BEST

SER 502 PROGRAMMING LANGUAGES
GROUP-20

ABDUL SAMAD KHAN
ANUSHA CHITOOR PREM
SALIL MALIK
VIJAYA MOUNIKA GADDE

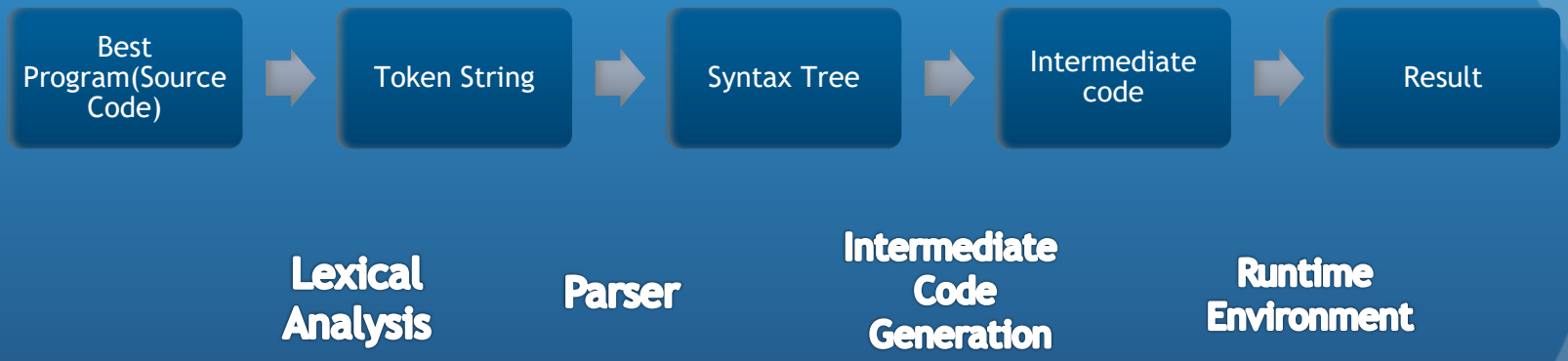
GLOSSARY

- Overview
- Tools Used
- Key Features
- Language Design
- Grammar
- BEST Compiler
- Intermediate Code Generation
- Runtime
- Execution
- Sample Program

OVERVIEW

- BEST Language is an imperative language mainly inspired from Java.
- It is easy to code because of the familiar keywords that are used and is aimed at first time programmers to make it easy to understand and implement.
- Runtime is written In Java and is simple, fast and efficient.
- It is a concise, limited and specialized language.
- Designed for beginners.

Overview


Best Program(Source Code)
Token String
Syntax Tree
Intermediate code
Result
Lexical Analysis
Parser
Intermediate Code Generation
Runtime Environment

Tools Used

- ANTLR is a language tool that provides a framework for constructing recognizers, Interpreters, Compilers and translators from a given grammar input. It is composed basically of 3 parts which are Lexer, Parser, Abstract Syntax tree(Parser).
- We used Antlr 4.7 for lexical analysis and parsing.
- We used Antlr and java for Intermediate code generation.
- We used java 10 to build the compiler and the runtime.

KEY FEATURES

- Simple Syntax
- Arithmetic Operators like +, -, *, /,%
- Relational Operators like >, >=,<,<=,<>
- Assignment Operator like =
- Handles complex expressions with Operator precedence.

KEY FEATURES

- DECISIONS CONSTRUCTS

  Whenever (CONDITION) then {PROGRAM}

  Whenever (CONDITION) then {PROGRAM} other {PROGRAM}

- LOOPS CONSTRUCTS

  'loop' VARIABLE '::' expression '--' condition '--' expression

  '{' program '}' ;

KEY FEATURES

- FUNCTION CONSTRUCTS

function : 'method' *VARIABLE '(' function_parameters ')' '{' program'}' ;*

call : 'calling function' *VARIABLE '('function_parameters ')'* ;

function_parameters : *(VARIABLE)(','VARIABLE)** | ;

- PRINT EXPRESSION

'print : 'print ' expression | 'print ' ""VARIABLE"" | 'print ‘ ""NUMBER"" ;

Grammar

```
grammar BEST;
start : program ;
program : program_block ';' program | program_block ';' ;
program_block : declaration | assignment | whenever | loop | function | call | print | ;
declaration : 'declare' VARIABLE ;
assignment : VARIABLE 'equals' expression ;
whenever : 'whenever' '(' condition ')' 'then' '{' program '}' |
          'whenever' '(' condition ')' 'then' '{' program '}' 'other' '{' program '}' ;
loop : 'loop' VARIABLE '::' expression '--' condition '--' expression '{' program '}' ;
function : 'method' VARIABLE '(' function_parameters ')' '{' program'}' ;
call : 'calling function' VARIABLE '('function_parameters ')'  ;
function_parameters : (VARIABLE)(','VARIABLE)* | ;
print : 'print ' expression | 'print ' '"'VARIABLE'"' | 'print ' '"'NUMBER'"' ;
condition : expression '=' expression | expression '<>' expression |
              expression 'lt' expression | expression 'gt' expression |
              expression 'lte' expression | expression 'gte' expression | boolean_expression ;
boolean_expression : 'True' | 'False' ;
expression : expression2 expression3 ;
expression3 : '+' expression2 expression3 | '-' expression2 expression3 | ;
expression2 : expression5  expression4 ;
expression4 : '*' expression5 expression4 | '/' expression5 expression4 |'%' expression5 expression4 | ;
expression5 : '(' expression ')' | NUMBER | VARIABLE | boolean_expression;
NUMBER :  [0-9]+;
VARIABLE : [a-z|A-Z]+ ;
WS : [ \t\r\n]+ -> skip ;
```

Language Design

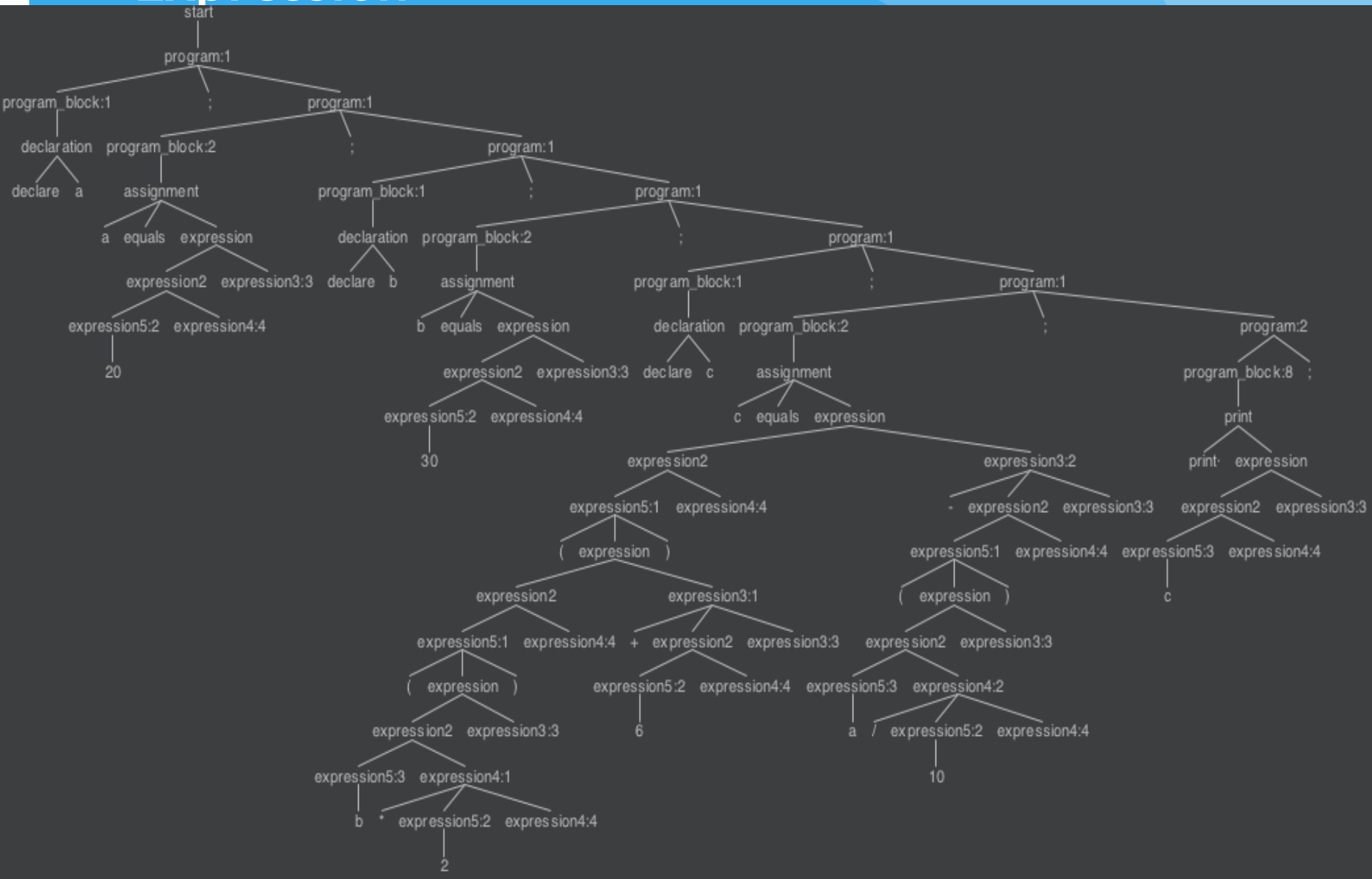
Grammar

- BEST Language has a Context Free Grammar.
- Grammar used is Extended Backus-Naur form or EBNF.
- Extended Backus-Naur form (EBNF) is a family of meta syntax notations, any of which can be used to express a context-free grammar.
- An EBNF consists of terminal symbols and non-terminal production rules which are the restrictions governing how terminal symbols can be combined into a legal sequence.

Example Program for expression Evaluation

```
declare a;

a equals 20;

declare b;

b equals 30;

declare c;

c equals ((b*2) + 6) - (a / 10);

print c;
```

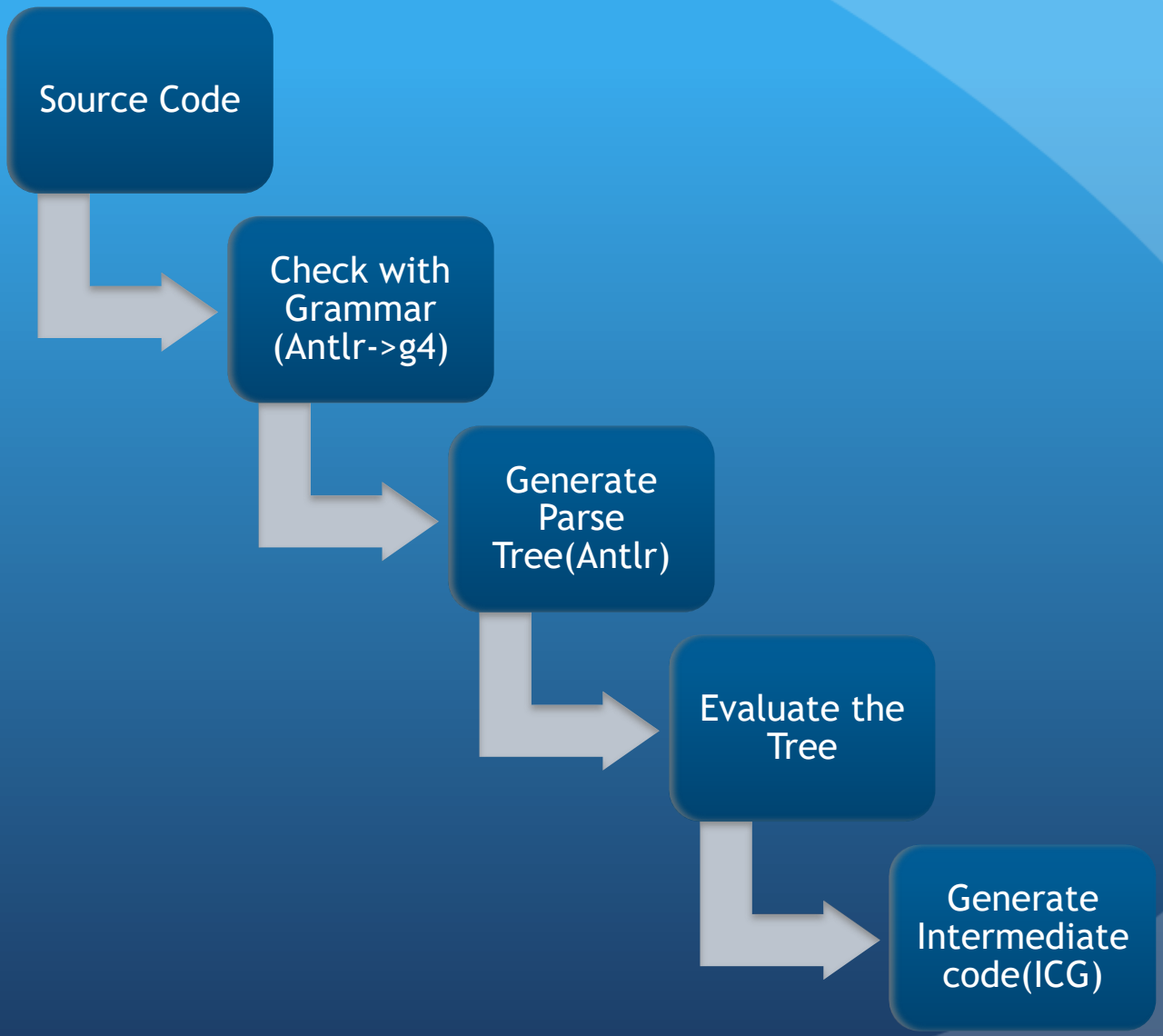
Parse Tree example for Evaluation of an Expression



Best Compiler

- Parsing of the grammar is done in a top down manner.
- The .g4 file is fed into the Antlr tool and the tool performs scanning and parsing of the file and thus produces the compiler and parser file.
- The ANTLR parser and scanner classes helps us in generating the parse tree.

Intermediate Code Generation



Best Runtime

- The intermediate .best file s read in the runtime module and every line is segmented into tokens.
- Intermediate code is interpreted line by line and it uses prefix notations.
- The complete runtime module is built on java 10

Token Literal Names

- null
- ';'
- 'declare'
- 'equals'
- 'whenever'
- '('
- ')'
- 'then'
- '{'
- '}'

Sample Program

```
method printReverseFunction(a)
{
    a equals a + 1;
    whenever( a <> 5 )
    then {
        calling function printReverseFunction(a);
        };
    print a;
};
declare a;
a equals 0;
calling function printReverseFunction(a);
```

Sample Byte Code

```
METHOD factorialIterative
MOV a
METHOD_START_factorialIterative
STORE c
PUSH 1
MOV c
STORE i
PUSH 1
MOV i
LOOP 0
PUSH i
PUSH a
LESS
LOOP_START_0
PUSH c
PUSH a
ADD
MOV c
PRINT START_PRINT
PUSH c
PRINT STOP_PRINT
PUSH i
PUSH 1
ADD
MOV i
LOOP_END_0
METHOD_END_factorialIterative
STORE a
PUSH 10
MOV a
CALL factorialIterative
MOV a
CALL_END_factorialIterative
```

Sample Result Code