Documentacion y pruebas analizador sintactico

Github: https://github.com/kelok3rik/Analizador-Sintactico

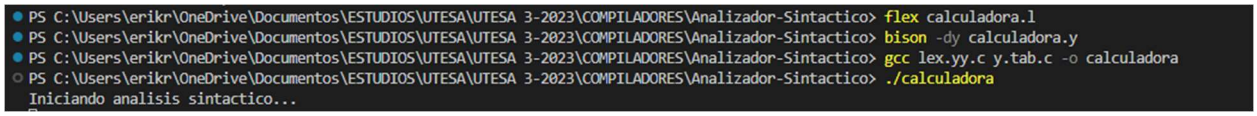
Ejecucion inicial

1) Generar el código C del analizador léxico con **flex calculadora.l**
   Esto generará un archivo llamado lex.yy.c.

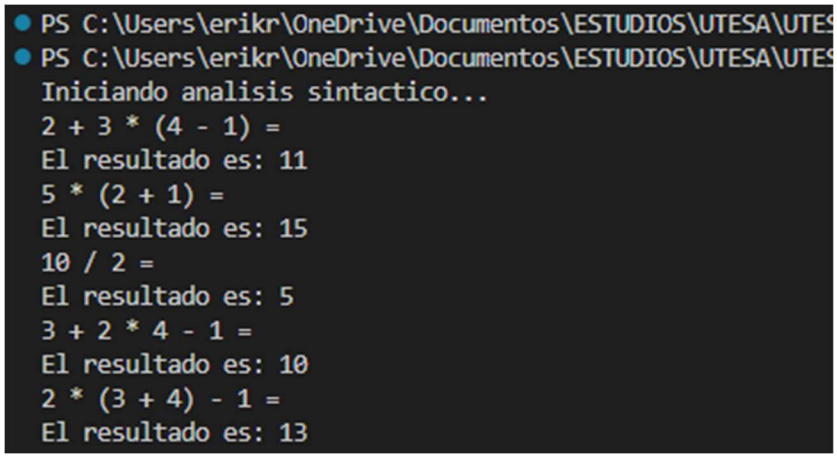
2) Generar el código C del analizador sintantico con **bison -dy calc.y**
   Esto generará los archivos y.tab.c y y.tab.h.

3) Compilar los archivos generados con **gcc lex.yy.c y.tab.c -o calculator**
   Esto compilará los archivos C generados en un ejecutable llamado calculator.

4) Ejecutar programa con **./calcultator**



Ejemplos de uso



Analizador sintactico

```
%{
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

void yyerror(const char *s);
int yylex(void);


%}

%token NUMBER
```

%{ ... %}: Esta sección permite incluir código C directamente en el archivo de Bison. En este caso, se incluyen las cabeceras estándar de C y se declaran las funciones yyerror y yylex.

%token NUMBER: Define el token NUMBER que será utilizado en el análisis léxico para representar los números.

```
%%

input:  /* empty */
    | input expr '=' { printf("El resultado es: %d\n", $2); }
    | input "salir" { printf("Saliendo del programa.\n"); exit(0); }
    ;

expr:   expr '+' term { $$ = $1 + $3; }
    | expr '-' term { $$ = $1 - $3; }
    | term { $$ = $1; }
    ;

term:   term '*' factor { $$ = $1 * $3; }
    | term '/' factor { $$ = $1 / $3; }
    | factor { $$ = $1; }
    ;

factor: NUMBER
    | '(' expr ')' { $$ = $2; }
    ;

%%
```

input: Regla que permite que haya múltiples expresiones. Se compone de expresiones y el símbolo (=) . Cuando se encuentra esta combinación, imprime el resultado.

expr: Define cómo se construyen las expresiones usando los operadores + y -.

term: Define cómo se construyen los términos usando los operadores * y /.

factor: Define los factores en las expresiones.

```
int main() {
    printf("Iniciando analisis sintactico...\n");

        yyparse();

    printf("Analisis sintactico completado.\n");
    return 0;
}

void yyerror(const char *s) {
    fprintf(stderr, "Error sintactico: %s\n", s);
}
```

main: Función principal que inicia el análisis sintáctico llamando a yyparse.

yyerror: Función que se llama en caso de errores sintácticos. Imprime un mensaje de error con información sobre el error.