North Western University

Compiler Design

(User manual)

Course Code: CSE-4104

Developed By:

Md.Reazul Islam Rifat

ID: 20201139010

Mohua Roy

ID: 20201100010

Md. Sayem Mollik

ID: 20201094010

Department Of CSE

North Western University, Khulna.

Contents: Page No

Abstract:

The aim of this project was the design and implementation of a lexer for programming language that is commonly referred to as a lexical analyzer. The parser is a necessary component of the compiler and interpreter for breaking down source code into tokens that can be processed on their own. This report sets out the design and implementation of this project in detail, including its symbol table.

Introduction:

The lexical analyzer is an initial step of a compiler or interpreter which, in order to perform subsequent analysis, will convert the source code into more manageable forms. The lexical analyzer's primary aim is to find and classify the linguistic units, or tokens, in a source code. The keywords, identifiers, numbers, operators, datatypes and punctuation symbols are included in these tokens.

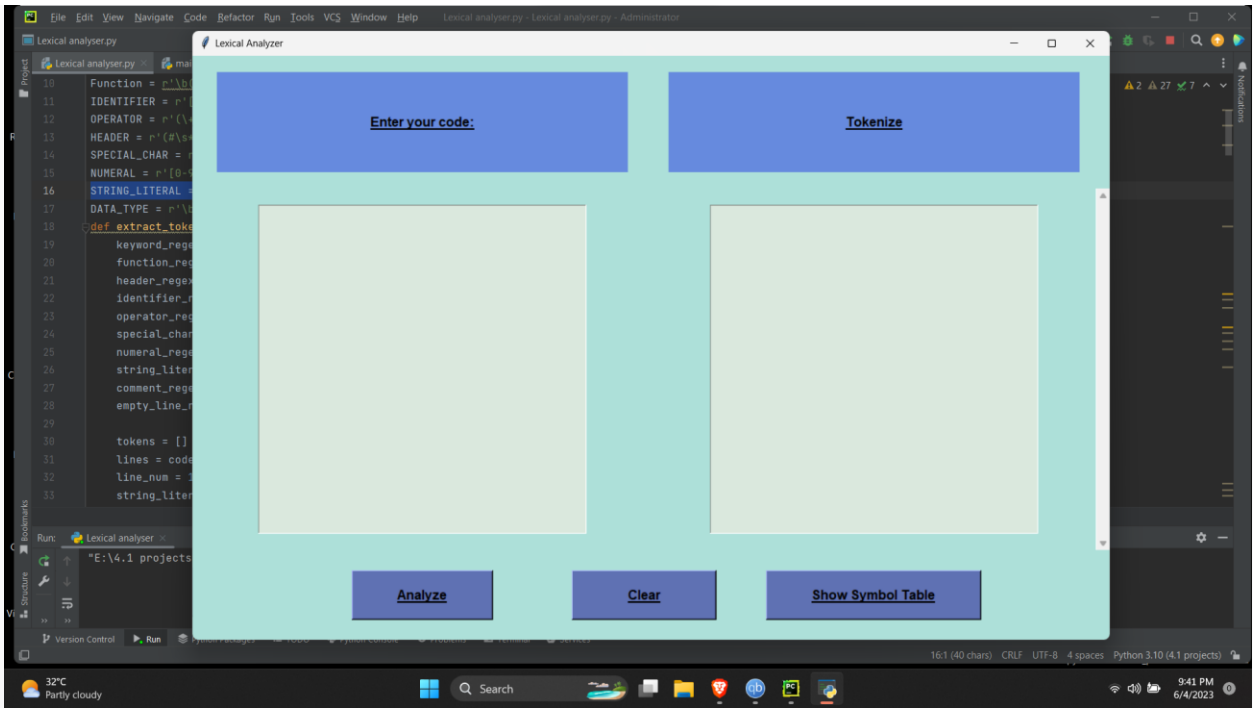
Objective:

The goal of this project is to create a program capable of analyzing the source code of a programming language and break it down into individual tokens. program should correctly identify and categorize tokens as keywords, identifiers, literals, operators and punctuation. It will also handle errors and provide Informative error message. The project aims to create an efficient and reliable tool to can be integrated with other

components of the compiler or interpreter for further analysis and source code processing.

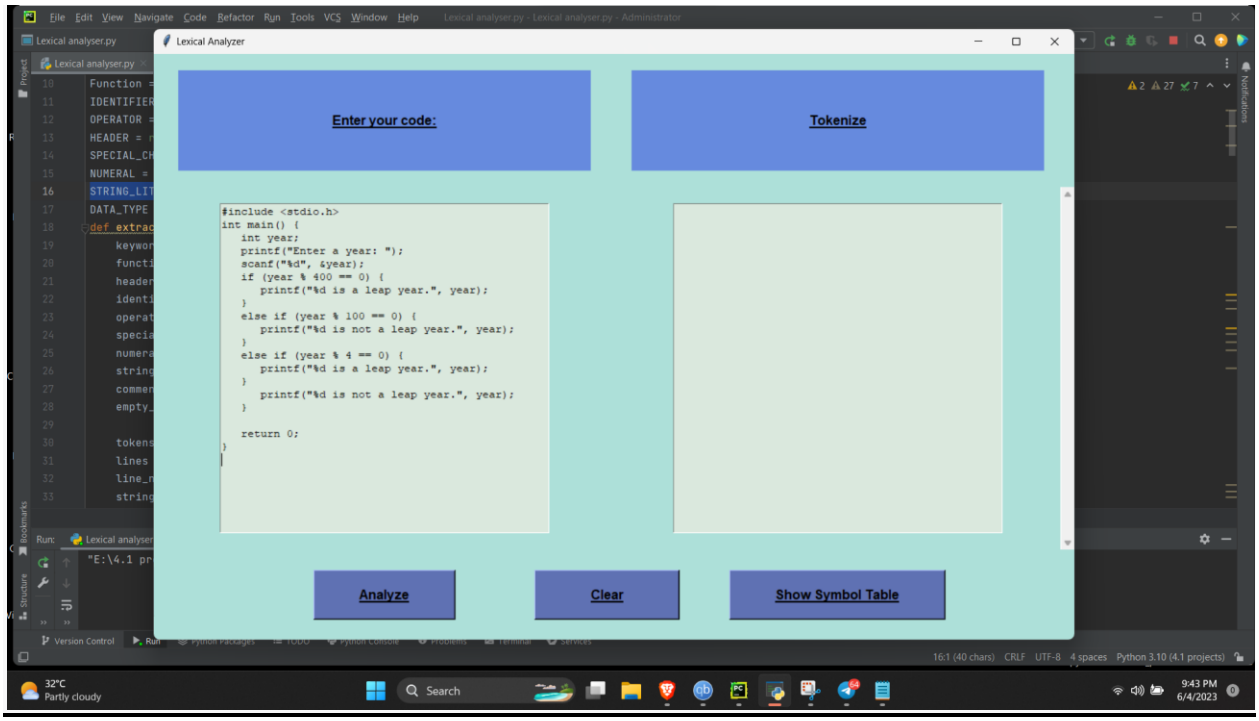
Design And Implementation:

The lexer has been implemented in the Python programming language with special use of the python tkinter library. To be able to match the various types of token, a set of c language code had to be created and then applied in sequential order to the source code. When selecting a longest matching token out of the input stream, the lexer follows the longest match rule. You'll find steps to input source code, view tokenization and display a symbol table.



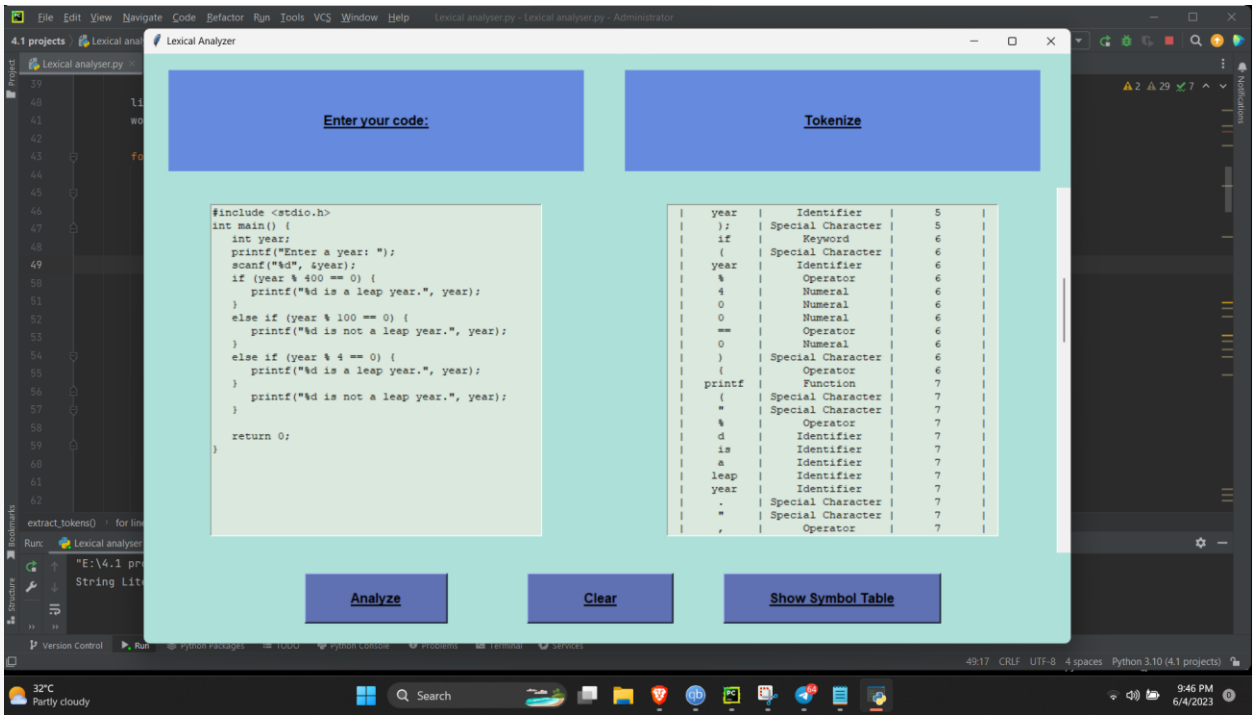
Interface of the project.

Input:



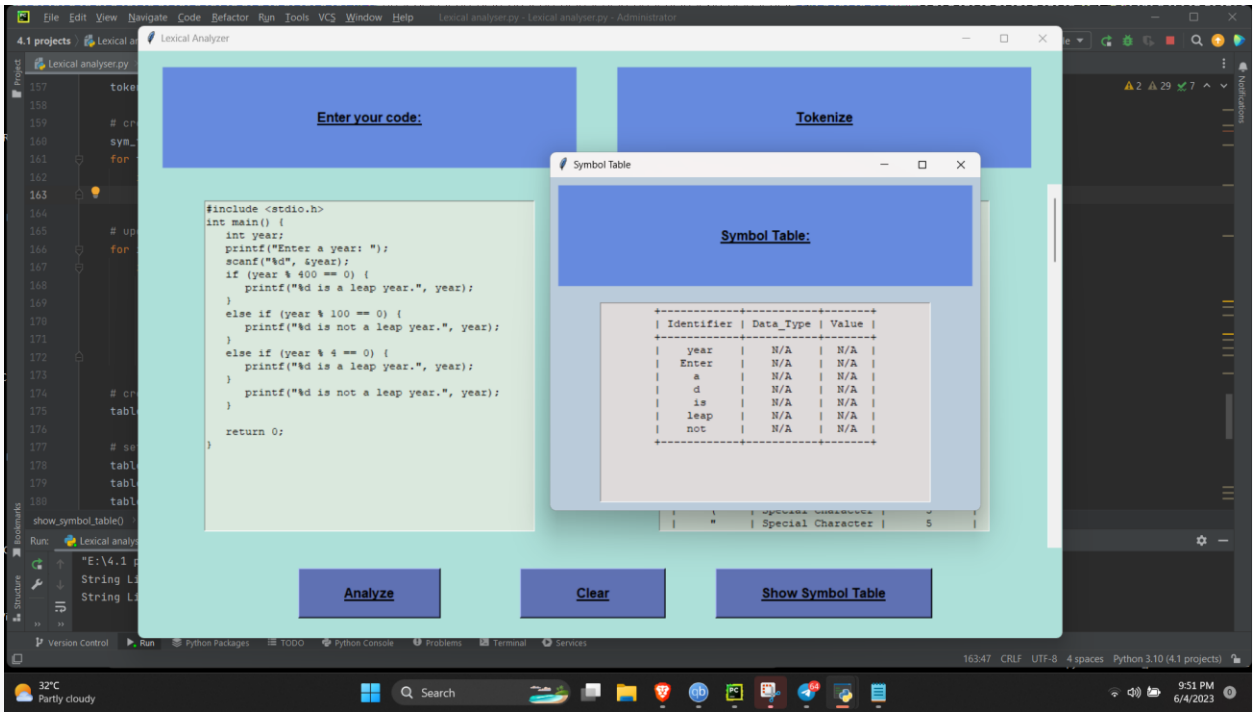
Enter your code field has been used to input source code. As source code, only the C language can be used. You can clear an input field by pressing the button.

Tokenization:



The lexer iterated over the input buffer character by character, recognizing and categorizing each token it encountered. The pattern for each of the various tokens, such as keywords, identifiers, numbers, strings symbols and so on, has been defined using source code. After clicking analyze, the whole process is handled. And for clearing the tokenization table, there is a simple Tokenizing button. In order to store and administer identifiers which have been encountered in tokenization, a table of symbols was kept by the lexer.

Symbol Table:



For holding source code identifier, a symbol is used. So in this symbol table hold the Identifiers, the value of each identifier.

Conclusion:

A lexer for the programming language has been successfully implemented in this project. The lexer was designed and implemented in such a way as to make it possible to recognise precisely and efficiently the symbol. In laying the basis for subsequent phases in compiler, this project achieved its goal of breaking source code into meaningful tokens.

Thanks To

Md. Shymon Islam

Lecturer

Department Of CSE, North Western University Khulna