SPL-1 Midterm Presentation, 2020

**Project Name** : Generating Abstract Syntax Tree

Course: Software Project Lab I

Course No: SE 305

Submitted by  
M. Mahtab Hossain JhalokBSSE Roll No. : 1103BSSE Session: 2018-2019

Supervised by  
 Abdus SatterDesignation: LecturerInstitute of Information TechnologyUniversity of Dhaka

**Table of Contents**

**1.Introduction …................................................................................4**

**1.1.Background study .........................................................................5-7**

**1.2.Challenges .................................................................................... 7-8**

**2.Project Overview ................................................................................ 8**

**3.User Manual .................................................................................10-15**

**4.Conclusion ........................................................................................ 16**

**5.References ......................................................................... ................16**

**Appendix............................................................................................... 16**

**1.Introduction**

Abstract syntax tree is a representation of tokens generated from statements and expressions in a programming language. With the AST, the interpreter or the compiler can generate machine code or evaluate an instruction.

The goal of my project is to read a c program and show the properties of classes and functions of that program such as variables , parameters , conditions , loop , return type etc .

1.1. Background study

# Background Study

# Lexical analysis

Lexical analysis, lexing or tokenization is the process of converting a sequence of characters (such as in a computer program or web page) into a sequence of tokens (strings with an assigned and thus identified meaning). A program that performs lexical analysis may be termed a *lexer*, *tokenizer*, or *scanner*, though *scanner* is also a term for the first stage of a lexer. A lexer is generally combined with a parser which together analyze the syntax of programming . A *lexeme* is a sequence of characters in the source program that matches the pattern for a token and is identified by the lexical analyzer as an instance of that token.

Common token names are

* identifier: names the programmer chooses;
* keyword: names already in the programming language;
* separator (also known as punctuators): punctuation characters and paired-delimiters;
* operator: symbols that operate on arguments and produce results;
* literal: numeric, logical, textual, reference literals

**Syntax**

In programming, syntax refers to the rules that specify the correct combined sequence of symbols that can be used to form a correctly structured program using a given programming language. Programmers communicate with computers through the correctly structured syntax, semantics and grammar of a programming language.

**Semantics**

Semantics is a linguistic concept separate from the concept of syntax, which is also often related to attributes of computer programming languages. The idea of semantics is that the linguistic representations or symbols support logical outcomes, as a set of words and phrases signify ideas to both humans and machines.

**1.2 Challenges**

Implementing a new software solution carries with it a number of challenges. The process can be overwhelming, confusing and lenthy. Implementing this project there are lot of challenges that I have faced. Some of them are:-

▪Handling large code for the first time  
▪Learning and understanding algorithm  
▪ Parsing a program

▪ Implementing Abstract Syntax Tree

**2. Plan of my project**

•Create a code to find functions of a c program

•Create a code to find parameters and variables which is used in a function

• Create a code to find conditions , loop of a program

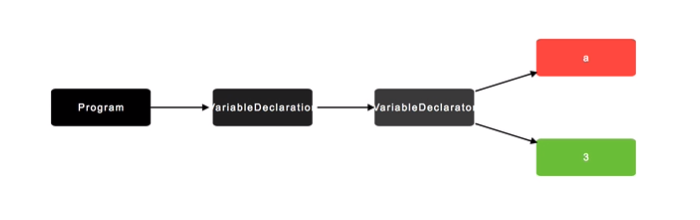
•Create a code to find those functions which are called from a function

• Merge all code and use abstract syntax tree algorithm to make abstract syntax tree code

3. A view of Abstract syntax tree

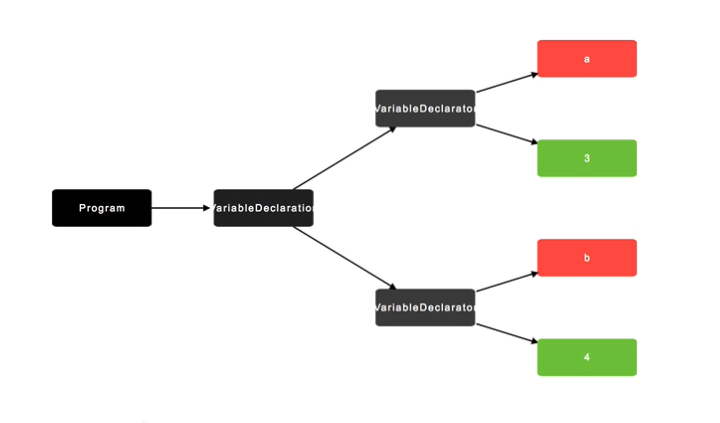
A variable declaration : int a = 3;

The AST of this declaration is :



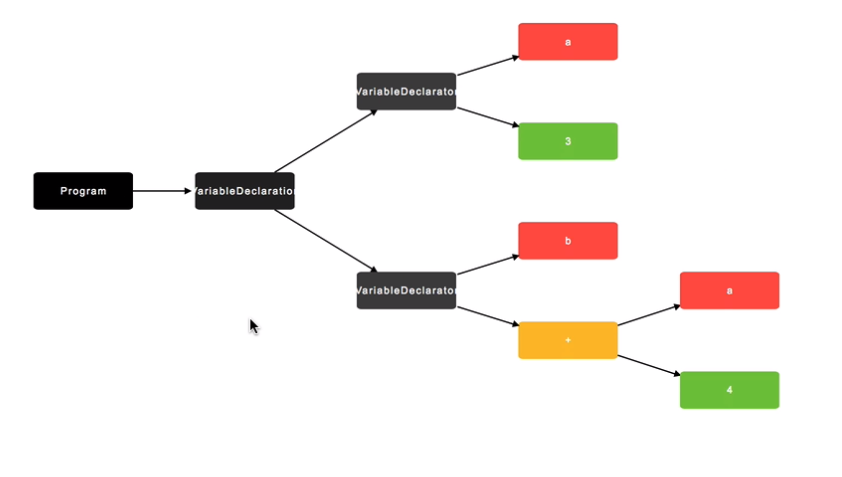
If we write : int a = 3 , b = 4 ;

Ast will be :



If we write : int a = 3 , b = a + 4;

Ast will be :



An AST of a program :



**4.Conclusion**

I think implementing this project will help me to improve my coding skill and I will learn how to handle large codes. I hope it will help me to deal with difficulties in future.

5. References

<https://egghead.io/lessons/javascript-introduction-to-abstract-syntax-trees?pl=abstract-syntax-trees-3582efe5>

<https://en.wikipedia.org/wiki/Lexical_analysis>

<https://www.techopedia.com/definition/3959/syntax>

<https://en.wikipedia.org/wiki/Abstract_syntax_tree>

**Appendix**In this project, I have implemented auto-commenting in a source code for first year student’s code. In future, I want to implement this project for a large code.