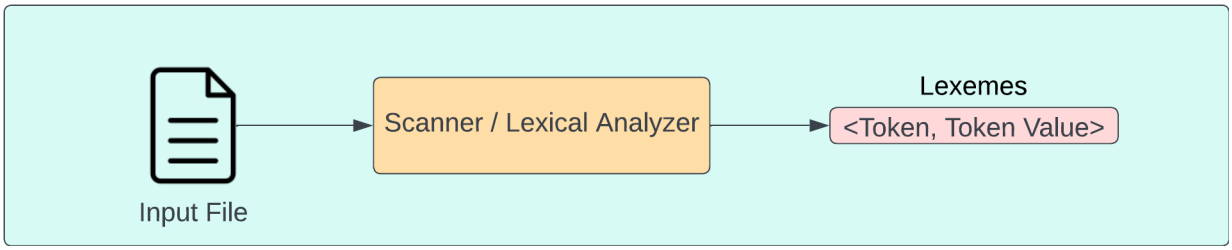
CS424
Compiler Construction

Assignment # 1
Report

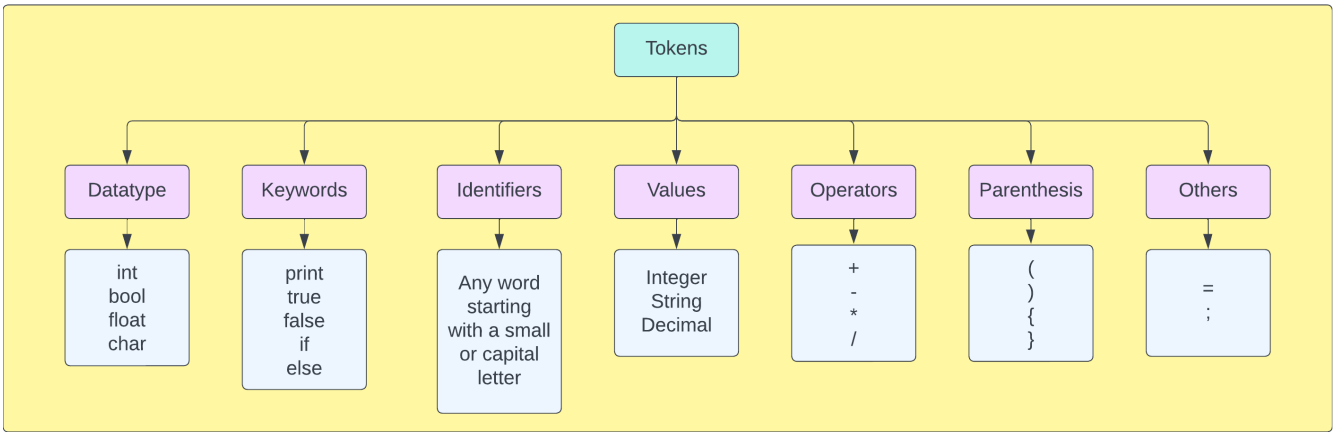
Name: Syed Muhammad Ashhar Shah
Reg No: 2020478

Scanner Design:



The scanner takes as input a source file that contains the contents that we want to read. The contents of the file are sent to our scanner also known as the lexical analyzer which then reads the contents of the file and breaks it down into lexemes. Each lexeme contains the token along with the detected tokens value.

Below is a graphical representations of the tokens that we have created for our scanner on which the contents of the source file will be mapped.



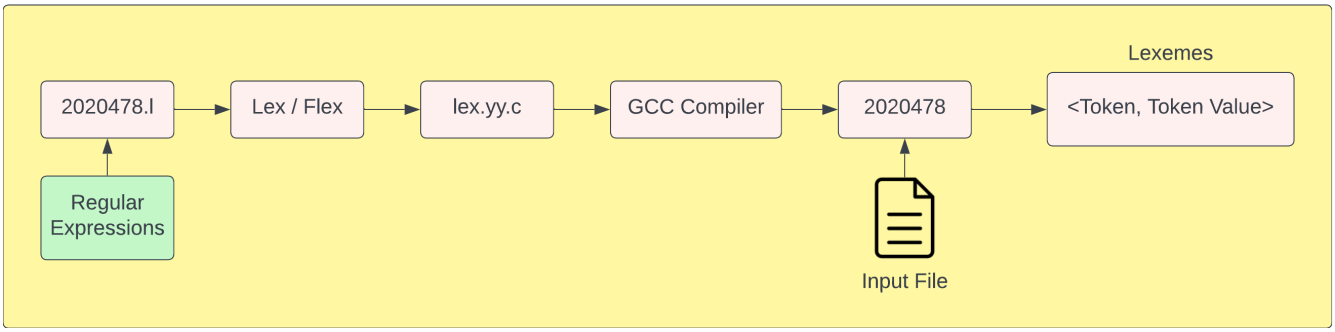
Our tokens will be categorized into 7 different types which are explained below:

1. Datatype: There are the reserved keywords that represent the type of a declared variable. It can be an int, bool, float, or char.
2. Keywords: These are the other reserved keywords that represent certain actions such as print,true,false,if, and else.
3. Identifiers: An identifier represents the name of the variable. For example, in the line int age = 3, age is the keyword.
4. Values: These represent the different values that can be provided to an identifier, in our program we only accept 3 types of values namely integers, decimals and strings.
5. Operators: These represent the different type of operators in our program that preform mathematical operations.
6. Parenthesis: We also have included tokens for parenthesis that are used to define the scope of a function or statement.
7. Others: We have also created two additional tokens, one for the assignment operator and another for the delimiter to identify the end of a piece of code.

Implementation Details

For the implementation we have created two files, one that uses the lex program and other that does not. In the program that does not use lex we simply open the input file in read only mode using the *open()* linux system call, read the contents of the file and store it in a character array. When all the conents of the file are read we then start tokenizing the input steam. We simple use if,else statements to check for the given token by performing comparision of the input stream alongside the reserved keywords using the *strcmp()* function.

In the file that uses the lex program. We first create a lex file with the .l extension that contains all our rules for the tokens which are defined are regular expression in the middle part of the file. We then use the *yylex()* function that performs lexical analysis on the input steam against the tokens defined by the regular expression. Below is a graphical representation of the process.



Testcase

Below I have attahed a sample test case for the MiniLang language in which I have created an integer variable and have checked the correctness of the value of the integer variable using a if-else statement block.

```
ashhar@ashhar-Lenovo-B590: ~/Desktop/Semester 8/Compiler Construction/Assignments/Assignment1/src/With_Lex
ashhar@ashhar-Lenovo-B590:~/Desktop/Semester 8/Compiler Construction/Assignments/Assignment1/src/With_Lex$ cat input.c
bool z = false;
if(z == true){
    print("z is true");
}
else{
    print("z is false");
}
ashhar@ashhar-Lenovo-B590:~/Desktop/Semester 8/Compiler Construction/Assignments/Assignment1/src/With_Lex$ make
lex 2020478.l
gcc lex.yy.c -o 2020478
./2020478 < input.c
TOKEN: Datatype, LEXEME: bool
TOKEN: IDENTIFIER, LEXEME: z
TOKEN: Assignment, LEXEME: =
TOKEN: Keyword, LEXEME: false
TOKEN: Delimiter, LEXEME: ;
TOKEN: Keyword, LEXEME: if
TOKEN: PARENTHESIS, LEXEME: (
TOKEN: IDENTIFIER, LEXEME: z
TOKEN: Assignment, LEXEME: =
TOKEN: Assignment, LEXEME: =
TOKEN: Keyword, LEXEME: true
TOKEN: PARENTHESIS, LEXEME: )
TOKEN: PARENTHESIS, LEXEME: {
TOKEN: Keyword, LEXEME: print
TOKEN: PARENTHESIS, LEXEME: (
TOKEN: STRING_VALUE, LEXEME: "z is true"
TOKEN: PARENTHESIS, LEXEME: )
TOKEN: Delimiter, LEXEME: ;
TOKEN: PARENTHESIS, LEXEME: }
TOKEN: Keyword, LEXEME: else
TOKEN: PARENTHESIS, LEXEME: {
TOKEN: Keyword, LEXEME: print
TOKEN: PARENTHESIS, LEXEME: (
TOKEN: STRING_VALUE, LEXEME: "z is false"
TOKEN: PARENTHESIS, LEXEME: )
TOKEN: Delimiter, LEXEME: ;
TOKEN: PARENTHESIS, LEXEME: }
ashhar@ashhar-Lenovo-B590:~/Desktop/Semester 8/Compiler Construction/Assignments/Assignment1/src/With_Lex$
```