CS424

GHULAM ISHAQ KHAN INSTITUTE OF ENGINEERING AND TECHNOLOGY, TOPI, SWABI

FACULTY OF COMPUTER SCIENCE AND ENGINEERING

COMPILER
ASSIGNMENT 01

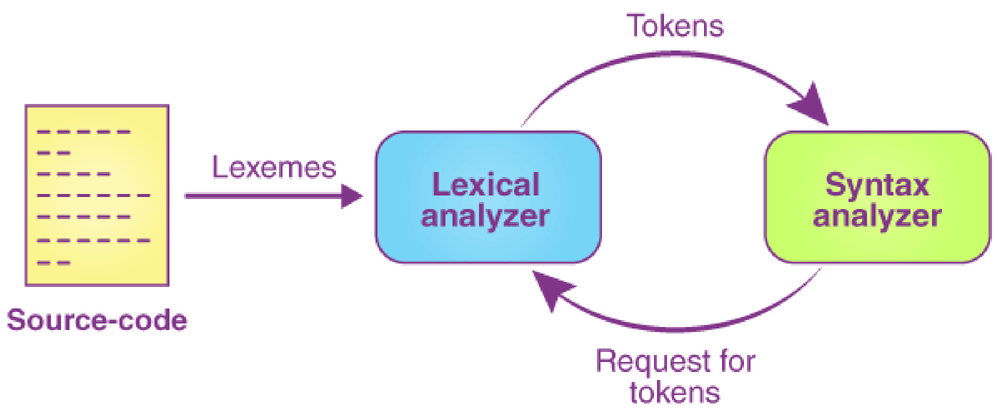
2020244 – MOHSIN ZIA

INSTRUCTOR: MR. USAMA ARSHAD
DATE: 20 FEB 2024

1. INTRODUCTION

The scanner is like a special tool used to understand what's inside code files. It helps to break down the code into smaller parts called tokens, which are like building blocks of the code. This scanner is also called a "lexical analyzer." When we use the scanner, we give it a file that has the code we want to understand. The scanner looks through this file carefully to find important parts. It breaks down the code into smaller pieces called "lexemes," which are like mini bits of code. Each lexeme has a special meaning, like a word or a number in the code.

This picture helps us see how the scanner reads and understands the code.



2. MINILANG REQUIREMENTS

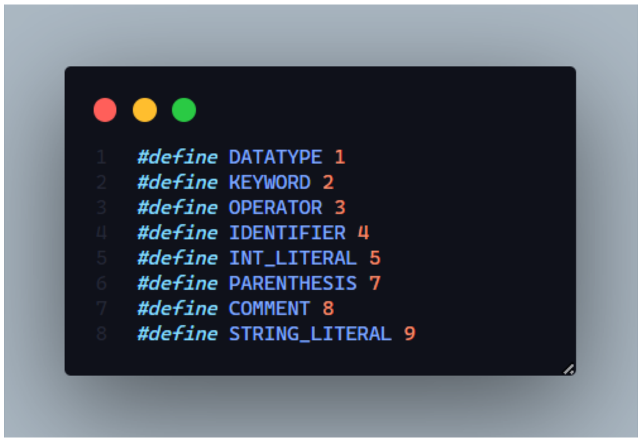
We've categorized our tokens into 8 different types to make them easier to understand. Here's what each type means:

- **DATATYPE**: They tell us what kind of information a variable holds. They can be things like "int" for whole numbers, "bool" for true or false values
- **KEYWORDS**: These are the special words that tell the computer to do something specific. For example, "print" tells the computer to show a message on the screen, "true" and "false" represent true or false values, and "if" and "else" help the computer make decisions.
- **IDENTIFIERS**: These are names we give to variables in our code. For instance, in the line "int Mohsin_123 = 123," "Mohsin_123" is the identifier.
- **LITERALS**: These are the actual pieces of information we give to our identifiers. In our program, we only accept values that are whole numbers (integers) or strings.
- **OPERATORS**: These symbols help us do math in our program. They include things like plus (+), minus (-), multiplication (*), and division (/).

- **PARANTHESIS**: These are tokens for symbols like "(" and ")", which help us group parts of our code together, especially in functions or statements.
- **COMMENTS**: The “//” is the comment sign.
- **OTHERS**: Anythings except these tokens are classified as others.

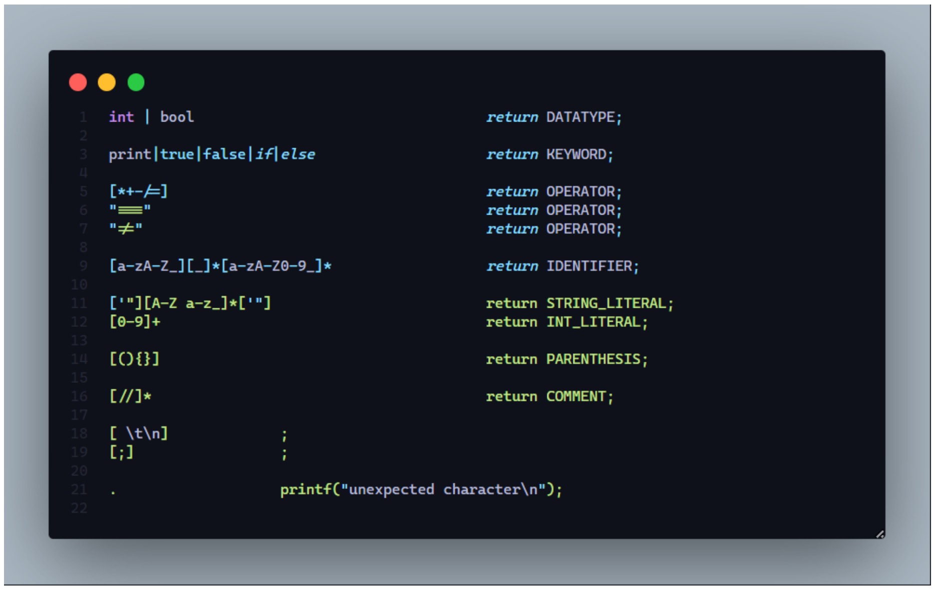
3. MINILANG IMPLEMENTATION

Firstly, I’ve created a header file that assigns each token a integer value.



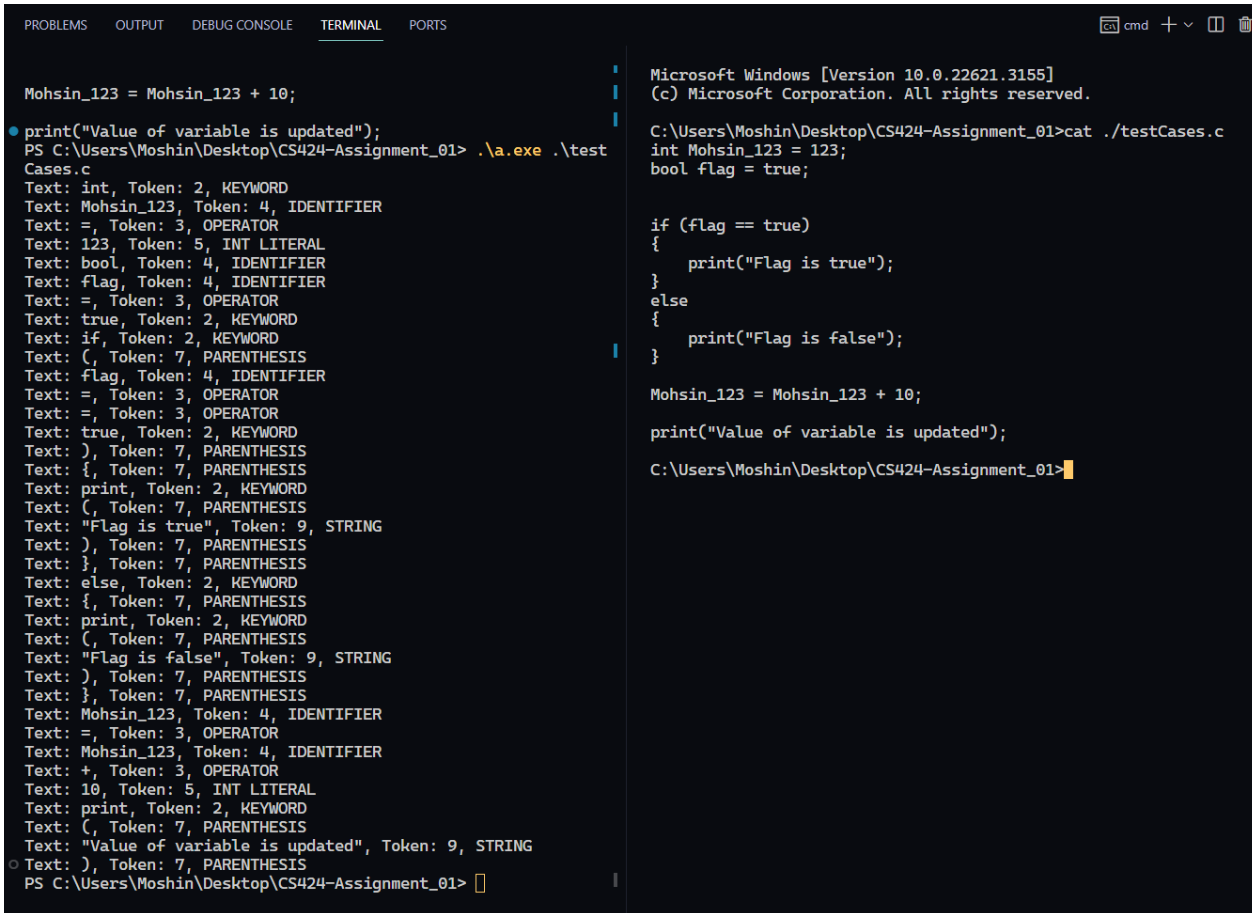
```
#define DATATYPE 1
#define KEYWORD 2
#define OPERATOR 3
#define IDENTIFIER 4
#define INT_LITERAL 5
#define PARENTHESIS 7
#define COMMENT 8
#define STRING_LITERAL 9
```

After that a lex file was created in which I’ve defined the rules for each requirement and the action we want to perform on that pattern being matched.



```
int | bool                                        return DATATYPE;

print|true|false|if|else                          return KEYWORD;

[*+-/=]                                           return OPERATOR;
"==="                                             return OPERATOR;
"≠"                                               return OPERATOR;

[a-zA-Z_][_]*[a-zA-Z0-9_]*                        return IDENTIFIER;

['"][A-Z a-z_]*['"]                               return STRING_LITERAL;
[0-9]+                                            return INT_LITERAL;

[(){}]                                            return PARENTHESIS;

[//]*                                             return COMMENT;

[ \t\n]               ;
[;]                   ;

.                     printf("unexpected character\n");
```

4. OUTPUT


PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
Mohsin_123 = Mohsin_123 + 10;

print("Value of variable is updated");
PS C:\Users\Moshin\Desktop\CS424-Assignment_01> .\a.exe .\test
Cases.c
Text: int, Token: 2, KEYWORD
Text: Mohsin_123, Token: 4, IDENTIFIER
Text: =, Token: 3, OPERATOR
Text: 123, Token: 5, INT LITERAL
Text: bool, Token: 4, IDENTIFIER
Text: flag, Token: 4, IDENTIFIER
Text: =, Token: 3, OPERATOR
Text: true, Token: 2, KEYWORD
Text: if, Token: 2, KEYWORD
Text: (, Token: 7, PARENTHESIS
Text: flag, Token: 4, IDENTIFIER
Text: =, Token: 3, OPERATOR
Text: =, Token: 3, OPERATOR
Text: true, Token: 2, KEYWORD
Text: ), Token: 7, PARENTHESIS
Text: {, Token: 7, PARENTHESIS
Text: print, Token: 2, KEYWORD
Text: (, Token: 7, PARENTHESIS
Text: "Flag is true", Token: 9, STRING
Text: ), Token: 7, PARENTHESIS
Text: }, Token: 7, PARENTHESIS
Text: else, Token: 2, KEYWORD
Text: {, Token: 7, PARENTHESIS
Text: print, Token: 2, KEYWORD
Text: (, Token: 7, PARENTHESIS
Text: "Flag is false", Token: 9, STRING
Text: ), Token: 7, PARENTHESIS
Text: }, Token: 7, PARENTHESIS
Text: Mohsin_123, Token: 4, IDENTIFIER
Text: =, Token: 3, OPERATOR
Text: Mohsin_123, Token: 4, IDENTIFIER
Text: +, Token: 3, OPERATOR
Text: 10, Token: 5, INT LITERAL
Text: print, Token: 2, KEYWORD
Text: (, Token: 7, PARENTHESIS
Text: "Value of variable is updated", Token: 9, STRING
Text: ), Token: 7, PARENTHESIS
PS C:\Users\Moshin\Desktop\CS424-Assignment_01>
```

```
Microsoft Windows [Version 10.0.22621.3155]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Moshin\Desktop\CS424-Assignment_01>cat ./testCases.c
int Mohsin_123 = 123;
bool flag = true;


if (flag == true)
{
    print("Flag is true");
}
else
{
    print("Flag is false");
}

Mohsin_123 = Mohsin_123 + 10;

print("Value of variable is updated");

C:\Users\Moshin\Desktop\CS424-Assignment_01>
```