LAB 9
Link github: https://github.com/teodoraarsene/Formal-Languages-and-Compiler-Design/tree/main/labs/lab9

Yacc Specification File

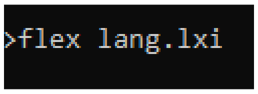
```
%{

#include <stdio.h>

#include <stdlib.h>


#define YYDEBUG 1

%}


%token INTEGER

%token STRING

%token CHAR

%token WHILE

%token FOR

%token IF

%token ELSEIF

%token ELSE

%token READ

%token PUTS

%token BREAK

%token RETURN

%token NEXT

%token END


%token plus

%token minus

%token mul
```

```
%token division
%token eq
%token equal
%token different
%token less
%token more
%token lessOrEqual
%token moreOrEqual

%token leftRoundBracket
%token rightRoundBracket
%token leftCurlyBracket
%token rightCurlyBracket

%token IDENTIFIER
%token NUMBER_CONST
%token STRING_CONST
%token CHAR_CONST

%start program

%%

program : declaration_list statements
declaration_list : declaration declaration_list | /*Empty*/
declaration : var_type IDENTIFIER equal_expression
equal_expression : eq expression | /*Empty*/
var_type : INTEGER | CHAR | STRING
expression : term sign_and_expression
```

```
sign_and_expression : sign expression | /*Empty*/

sign : plus | minus | mul | division

term : IDENTIFIER | constant

constant : NUMBER_CONST | STRING_CONST | CHAR_CONST

statements : statement statements | /*Empty*/

statement : simple_stmt | struct_stmt

simple_stmt : assignment_stmt | input_output_stmt

struct_stmt : if_stmt | while_stmt

assignment_stmt : IDENTIFIER eq expression

input_output_stmt : READ leftRoundBracket term rightRoundBracket | PUTS leftRoundBracket term rightRoundBracket

if_stmt : IF leftRoundBracket condition rightRoundBracket leftCurlyBracket statements rightCurlyBracket else_stmt

else_stmt : ELSE leftCurlyBracket statements rightCurlyBracket | /*Empty*/

while_stmt : WHILE leftRoundBracket condition rightRoundBracket leftCurlyBracket statements rightCurlyBracket

condition : expression relation expression

relation : equal | different | less | more | lessOrEqual | moreOrEqual


%%


yyerror(char *s)

{

        printf("%s\n",s);

}


extern FILE *yyin;


main(int argc, char **argv)

{
        if(argc>1) yyin :  fopen(argv[1],"r");

        if(argc>2 && !strcmp(argv[2],"-d")) yydebug: 1;

        if(!yyparse()) fprintf(stderr, "\tProgram is syntactically correct.\n");

}
```

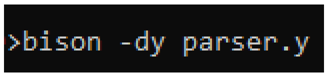
Demo

We first run the command:



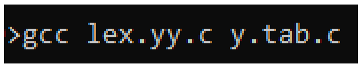
```
>flex lang.lxi
```

Then we run:



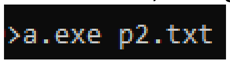
```
>bison -dy parser.y
```

And:



```
>gcc lex.yy.c y.tab.c
```

An executable was created after the second command, so now we can run the program.
We have 4 examples for which we can run the program (p1.txt, p2.txt, p3.txt and p1err.txt)
In this demo, I am going to run the program for p2.txt, using the following command:



```
>a.exe p2.txt
```

Where a.exe being the generated executable.