Github: https://github.com/CatalinaArba/LFTC/tree/main/Lab9

# Documentation LEX and YACC

## 1. Introduction:

Flex and Bison are tools used in conjunction for lexical analysis (Flex) and parsing (Bison) in the construction of compilers and interpreters. They facilitate the process of creating a scanner and a parser, respectively, for processing programming language source code.

## 2. Lex and Flex:

Lex typically refers to the original lexical analyzer generator, while its modern counterpart is Flex (Fast Lexical Analyzer Generator). Flex is an enhanced version of Lex, providing more features and better performance.

Flex (Fast Lexical Analyzer Generator) is a tool for generating lexical analyzers (scanners) based on regular expressions. It reads an input file containing specifications in the form of regular expressions and corresponding actions. The output is a C source code file that can be compiled to create a scanner.

In our case, the Flex file is named myscanner.lxi. The specified regular expressions define the rules for identifying tokens in your programming language.

To generate the scanner, you use the following command in the command prompt:

**$ flex myscanner.lxi**

```
E:\GitHub\LFTC\Lab9>flex myscanner.lxi
```

This command processes the Flex file and generates a C source code file named lex.yy.c.

## 3. Bison:

Yacc is the original parser generator, and its modern equivalent is Bison. Bison offers improvements and additional features over Yacc and is widely used for generating parsers in the construction of compilers and interpreters.

Bison is a parser generator that takes a formal grammar file as input and generates a parser in C. The grammar file contains rules specifying the syntax of the programming language.

In our case, the Bison file is named parser.y. It defines the grammar rules for your language, including how different language constructs are parsed.

To generate the parser, you use the following command in the command prompt:

**$ bison -d parser.y**

```
E:\GitHub\LFTC\Lab9>bison -d parser.y
```

This command processes the Bison file and generates two files: parser.tab.c (the parser source code) and parser.tab.h (the parser header file).

## 4. Compilation:

After generating the scanner and parser source code files, you need to compile them along with your main program. In your case, the main program is named a.exe.

Use the following commands in the command prompt to compile the files:

**$ gcc lex.yy.c parser.tab.c -o a.exe**

```
E:\GitHub\LFTC\Lab9>gcc lex.yy.c parser.tab.c
```

This command compiles the generated scanner and parser files along with your main program, producing an executable file named a.exe.

## 5. Running the Program:

To run your program with an input file (e.g., p1.in), use the following command:

**$ ./a.exe p1.in**

or

**$ ./a.exe p2.in**

or

**$ ./a.exe p3.in**

This assumes that your input file is named p1.in. Adjust the filename accordingly.


## 6. Conclusion:

Flex and Bison are powerful tools for building compilers and interpreters. They streamline the process of lexical analysis and parsing, making it easier to develop robust language processors. The commands provided in this documentation illustrate the typical workflow for using Flex and Bison in conjunction.