Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Organización de Lenguajes y Compiladores 1 – “C”
Laboratorio
Aux. José Diego Pérez Toralla

Proyecto 2
Manual Técnico

Nombre: Eduardo Alexander Reyes Gonzalez
Carnet: 202010904

Contenido

Objetivos

Objetivo General

Que el usuario comprenda el programa y su funcionamiento

Objetivos Específicos

- Demostrar por medio de la programación la realización del método del árbol

- Mostrar debidos reportes al usuario

Alcances

Que Las personas se interesen en el programa y lo puedan utilizar en un futuro

Requisitos mínimos

Requisitos de hardware

- Equipo Pentium II o superior
- Mínimo 64 Mb en memoria RAM
- Sistema Operativo win 98 o superior
- Resolución grafica mínimo 800 * 600

Requisitos de Software

- Java virtual Machine
- Navegador a internet
- NetBeans o un IDE de java

Clases Utilizadas

1) **AST:** Esta clase es encarga de almacenar toda la información del árbol sintáctico, además de ejecutar cada instrucción que llega.

2) **Instruccion:** Es la clase que indica que una ejecución es una de tipo instrucción. Especialmente las funciones que no retornan ningún valor.

3) **Expresion:** Es la clase que indica que una ejecución es una de tipo expresión. Especialmente las funciones que retornan valores de tipo Expresión.

4) **LISTA_EJECUCIONES:** Es la clase que guarda la línea, columna y nombre de la función que se ejecuta.

5) **NODO_GRAFICA:** En esta clase se crean los nodos utilizados para la creación de la gráfica del árbol sintáctico

6) **NODO_REPORTE_ERROR:** Es la clase encargada de crear los nodos utilizados en la grafica del reporte de errores.

7) **NODO_TABLA_SIMBOLOS:** Es la clase encargada de crear los nodos utilizados en la grafica de la tabla de símbolos.

8) **TABLA_FUNCIONES_Y_VARIABLES:** Es la clase del entorno de trabajo, en esta clase se almacenan las variables, listas, vectores y funciones del programa

9) **FUNCION:** En esta clase guarda objetos función y métodos que se declaran en el programa

10) **VARIABLE:** En esta clase se crean objetos de tipo Variable y es utilizada para crear las variables declaradas en el programa

11) **VECTOR:** En esta clase se crean objetos de tipo vector, y es utilizada para crear los vectores declarados en el programa

12) **LISTA:** En esta clase se crean objetos de tipo Lista, y es utilizada para crear las listas declaradas en el programa

13) **METODO:** En esta clase se crean objetos de tipo Método, y es utilizado para crear los métodos y listas declarados en el programa

14) **PARAMETRO:** Esta clase crea los objetos de tipo parámetro utilizado en las funciones y métodos.

15) **Tipo:** Esta clase es la principal para la declaración de variables, vectores, listas etc. Y gracias a esta se pueden declarar valores de tipo entero, booleano, doublé, string, void.

16) **CASTEOS:** Esta clase es la principal para guardar los objetos de tipo expresión que y realizar operaciones de tipo casteo.

17) **IF:** Esta clase crea objetos de instrucción de tipo IF, gracias a esta clase se pueden realizar operaciones de tipo if.

18) **PRINT:** Esta clase crea instrucciones de tipo Print, y es utilizada para poder ejecutar cada instrucción print en el programa

19) **WHILE**: Esta clase crea instrucciones cíclicas de tipo while, y se encarga de ejecutar todas las instrucciones de este tipo del programa.

20) **FOR:** Esta clase crea las instrucciones de tipo for, y es la encargada del funcionamiento de estas instrucciones.

21) **DO_WHILE:** Esta clase crea las instrucciones cíclicas de tipo dowhile, y se encarga de realizar la ejecución de cada instrucción de este tipo que se declara en el programa.

22) **SWITCH:** Esta clase se encarga de crear el objeto switch, el cual se encarga de recorrer cada objeto case y ejecutar el switch correctamente.

23) **CASE:** Esta clase se encarga de crear cada objeto case, que es utilizado en la clase switch. Es el encargado de almacenar la información necesaria para hacer la comparación con cada case.

24) **TO_LOWER:** Esta clase se encarga de crear cada expresión to_lower, que es declarado en el programa.

25) **TO_UPPER:** Esta clase se encarga de crear cada expresión toupper, que es declarado en el programa.

26) **LENGTH:** Esta clase se encarga de crear cada expresión length, que es declarado en el programa.

27) **TRUNATE:** Esta clase se encarga de crear cada expresión truncate, que es declarado en el programa.

28) **ROUND:** Esta clase se encarga de crear cada expresión round, que es declarado en el programa.

29) **TYPEOF:** Esta clase se encarga de crear cada expresión typeof, que es declarado en el programa.

30) **TOSTRING:** Esta clase se encarga de crear cada expresión tostring, que es declarado en el programa.

31) **BREAK:** Esta clase se encarga de crear cada instrucción break declarada en el programa. Esta clase modifica su parámetro break para dar a entender a las otras clases que se ejecutó una función break.

32) **CONTINUE:** Esta clase se encarga de crear cada expresión continue, que es declarado en el programa.

RETURN: Esta clase se encarga de crear cada expresión return, que es declarado en el programa. Esta clase retorna valores.

33) **TOCHARARRAY:** Esta clase se encarga de crear cada expresión tochararray, que es declarado en el programa. Y se encarga de retornar una lista el cual será guardada en un objeto lista.

34) **OPERACIÓN_TERNARIA:** Esta clase se encarga de ejecutar las operaciones ternarias ejecutadas en el programa.

35) **OPERACIÓN_UNARIA:** Esta clase se encarga de ejecutar las operaciones que poseen un solo signo, tales como la negación, indicar un numero negativo, entre otras.

36) **OPERACIONES:** Esta clase se encarga de ejecutar las operaciones que se encuentran en el programa, tales como operaciones aritméticas, lógicas, relacionales.

37) **Valor:** Esta clase se encarga de buscar cada variables, lista o vector y retorna el valor del mismo.

38) **ASIGNACION_VARIABLE:** Esta clase se encarga de la ejecución de la instrucción de asignación de variables que se encuentran en el programa.

39) **DECLARACION_VARIABLE:** Esta clase se encarga de la ejecución de la instrucción de declaración de variables y se encarga de guardarlos donde deben estar.

40) **VALIDAR_EXISTE_VARIABLE;** Esta clase se encarga de validar la existencia de variables y se encarga de retorna el objeto de la variable si es que existe.

41) **DECLARACION_VECTOR_TIPO1:** Esta clase es la encargada de declarar los vectores y guardarlos en el entorno correspondiente.

42) **ASIGNACION_VECTOR:** Es la clase que se encarga de asignar valores en un vector de una posición determinada.

43) **VALIDAR_EXISTE_VECTOR:** Se encarga de realizar la validación correspondiente a los vectores, si existe retorna el objeto del vector, si no existe retorna un objeto indefinido.

44) **DECLARACION_METODO:** Se encarga de realizar la declaración de los métodos que se declaran en el programa.

45) **LLAMADA_METODO:** Se encarga de ejecutar cada instrucción del método al que es llamado.

46) **LLAMADA_METODO_EXPRESION:** Se encarga de ejecutar cada expresión del método al que es llamado, con la diferencia que este retorna valores y pueden ser asignados a una variable.

47) **LLAMADA_MAIN:** Se encarga de que ejecute la función principal del programa.

48) **VALIDAR_EXISTE_FUNCION:** Se encarga de validar si la función a la que se quiere llamar existe.

49) **DECLARACION_PARAMETRO:** Se encarga de declarar los parámetros de las funciones y métodos.

50) **DECLARACION_LISTA_TIPO1:** Se encarga de hacer la declaración de las listas en el respectivo ámbito.

51) **AGREGAR_A_LISTA:** Se encarga de acceder a la lista que se esta llamando, y agregar valores a ella.

52) **VALIDAR_EXISTE_LISTA:** Esta clase se encarga de validar si la lista a la que se está llamando existe. Para posteriormente guardar valores.

53) **ASIGNACION_LISTA:** Esta clase se encarga de guardar un valor especifico en la posición de la lista especificado.

54) **DECLARACION_LISTA_TIPO2:** Se encarga de realizar la declaración de lista utilizando la instrucción tochararray, el cual recibe una lista y crea la lista correspondiente

Menú del sistema

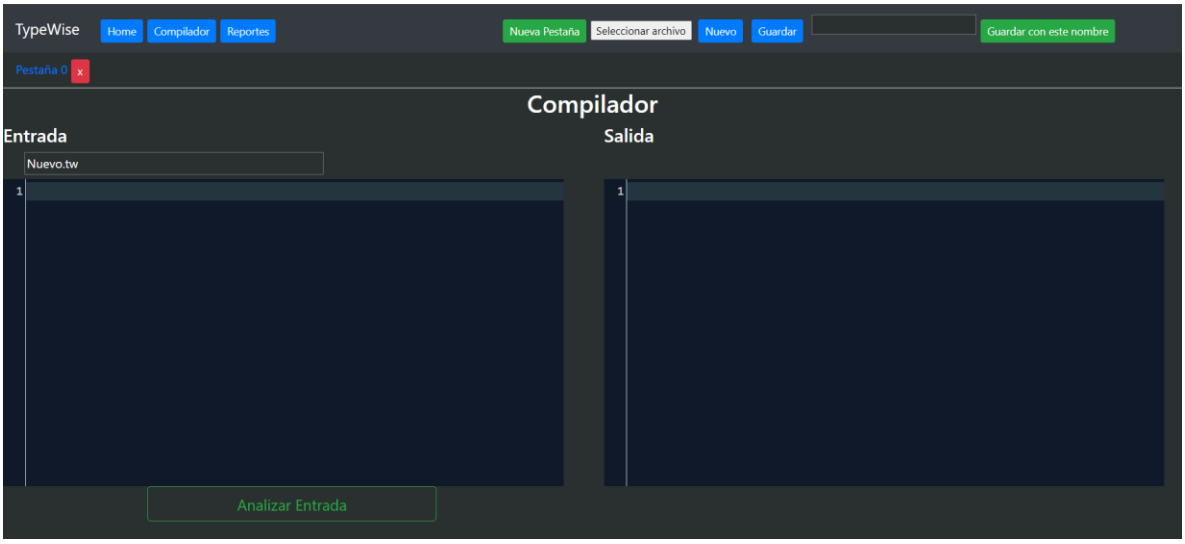
Al ingresar al programa se encontrará con una interfaz gráfica en la cual se encuentran una ventana en la que podemos observar la información del programa, y 2 botones, el cual es el botón de home, y el botón de Compilador.

Compilador y Editor

Al presionar el botón compilador, nos mandara a la ventana más importante de nuestro programa el cual es nuestro compilador.

En esa ventana se podrá observar 2 cuadros de texto, uno para entrada y otro para salida, en el cuadro de entrada podemos ingresar el código que deseamos ejecutar, y para ejecutarlo presionamos el botón Ejecutar que se encuentran en la parte de abajo del cuadro de entrada.

También poseemos en la barra de navegación superior, 3 botones: Home, Compilador y Reportes.



También encontraremos en la parte superior unos botones el cual es nuestra función de Editor, El cual nos permitirá:

1) crear nuevas pestañas
2) abrir un archivo
3) Nuevo
4) Guardar
5) Guardar Como

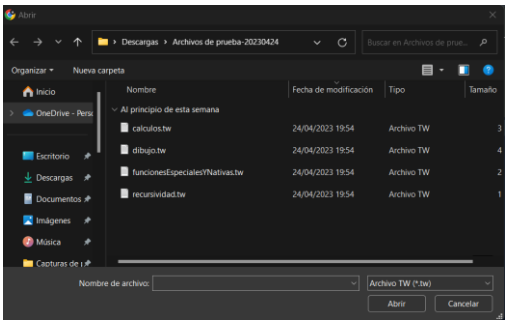
1) Crear nuevas pestañas

Las pestañas funcionan como en cualquier navegador, creamos una pestaña y podemos escribir nuestro código, y con total libertad podemos movernos a otras pestañas si perder el código.

Para el funcionamiento de esta función, se creo un script el cual se encarga de crear una pestaña nueva cada vez que se presiona el botón, también a ese botón se la agrega una función en específico que se encarga de obtener la posición en la que se encuentra y mandar a llamar el contenido de la lista correspondiente a los valores de entrada, y se le agrega un botón con una x para poder cerrar la pestaña abierta.

2) Abrir un archivo

Para abrir un archivo, presionamos el botón Seleccionar Archivo, y este nos mostrara una ventana en la que podemos abrir un archivo con extensión .TW. Debido a que en el navegador no se puede acceder a la información del computador, y utilizarlo a nuestro favor, se le envía el archivo y se encarga de obtener el texto y colocarlo en el textarea.



3) Nuevo

Si presionamos el botón Nuevo, este vaciara el cuadro de texto de Entrada y colocara como nombre nuevo.tw, asi podemos vaciar cuadros de texto de manera rápida.

4) Guardar Como

En la parte superior aparece un cuadro de texto en el cual podemos escribir el nombre con el que deseamos guardar nuestro documento, al presionar el botón guardar como, tomara el nombre de ese cuadro de texto y descargara un archivo con ese nombre. Debido a que no se puede acceder a la memoria del usuario, este solo puede descargar el archivo.

4) Guardar

Si presionamos el botón Guardar, descargara un archivo con el nombre que aparece arriba de la Entrada, y quedara guardado en nuestro dispositivo.

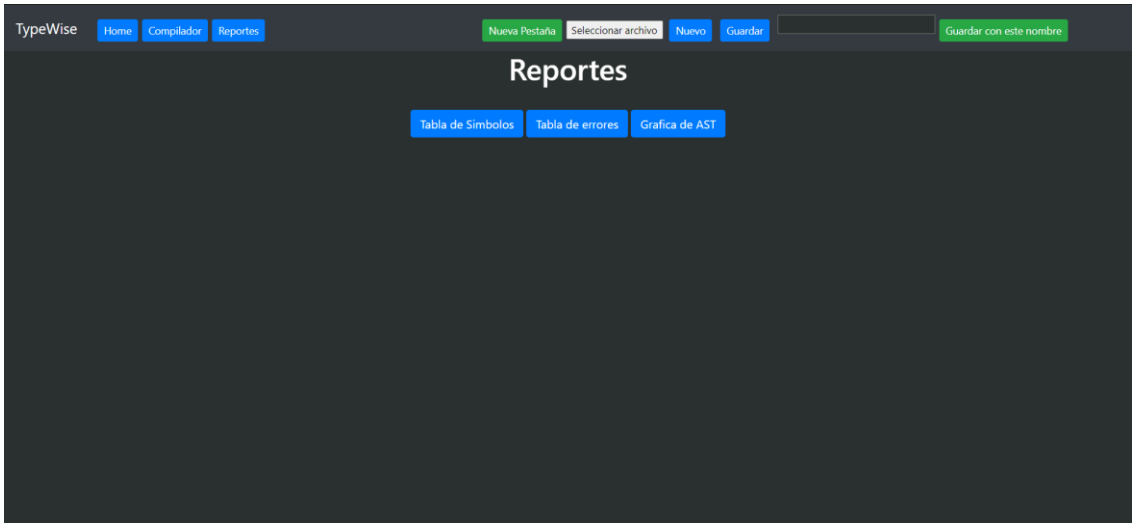
Ejecutar

En la parte del compilador, podemos realizar solo 1 acción la cual es ejecutar, esta opción analizará la información que este en el cuadro de entrada, y la respuesta de esa información podrá ser observada en cuadro que se encuentra en la parte derecha el cual es el cuadro de salida. Toma el valor del textarea que y lo envía a través de una petición al servidor. El servidor ejecuta el parser y obtiene un árbol sintáctico, el cual se recorre y se ejecuta. También obtiene los nodos de las graficas y como respuesta se envía al navegador un json el cual posee la salida del programa, y el código de las 3 graficas. AST, Errores y tabla de símbolos.

Reportes

Si presionamos el botón de reportes, este nos mostrara una ventana en la que podemos observar 3 botones, los cuales son:

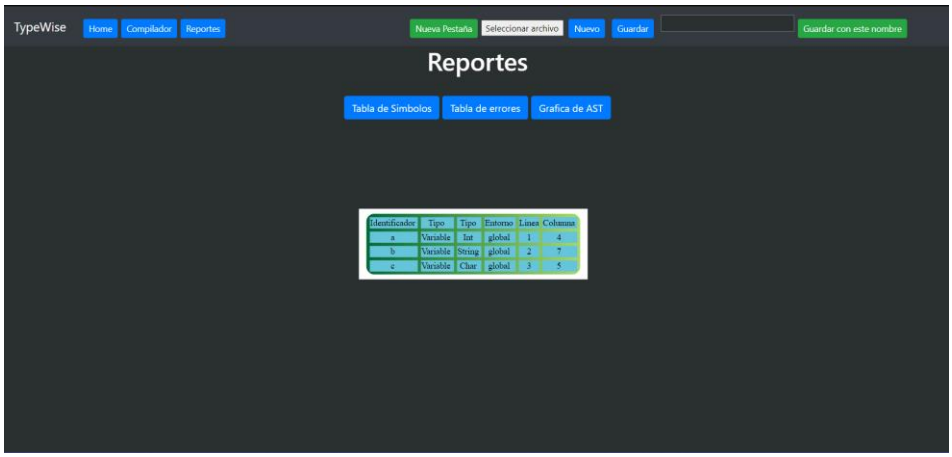
1) Tabla de Símbolos
2) Tabla de errores
3) Grafica de AST



Para poder muestra la ventana de reportes se oculta todo el div del compilador y se muestra el div de reportes, eso para evitar cargar de nuevo la pagina y se pierdan las pestañas ya abiertas.

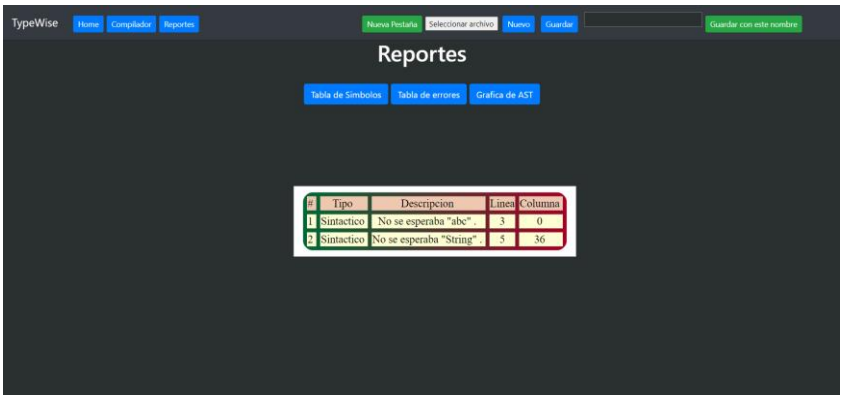
1) Tabla de símbolos

Este botón tomará el código de nuestra gráfica, y la convertirá en una imagen la cual será mostrada de manera inmediata en nuestra pantalla, esta imagen representa la tabla de todas las variables y funciones que hemos declarado.



2) Tabla de errores

Este botón tomara el código de nuestra gráfica, y la convertirá en una imagen la cual será mostrada de manera inmediata en nuestra pantalla, esta imagen representa la tabla de todos los errores que puedan existir en nuestro código.



2) Grafica de AST

Este botón tomará el código de nuestra gráfica, y la convertirá en una imagen la cual será mostrada de manera inmediata en nuestra pantalla, esta imagen representa el árbol de análisis sintáctico, el cual posee la estructura de todas las instrucciones correctas que declaremos en nuestro código.