

UNIVERSIDAD NACIONAL DEL ALTIPLANO



BIYACC

Facultad: Ingeniería Mec. Eléctrica, Electrónica y Sistemas

Escuela Profesional: Ingeniería de Sistemas

Curso: compiladores

Docente: Fernadez Chambi Mayenka

Integrantes:

- Julio Harold Barra Juli

Semestre: sexto

Perú - 2022

Directivas y declaración de variables:

Las líneas `%{ }` y `%}` se utilizan para incluir código C en el archivo de especificación de Bison.

Las líneas `#include <stdio.h>` y `#include <stdlib.h>` son directivas para incluir los encabezados de las bibliotecas estándar de C.

`extern int yylex()` y `extern void yyerror(const char* msg)` son declaraciones externas de funciones definidas en otro lugar.

Declaración de la unión y los símbolos:

`%union` se utiliza para definir una unión que contiene una variable de tipo `float` llamada `f`.

`%token <f> NUM` declara un símbolo `NUM` que se asocia con un valor de tipo `float`.

`%type <f> E T F` declara los tipos de retorno para las reglas gramaticales `E`, `T` y `F`.

Reglas gramaticales:

`S` es el símbolo inicial y representa la expresión completa. En este caso, simplemente imprime el resultado de la expresión.

`E`, `T` y `F` son reglas gramaticales que definen la estructura de las expresiones aritméticas.

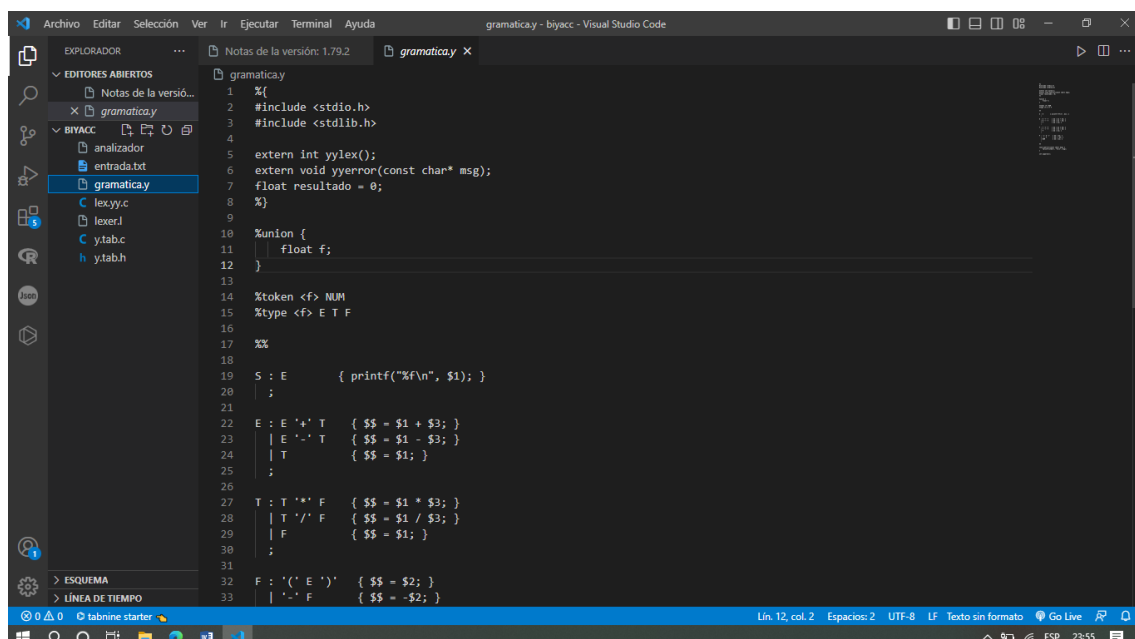
Las reglas están definidas usando la notación de producción. Por ejemplo, `E : E '+' T` indica que una expresión `E` puede ser la suma de otra expresión `E` más un término `T`.

Las acciones asociadas con las reglas se definen entre llaves `{ }`. Por ejemplo, `$$ = $1 + $3` asigna el resultado de la suma de `$1` y `$3` a `$$`.

Funciones `yyerror` y `yyparse`:

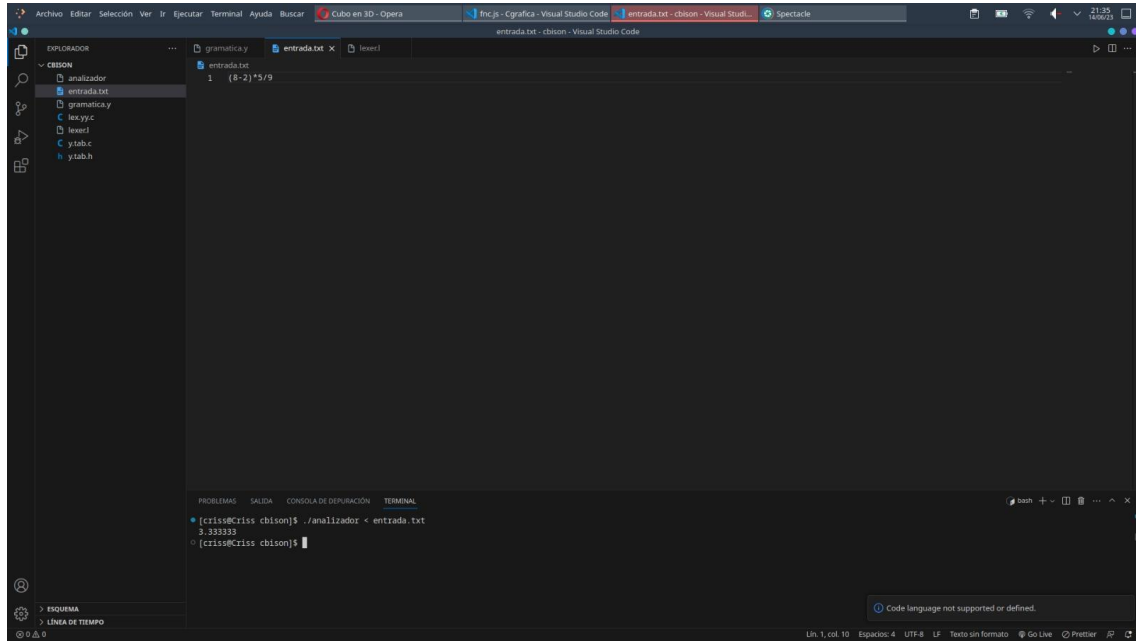
`yyerror` es una función definida por el usuario que se llama cuando se produce un error de análisis.

`yyparse` es una función generada por Bison que inicia el análisis sintáctico.



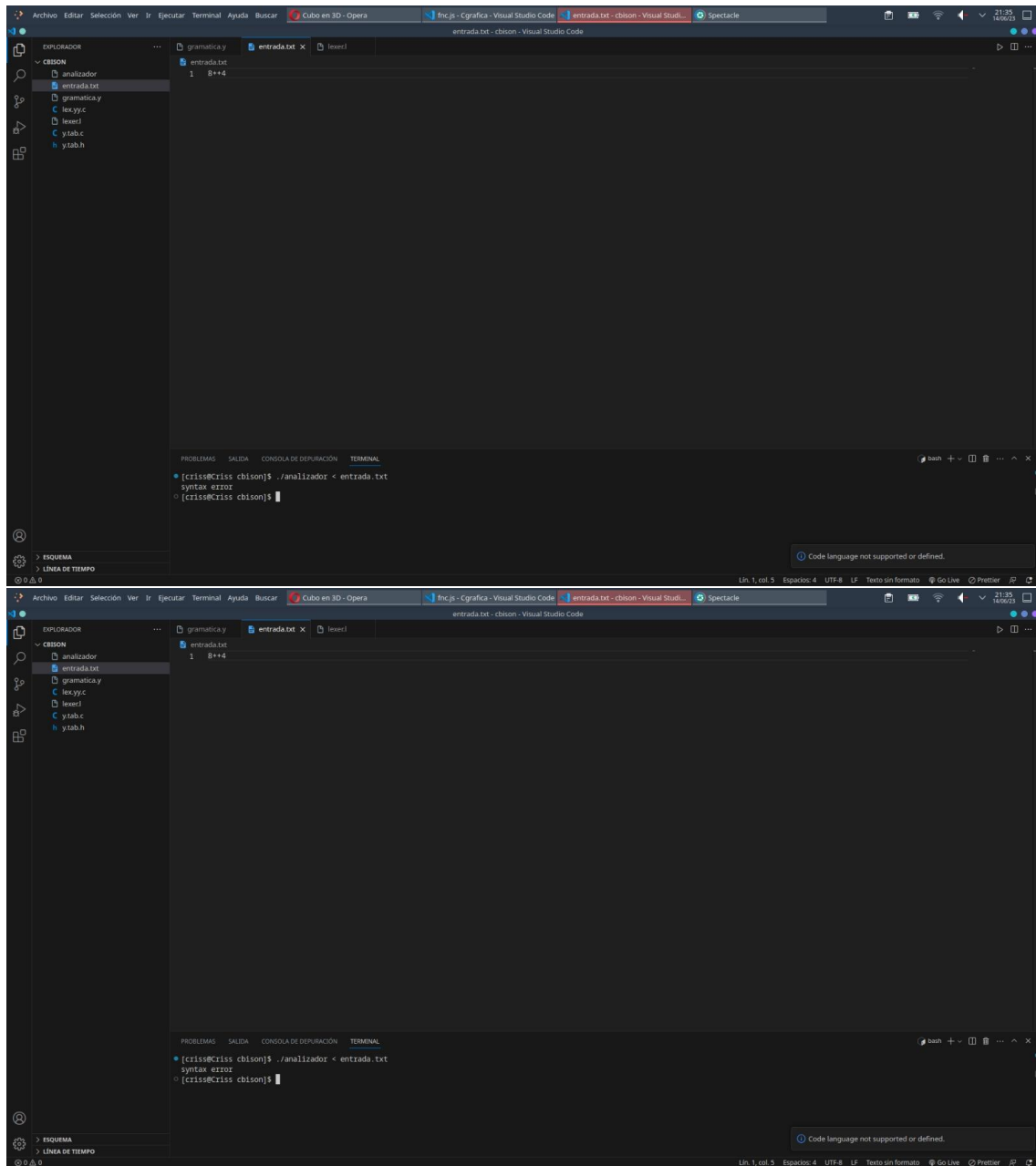
```
1  %{
2  #include <stdio.h>
3  #include <stdlib.h>
4
5  extern int yylex();
6  extern void yyerror(const char* msg);
7  float resultado = 0;
8  %}
9
10 %union {
11     float f;
12 }
13
14 %token <f> NUM
15 %type <f> E T F
16
17 %%
18
19 S : E          { printf("%f\n", $1); }
20   ;
21
22 E : E '+' T    { $$ = $1 + $3; }
23   | E '-' T    { $$ = $1 - $3; }
24   | T          { $$ = $1; }
25   ;
26
27 T : T '*' F    { $$ = $1 * $3; }
28   | T '/' F    { $$ = $1 / $3; }
29   | F          { $$ = $1; }
30   ;
31
32 F : '(' E ')'  { $$ = $2; }
33   | '-' F      { $$ = -$2; }
```

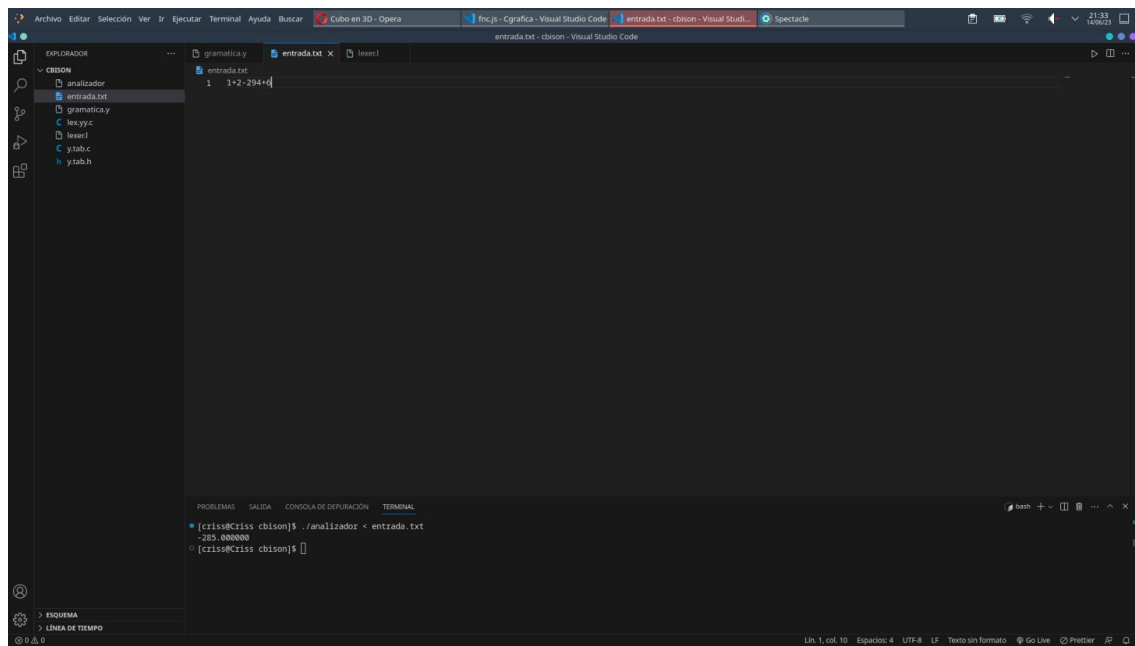
PRUEBAS



The screenshot shows the Visual Studio Code interface with the following details:

- Explorer:** A folder named 'CBISON' is expanded, showing files 'analizador', 'entrada.txt', 'gramatica.y', 'lex.yyc', 'lexer.l', 'y.tab.c', and 'y.tab.h'.
- Editor:** The file 'entrada.txt' is open, containing the text `(8-2)*5/9`.
- Terminal:** The terminal window at the bottom shows the command `[criss@criss cbison]$./analizador < entrada.txt` and its output `8.88889`.
- Status Bar:** The bottom status bar indicates 'Lin. 1, col. 10', 'Español', 'UTF-8', 'LF', 'Texto sin formato', 'Go Live', 'Prettier', and '9/9'.





LINK DEL REPOSITORIO

<https://github.com/haroldbarra/biyacc.git>