

Universidad de San Carlos de Guatemala.
Facultad de ingeniería
Ingeniería en Ciencias y Sistemas
Organización de compiladores y lenguajes 1.
Aux. Kevin López.

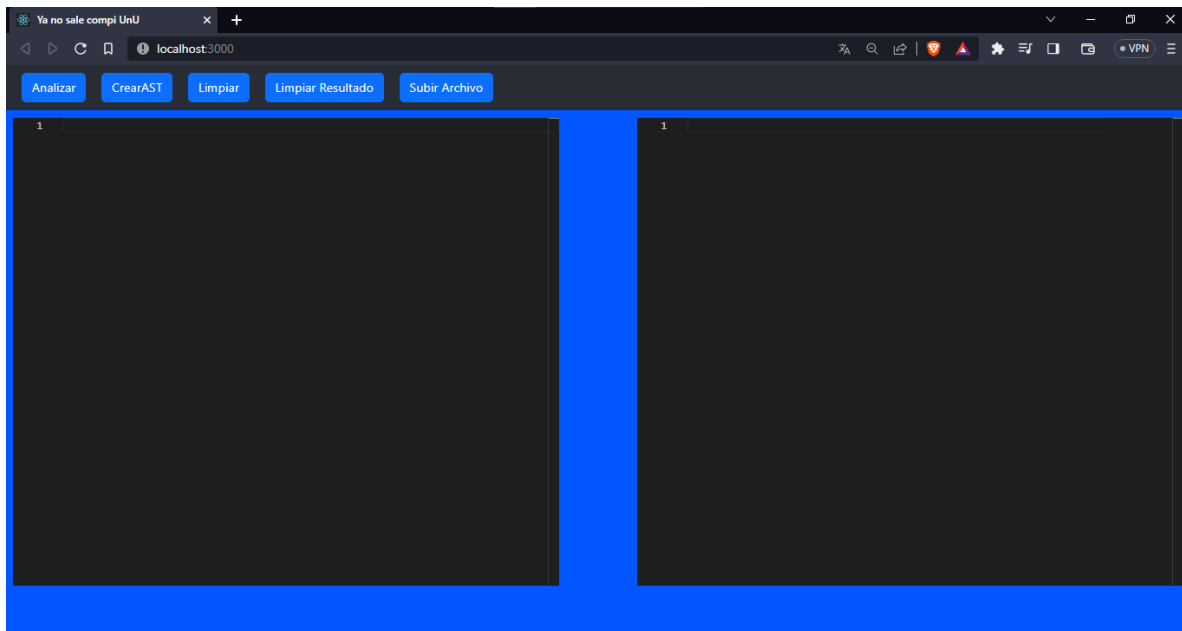
Proyecto 2//Manual de usuario.

Kevin Estuardo
Secaída Molina
201602404

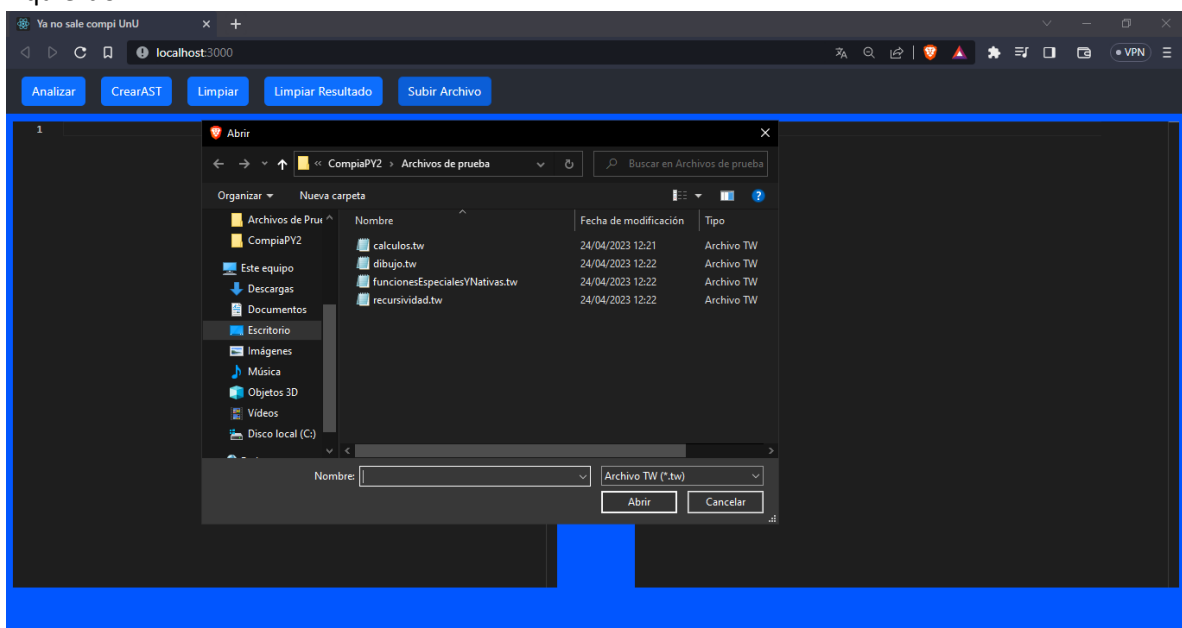
El siguiente manual da indicaciones al usuario de cómo puede y debería de utilizar el programa para que este de los resultados óptimos.

El usuario contará con una aplicación web donde podrá visualizar las 2 áreas de texto y 5 botones.

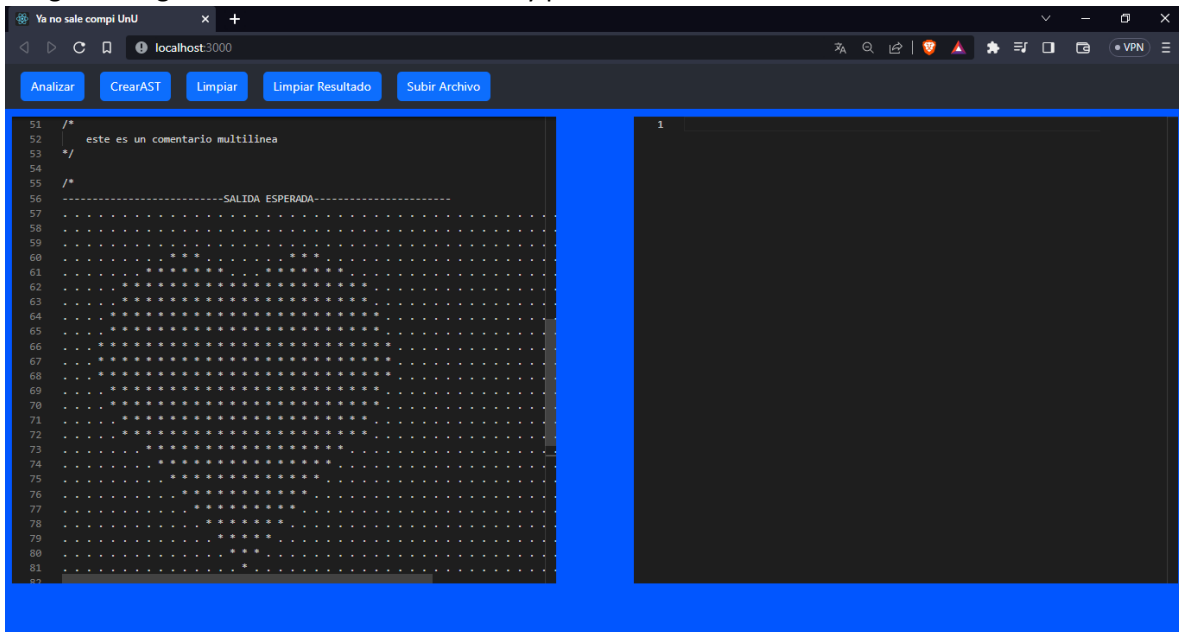
El área de texto del lado izquierdo se utiliza para ingresar texto y el área de texto del lado derecho es donde se mostrará el resultado obtenido luego de presionar el botón analizar.



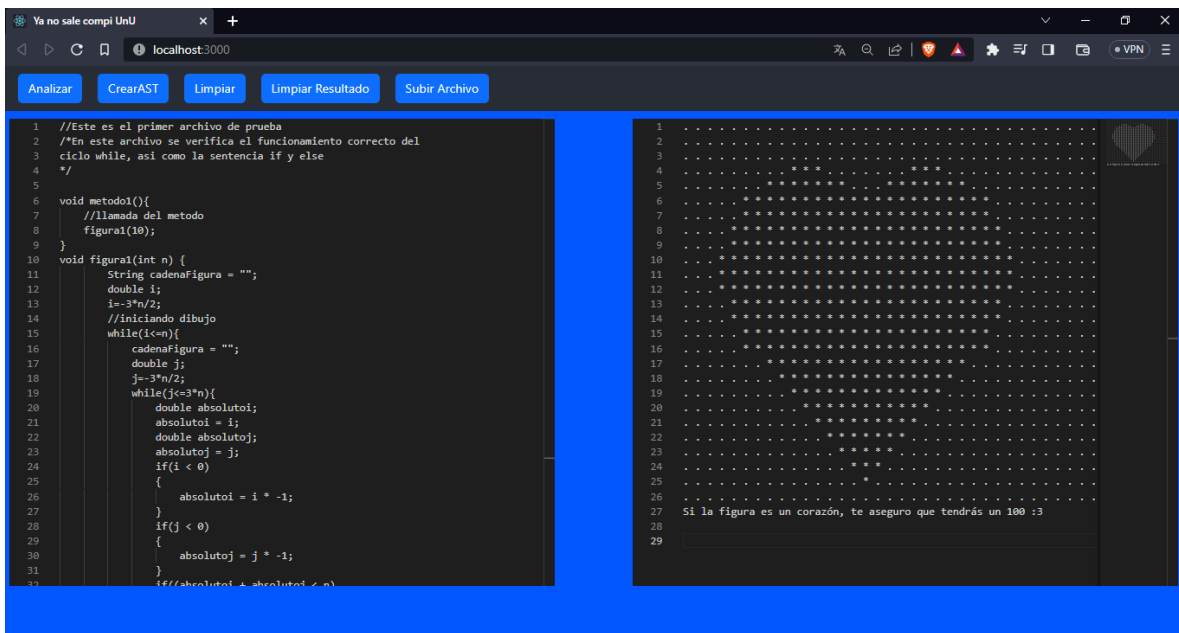
También cuenta con otros botones pero ahorita nos enfocaremos en el botón Subir archivo, el cual se utilizar para cargar archivos con extensión .tw para facilitar el ingreso de texto al área del lado izquierdo.



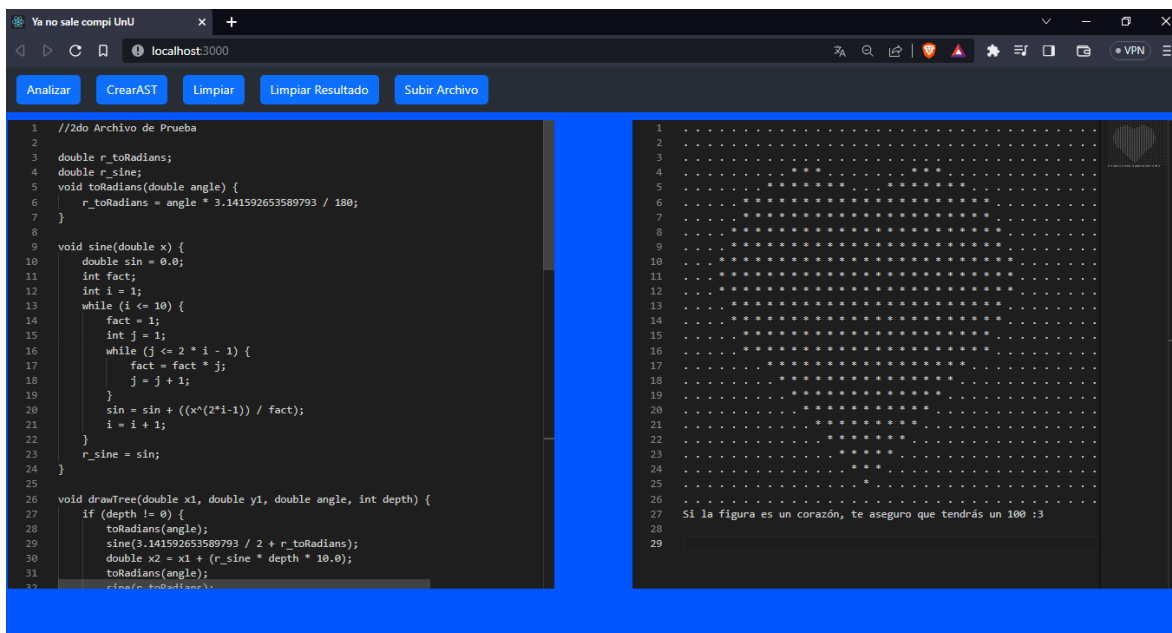
Luego de cargar el archivo este se mostrara y podrá editarlo.



Luego al momento de presionar el botón de analizar este recorrerá el texto e imprimirá el resultado del lado derecho.



Aquí se subió otro archivo para comprobar que la carga si funcionaba y sustituía la carga que ya estaba ahí.

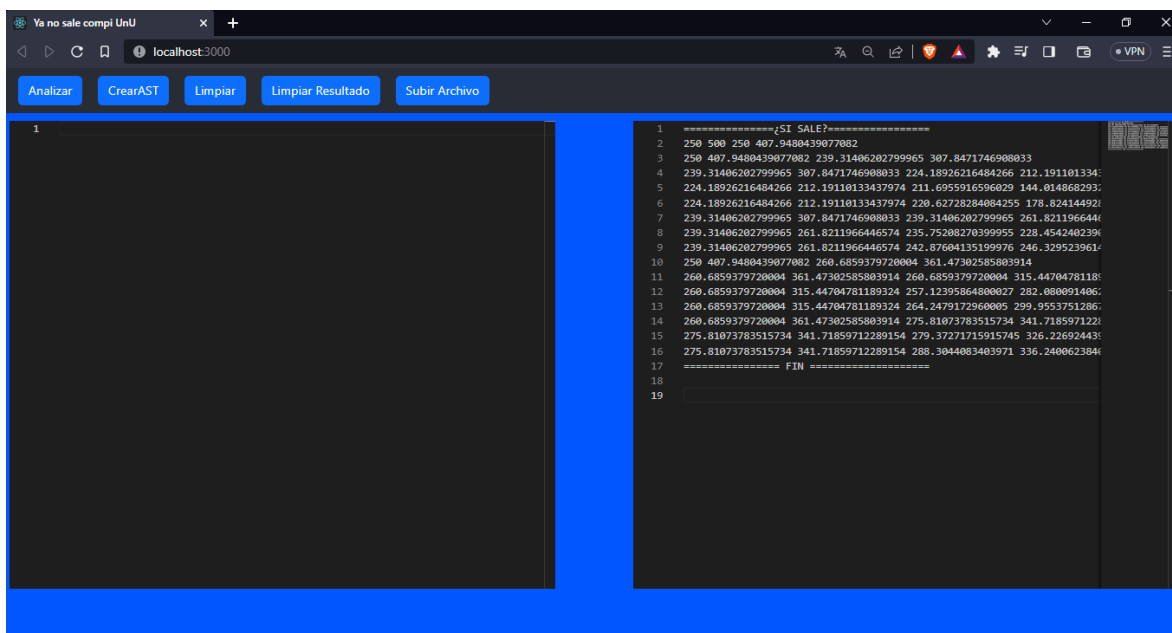


The screenshot shows a web application interface with a dark theme. At the top, there's a browser window with the address bar showing 'localhost:3000'. Below the browser window, there are five buttons: 'Analizar', 'CrearAST', 'Limpiar', 'Limpiar Resultado', and 'Subir Archivo'. The main area is divided into two panels. The left panel contains a code editor with the following C code:

```
1 //2do Archivo de Prueba
2
3 double r_toRadians;
4 double r_sine;
5 void toRadians(double angle) {
6     r_toRadians = angle * 3.141592653589793 / 180;
7 }
8
9 void sine(double x) {
10     double sin = 0.0;
11     int fact;
12     int i = 1;
13     while (i <= 10) {
14         fact = 1;
15         int j = 1;
16         while (j <= 2 * i - 1) {
17             fact = fact * j;
18             j = j + 1;
19         }
20         sin = sin + ((x^(2*i-1)) / fact);
21         i = i + 1;
22     }
23     r_sine = sin;
24 }
25
26 void drawTree(double x1, double y1, double angle, int depth) {
27     if (depth != 0) {
28         toRadians(angle);
29         sine(3.141592653589793 / 2 + r_toRadians);
30         double x2 = x1 + (r_sine * depth * 10.0);
31         toRadians(angle);
32         sine(r_toRadians);
33     }
34 }
```

The right panel shows the output of the program, which is a text-based representation of a tree structure. The output is a large block of text with many lines of numbers and symbols, including a heart symbol at the end.

También cuenta con los botones limpiar el cual limpia el área de texto del lado izquierdo y el limpiar resultado borra el área de texto derecho.



The screenshot shows the same web application interface as before, but after clicking the 'Limpiar' button. The left panel is now empty, and the right panel shows a large block of text with many lines of numbers and symbols, including a heart symbol at the end. The text is the same as the one in the previous screenshot, but it appears to be a different output or a different part of the same output.

Por ultimo mostramos el resultado de presionar el botón de crear AST en el cual podemos visualizar como fue la ejecución del código ingresado.

The image displays a screenshot of an IDE interface, likely Visual Studio Code, showing the results of creating an AST (Abstract Syntax Tree) for a code snippet. The top panel shows the AST tree structure, which is a hierarchical representation of the code's syntax. The middle panel shows the execution results in the console, including the type and value of variables, and the execution of instructions. The bottom panel shows the code snippet being analyzed, which is a JavaScript function that calculates the sine of an angle in radians.

The AST tree structure is as follows:

```
graph TD
    Root[AST] --> Declaration[DECLARACION]
    Root --> Method[DEC METODO]
    Root --> Main[MAIN]
    Declaration --> ValCadena[VAL CADENA]
    ValCadena --> ValCadenaValue["tipo: 'VAL CADENA', valor: '=====¿SI SALE?=====', linea: 42, columna: 11"]
    Method --> Instruccion[INST INSTRUCCION]
    Instruccion --> InstruccionValue["LLAMADA_METODO instrucciona al inicio del recorrerInstrucciones"]
    Main --> Instruccion
    Instruccion --> InstruccionValue["INST_PRINT instrucciona al inicio del recorrerInstrucciones"]
    Instruccion --> ValCadena
    ValCadena --> ValCadenaValue["tipo: 'VAL CADENA', valor: '===== FIN =====', linea: 44, columna: 11"]
```

The console output shows the execution results:

```
tipo: 'VAL CADENA',
valor: '=====¿SI SALE?=====',
linea: 42,
columna: 11
} expression
LLAMADA_METODO instrucciona al inicio del recorrerInstrucciones
INST_PRINT instrucciona al inicio del recorrerInstrucciones
VAL CADENA valor
{
  tipo: 'VAL CADENA',
  valor: '===== FIN =====',
  linea: 44,
  columna: 11
} expression
Archivo creado
```

The bottom panel shows the code snippet being analyzed:

```
1  =====¿SI SALE?=====
2  250 500 250 407.9480439077082
3  250 407.9480439077082 239.314
4  239.31406202799965 307.847174
5  224.18926216484266 212.191108
6  224.18926216484266 212.191108
7  239.31406202799965 307.847174
8  239.31406202799965 261.821194
9  239.31406202799965 261.821194
10 250 407.9480439077082 260.488
11 260.6859379720004 361.4730251
12 260.6859379720004 315.4470471
13 260.6859379720004 315.4470471
14 260.6859379720004 361.4730251
15 275.81073783515734 341.718597
16 275.81073783515734 341.718597
17 ===== FIN =====
18
19
```

The console also shows the execution results of the code snippet, including the type and value of variables, and the execution of instructions.