We can add many features to ROCH which current high level languages have.
- Since this is a language developed for beginners, there's a lot of scopr for improvement.
- We can add a complex data-structure to ROCH
- We can add functions to ROCH
- More custom looping constructs can be added