**SPL-1 Final Presentation, 2020**

**Project Name : Generating Abstract Syntax Tree**

**Course: Software Project Lab I**

**Course No: SE 305**

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

Supervised by  
 ***Abdus Satter*Designation: *Lecturer*Institute of Information Technology  
University 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. They are -

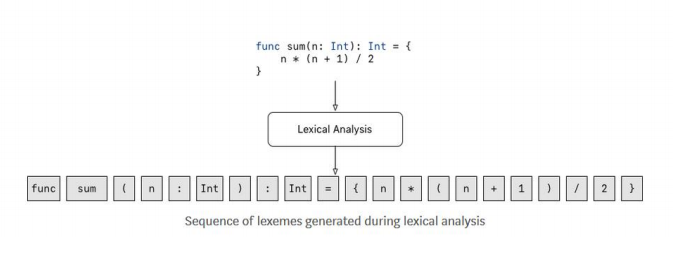
1. Functions : A program always have some functions to execute . A syntax tree must find those functions.
2. Return type : Finding the return data type of that function.
3. Parameter : A function may use some parameter for its work .Show its parameter .
4. Loops : A program uses many types of loop for doing its job easily . This program will also detect those .
5. Variable : Finding all the variables of a program .
6. A function calling another function : Show the functions properties which is called by another function .

1.1. Background study

# Background Study

# Lexical analysis

The first phase of making this tree is Lexical analysis, lexing or tokenization . It 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.

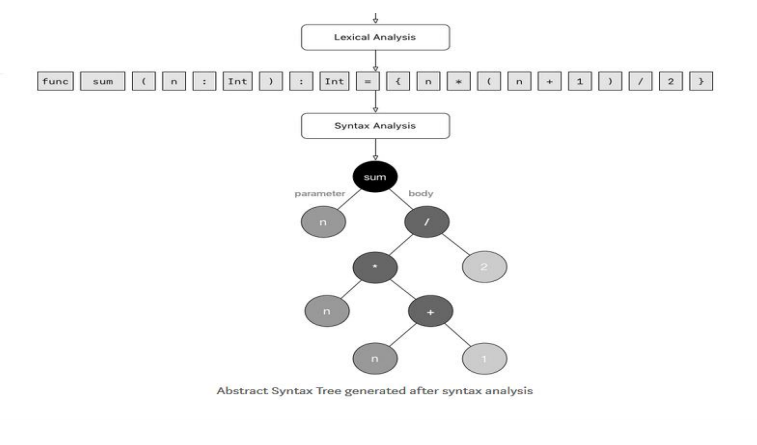


Common token names are

* 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 Analysis**

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:-

1. Handling large code for the first time  
2. Learning and understanding algorithm  
3. Parsing a program

4. Implementing Abstract Syntax Tree

5. Parsing character by character was a very labourios job

6. Dividing the entire source code into tokens

7. Lack of resources in the internet gave a lot of difficulties.

8. Representing the code into an abstract Syntax Tree was very confusing and it took a lot of time.

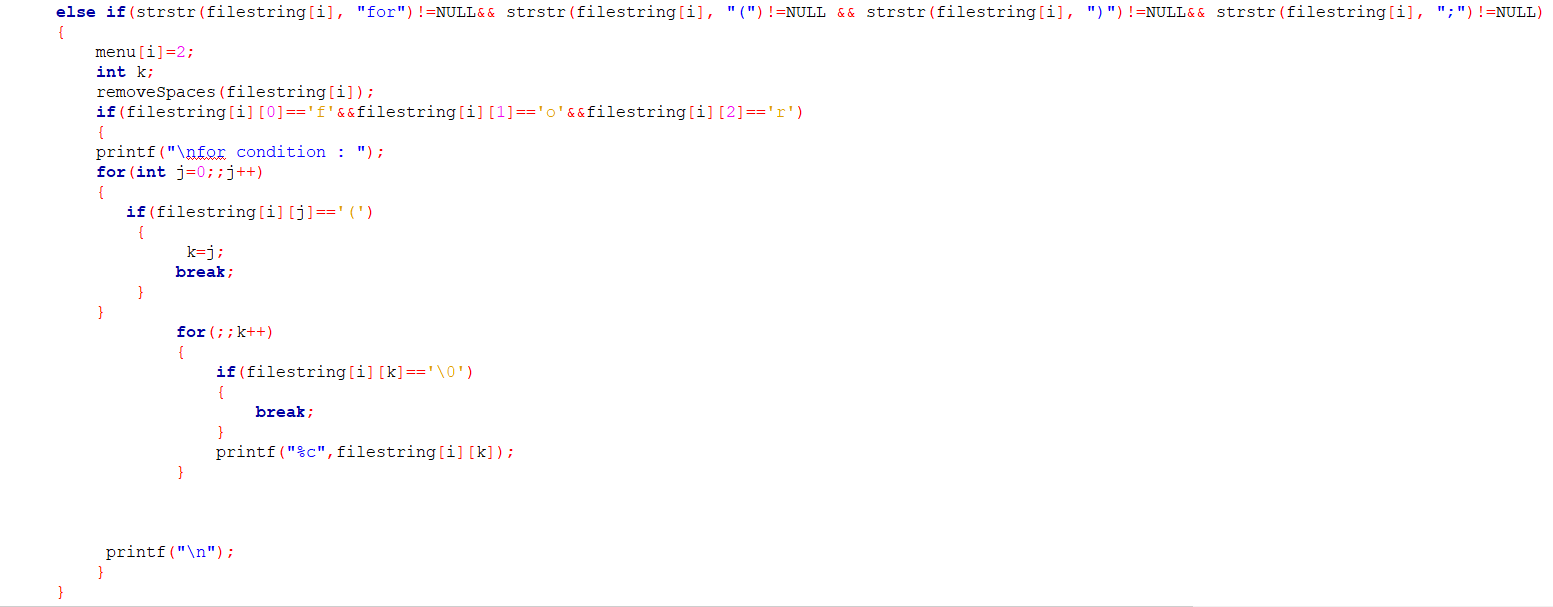
**2. Project Overview**

The project has mainly two steps :

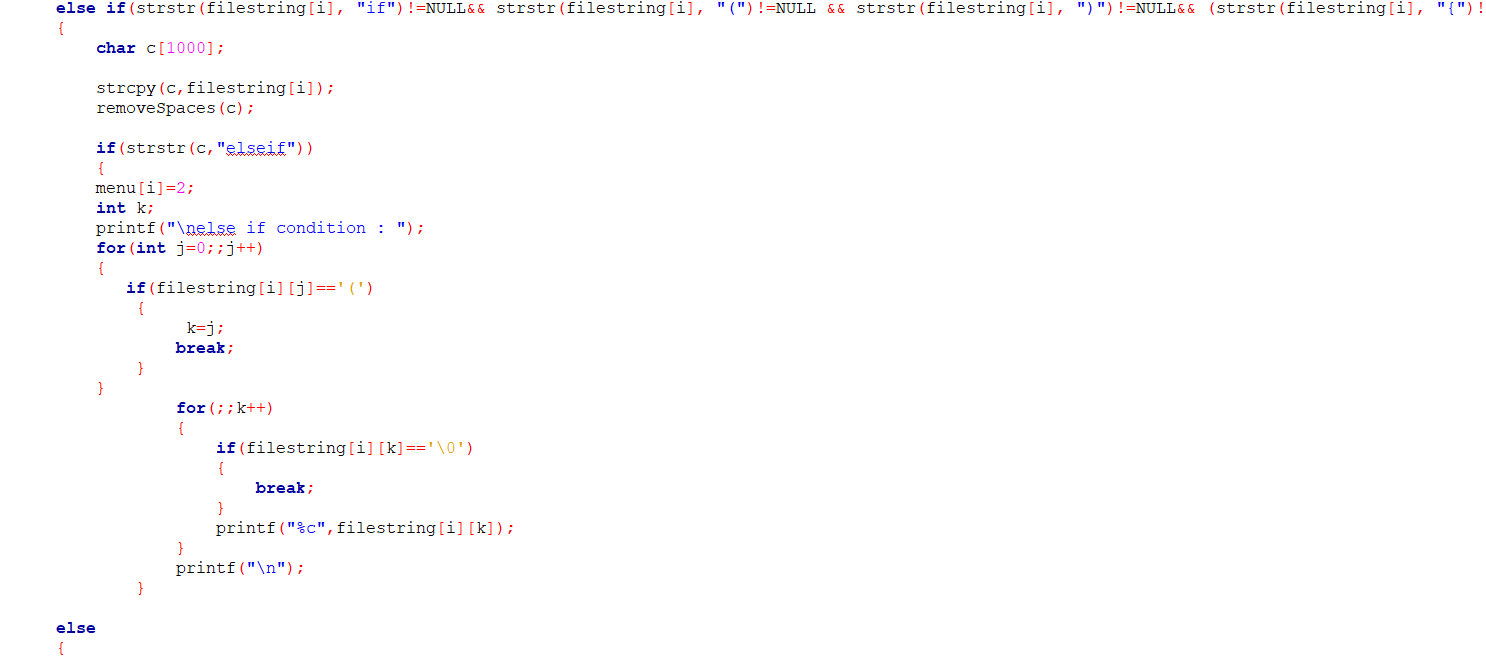
1. Lexical Analysis
2. Systactical Analysis

Lexical Analysis :

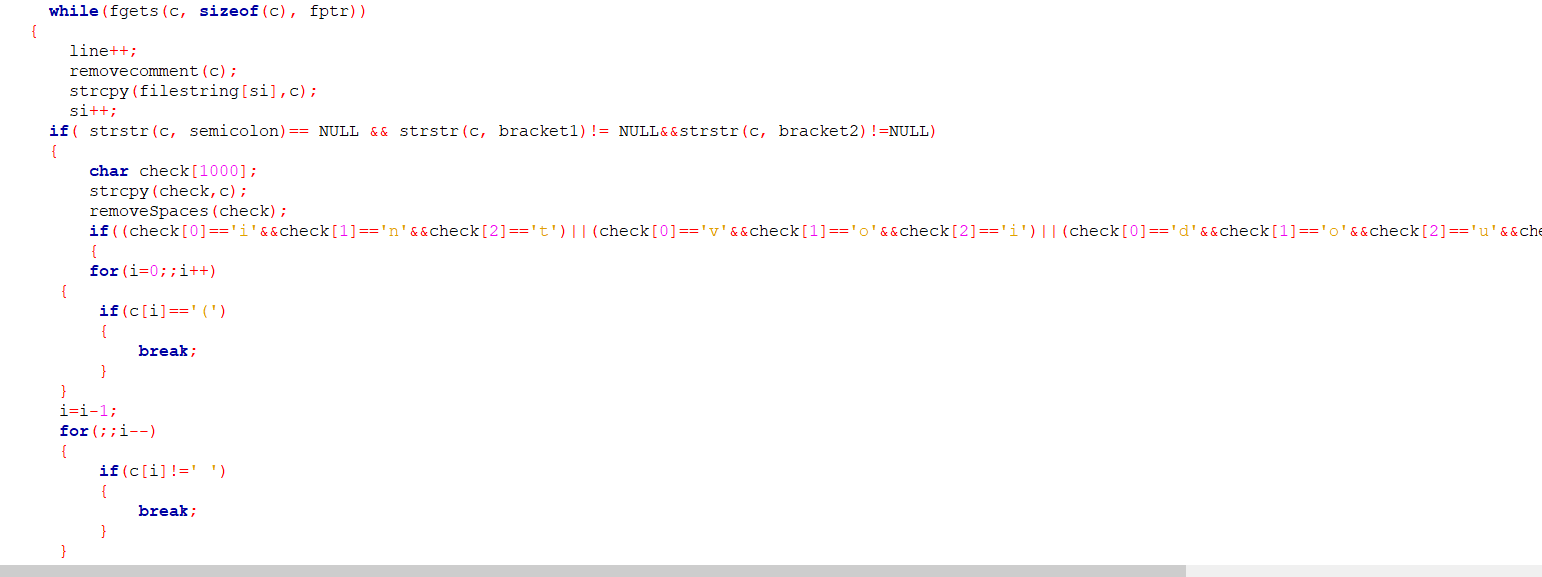
The source code is first converted into tokens so that the syntactical analysis can be done on the Lexed Code. Several Steps were taken during these time. Some of the steps are illustrated below:



Here, we can see for loops are being lexed from the source code and bracket semicolon and endline are being used for the cause



Here, we have detected if condition and converted them into tokens for further use.



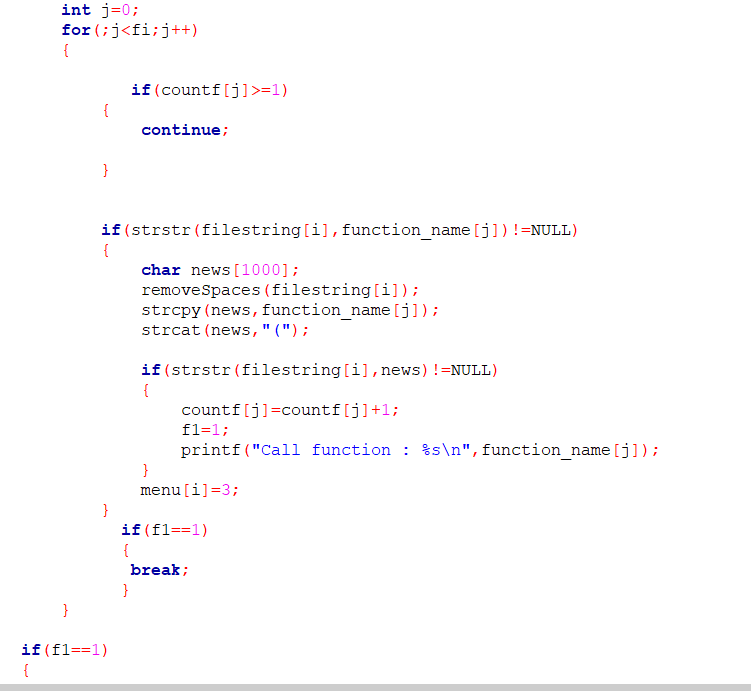
Here, we have detected functions from a program .

Next let us show about the syntactical analysis of the lexed code.

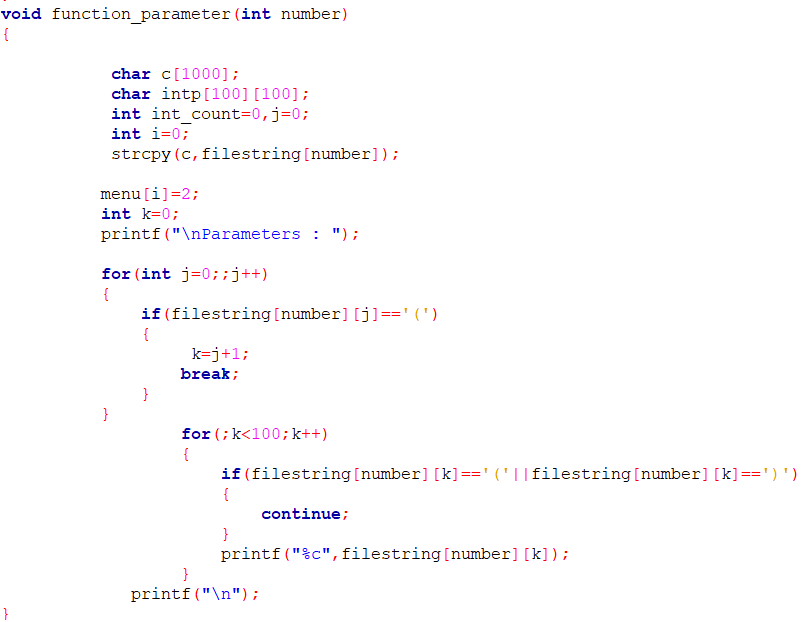
Syntactical Analysis :

The lexed code will be analysed syntactically and an abstract syntax tree will be formed with the tokens. Any irregularities with the Context Free Grammar will be determined from here.

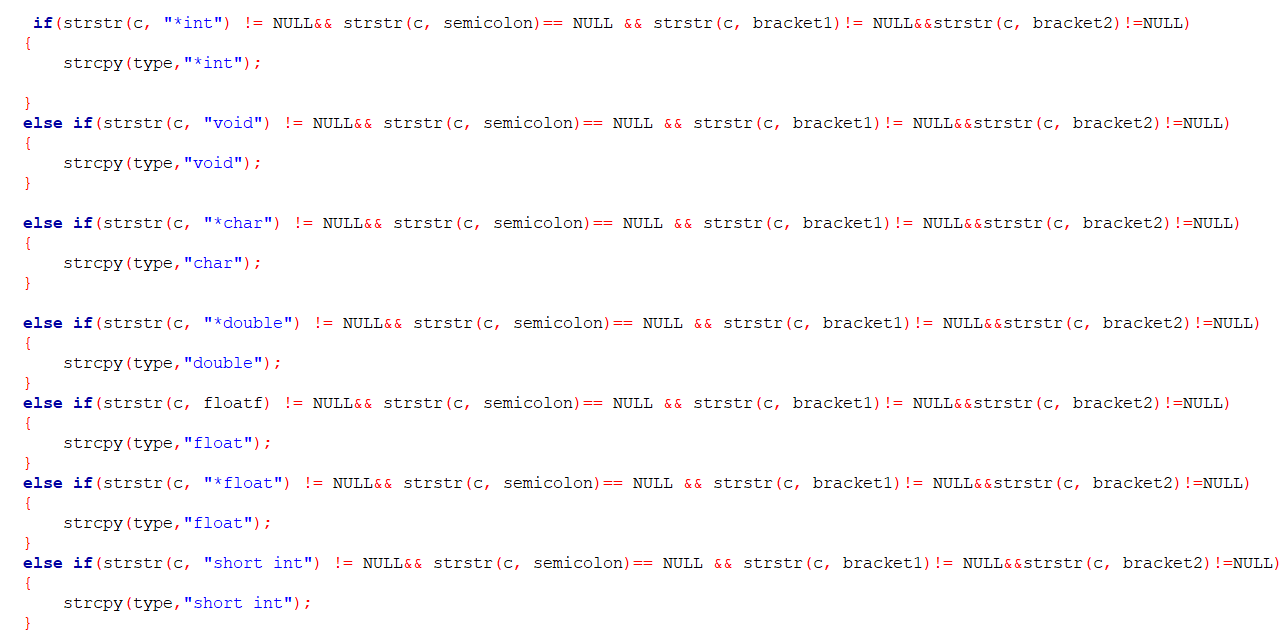
Some significant functions of the code is shown below and explained briefly.



Here we have detect the function name which is called from a function. We use some condition and loops for this job.



By this code we show the parameters of a function .



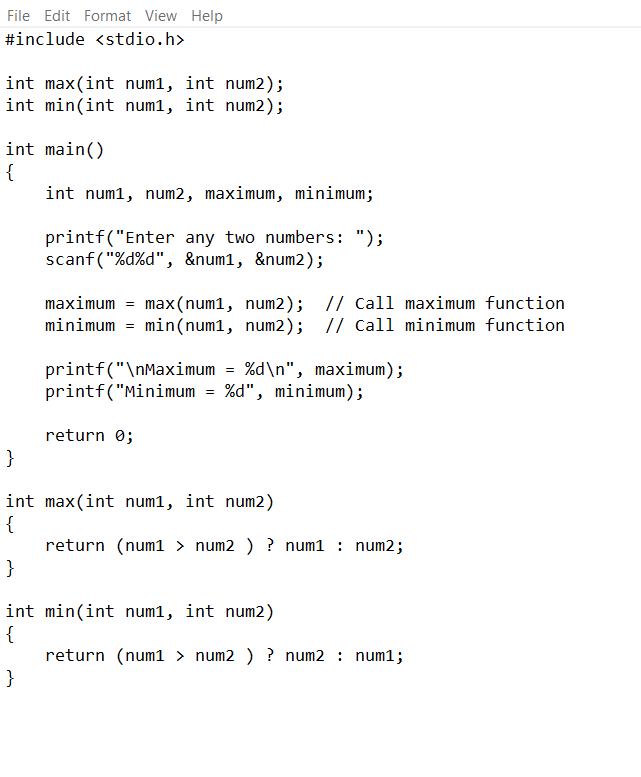
By this we detect the return type of a function .

**User Manual**

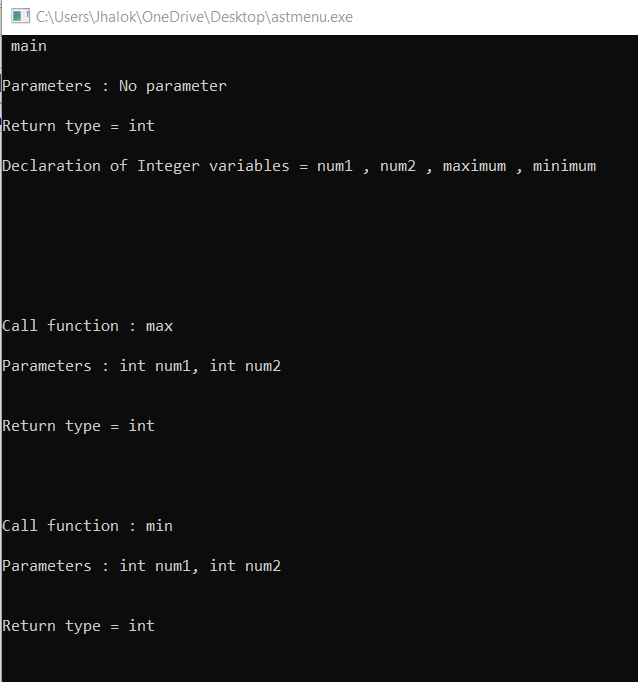
Input will be taken form a file .

Example 1 :

We take a c program which show the maximum and minimum from 2 numbers.

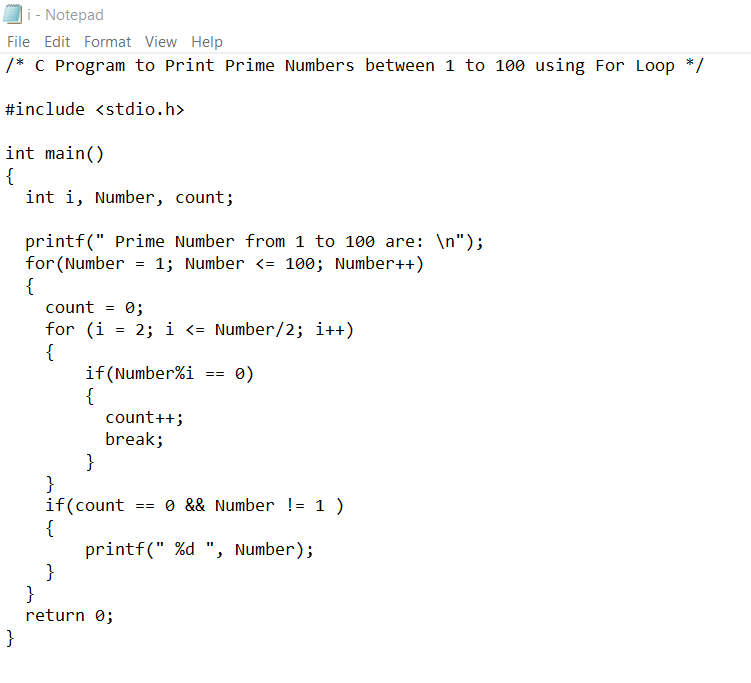


output will be :

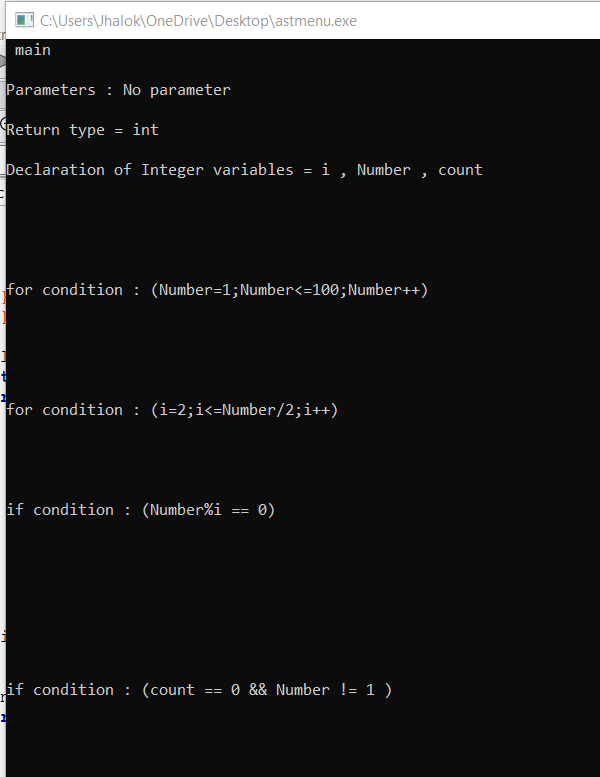


Example 2 :

We take a c program . It prints all the prime numbers from 1 to 100.



Output :



**Conclusion**

The project helped me understand how the compiler works and it has helped me so much in a number of ways. It taught me how to be more manageable while coding and how to handle large lines of code. . This project was quiet challenging as I didn ’t have any knowledge about computer compilers before doing this project and I gained a lot of experience from it. Hope this knowledge will help me in my future compiler and parsing related projects. I want to thank my supervisor for guiding me a lot during this project.

**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**

I could complete upto Syntactical Analysis. I have plans for this project in the future. I would like to generate the object code from the source code and also detect errors from the source code