Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Organización de Lenguajes y Compiladores 1 N
Aux Emely García
Maynor Octavio Piló Tuy
201531166

SYSCOMPILER

Manual Técnico

Descripción

SYSCOMPILER es un lenguaje de programación que consiste en un lenguaje de programación para que los estudiantes, del curso de Introducción a la Programación y Computación 1, aprendan a programar y tener conocimiento de todas las generalidades de un lenguaje de programación. Cabe destacar, que este lenguaje será utilizado para generar sus primeras prácticas de laboratorio del curso antes mencionado. Y para la programacion en general .

Requerimientos mínimos del sistema

El sistema debe de tener los siguientes requerimientos instalados:

- Windows 7,8,10
- graphviz-2.49.3 (64-bit)
- nodejs v14.17.6
- jison v0.4.18

1. **Funcionamiento de la aplicación:**
   El programa se maneja con una arquitectura Cliente-Servidor, con el objetivo de que pueda separar los servicios administrados por el intérprete, de la aplicación cliente que se mostrará al usuario final.

Arquitectura Cliente Servidor.

1.1 Entornos de trabajo:

El programa cuenta con los siguientes entornos de trabajo.

- Editor: proporciona ciertas funcionalidades, características y herramientas que serán de utilidad al usuario. La función principal del editor será el ingreso del código fuente que será analizado. En este se podrán abrir diferentes archivos al mismo tiempo y deberá mostrar la línea actual.

- Funcionalidades: el programa tiene las siguientes funcionalidades; Crear archivos: El editor deberá ser capaz de crear archivos en blanco. Abrir archivos: El editor deberá abrir archivos .sc . Guardar el archivo: El editor deberá guardar el estado del archivo en el que se estará trabajando. Eliminar pestaña: permitirá cerrar la pestaña actual.

- Características: Múltiples Pestañas: crear nuevas pestañas con la finalidad de ver y abrir los archivos de prueba en la aplicación.

- Herramientas: Ejecutar: hace el llamado al intérprete, el cual se hace cargo de realizar los análisis léxico, sintáctico y semántico, además de ejecutar todas las sentencias.

- Reportes: Reporte de Errores: muestra todos los errores encontrados al realizar el análisis léxico, sintáctico y semántico. Generar Árbol AST (Árbol de Análisis Sintáctico): se genera una imagen del árbol de análisis sintáctico que se genera al realizar los análisis. Reporte de Tabla de Símbolos: Se muestran todas las variables, métodos y funciones que han sido declarados dentro del flujo del programa.

- Área de consola En esta área se mostrarán los resultados, mensajes y todo lo que sea indicado dentro del lenguaje.

2. Descripción del Lenguaje Syscompiler:

Case insensitive	El lenguaje no distingue entre mayúsculas o minúsculas.
Comentarios	Comentarios de una línea Estos comentarios deberán comenzar con // y terminar con un salto de línea. Comentarios de una línea Estos comentarios deberán comenzar con /* y terminar con */.
Tipos de Datos	Los tipos de dato que soportará el lenguaje en concepto de un tipo de variable se definen a continuación: Entero, Booleando, Cadena, Carácter, Double.
Operadores Aritméticos	Suma, Resta, Multiplicación, División, Potencia, Módulo, Negación Unaria.
Operadores Relacionales	Igualación , Diferenciación, Menor que , Menor o igual que, Mayor que, Mayor o igual que.
Operador Ternario	Este operador se le considera como un atajo para la instrucción 'if'.
Operadores Lógicos	OR , AND, NOT.
Declaración y asignación de variables	Una variable deberá de ser declarada antes de poder ser utilizada. Todas las variables tendrán un tipo de dato y un nombre de identificador. Las variables podrán ser declaradas global y localmente.
Casteos	El lenguaje aceptará los siguientes casteos: Int a double Double a Int Int a String Int a Char Double a String char a int Char a double
Incremento y Decremento	Los incrementos y decrementos nos ayudan a realizar la suma o resta continua de un valor de uno en uno.
Vectores	Los vectores son una estructura de datos de tamaño fijo que pueden almacenar valores de forma limitada, y los valores que pueden almacenar son de un único tipo; int, double, boolean, char o string.

Listas Dinámicas	Las listas son una estructura de datos que pueden crecer de forma iterativa y pueden almacenar hasta N elementos de un solo tipo; int, double, boolean, char o string.
if	La sentencia if ejecuta las instrucciones sólo si se cumple una condición. Si la condición es falsa, se omiten las sentencias dentro de la sentencia.
Switch Case	Switch case es una estructura utilizada para agilizar la toma de decisiones múltiples, trabaja de la misma manera que lo harían sucesivos if.
While	El ciclo o bucle While, es una sentencia que ejecuta una secuencia de instrucciones mientras la condición de ejecución se mantenga verdadera.
For	El ciclo o bucle for, es una sentencia que nos permite ejecutar N cantidad de veces la secuencia de instrucciones que se encuentra dentro de ella.
Do-While	El ciclo o bucle Do-While, es una sentencia que ejecuta al menos una vez el conjunto de instrucciones que se encuentran dentro de ella y que se sigue ejecutando mientras la condición sea verdadera.
Break	La sentencia break hace que se salga del ciclo inmediatamente, es decir que el código que se encuentre después del break en la misma iteración no se ejecutara y este se saldrá del ciclo.
Continue	La sentencia continue puede detener la ejecución de la iteración actual y saltar a la siguiente. La sentencia continue siempre debe de estar dentro de un ciclo, de lo contrario será un error.
Return	La sentencia return finaliza la ejecución de un método o función y puede especificar un valor para ser devuelto a quien llama a la función.
Funciones	las funciones serán declaradas definiendo primero su tipo, luego un identificador para la función, seguido de una lista de parámetros dentro de paréntesis.
Métodos	los métodos serán declarados haciendo uso de la palabra reservada 'void' al inicio, luego se indicará el identificador del método, seguido de una lista de parámetros dentro de paréntesis.

Llamadas metodos	La llamada a una función específica la relación entre los parámetros reales y los formales y ejecuta la función.

3. Principales Clases, funciones y métodos de la aplicación.

Paquete:		**Error**
Carpeta que contiene las clases de guardar y recibir los errores dectectados durante la gramatica.		
Clases	Error.js	Clase que contene el objeto error.
	Lista_errores	Array de errores detectador durante la ejecucion del codigo fuente.

Paquete:		**Expresion**
Fuente utilizada para las expresiones que se ingresen del codigo fuente.		
Clases	Acceso.ts	Clase generada para acceder a las variables guardadas.
	AccesoLista.ts	Clase generada para acceder a una lista que se ha guardado anteriormente.
	AccesoVector.ts	Clase generada para acceder a un vector que se haya guardado.
	Aritmetica.ts	Clase utilizada para realizar todas las operaciones aritméticas que soporta el lenguaje.
	Expresion.ts	Clase que se encarga de operar las expresiones que recibe el lenguaje.
	Ind_dec.ts	Clase encargada de realizar el auto incremental de una variable o el decremento de la misma.
	Logicos.ts	Clase que se encarga de las operaciones logicas que soporta el lenguaje.
	Primitivo.ts	Clase que se encarga de verificar los datos que se ingresan y que soporta el lenguaje.
	Relacional.ts	Clase encargada de las operaciones relacionales que soporta el lenguaje.
	Retorno.ts	Clase que contiene los tipos de datos que soporta el lenguaje.

Paquete:		**Instrucción**
Paquete utilizado para realizar todas las instruciiones		
Clases	Asignar.ts	Clase encargada de asignar variables, listas y vectores declaradas en el codigo fuent.
	AsignarLsitas.ts	Clase que se encarga de asignar listas declaradas en el codigo fuente.
	AsignarVector.ts	Clase que se encarga de asignar vectores declaradas en el codigo fuente.
	Break.ts	Clase que contiene la funcion break.
	Casteo.ts	Clase encargada de realizar el casteo declarado en el codigo fuente y soportado por el lenguaje.
	Continue.ts	Clase que contiene la funcion continue.
	Declararar.ts	Clase encargada de declarar las variables que se ingresan en el codigo fuente.
	Declararlista.ts	Clase que se encarga de declarar listas.
	DeclrararVector.ts	Clase encargada de declarar vectores.
	DoWhile.ts	Clase que realiza el proceso de ciclo DoWhile.
	For.ts	Clase que realiza el proceso de ciclo For.
	Funcion.ts	Clase que encargada de guardar funciones declaradas y metodos declarados.
	If.ts	Clase que realiza el proceso de la sentencia if.
	Instrucción.ts	Clase que declara codigo de fuente como instrucciones.
	Return.ts	Clase encargada de realizar la funcion Return.
	StartWith	Clase principal con que se inicializa la ejecución del Código

Paquete:		**Mas**
Contiene las clases y objetos en donde se guardan las variables, funciones, metodos que se ingresan en el codigo fuente.		
Clases	**Ambito.ts**	Clase que se encarga de verificar el ámbito de cada variable ingresada. Y la recuperación de variables.
	ListaTabla.ts	Se encarga de guardar todas las variables, métodos, funciones que se ingresan.

	Simbolo.ts	Es el objeto que contiene los símbolos que se ingresa en el Código fuente.