Welcome To

My Presentation

On Building

Abstract Syntax Tree

AST

Submitted By-

Mosamma Sultana Trina

Roll : BSSE 1313
Session : 2020-21

Supervised By-

Dr. Sumon Ahmed

Associate Professor
Institute of Information Technology

What is Abstract Syntax Tree?

- An Abstract Syntax Tree (AST) is a data structure that represents the syntactic structure of a program.

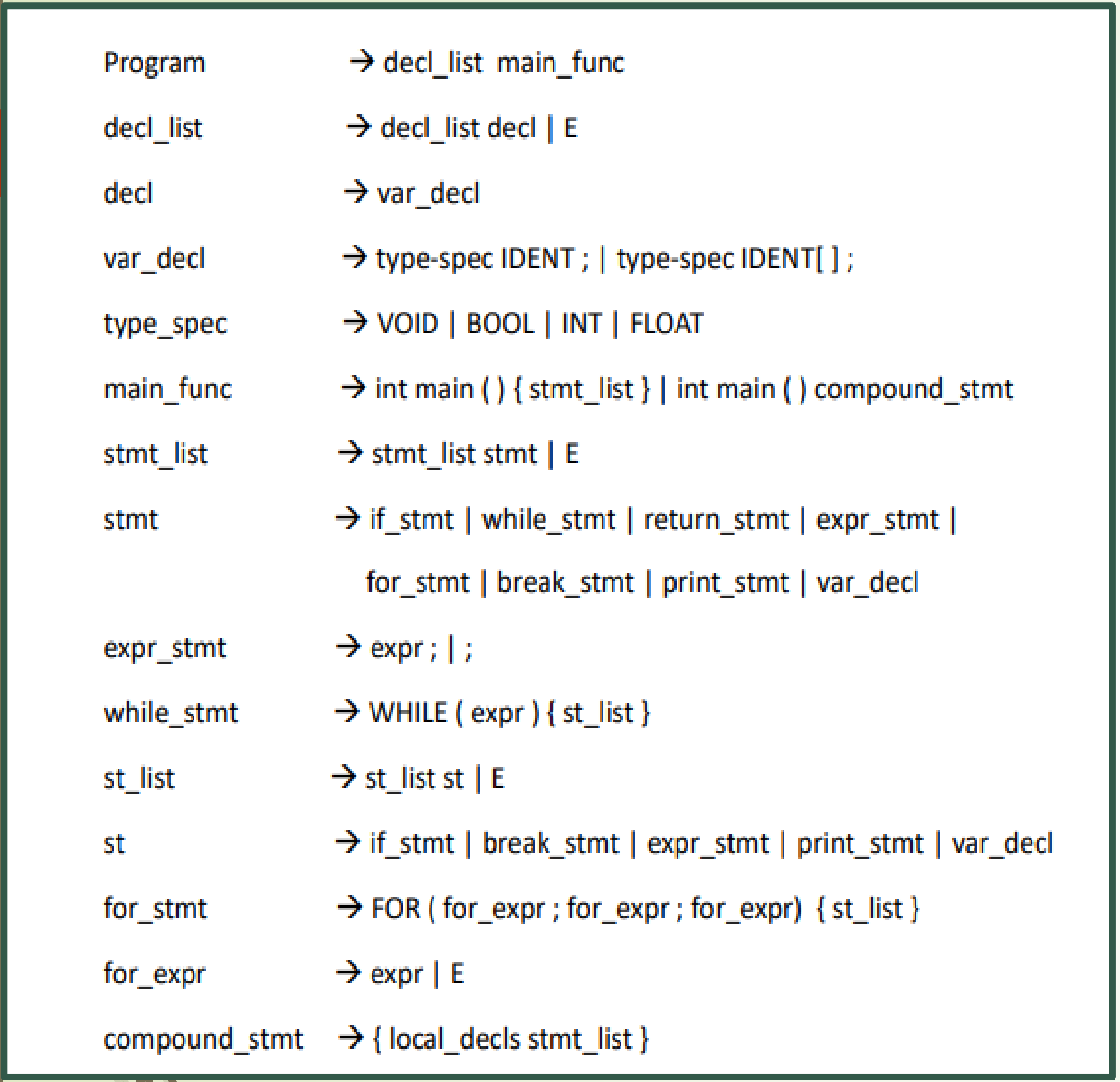
- To put it simply, an AST is like a blueprint of a program's syntax. It provides a more structured and organized way of representing the code than the raw source code itself.

About My Project :

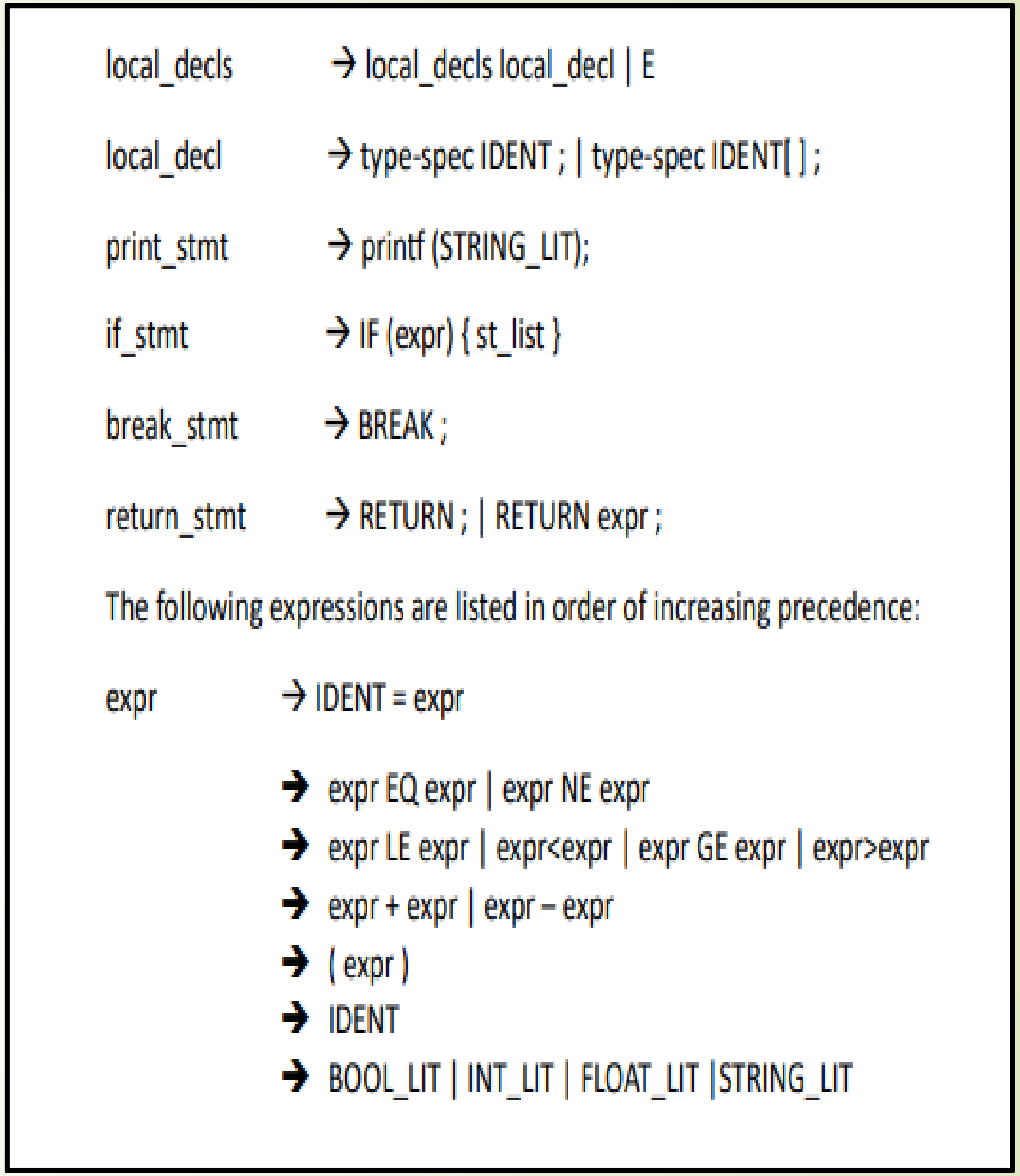
My Project is to build Abstract Syntax Tree for C source code.

Steps:

1. Lexical Analysis
2. Parse Tree
3. Abstract Syntax Tree



```
Program          → decl_list  main_func
decl_list        → decl_list decl | E
decl             → var_decl
var_decl         → type-spec IDENT ; | type-spec IDENT[ ] ;
type_spec        → VOID | BOOL | INT | FLOAT
main_func        → int main ( ) { stmt_list } | int main ( ) compound_stmt
stmt_list        → stmt_list stmt | E
stmt             → if_stmt | while_stmt | return_stmt | expr_stmt |
                   for_stmt | break_stmt | print_stmt | var_decl
expr_stmt        → expr ; | ;
while_stmt       → WHILE ( expr ) { st_list }
st_list          → st_list st | E
st               → if_stmt | break_stmt | expr_stmt | print_stmt | var_decl
for_stmt         → FOR ( for_expr ; for_expr ; for_expr)  { st_list }
for_expr         → expr | E
compound_stmt    → { local_decls stmt_list }
```



```
local_decls      → local_decls local_decl | E
local_decl       → type-spec IDENT ; | type-spec IDENT[ ] ;
print_stmt       → printf (STRING_LIT);
if_stmt          → IF (expr) { st_list }
break_stmt       → BREAK ;
return_stmt      → RETURN ; | RETURN expr ;
```

The following expressions are listed in order of increasing precedence:

```
expr         → IDENT = expr
             →  expr EQ expr | expr NE expr
             →  expr LE expr | expr<expr | expr GE expr | expr>expr
             →  expr + expr | expr – expr
             →  ( expr )
             →  IDENT
             →  BOOL_LIT | INT_LIT | FLOAT_LIT |STRING_LIT
```

Data Structures:

1. Array of Structure
2. Linked List

Algorithms:

1. LL (Left-to-Right, Leftmost derivation)
2. Depth first

Techniques:

1. Top-down parsing
2. Tree Representation : Left-Child Right-Sibling

Lines of code:

1.	Lexical Analysis	--	589
2.	Parse Tree	--	1702
3.	Abstract Syntax Tree	--	187
4.	Main	--	23

Total = 2501 lines

Challenges:

1. Handling code with more than 2000 lines is difficult for me
2. Grammar Handling
3. Parsing Complexity
4. Handling Nested "if" Statements

Thank you