North Western University

Compiler Design

(User manual)

Course Code: CSE-4104

Developed By:

Chandan Bhowmick

20201102010

Kulsum Ara Eti

20201103010

Suraiya Bithi

20201116010

Mahmuda Rahman Mou

20201080010

Department: CSE

4th year, 1st semester

Contents

Abstract:

The purpose of this lab project was to design and implement a lexical analyzer, also known as a lexer, for a programming language. The lexer is an essential component of a compiler or interpreter, responsible for breaking down the source code into tokens that can be processed further. This report discusses the design choices, implementation details, and showing symbol table of this project.

Introduction:

The lexical analyzer is the first phase of a compiler. It takes modified source code from language preprocessors that are written in the form of sentences. The lexical analyzer breaks these syntaxes into a series of tokens, by removing any white space or comments in the source code. If the lexical analyzer finds a token invalid, it generates an error. The lexical anayer works closely with the syntax analyzer.

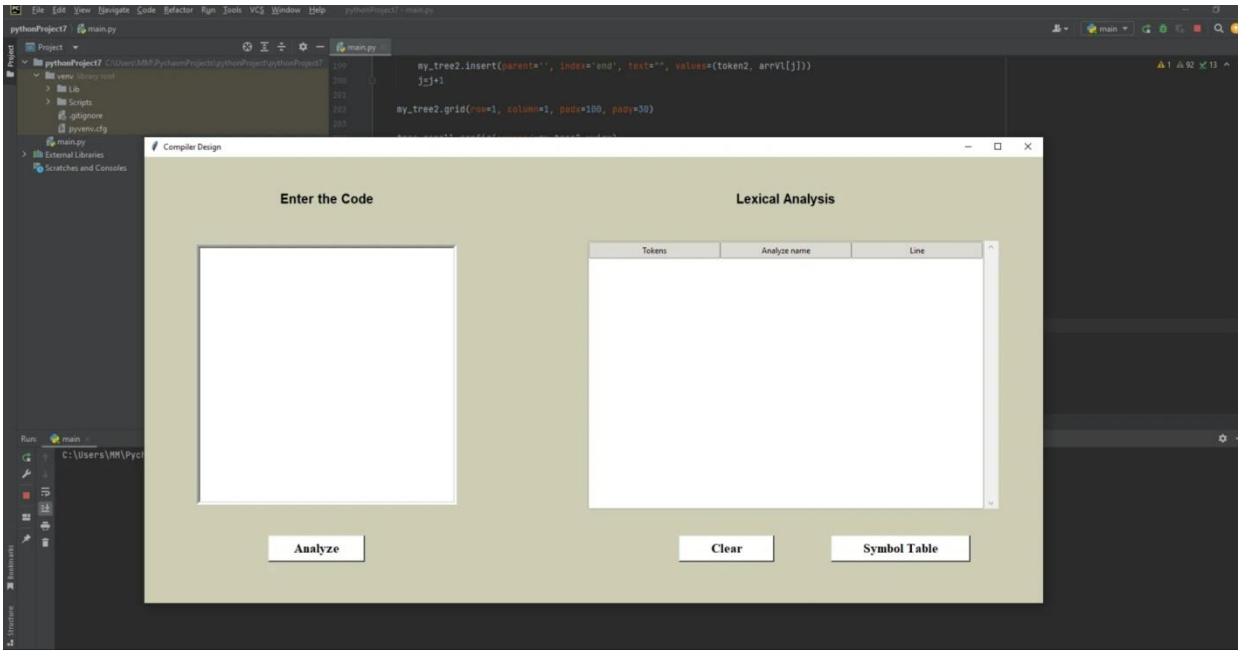
Objectives:

The objective of this project is to create a program that can analyze the source code of a programming language and break it down into individual tokens. The program should accurately recognize and categorize tokens such as keywords, identifiers, literals, operators, and punctuation symbols. It should also handle errors and provide informative error messages. The project aims to create an efficient and reliable tool that can integrate with other components of a compiler or interpreter for further analysis and processing of the source code.

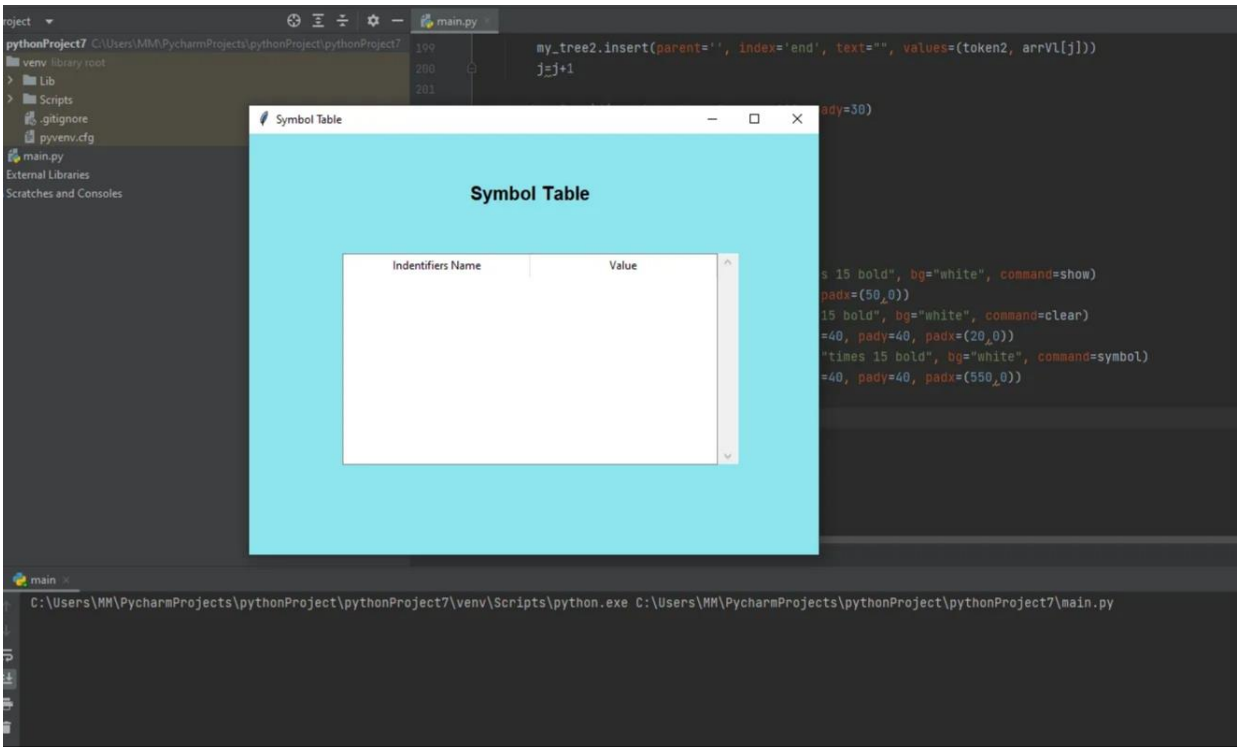
Design And Implementation:

The lexer was implemented using the Python programming language and special use of python tkinter library. The design involved creating a set of c language code to match the different token types and then applying them to the source code in a sequential manner. The lexer followed the longest match rule, where it selects the longest matching token from the input stream.

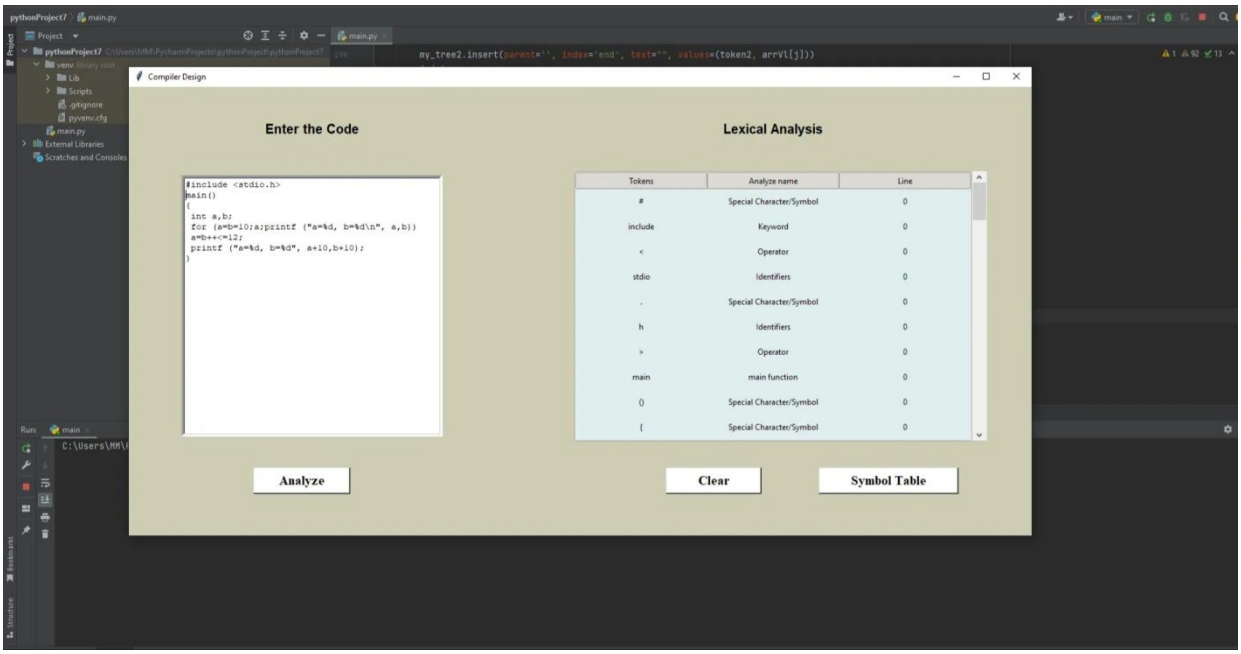
Firstly created the structure enter the code & lexical analysis

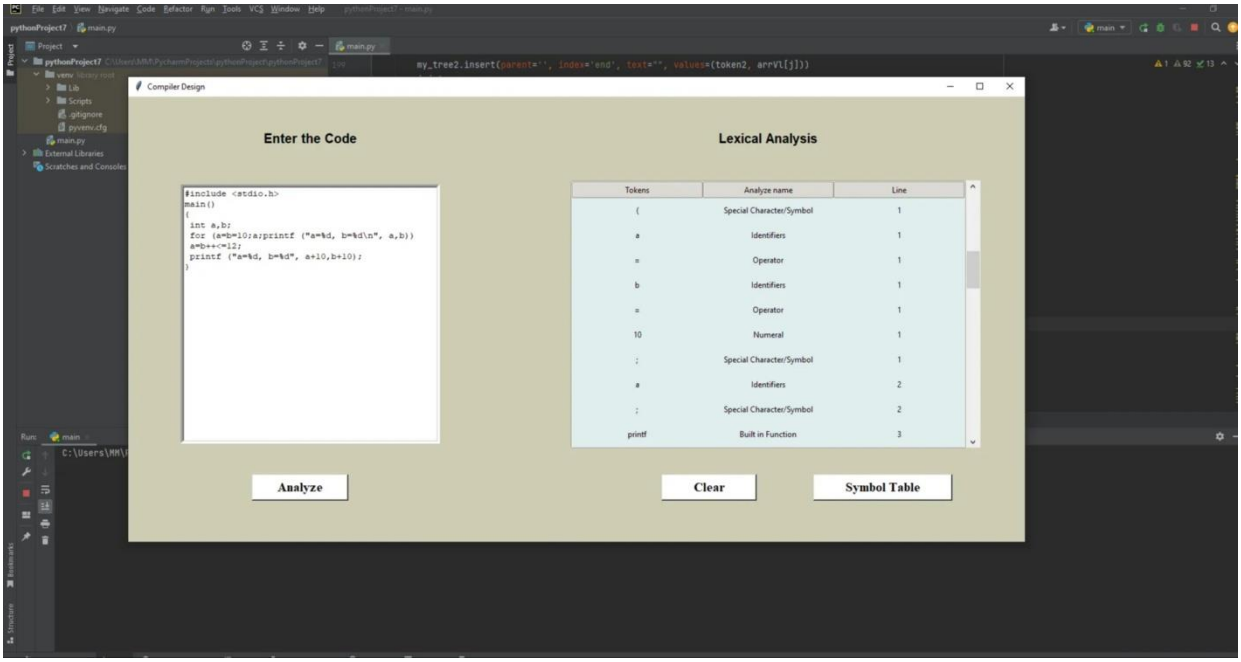


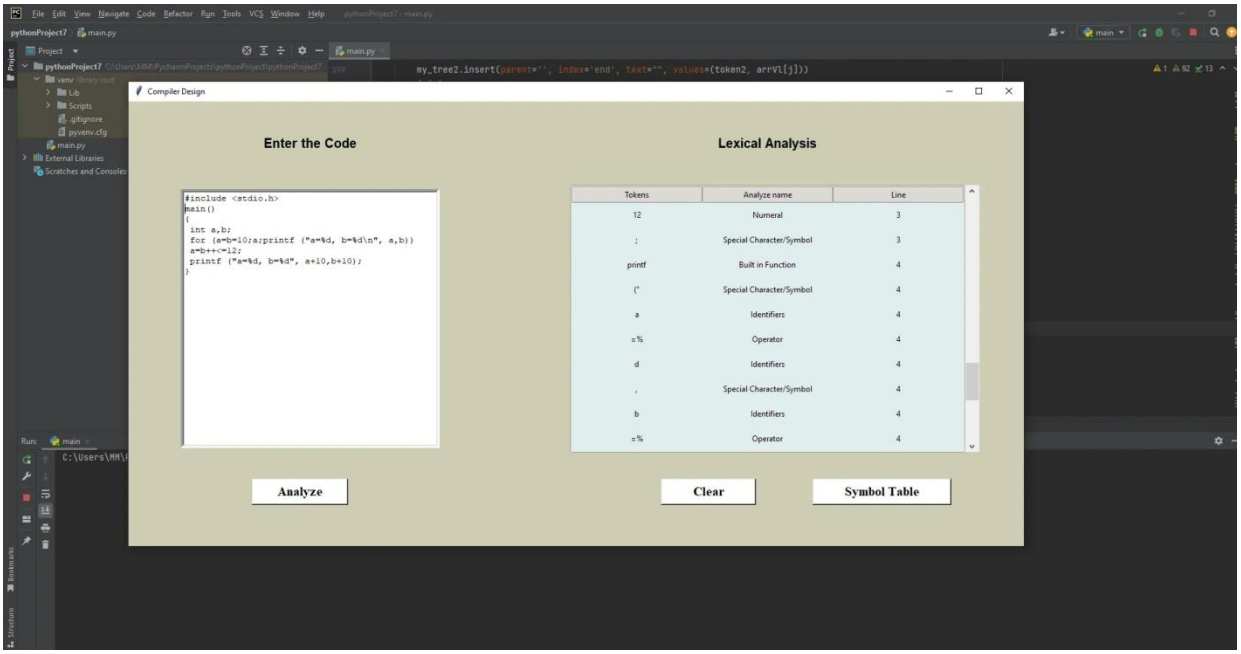
This picture shown in symbol table:



Then this picture shown in a code and lexical analysis:





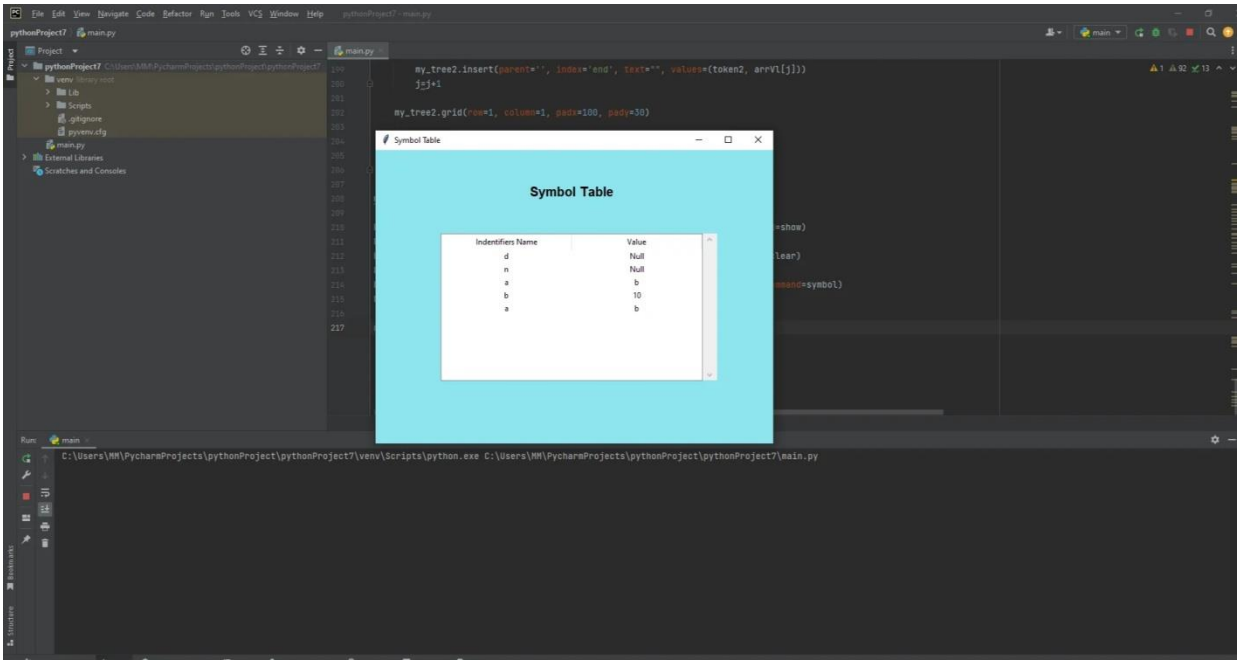


The lexer iterated over the input buffer character by character, recognizing and categorizing each token it encountered. Source code were utilized to define the patterns for various tokens, such as keywords, identifiers, number, strings symbols etc. Whole process handled after clicking analyze button. And there is clear tokenize button for clear the tokenize table. The lexer maintained a symbol table to store and manage identifiers encountered during tokenization.

Symbol Table:

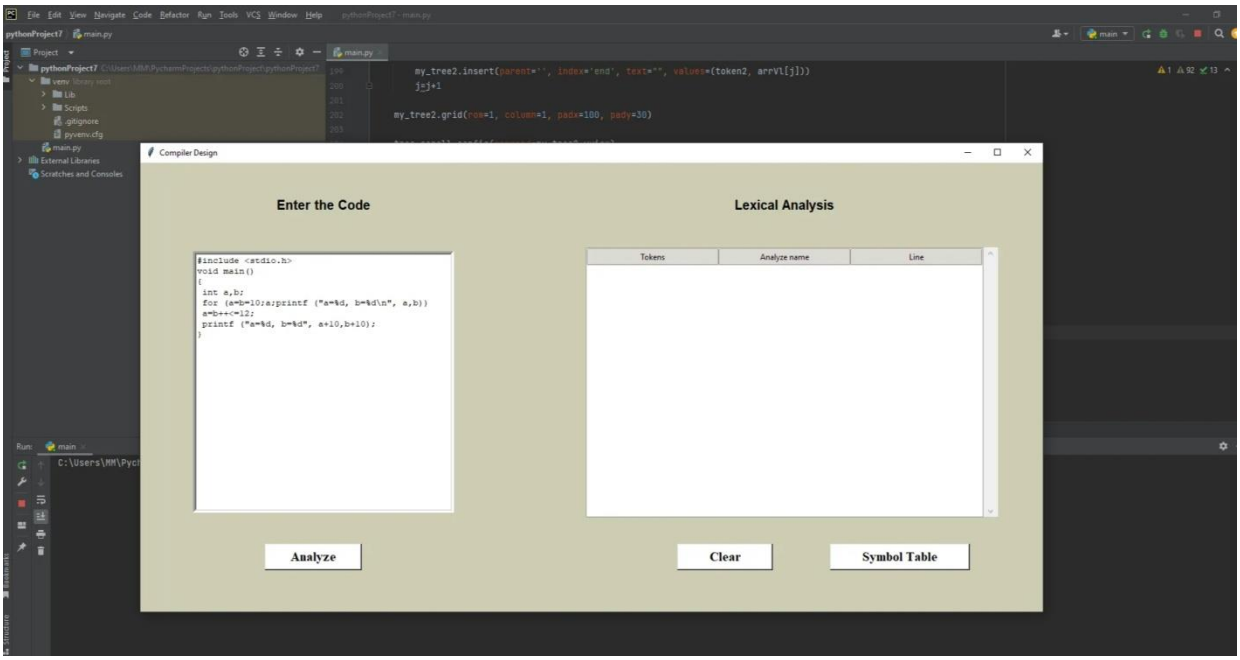
Symbol table is an important data structure created and maintained by compilers in order to store information about the occurrence of various entities such as variable names, function names, objects, classes, interfaces, etc. Symbol table is used by both the analysis and the synthesis parts of a compiler.

Then this picture shown in symbol table Indentifiers name & value.



Symbol is used for hold the identifiers of the source code. So in this symbol table hold the identifiers and the value of each identifiers.

Finally enter the code analyze



Conclusion:

In this project successfully implemented a lexer for a programming language. The design and implementation of the lexer allowed for accurate and efficient token recognition. The project achieved its objective of breaking down the source code into meaningful tokens, laying the foundation for subsequent phases of the compiler.

Thanks to

Md. Shymon Islam

Lecturer

Department Of CSE

North Western University Khulna