DESARROLLO DE UN LENGUAJE DE PROGRAMACIÓN CON FINES EDUCATIVOS BASADO EN LA FILOSOFÍA DE LOGO

YEHISON FABIÁN BECERRA RODRÍGUEZ
OSCAR ALBERTO CARRILLO ROZO

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FISICO-MECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2008

DESARROLLO DE UN LENGUAJE DE PROGRAMACIÓN CON FINES EDUCATIVOS BASADO EN LA FILOSOFÍA DE LOGO

YEHISON FABIAN BECERRA RODRÍGUEZ
OSCAR ALBERTO CARRILLO ROZO

Proyecto de Grado para optar al Título de
Ingeniero de Sistemas

Director
Ing. JORGE HERNANDO RAMÓN SUÁREZ

Codirectora
Dra. MARTHA VITALIA CORREDOR MONTAGUT

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FISICO-MECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2008

A mi madre, que siempre estuvo a mi lado y me brindó toda su ayuda

A mi tía Edilia, que ha sido como una segunda madre en mi vida

A mi hermana, que ha estado presente en momentos difíciles

A mi padre, por su apoyo

Y a mis amigos incondicionales.

Yehisón Becerra

A Dios por brindarme la fortaleza de concluir este proyecto

A mi esposa, mi compañera incansable en el caminar de la vida.

A mi familia, por su apoyo constante

A mis amigos, que han estado en momentos difíciles

Oscar Carrillo

AGRADECIMIENTOS

Los autores expresan sus agradecimientos a:

Ing. Jorge Ramón Suárez, Director del proyecto, por su constante empeño para que termináramos este proyecto.

Dra. Martha Vitalia Corredor, codirectora del proyecto, por sus sugerencias y correcciones para entregar un buen trabajo de grado.

Escuela de Ingeniería de Sistemas – UIS, por la formación íntegra como ingenieros.

D.I. Diana Rueda – D.I. Santiago Aguillón, por su aporte en el diseño gráfico del software resultado de este proyecto.

Grupo E.R.A. – UIS, por su aporte en la construcción del móvil físico necesario en el proyecto.

Todas aquellas personas que de una u otra forma brindaron su apoyo para la culminación de este proyecto.

Yehison Becerra expresa sus agradecimientos a:

Dios por todas las oportunidades y bendiciones brindadas.

Mis padres por su apoyo. Sobre todo a mi madre, que ha sido la persona que ha estado a mi lado en las buenas y en las malas y ha soportado mis enojos y mi terquedad.

Mi tía Edilia por todas las atenciones recibidas y por ese calor maternal.

Mi hermana y a mi prima, que siempre estuvieron a mi lado.

Yurani, que durante la carrera fue mi compañera sentimental, y tuvo que soportarme por largo tiempo.

Mi nona Argemira, nona Matilde y nono Cipriano, por el gran amor que me brindan.

Mi tío Carlos, que me ha dado la mano cuando la he necesitado.

Tatiana y a su familia, que durante los últimos años me hicieron sentir parte de la familia de ellos con su apoyo incondicional.

Mis familiares, que han estado pendientes de mis estudios y, por ende, de mi vida.

Mis grandes amigos de parranda: Darwin, Samuel y Gerson.

Mis grandes amigos de estudio y trabajo: Oscar Murillo y Sergio Gélvez.

Mi compañero de grado: Oscar Carrillo, por su gran amistad y por haber logrado superar las barreras de distancia para culminar esta etapa.

Todas las personas que de una manera u otra, han dejado huella en mi vida y también, a las personas que me han complicado la vida, ya que me han hecho una mejor persona.

Oscar Carrillo expresa sus agradecimientos a:

Mis tíos Pedro, Geña, Helena y Martha por la formación humana que me han brindado y por su apoyo durante el transcurso de mi carrera.

Rubiela, mi esposa, por ser el ángel que me cuida y me inspira para hacer bien las cosas.

Mi nona Socorro por la confianza depositada en mí y brindarme su apoyo.

Demás familiares, por darme ánimo y fuerzas.

Mis compañeros de trabajo y amigos, quienes a lo largo de mi carrera han dejado su rastro, para hacerme un mejor profesional.

TABLA DE CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

LISTA DE ANEXOS

TÍTULO:

DESARROLLO DE UN LENGUAJE DE PROGRAMACIÓN CON FINES EDUCATIVOS BASADO EN LA FILOSOFÍA DE LOGO*

AUTORES:
BECERRA RODRÍGUEZ, Yehison Fabián
CARRILLO ROZO, Oscar Alberto**

PALABRAS CLAVE:
Compilador, Intérprete, Logo, DirectX, Prototipos, Lenguaje, Educación, Robot

DESCRIPCIÓN:

Este documento describe el proceso de investigación realizado para implementar un Lenguaje de Programación estructurado, orientado hacia el ambiente educativo y su respectivo intérprete. El Lenguaje de Programación Educativa (LPE) desarrollado cuenta con un entorno de programación, que permite la ejecución de los programas diseñados en un simulador 3D y a su vez establece comunicación con un móvil físico inalámbrico a través del puerto serial. La base gramatical para la construcción de LPE es el lenguaje de programación LOGO que por sus características pedagógicas es ideal en el ambiente educativo.

La metodología de desarrollo que se siguió fue Evo, que permitió la entrega de prototipos que fueron evolucionando hasta llegar al producto final; el primer prototipo fue un analizador que reconocía la gramática del lenguaje implementado, el segundo y tercer prototipo fueron el entorno de visualización en 2D y 3D respectivamente, la cuarta entrega añadió un módulo de comunicación a través del puerto serial y finalmente se implementó un entorno gráfico orientado hacia la población estudiantil entre 7 y 12 años.

Por medio de este proyecto se consiguió unificar las fortalezas de las áreas de Ingeniería de Sistemas, Ingeniería Electrónica y Diseño Industrial. Se recomienda que para futuras mejoras, se implementen metodologías de enseñanza para los diferentes niveles educativos y se logre difundir el uso del software resultado del presente trabajo.

---

* Proyecto de grado en modalidad de Investigación

** Facultad de Ingenierías Fisico-Mecánicas, Escuela de Ingeniería de Sistemas e Informática. RAMÓN SUÁREZ, Jorge Hernando

TITLE:

DEVELOPMENT OF A PROGRAMMING LANGUAGE FOR EDUCATIONAL PURPOSES BASSED ON THE LOGO PHILOSOPHY[*]

AUTHORS:
BECERRA RODRÍGUEZ, Yehison Fabián
CARRILLO ROZO, Oscar Alberto[**]

KEYWORDS:
Compiler, Interpreter, Logo, DirectX, Prototypes, Language, Education, Robot

DESCRIPTION:

This document describes the research process undertaken to implement a structured programming language, oriented to the educational environment and their interpreter. The developed language "Lenguaje de Programación Educativa" (LPE) has a programming environment that allows the execution of designed programs in a 3D simulator and provides physical communication with a wireless mobile through the serial port. The basis for the grammatical construction of LPE is the programming language Logo which by its nature is ideal for teaching in the educational environment.

The followed development methodology was Evo, which allowed the delivery of prototypes that were moving up to the final product, the first prototype was an analyzer that recognized the grammar of the language implemented, the second and third were the prototype environment visualization 2D and 3D respectively, the fourth delivery added a communication module through the serial port and finally implemented a graphical environment-oriented student population between 7 and 12 years.

Through this project was unified the strengths of the areas of Systems Engineering, Electronic Engineering and Industrial Design. It is recommended that for future improvements are implemented teaching methodologies for different levels of educational attainment and it will achieve the use of the software result of this work.

---

[*] Graduation Project in the category of Investigation

[**] Physical – Mecanical Engineering's Department, Systems and Informatics Engineering School. RAMÓN SUÁREZ, Jorge Hernando

INTRODUCCION

Cada día el uso de computadores es más frecuente, lo que conlleva a la aparición de más problemas a solucionar computacionalmente (refiriéndose como problema a situaciones de estudio y no solamente a una situación negativa). En el ámbito de la educación no ocurre algo diferente, pues mediante diversas investigaciones se ha logrado mostrar que los niños refuerzan su aprendizaje cuando este proceso se desarrolla con apoyo en los juegos.

El uso racional de las tecnologías de la información y la comunicación (TICs) en el ambiente educativo permite a los niños reforzar y ampliar sus conocimientos, puesto que se da espacio a su imaginación y creatividad a través del uso de herramientas computacionales, como el software educativo y la robótica básica.

Al hablar de estas dos áreas de desarrollo de la computación para la educación, cabe resaltar que la primera es de la que más uso se ha podido visualizar en el país, pero contrario a lo que se podría suponer, no ha tenido una base muy robusta sobre la cual estandarizar las aplicaciones existentes.

En el área educativa la robótica no ha experimentado mayor desarrollo en comparación con el área industrial, tal vez sea porque no se contaba con elementos de fácil manejo para que los alumnos de nivel primario y secundario pudieran realizar sus proyectos de una manera sencilla, sin especiales conocimientos en la electrónica, problema que ha estado desapareciendo con el surgimiento de interfaces del computador para el control de elementos externos. En este sentido,

> "Cuando se habla de robótica educativa, se refiere a que sean los propios usuarios (especialmente niños) los que construyan sus modelos, y los hagan funcionar. Durante muchos años LOGO fue un lenguaje que cumplía con las características de un lenguaje educativo en el aula, procurando que los niños movieran distintos elementos en la pantalla, animando diversas figuras, etc. Ese desarrollo estuvo impulsado por la importancia pedagógica que se le brinda a dicha herramienta, como elemento motivador del desarrollo cognoscitivo"[1].

Sin embargo, con los avances tecnológicos actuales no es posible quedarse en esta etapa, hay que ir más allá y pensar no sólo en construir proyectos que muestren imágenes en pantalla, sino lograr darles vida de forma que el estudiante logre apreciar sus avances de una manera real.

[1] http://www.argos.edu.uy/fundrobotica.htm

Si un niño se emociona al ver culminado un dibujo (tarea de clase) o cuando logra pintar en pantalla, ¿cómo se sentirá al ver que es capaz de manipular un elemento físico y lograr que se mueva a su gusto?

Es por lo anterior, que el desarrollo de este proyecto de Ingeniería de Sistemas y otros dos proyectos de Ingeniería Electrónica y Diseño Industrial, pretenden profundizar en el campo de la robótica educativa, esto es, situar al alcance de niños y jóvenes una herramienta software que controle dispositivos externos a la computadora (dispositivo físico, real) de acuerdo a las órdenes que se le indiquen y así colaborar en parte en el desarrollo educativo de ellos que son el futuro de una sociedad con deseos de desarrollarse.

En el documento aquí presentado se muestra lo más relevante del proceso de desarrollo del Lenguaje de Programación Educativo – LPE, como es un resumen de las referencias teóricas necesarias para la comprensión del proceso de creación de un intérprete basado en la filosofía LOGO con un entorno de ejecución en 3D, la metodología para la construcción del producto software, los resultados obtenidos y por último las conclusiones y recomendaciones obtenidas durante el desarrollo del mismo.

1. PRESENTACIÓN DEL PROYECTO

1.1 OBJETIVOS

1.1.1 Objetivo general

Desarrollar un intérprete de comandos en español, que facilite la interacción con un móvil lógico y uno físico, orientado a enseñar programación mediante un enfoque lúdico.

1.1.2 Objetivos específicos

1.1.2.1. Generar un intérprete de un lenguaje de programación estructurado capaz de resolver expresiones numérico-lógicas, que acepte la creación de rutinas con parámetros de entrada / salida y sentencias de condición y bucles, para el desarrollo de la creatividad, imaginación e inteligencia de los usuarios que desean ingresar al campo de la programación, de modo que los algoritmos implementados por medio de este lenguaje se conviertan en movimientos planeados de un móvil físico.[2]

1.1.2.2. Diseñar e implementar un entorno gráfico que simule el comportamiento del móvil físico, de manera que éste no sea necesario al momento de validar el código generado por el usuario, logrando así previsualizar los movimientos que realizará el móvil en el momento de su ejecución.

1.1.2.3. Crear una interfaz de comunicación entre el computador y el móvil físico, para lograr que este último se comporte como el usuario especifica en el código mediante el envío de mensajes por puerto.

[2] Por ejemplo un móvil en forma de tortuga, o carro, o robot objeto de otro proyecto

1.2 DESCRIPCIÓN DEL PROYECTO

En el ámbito de la educación virtual, se han logrado muchos desarrollos y avances que encajan con las necesidades de estudio actuales, pero la educación por PC tiene una historia más exquisita que pocos conocen, así que regresando en el tiempo, en una época donde no existía tanta tecnología, surgió LOGO, una herramienta capaz de obedecer órdenes de los usuarios y representar las órdenes en la pantalla en un entorno gráfico.

Se podría considerar a LOGO como un lenguaje de programación, sin embargo, no es tan complejo como los lenguajes conocidos, pero que proporciona beneficios desde el punto de vista educativos; este lenguaje proporciona a los estudiantes un ambiente matemático que favorece el desarrollo de habilidades para analizar y resolver problemas, y la introducción al mundo de la programación y geometría, en un entorno que motiva la creatividad y la intuición.

El software que se desarrolló se fundamentó en la filosofía del lenguaje LOGO, dado que hasta ahora ha sido un lenguaje utilizado con éxito en la educación. Sin embargo, los comandos y las instrucciones cambiaron teniendo en cuenta algunos aspectos del LOGO que han logrado motivar a niños y jóvenes en el uso del computador, como es el uso de un entorno gráfico.

Como resultado del proyecto se buscó ofrecer un software que no sólo pretende familiarizar al usuario con los lenguajes de programación, sino ofrecer un entorno que permita mejorar la ubicación espacial (especialmente en niños), desarrollar la creatividad y afianzar los conocimientos geométricos.

El entorno gráfico se mejoró y estuvo orientado por un proyecto de grado de la Escuela de Diseño Industrial[3] de la Universidad Industrial de Santander, porque se busca que el usuario sienta que está jugando y no programando, pues con los trabajos en LOGO se han obtenido muy buenos resultados ya que se aprende jugando. Adicionalmente con las gráficas se busca que el objeto “móvil”, pueda ser cambiado por el usuario, así como el espacio en donde éste transita. Estas características hacen justificable el desarrollo del proyecto puesto que se busca sea un aporte importante a la educación.

De otro lado, el sistema también crea un lazo con el mundo exterior, pues se implementan algunas de las funciones del móvil lógico para operar con uno físico que estuvo a cargo del grupo de Electrónica Recreativa Avanzada – ERA de la Universidad Industrial de Santander, de forma que el usuario, no sólo ve una animación de su trabajo, sino que puede comprobar que en realidad está

[3] AGUILLÓN VESGA Edgar Santiago y RUEDA TRIANA Diana Marcela DISEÑO Y CONSTRUCCIÓN DE UN ROBOT MÓVIL NO AUTÓNOMO, PROGRAMABLE POR MEDIO DE UN SOFTWARE BASADO EN LA FILOSOFÍA DE LOGO. [Libro]. - 2007.

consiguiendo los logros que se propuso antes de iniciar una tarea específica, situación que le permite elevar su autoestima, lo que agrega valor al producto como herramienta de apoyo en la enseñanza y al aprendizaje.

Así, el proyecto permite motivar e ingresar al usuario en el mundo de la electrónica educativa (robótica) familiarizándolo con funciones básicas para mover un objeto físico a través de órdenes en un lenguaje que puede recordar.

2. MARCO CONCEPTUAL

2.1 COMPILADORES E INTÉRPRETES

2.1.1 Procesadores de lenguaje

Existe un vacío natural en la comunicación entre el hombre y la maquina. El hardware de los computadores opera en un nivel muy automático en términos de bits y registros, mientras que las personas tienden a expresarse en términos de lenguaje natural como el español o en notación matemática. Este vacío en la comunicación es usualmente llenado por medio de un lenguaje artificial que permite al hombre expresarse con un conjunto de palabras bien definido, sentencias y fórmulas que pueden ser entendidas por un computador. Para conseguir esta comunicación, el hombre debe poseer un manual de usuario, que explica los componentes y significados permitidos por el lenguaje, y el computador poseerá un software por medio del cual puede tomar sartas de caracteres que representan los comandos o programas escritos en un lenguaje de programación y traducir esta entrada internamente en unos patrones de bits requeridos para llevar a cabo las intenciones del usuario.

Los lenguajes de computación varían ampliamente en complejidad. Pueden encontrarse, por ejemplo:

- Lenguajes de máquina que son interpretados por el hardware o microprogramas de la máquina.
- Los lenguajes ensamblador o de “bajo nivel” que en gran parte son un espejo del conjunto de instrucciones de un computador particular.
- Los lenguajes de comandos que son usados para interactuar con un sistema operativo.
- Lenguajes de alto nivel, como FORTRAN, C ++, JAVA, etc., los cuales tienen una estructura compleja y no dependen del conjunto de instrucciones o sistema operativo de una máquina en particular.

El término “procesador de lenguaje” describe a los programas que permiten que el computador entienda los comandos y entradas suministradas por el usuario. Al respecto, existen dos tipos de programas procesadores de lenguaje: intérpretes y traductores[4].

[4] LEWIS II, P., ROSENCRANTZ, D. J. et al. Compiler Design Theory. Addison Wesley. USA. 1978

Un *intérprete* es un software que acepta como entrada un programa escrito en un lenguaje de computador llamado *lenguaje fuente* y realiza las operaciones indicadas por el programa.

Un *traductor* es un software que acepta como entrada un programa escrito en un *lenguaje fuente* y produce como salida otra versión de ese programa escrito en otro lenguaje llamado *lenguaje objeto.* Usualmente el lenguaje objeto es el lenguaje de máquina de algún computador. Los traductores están principalmente divididos en *ensambladores* que traducen lenguajes de bajo nivel a código binario que entiende la máquina y en *compiladores* que traducen lenguajes de alto nivel en lenguaje de bajo nivel (ensamblador).

La construcción de compiladores e intérpretes nace ante la necesidad de establecer una comunicación con los dispositivos internos de cómputo de manera efectiva y con el menor consumo de tiempo, dado que comenzaban a surgir computadores con fines comerciales. Esto sucede a finales de la década de los 50's y su nacimiento viene acompañado con el de lenguajes de alto nivel como FORTRAN y COBOL, que permitían realizar esta tarea. Con estos lenguajes nacen los compiladores.

Un compilador se define como un traductor que facilita la comunicación entre el programador y la máquina, por medio de un proceso de transformación. Durante este proceso, el compilador lee un programa escrito en un lenguaje (fuente), y lo traduce a un programa equivalente en otro lenguaje (objeto).

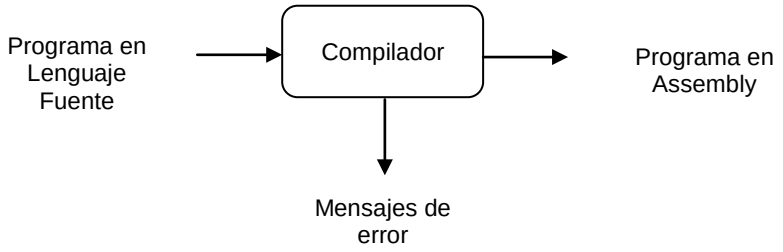


Figura 1. Proceso de transformación en un Compilador[5]

[5] AHO Alfred V., SETHI Ravi y ULLMAN Jeffrey Compilers: Principles, Techniques, and Tools [Libro]. - China : Addison Wesley, Pearson Education, Inc, 1986.

2.1.2 El contexto de un compilador

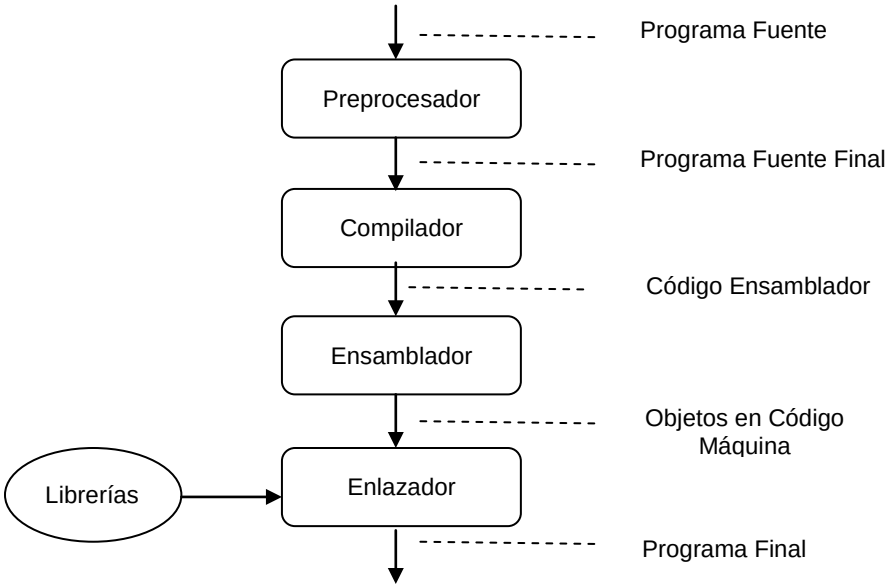


Figura 2. Contexto de un Compilador[6]

Para crear un programa destino ejecutable, además de un compilador, pueden ser necesarios otros programas como:

- *Preprocesador*: Que se encarga de recopilar el programa fuente que se encuentre disperso en diferentes módulos y expandir las funciones convirtiéndolas en instrucciones ejecutables.

- *Ensamblador*: Necesario para poder obtener el programa final en código binario que entienda el computador, de tal manera que sea ejecutable.

- *Enlazador*: Esta herramienta toma las direcciones de los objetos compilados en la máquina como librerías y las incluye en el nuevo código para crear el nuevo programa ejecutable.

2.1.3 Fases de un compilador

Los procesos llevados a cabo durante la compilación pueden clasificarse en dos grandes etapas, que son el “Análisis y Síntesis”.

[6] AHO Alfred V., SETHI Ravi y ULLMAN Jeffrey Compilers: Principles, Techniques, and Tools [Libro]. - China : Addison Wesley, Pearson Education, Inc, 1986.

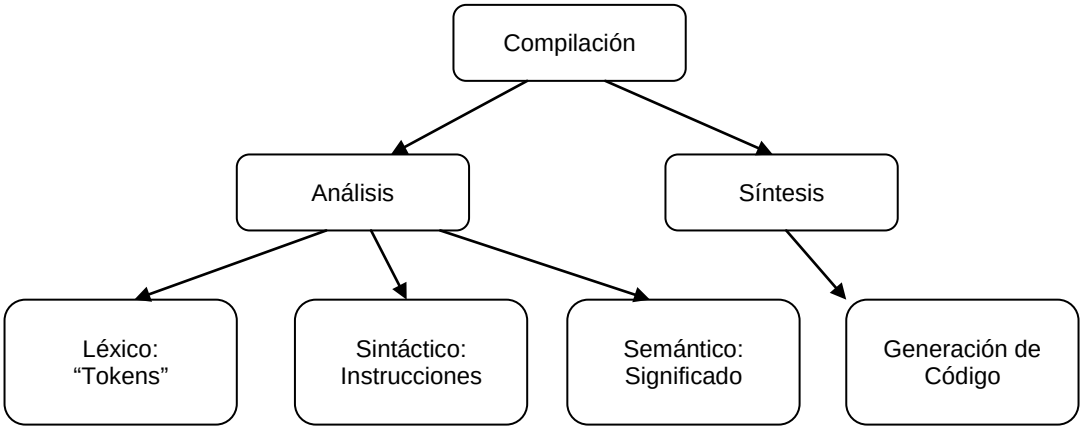


Figura 3. Etapas de la Compilación[7]

1. ***Etapa de análisis***: Durante esta etapa se divide el programa en lexemas reconocidos dentro de la gramática del lenguaje y se verifica que estén en el orden esperado y sean coherentes

   a. *Análisis léxico*: divide el código fuente en los componentes léxicos o "tokens". Cada token es una pequeña estructura de datos que indican el comienzo, longitud, fin, tipo y valor de una "*palabra*" significativa en el texto o lenguaje de programación.

   Si se toma como ejemplo la entrada:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
g	i	r	a	d	e	r	e	c	h	a		9	0	\	n
17	18	19	20	21	22	23	24	25	26	27					
a	v	a	n	z	a		1	0	\	n					
28	29	30	31	32	33	34	35	36							
b	a	j	a	l	a	p	i	z							

   Este código puede ser tokenizado por un analizador léxico de la siguiente manera:
   - Desde el punto de vista del analizador léxico, `giraderecha` es un identificador, comienza en la posición 1 y tiene 11 caracteres de longitud.
   - Un espacio en blanco comienza en la posición 12.
   - Un número comienza en la posición 13 y tiene 2 caracteres de longitud.
   - Una nueva línea comienza en la posición 15 y tiene 2 caracteres de longitud (es representado por \n).

[7] GONZÁLEZ Michael y GUTIERREZ Javier Introducción a la Tecnología de Compiladores [En línea]. - 17 de Enero de 2008. - http://www.ctr.unican.es/asignaturas/lan/compila-2en1.pdf.

- `avanza` es un identificador que comienza en la posición 17 y tiene 6 caracteres de longitud.
- Un espacio en blanco comienza en la posición 23.
- Un número comienza en la posición 24 y tiene 2 caracteres de longitud.
- Una nueva línea comienza en la posición 26 y tiene 2 caracteres de longitud.
- `bajalapiz` es un identificador que comienza en la posición 28 y tiene 9 caracteres de longitud.

b. *Análisis sintáctico*: agrupa los tokens reconocidos en instrucciones que hagan parte de la gramática del lenguaje, revisa que el orden en que estén los lexemas sea el correcto, y esto lo hace con la ayuda de las expresiones regulares y más formalmente con la definición del lenguaje en BNF (Backus Naur Form).

   La notación BNF describe la estructura gramática de los programas que están sintácticamente correctos, siendo de gran utilidad para el programador por tener a mano una forma precisa y ágil de la especificación del lenguaje y también por brindarle la posibilidad de desarrollar herramientas que generen los analizadores basándose en la lectura del BNF.

c. *Análisis semántico*: verifica que las instrucciones evaluadas durante el análisis sintáctico posean un significado válido, generalmente se trata de comprobar los tipos de datos y sus restricciones en las operaciones que se realicen.

Las etapas de análisis léxico y sintáctico de un compilador también son utilizadas en otras aplicaciones como por ejemplo: los editores de texto presentes en los IDE (Entorno de Desarrollo Integrado) de los lenguajes de alto nivel que actualmente permiten revisar el código a medida que se digita y los correctores ortográficos y gramaticales que hacen parte de las suites de oficina estilo Microsoft Word.

2. ***Etapa de síntesis***: con base en las instrucciones internas del compilador, se genera el programa ejecutable para el sistema operativo y/o arquitectura destino.

   a. *Generación de código intermedio*: las instrucciones analizadas en la etapa de análisis se convierten en código más fácil de llevar al programa objeto. Una forma posible es “el código de tres direcciones” que consiste en llevar todas las instrucciones a sentencias de máximo tres operandos.

b. *Optimización de código*: se revisa el código intermedio en busca de sentencias, que al reescribirlas, produzcan un programa objeto que se ejecute en menos tiempo.

c. *Generación de Código*: en esta fase se genera el código objeto, que la mayoría de veces es código ensamblador o código de máquina.

En la figura 4 se muestran las distintas fases que componen las etapas de análisis y síntesis del proceso de compilación.

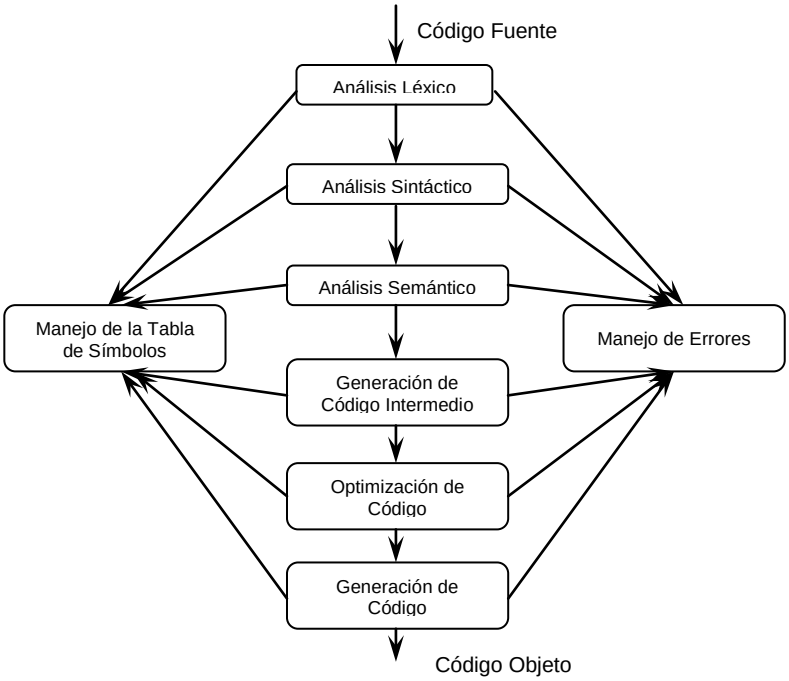


Figura 4. Fases de un Compilador[8]

2.2 LENGUAJE LOGO

Dentro del estudio de las relaciones entre la informática y el desarrollo cognitivo es conveniente dedicar especial atención a LOGO. Este ambiente de aprendizaje ha creado muchas expectativas y generado muchos interrogantes entre los que se interesan por los usos educativos del computador.

[8] AHO Alfred V., SETHI Ravi y ULLMAN Jeffrey Compilers: Principles, Techniques, and Tools [Libro]. - China : Addison Wesley, Pearson Education, Inc, 1986.

Es importante revisar cuales han sido las circunstancias bajo las cuales se han dado los éxitos o fracasos de este lenguaje de programación, de modo que se pueda obtener provecho de las experiencias vividas por otros países en la aplicación de este ambiente y analizar los beneficios para la educación Colombiana al apoyarse en LOGO.

2.2.1 Historia y evolución

Nació en los años 70, creado por Seymour Papert[9] en el laboratorio de Inteligencia Artificial del Instituto Tecnológico de Massachusetts de los Estados Unidos de América. Es llamado “El lenguaje para aprender". El objetivo de este lenguaje era introducir a los niños en los conceptos de programación para entender la lógica y el funcionamiento de las máquinas y conocer cómo se comportan, usando un lenguaje sencillo y casi natural, que facilitara a su vez la labor del educador. Se diseñó para ser fácil de aprender, fácil de usar y fácil de leer, pero tan flexible y potente que pudiera afrontar problemas complejos.

El uso extendido de Logo empezó con la llegada de los computadores personales a finales de los 70’s. Se reforzó con la publicación en 1980 del libro de Seymour Papert “*Mindstorms”*, lo que provocó que miles de maestros alrededor del mundo se entusiasmaran por el potencial intelectual y creativo de Logo. Este entusiasmo alimentó el auge de Logo a inicios de los 80’s.

Según su creador, LOGO debe considerarse como una nueva modalidad o filosofía educativa que propicia el ambiente de aprendizaje, no como un simple lenguaje de programación. Logo ha cambiado con el paso del tiempo en busca de mantenerse en medio del rápido desarrollo de la tecnología de los computadores. Los cambios drásticos de este lenguaje han permitido que pueda ser utilizado por un amplio grupo de usuarios: principiantes, en presentaciones multimedia accesibles incluso a niños pequeños, y usuarios experimentados en simulaciones y exploraciones complejas de proyectos sofisticados.

2.2.2 Características

De todas las presentaciones que se pueden encontrar de LOGO la más utilizada en la educación, es la de La Tortuga*,* llamada "*Gráficos de la Tortuga*". Debe su

[9] Seymour Papert. Pionero de la inteligencia artificial y pensador influyente sobre cómo el uso de las computadoras puede cambiar las maneras de aprendizaje. Nació en Pretoria, Sudáfrica el 1 de marzo de 1928. Es considerado como destacado científico computacional y educador. Es licenciado en Matemáticas en el Instituto Tecnológico de Massachusetts. Papert trabajó con el psicólogo educativo Jean Piaget en la Universidad de Ginebra desde 1959 hasta 1963 una colaboración que condujo a Papert a considerar el uso de las Mátemáticas al servicio del entendimiento de cómo los niños piensan y aprenden. (*Tomado de http://www.wikipedia.com.*)

nombre al personaje principal que es una tortuga animada (Ver figura 5), que se mueve bajo el control de un ordenador. La Tortuga va dibujando en la pantalla a medida que recibe órdenes como: *avanza, retrocede, bajalapiz, subelapiz, giraderecha, giraizquierda*, etc. Así un niño se convierte en constructor de sus propias estructuras mentales.

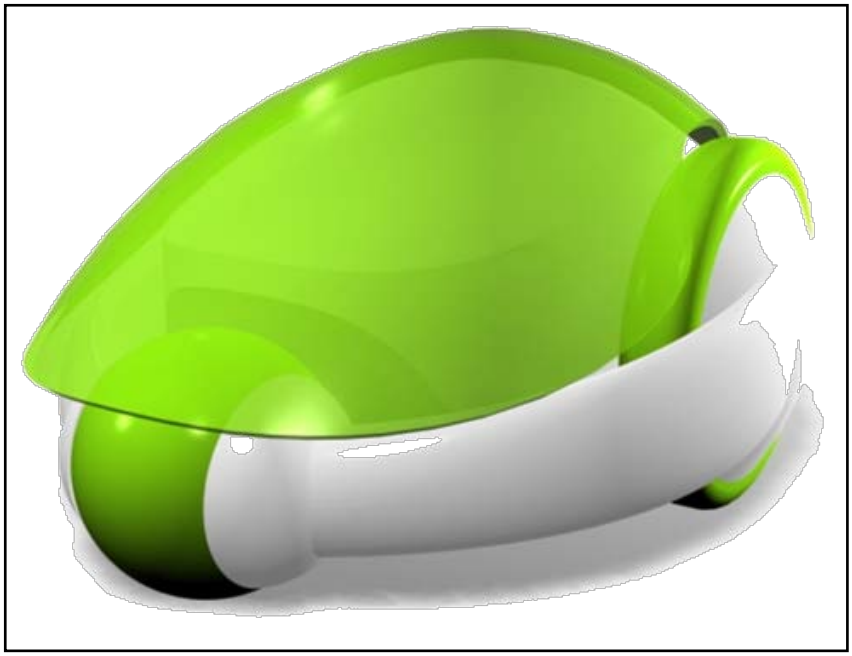


Figura 5. Modelo de la tortura resultado del presente proyecto[10]

2.2.3 Ventajas de Logo

Construcción de Conceptos
Cuando el alumno hace la programación de un fenómeno por simple que sea, debe para hacerla, buscar bibliográficamente los conceptos que concierne el fenómeno en el que trabaja, y aprender mientras construye.

Manejo del Error
Según Papert, la educación convencional enseña que el error es malo y no encamina a que el estudiante reflexione sobre dicho error. En un ambiente LOGO el error se considera como una fuente de aprendizaje; el análisis de este y su corrección en los programas son elementos fundamentales en el proceso requerido para lograr un determinado objetivo.

[10] AGUILLÓN VESGA Edgar Santiago y RUEDA TRIANA Diana Marcela. Diseño y Construcción De Un Robot Móvil no Autónomo, Programable por Medio de un Software Basado En La Filosofía De Logo. [Libro]. - 2007

Horacio C. Reggini[11] en su publicación "Alas para la mente" resume la importancia de LOGO en el proceso educativo, afirmando que su inclusión en la escuela permite la obtención de logros tales como: mejoramiento de la auto-concepción, desarrollo de habilidades para solucionar problemas; desarrollo de la creatividad; desarrollo de la capacidad para analizar y dividir problemas en sub-problemas; y desarrollo de la capacidad de síntesis enlazando sub-procedimientos en sub-procedimientos cada vez más complejos.

Además de lo anterior se pueden citar otras ventajas entre las que se destacan las siguientes:

- Está en castellano y es tan sencillo que se diseñó para que fuera utilizado principalmente por niños sin conocimientos en computación.
- La sencillez con el trabajo de gráficos.
- La recursión.
- La modularización; ésta permite, como dice Papert, a cada niño hacer procedimientos del tamaño de su mente.
- Es fácil conseguir aplicaciones didácticas en Matemáticas y en general en asignaturas de Ciencias, Diseño y Música.
- La enseñanza Logo permite incrementar la inteligencia y desarrollar destrezas que facilitan que el niño aprenda a aprender y pensar.
- La interactividad del lenguaje y los ambientes Logo, permiten construir un diálogo entre niño y computador. Ello le permite visualizar en cada instante el resultado del programa en construcción y así editarlo, corregirlo y volverlo a ejecutar.
- Posee una gran ventaja educativa. Basta conocer escasas instrucciones primitivas como *avanza, retrocede, giraizquierda, giraderecha,* para desarrollar pequeños proyectos.

2.2.4 Experiencias con Logo

La presencia de LOGO en la educación tiene una larga trayectoria histórica, que viene avalada por las múltiples experiencias llevadas a cabo prácticamente en todo el mundo a lo largo de sus casi tres décadas de existencia. Es por eso que la

[11] Investigador ampliamente conocido internacionalmente por sus trabajos en el campo de computadoras y educación. Graduado de ingeniero de la Universidad Nacional del Sur. En 1960 organizó el Grupo de Estudio de Aplicaciones de Computadoras (GEAC) en la Facultad de Ingeniería (UBA), y en 1966 se relacionó con el proyecto MAC del Instituto Tecnológico de Massachussets. Ha cumplido una ininterrumpida labor docente en diversas universidades y es miembro de la Academia Nacional de Ciencias Exactas, Físicas y Naturales. Autor del libros Alas para la mente -introducción al sistema Logo de computación-, e Ideas y formas, en el que desarrolla su nuevo sistema Logo tridimensional.

inclusión de LOGO como una herramienta educativa más para trabajar y adquirir los hábitos, destrezas y habilidades curriculares ofrece amplias perspectivas en la orientación de los aprendizajes. A continuación se muestran algunas de las muchas aplicaciones que se le han dado a este ambiente de aprendizaje y los resultados obtenidos con la experimentación de estos.

Logo en Latinoamérica

Caso Argentina[12]

Al seleccionar un tema de enseñanza dentro de cualquier ámbito del conocimiento siempre estará presente el enfoque conceptual. Un ejemplo clásico sobre este enfoque se tiene en la enseñanza de la geometría (especialmente en el nivel primario); es posible partir del punto para definir la recta como sucesión o conjunto de puntos o partir de la recta para definir el punto como intersección de rectas. Esta dualidad discreto-continua es un punto fundamental de la matemática, cuya historia es rica en modelos que construyen lo continuo a través de lo discreto y viceversa.

En la enseñanza de la matemática, la única forma de romper inicialmente con la situación dialéctica antes descrita es tomando el desarrollo madurativo del sujeto como eje epistemológico para su construcción, pero cuidando de favorecer el juego de oposiciones. La experiencia se llevó a cabo en los talleres experimentales de matemática en una escuela de nivel medio en la ciudad de Buenos Aires con alumnos de 13 a 14 años, los cuales ya tenían cuando menos dos años de experiencia trabajando en un taller de matemática experimental utilizando una metodología constructivista. Se observó que para resolver los problemas, los alumnos parten de nociones netamente intuitivas (no hacen explícitas las herramientas matemáticas adquiridas dentro de las clases formales de matemática), poniendo de manifiesto la discrepancia entre la teoría adquirida y los instrumentos conceptuales utilizables en la búsqueda de la solución.

Al final se determinó que la riqueza de un lenguaje geométrico y procedimental como LOGO y la intuición que todo adolescente posee de los movimientos en un plano, permitió "hacer geometría sin estar el docente informando sobre geometría".

Caso Costa Rica[13]

[12] MURARO, Susana, “LOGO: INSTRUMENTO PARA LA CONSTRUCCION DE NOCIONES LOGICO-MATEMATICAS”. Boletín de Informática Educativa, Vol. 3 N° 1, Universidad Nacional de Colombia, 1990.

[13] FONSECA, Clotilde – SCHAFFER, Marilyn. “¿Por qué LOGO? Una respuesta de Costa Rica”. Boletín de Informática Educativa, Vol. 3 N° 1, Universidad Nacional de Colombia, 1990.

El interés de incorporar LOGO al sistema costarricense de educación primaria, se fundamentó en formar una nueva generación de niños y docentes que estén en mejor capacidad de enfrentar los retos del futuro. Desde sus orígenes, el Programa de Informática Educativa fue concebido como un proyecto cuyo punto focal no es el computador en sí, sino el sistema de aprendizaje que se crea gracias a su uso inteligente, y que tiene por destinatario al niño. Lo que se perseguía era desarrollar un ambiente de enriquecimiento para estudiantes y docentes y generar un ámbito que no sólo nutra el aprendizaje del estudiante sino que incluso estimule a los maestros a comprender cómo se generan los procesos cognitivos de sus estudiantes. Fue un programa pensado para las escuelas de primaria, donde se introdujeron equipos de computo para incentivar la capacidad creadora y el desarrollo de procesos cognitivos en los escolares del país, seleccionando a LOGO como la herramienta central de esta ambiciosa tarea. Fue desarrollado alrededor de 1990, y alcanzando a implementarse aproximadamente en el 45% del la población total de escolares del país.

Se realizó un entrenamiento de docentes en módulos de tres semanas de duración a tiempo completo, entrenamiento recibido por los maestros incorporados al programa por lo menos una vez al año. Se instalaron laboratorios, cada uno con 20 máquinas; los niños asistían al laboratorio dos veces por semana y desarrollaban su trabajo con el computador de dos en dos. El trabajo se desarrolló en el área de ciencias implementándose en un trabajo gráfico sobre "las plantas" hecho por estudiantes de VI Grado, quienes ilustraron gráficamente el proceso de nacimiento, crecimiento, reproducción y muerte de una planta. En la elaboración de este proyecto, tuvieron necesidad de leer (uso del lenguaje) sobre el ciclo de la vida de una planta. Luego, tomar decisiones sobre el material que requería para explicar en la pantalla y cómo crear las imágenes que aparecen allí (resolución de problemas). Al hacerlo, tuvo que calcular distancias (matemáticas), usar el concepto de recursividad para repetir patrones, usar conceptos geométricos para crear líneas y lograr ángulos, y utilizar el sistema de coordenadas para ubicar las formas definidas en las posiciones deseadas.

Logo en Colombia

La aparición de LOGO en Colombia a finales de la década del setenta, fue enmarcada con la presencia de los micro-computadores en los colegios e intensificada significativamente en la década del ochenta.

Un aspecto importante para resaltar es la estrecha relación que se encuentra entre el acceso a la tecnología y el nivel socio-económico de los educandos; esto se pudo ver con la concentración del fenómeno tecnológico en los planteles privados dada su solvencia. Parecería obvio pensar que los usuarios deberían ser prioritariamente los millones de niños vinculados a los planteles públicos de las

áreas rurales y urbanas de nuestro país, pues si el computador favorece el desarrollo intelectual de los estudiantes, entonces sus beneficios serían mayores en donde los problemas y deficiencias fueran igualmente mayores.

Luego de la adquisición de los equipos, se debía justificar la inversión y se llevó a una búsqueda infructuosa de software en español. La mayoría del software encontrado estaba enfocado a aplicaciones para la administración del plantel (notas, inventarios, nóminas) y unos cuantos programas para refuerzo (ejercitación y práctica) de ciertas áreas del conocimiento. Dentro de dicha búsqueda, LOGO apareció no como la alternativa educativa buscada, sino como la única que se podía utilizar de inmediato mientras aparecía el software deseado que se enmarcara dentro de una concepción educativa que favoreciera el desarrollo del individuo.

Ejemplos de planteles públicos en los que se instauró LOGO y se obtuvieron resultados satisfactorios son:

Experiencias Con Logo En Varios Planteles Públicos

La experiencia expuesta a continuación fue una investigación desarrollada por el Instituto SER[14] centrado exclusivamente en planteles públicos y a nivel de educación básica primaria.

La Escuela Veredal Juan XXIII. La visita que Seymour Papert realizó a Colombia durante el gobierno de Betancur, despertó el interés del Instituto en la realización de un proyecto de investigación que permitiera analizar en detalle algunas de las "bondades" de LOGO, expuestas por su creador.

Las Escuelas Rurales de Nemocón. Después de finalizar el estudio en la Escuela Juan XXIII de la vereda de Susatá, se consideró pertinente vincular otras tres escuelas a un proceso similar que permitiera replicar la experiencia anterior, hacer los ajustes y modificaciones necesarios y de esta manera consolidar, de la mejor forma posible, la experiencia iniciada. El proceso de selección de las tres escuelas deseadas fue bastante complicado, dado que prácticamente todos los docentes deseaban ser incluidos en la nueva fase del proyecto. Debido a que varios directores de escuela habían organizado "excursiones" con los estudiantes para conocer los computadores de la escuela de Susatá, su motivación permitió seleccionar los planteles que consideraban los computadores como "algo" que realmente deseaban para su escuela y no como unas herramientas donadas pero que realmente ni esperaban ni necesitaban. En otras palabras, los computadores

[14] El Instituto SER de Investigación es una entidad privada, sin fines lucrativos, creado en 1973 por un grupo de académicos y formadores de empresa en Colombia. http://institutoser.uniandes.edu.co/

entregados a estas escuelas fueron vistos como algo que realmente deseaban para utilizar con los estudiantes.

Las Escuelas del Distrito Especial de Bogotá. La Alcaldía de Bogotá y específicamente la Secretaría de Educación del Distrito Especial, inició en Abril de 1989 un proceso tendiente a ofrecer a las escuelas públicas acceso a computadores y al lenguaje LOGO. Para tal efecto, la Secretaría solicitó la colaboración del Instituto SER en dicho proceso, el cual contempló las siguientes acciones:

- Capacitación de los docentes de 40 planteles (ambas jornadas) de educación básica primaria. Dicha capacitación se realizó a través de talleres de aproximadamente 40 horas de duración en donde los docentes, además de conocer los equipos (historia, partes y funciones), interactuaron directamente con el lenguaje LOGO.

- Una vez capacitados los docentes en el uso de los equipos y del lenguaje LOGO, se realizó la dotación de las escuelas. En promedio la Secretaría entregó entre 10 y 14 equipos ATARI 65 XE con los respectivos paquetes de LOGO, unidades de diskette y una impresora por plantel.

- Seguimiento de la experiencia. Dentro de las acciones planteadas se consideró pertinente realizar visitas a los planteles para identificar los principales problemas o interrogantes que tienen los docentes. De igual forma, se programaron seminarios-taller de profundización en relación con el manejo del lenguaje LOGO.

A pesar de esta experiencia, las visitas realizadas a los planteles que alcanzaron a recibir los equipos durante el segundo semestre de 1989, permitió afirmar que, como en el caso de Nemocón, existía una gran motivación por parte de docentes y alumnos en relación con el lenguaje LOGO.

2.2.5 El uso del lenguaje Logo y el desarrollo del pensamiento

Hace ya más de treinta años fue diseñada la primera versión del lenguaje LOGO por un equipo de investigadores bajo la dirección de Seymour Papert. La creación de este lenguaje de computador estaba inscrita dentro de un marco filosófico-pedagógico inspirado en la obra de Jean Piaget. Uno de los elementos centrales de este marco de referencia es la concepción del ser humano como un constructor de su propio conocimiento. Esta concepción es difícil entenderla sin contraponerla con otra posición incompatible y con la cual "compita".

Una de estas puede ser la propuesta por Federic Skinner[15], quien concibe al ser humano, en el momento de su nacimiento, como un organismo biológico desprovisto de cualquier otra propiedad que no sean algunos reflejos y la posibilidad de actuar (o emitir respuestas) ante los estímulos que el medio ambiente le ofrece. Estas respuestas son modificadas en función de sus relaciones con estos estímulos. La "suma" de todas esas modificaciones acaecidas hasta un momento dado en la vida del organismo, tiene como resultado el repertorio de respuestas que el organismo presenta en ese momento.

Por su parte Piaget, concibe al ser humano, en el momento de su nacimiento, como un organismo biológico con unas estructuras iniciales (respuestas preestablecidas) que se modifican gracias al funcionamiento de la inteligencia. Ese funcionamiento se da en dos grandes procesos: el equilibrio (de la estructura con el mundo circundante) y la organización (de todas las estructuras dentro de una "totalidad mental"). El equilibrio tiene dos subprocesos: la asimilación (o modificación del mundo externo "para que encaje" en las estructuras mentales del sujeto) y la acomodación (o modificación de las estructuras “para eliminar la discrepancia” entre éstas y el mundo circundante).

Desde el punto de vista conductista el computador es un instructor, y el usuario es un sujeto pasivo que se limita a reaccionar ante el medio y modificar su comportamiento en función de él.

El LOGO al igual que Piaget, concibe al ser humano como un sujeto activo que, en lugar de "aprehender" el mundo, "construye teorías" acerca de él. El sujeto interactúa continuamente con su mundo, y aprende de él por ser éste el objeto de su conocimiento. Por eso LOGO está dentro de la categoría de software para "uso interactivo del computador", porque lo que intenta ofrecer es conjunto de objetos de conocimiento para que el usuario construya teorías cada vez mejor equilibradas.

Papert, considera que el uso del LOGO desarrolla en el niño la capacidad de crear nuevas teorías (cada vez más complejas) acerca del mundo, ponerlas a prueba y perfeccionarlas en función de los resultados.

2.2.6 Logo ante el futuro

Aunque LOGO ha estado un poco estancado, con la difusión de LOGO en Internet, la aparición de nuevas versiones y la experiencia acumulada a lo largo de

[15] Burrhus Frederic Skinner. (20 de marzo de 1904 - 18 de agosto de 1990) Psicólogo y autor norteamericano. Condujo un trabajo pionero en psicología experimental y defendió el conductismo, que considera el comportamiento como una función de las historias ambientales de refuerzo. Escribió trabajos controvertidos en los cuales propuso el uso extendido de técnicas psicológicas de modificación del comportamiento, principalmente el condicionamiento operante, para mejorar la sociedad e incrementar la felicidad humana, como una forma de ingeniería social.

todo este tiempo de experimentación, parece que LOGO tendrá un gran futuro en la informática educativa de las aulas. En muchos países europeos y americanos LOGO ha recibido un fuerte apoyo a nivel institucional.

A modo de ejemplo a nivel mundial, se puede citar el caso de Costa Rica expuesto anteriormente, país que inició en 1988 el Programa de Informática Educativa para la Educación Primaria, desarrollado de manera conjunta entre el Ministerio de Educación Pública y la Fundación Omar Dengo[16], con la finalidad de apoyar el aprendizaje de las disciplinas curriculares básicas mediante LOGO, usando la versión *LOGO Writer* en sus primeros tiempos y *MicroWorlds* en la actualidad.

Además continuamente se realizan reuniones monográficas sobre LOGO alrededor del mundo, implicando a las instituciones de los países donde se desarrollan:

- En Iberoamérica se celebra un congreso en informática educativa cada dos años, el próximo será en Venezuela en 2008[17], cabe destacar que en su encuentro de 1995 realizado en Brasil estuvo centrado en las posibilidades de LOGO frente al desarrollo acelerado de otros recursos tecnológicos como *Windows*, multimedia, robótica y telemática.
- A nivel europeo existe también una reunión bianual, que se conoce con el nombre de *Eurologo*[18] pero que, al igual que el iberoamericano, suele tener una proyección mundial. El próximo congreso tendrá lugar este año en Bratislava (Eslovaquia) y el último se realizó en Warsaw (Polonia) en 2005. Estas reuniones tienen ya una década de antigüedad y están sirviendo para que LOGO se afiance cada vez con más fuerza en los sistemas educativos europeos, hasta el punto de calar hondo en países con escasa tradición informática.
- Y finalmente, en EEUU donde tienen lugar reuniones anuales desde los comienzos mismos de la aparición del lenguaje hasta la actualidad.

2.3 GRÁFICOS 3D - DIRECT X

Las personas que han tenido contacto con video juegos u otras aplicaciones desarrolladas para manejar gráficos han oído alguna vez la palabra DirectX. La mayoría de los usuarios desconoce su función y sólo se limitan a instalarlo pues en realidad para ellos tampoco es necesario conocer de qué se trata, con su ayuda en la ejecución de los juegos es suficiente. Es por esto que se cree conveniente hacer una descripción más amplia de él dado que el entorno de ejecución de LPE está implementando utilizando éstas librerías.

[16] http://www.fod.ac.cr
[17] http://ares.unimet.edu.ve/ribie/ribie.htm
[18] http://www.eurologo.org

Microsoft DirectX es una colección avanzada de interfaces de programación de Aplicaciones (APIs) integrada a los sistemas operativos Microsoft Windows; este conjunto de APIs mantiene una plataforma de desarrollo de aplicaciones multimedia estándar para aplicaciones Windows, permitiéndoles a los programadores de software acceder al hardware especializado sin tener que escribir código específico de cada tipo de hardware.

Cuando una aplicación o un juego son escritos para DirectX, el programador no debe preocuparse por exactamente qué tarjeta de sonido o por el adaptador gráfico que el usuario final podría tener en su máquina, DirectX se encarga de eso por él.

DirectX juega un papel en muchas funciones, incluyendo renderización 3D, reproducción de video, interfaces para joysticks y ratones, gestión de redes para multi-jugador y muchos más.

¿Porque DirectX?

DirectX proporciona a los programadores una manera estandarizada y amigable de acceder a los recursos de la computadora para programar aplicaciones y juegos aprovechando las últimas tecnologías de hardware de manera generalizada.

Otra de las principales ventajas de trabajar con DirectX es que no sólo soporta la parte de gráficos y de 3D sino posee todas las herramientas para construir aplicaciones completas de alto nivel de una manera en la que el hardware no es una limitación, sino que el programador solo debe conocer el API y este es el que se encarga de saber cómo realmente funcionan los distintos tipos de hardware.

2.3.1 Funcionamiento de DirectX

Básicamente el programador tiene a mano el API de DirectX de donde se tienen las funciones con las que se dispone para programar la aplicación, este API se comunica con un HAL (capa de abstracción del hardware) diciéndole a ésta lo que la aplicación está queriendo hacer, el HAL es el que realmente conoce que es lo que el hardware puede o no realizar y cómo lo realiza así que es éste el que se comunica con el hardware y de acuerdo a esto le asigna el trabajo, en caso de que el hardware no sea compatible con alguna de las funcionalidades de la versión de DirectX que se esté utilizando existe un HEL (capa de emulación de hardware) que se encarga de emular las funciones que el hardware no puede manejar.

De esta manera se puede mantener la compatibilidad de DirectX con todo tipo de hardware más viejo y permite utilizar aplicaciones nuevas con este hardware, pero con cierta penalización en rendimiento o en calidad.

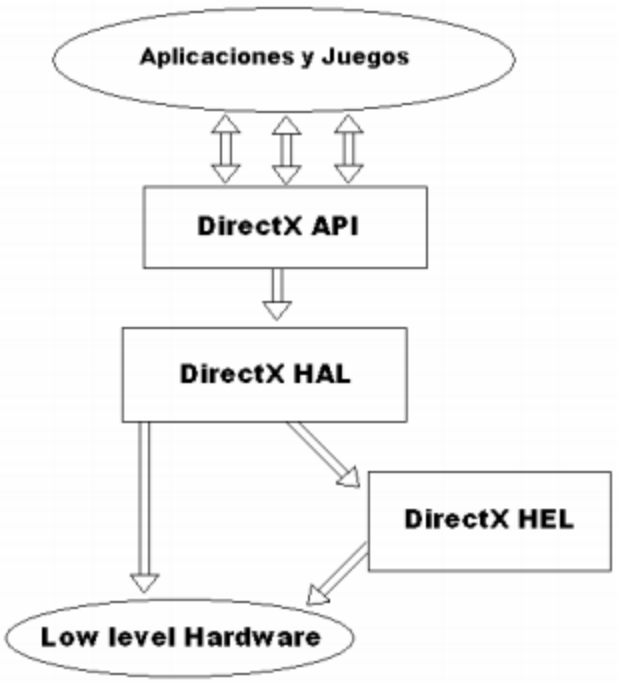


Figura 6. Comunicación DirectX - HAL[19]

2.3.2 Metas de DirectX

Una de las razones primarias para crear DirectX era promover el desarrollo de los juegos para la plataforma Windows. La mayoría de juegos desarrollados para el computador personal eran basados en MS-DOS. Sin embargo, al diseñar los juegos basados en DOS, los diseñadores, debían programar varias implementaciones de hardware para una variedad de tarjetas que complicaban la instalación.

DirectX fue desarrollado para garantizar a los programas basados en Windows un alto rendimiento en el acceso en tiempo real al hardware disponible en los computadores. Éste provee interfaces consistentes entre los desarrolladores de hardware y software para dar a los programadores acceso de bajo nivel a las capacidades del hardware, como acceso directo a la memoria de video, acceso a

[19]URRAZA Juan de Microsoft DirectX [En línea]. 2003. - http://www.jeuazarru.com/docs/DirectX.pdf.

la aceleración de video y sonido por hardware, etc., todo esto mediante una capa de abstracción entre el software y los distintos dispositivos de hardware.

A partir de este punto, DirectX fue evolucionando, agregando cada vez más funcionalidades, y mejorando otras. En general DirectX ha venido ganando mucha popularidad en los últimos años entre los programadores de videojuegos, en especial desde que Microsoft comenzó a mejorar la parte de Direct3D haciéndolo más amigable como eficiente, y se ha vuelto una seria competencia para OpenGL[20], otro de los estándares gráficos de gran aceptación.

2.3.3 Componentes de DirectX

El API de DirectX está conformado por los siguientes componentes:

- *DirectDraw* es un administrador de la memoria de video. Usando DirectDraw, un programa puede manipular la memoria de video con facilidad, mientras aprovecha todo tipo de capacidades de los tipos diferentes de adaptadores de video sin depender de un tipo particular de hardware.

- *Direct3D* es un conjunto de servicios gráficos 3D que se encarga de brindar la aceleración avanzada de los aceleradores hardware.

- *DirectSound* permite la aceleración por hardware de sonidos 3D utilizando sistemas surround y un subconjunto de la norma EAX de Creative.

- *DirectInput* proporciona acceso rápido y consistente a joysticks y dispositivos de mano, permite configuraciones personalizadas de calibración y drivers.

- *DirectPlay* simplifica los servicios de comunicación, permitiendo a las aplicaciones el envío de mensajes sin importar el medio, protocolo o servicio que se esté utilizando para conectarse a la red.

- *DirectMusic* proporciona el manejo de música aprovechando las características del hardware.

- *DirectX HAL* presenta las capacidades del hardware de forma genérica y no dependiente del hardware.

- *DirectX HEL* es la capa de emulación del hardware y permite desarrollar aplicaciones sin tener que preocuparse por las funcionalidades que tiene

[20] OpenGL es una especificación estándar que define una API multilenguaje y multiplataforma para escribir aplicaciones que produzcan gráficos 2D y 3D desarrollada por Silicon Graphics Inc. Su nombre viene del inglés **Open G**raphics **L**ibrary, cuya traducción es *biblioteca de gráficos abierta.*

disponibles el hardware y mantiene la compatibilidad con el hardware más antiguo.

2.3.4 Evolución de DirectX

Hace largo tiempo que Microsoft lanzó la primera versión de DirectX, hablando de DirectX 3. A ésta le siguieron DirectX 5, 6 y 7, cada uno con distintas variaciones de funcionamiento, que lograron colocar a DirectX como un estándar dentro del entorno Windows.

DirectX 7.0 fue lanzado en septiembre de 1999. Agregó un soporte mejorado para gráficos 3D y para sonido, además de mejorar notablemente el rendimiento.
Aceleración por hardware para DirectMusic en los adaptadores que soportan la tecnología. También introdujo al API de Direct3D la capacidad de aprovechar transformación, iluminación y compresión de texturas por hardware.
Mejoras en la aceleración de sonidos por hardware y los algoritmos de sonido 3D mejorados usando tecnología EAX de Creative. También incluyó soporte para Visual Basic en el API, permitiendo a programadores utilizar el API en un ambiente más familiar.

La verdadera revolución comenzó a partir de DirectX 8, con la inclusión de las instrucciones para shaders[21]. Hasta ese momento, los programadores estaban atados a las funciones predeterminadas que podía llevar a cabo la GPU[22]. A partir de la versión 8, la libertad a la hora de desarrollar escenas en tres dimensiones creció exponencialmente, gracias al concepto de “shaders programables”. Pero seguía habiendo una limitación, y ésta estaba dada por los recursos de programación implicados: variables, constantes, registros. Más precisamente, por la cantidad de cada uno de ellos.
DirectX 8.0 cambio radicalmente en la arquitectura de video, DirectDraw y Direct3D se fusionan, se introduce un modelo de “shaders” para manejar el procesado de vértices y de píxeles, con funciones fijas o con un lenguaje parecido al código ensamblador. Esto significa que los programadores ya no solo tenían un API que les brindaba toda la capacidad del hardware, sino que ellos mismos podían de una manera básica, programar la tarjeta de video, para que produzca los efectos visuales que ellos querían lograr, también hubo una mayor integración entre DirectSound y DirectMusic.
Se integró la habilidad de insertar filtros de procesamiento dentro del pipe-line de audio para lograr efectos especiales Reverb, Chorus, etc.

[21] Un shader es un procedimiento de sombreado e iluminación que permite al programador especificar el renderizado de un pixel.
[22] Graphics Processing Unit – Unidad de Procesamiento Gráfico.

Grandes cambios estructurales en DirectPlay, se adoptó el modelo “Push” remplazando al “poll”, permitiendo el uso efectivo de las capacidades de multilectura, también se agregó soporte para conversaciones de voz.
Se introduce DirectShow como parte de DirectX, DirectShow proporciona al programador un API unificado para manejar las capacidades de reproducción de video de los diversos tipos de hardware disponible en el mercado.

Con la salida de DirectX 9.0c (hubo varias versiones entre medio) este problema ya no suponía una complicación para el programador. El principal cambio en esta última versión de DirectX es la introducción del HLSL (High Level Shading Language), con la introducción de este lenguaje basado en C, los programadores pueden crear desde animaciones ultra realistas, hasta efectos visuales sorprendentes, sin tener que preocuparse por el tipo especifico de hardware que se esté utilizando.

Se fueron agregando también, nuevas técnicas de procesamiento grafico, como son el Depth Adaptive Tessellation (reducción dinámica en la complejidad de objetos según distancia) y el displacement mapping (capacidad de dar profundidad a un objeto 3D de acuerdo a un mapa de colores).
Ahora en verdad se puede hablar de una verdadera GPU, ya que con la introducción del HLSL que permite saltos, loops y manejo de subrutinas, se tiene un procesador grafico totalmente programable.

En esta versión, las utilidades nuevas del API facilitan la visualización y manipulación de texturas así como unos plug-ins actualizados para exportar – importar modelos 3D de aplicaciones como 3D Studio.
DirectPlay tuvo mejoras en cuestiones de seguridad, también incluye el soporte para Internet Protocol v.6.
DirectInput no sufrió cambios en el API, pero se trabajo para lograr mayor compatibilidad con el hardware actual.
DirectSound tuvo mejoras en el rendimiento.

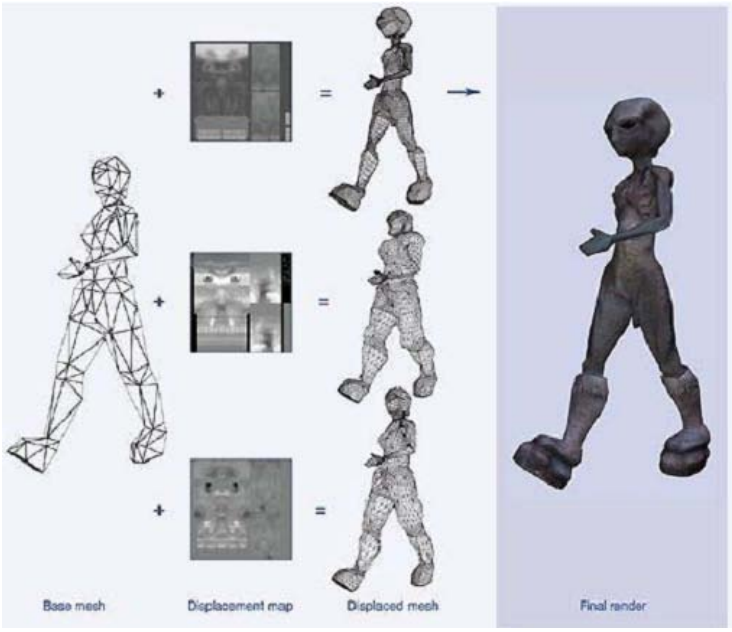
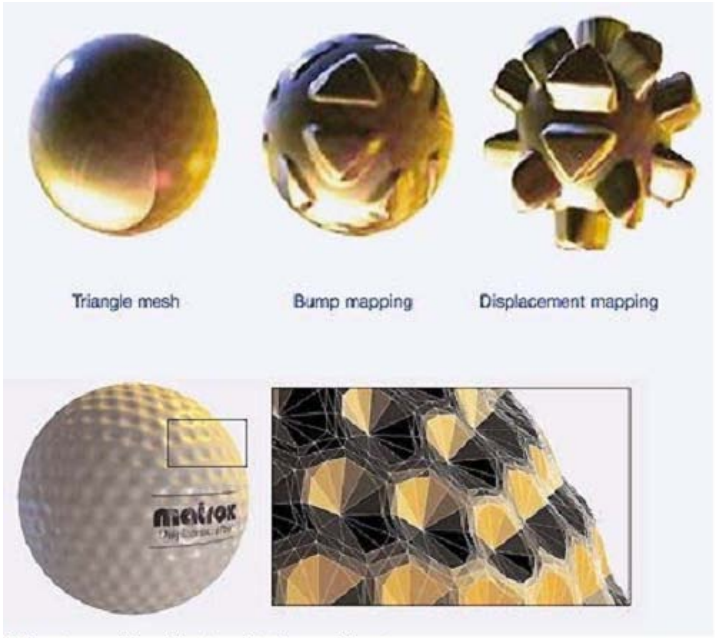


Figura 7. Generación de diferentes modelos 3D basados en un modelo simple usando Displacement Mapping[23]



Diferencias entre Bump Mapping y Displacement Mapping.
Notar que el Displacement Mapping crea geometría 3D real sobre el modelo base, mientras que el Bump Mapping utiliza efectos de sombreado para crear el efecto de geometría 3D.

Figura 8. Diferencias entre Bump Mapping y Displacement Mapping[24]

[23]URRAZA Juan de Microsoft DirectX [En línea]. 2003. - http://www.jeuazarru.com/docs/DirectX.pdf.
[24] Ibíd.

2.3.5 El comienzo de una nueva generación

DirectX 10, junto al modelo de shaders unificados, constituyen el futuro inminente en el campo del procesamiento de gráficos tridimensionales. Por supuesto que el hardware tendrá que evolucionar en forma acorde para no limitar a los programadores a la hora de trabajar con este API. En tal caso, no hace falta reiterar que las posibilidades son enormes, mucho mayores a lo que se estipula hoy en día.

Por todo lo anterior, DirectX 10 promete mejoras que parecían inimaginables hace un tiempo atrás, todo parece indicar que el realismo en un juego de ordenador tendrá un avance muy significativo, siempre y cuando los desarrolladores hagan las cosas como corresponde.

2.4 INGENIERÍA DEL SOFTWARE

2.4.1 Administración evolutiva de proyectos

Evo, es el método iterativo para diseño ágil de aplicaciones más antiguo[25], fue creado por Tom Gilb. Se le llama también Desarrollo Evolutivo, Administración Evolutiva, Administración de Proyectos Manejado por Requerimientos e Ingeniería Competitiva. En 1976 Gilb trató temas de desarrollo iterativo y gestión evolutiva en su clásico "Software Metrics"[26], texto que acuñó el concepto e inauguró el campo de las métricas de software. En 1981 publicó "Evolutionary Development" en ACM[27] Software Engineering Notes y "Evolutionary Delivery versus the 'Waterfall Model'" en ACM Sigsoft Software Requirements Engineering Notes.

En las breves iteraciones de Evo, se efectúa un progreso hacia las máximas prioridades definidas por el cliente, liberando algunas piezas útiles para algunos participantes y solicitando su feedback[28]. Esta es la práctica que se ha llamado Planeamiento Adaptativo Orientado al Cliente y Entrega Evolutiva. En Evo se espera que cada iteración constituya una re-evaluación de las soluciones en procura de la más alta relación de valor contra costo, teniendo en cuenta tanto la realimentación como un amplio conjunto de estimaciones métricas. Evo requiere, igual que otros MA's[29], activa participación de los clientes. Todo debe cuantificarse. A diferencia de otros MA's como Agile Modeling, donde la

[25] REYNOSO Carlos Métodos Heterodoxos en Desarrollo de Software [En línea]. - Microsoft Corporation, 26 de Junio de 2006. – http://www.microsoft.com/spanish/msdn/arquitectura/roadmap_arq/heterodox.mspx
[26] GILB Tom Software Metrics [Libro]. - [s.l.] : Chartwell-Bratt, 1976
[27] ACM acrónimo de "Association for Computing Machinery".
[28] Retroalimentación
[29] Métodos Ágiles.

metodología es puntillosa pero discursiva, en Evo hay una especificación semántica y una pragmática rigurosa, completamente alejadas del sentido común, pero con la fundamentación que les presta derivarse de prácticas productivas suficientemente probadas.

Los diez principios fundamentales de Evo son:

1) Se entregarán temprano y con frecuencia resultados verdaderos, de valor para los participantes reales.
2) El siguiente paso de entrega de Evo será el que proporcione el mayor valor para el participante en ese momento.
3) Los pasos de Evo entregan los requerimientos especificados de manera evolutiva.
4) No se pueden saber cuáles son los requerimientos por anticipado, pero se pueden descubrir más rápidamente intentando proporcionar valor real a participantes reales.
5) Evo es ingeniería de sistemas holística (todos los aspectos necesarios del sistema deben ser completos y correctos) y con entrega a un ambiente de participantes reales (no es sólo sobre programación; es sobre satisfacción del cliente).
6) Los proyectos de Evo requieren una arquitectura abierta, porque habrá que cambiar las ideas del proyecto tan a menudo como se necesite hacerlo, para entregar realmente valor a los participantes.
7) El equipo de proyecto de Evo concentrará su energía como equipo hacia el éxito del paso actual. En este paso tendrán éxito o fracasarán todos juntos. No gastarán energías en pasos futuros hasta que hayan dominado los pasos actuales satisfactoriamente.
8) Evo tiene que ver con aprendizaje a partir de la dura experiencia, tan rápido como se pueda: qué es lo que verdaderamente funciona, qué es lo que realmente entrega valor. Evo es una disciplina que hace confrontar los problemas tempranamente, pero que permite progresar rápido cuando probadamente se ha hecho bien.
9) Evo conduce a una entrega temprana, a tiempo, tanto porque se lo ha priorizado así desde el inicio, y porque se aprende desde el principio a hacer las cosas bien.
10) Evo debería permitir poner a prueba nuevos procesos de trabajo y deshacernos tempranamente de los que funcionan mal.

El modelo de Evo consta de cinco elementos mayores[30]:

1) Metas, Valores y Costos – Cuánto y cuántos recursos. Las Metas y Valores de los Participantes se llaman también, según la cultura, objetivos, metas

[30] GILB Kai Evolutionary Project Management & Product Development [Libro]. - [s.l.] : http://www.gilb.com/Download/EvoProjectMan.pdf, 2006.

estratégicas, requerimientos, propósitos, fines, ambiciones, cualidades e intenciones.
2) Soluciones – Banco de ideas sobre la forma de alcanzar Metas y Valores dentro del rango de los Costos.
3) Estimación de Impacto – Mapear las Soluciones a Metas y Costos para averiguar si se tienen ideas adecuadas para lograr las Metas dentro de los Costos.
4) Plan Evolutivo – Inicialmente una idea general de la secuencia a desarrollar y evolucionar hacia las Metas. Los detalles necesarios evolucionan junto con el resto del plan a medida que se desarrolla el producto/servicio.
5) Funciones – Describen qué hace el sistema. Son extremadamente secundarias, más de lo que se piensa, y deben mantenerse al mínimo.

Cuando se inicia el ciclo, primero se definen los Valores y Metas del Participante; esta es una lista tradicional de recursos tales como dinero, tiempo y gente.

Una vez que se comprende hacia dónde se quiere ir y cuándo se podría llegar ahí, se definen Soluciones para lograrlo. Utilizando una Tabla de Estimación de Impacto, se realiza la ingeniería de las Soluciones para satisfacer óptimamente las Metas y Valores de los Participantes.

A medida que se desenvuelve el proyecto, se obtiene realimentación en tiempo real sobre las mejoras que implica la Entrega Evolutiva sobre las Metas y Valores del Participante, así como sobre el consumo de Recursos. Esta información se usa para establecer qué es lo que está bien y lo que no, cuáles son los desafíos y qué es lo que no se sabía desde un principio; también se aprende sobre las nuevas tecnologías y técnicas que no estaban disponibles cuando el proyecto empezó. Se ajusta luego todo según se necesite, pero sin detallar las Soluciones o las Entregas Evolutivas hasta que se esté próximo a la entrega. Por último vienen las Funciones y Sub-Funciones, de las que se razona teniendo en cuenta que en rigor, en un nivel puro, son de hecho Soluciones a Metas.

Tom Gilb distingue claramente entre los Pasos de la Entrega Evolutiva y las iteraciones propias de los modelos iterativos tradicionales o de algún método ágil como RUP. Las iteraciones siguen un modelo de flujo, tienen un diseño pre-establecido y son una secuencia de construcciones definidas desde el principio; los pasos evolutivos se definen primero de una manera genérica y luego se van refinando en cada ciclo, adoptando un carácter cada vez más formal. Las iteraciones no se realizan sólo para corregir los errores de código mediante refinamiento convencional, sino en función de la aplicación de mejoras.

2.4.2 Metodología de desarrollo

Haciendo un análisis de algunas de las metodologías existentes en el momento y teniendo en cuenta que se desea un desarrollo ágil de la aplicación, se ha elegido tomar una metodología que permita dar soporte a este tipo de proyectos y que además estipule la entrega del producto como una serie de avances en los que el usuario forma parte del desarrollo de nuevas características y funcionalidades no estipuladas en el comienzo del desarrollo.

Entre las metodologías consultadas, la que más cumplió con estas perspectivas fue el Prototipado Evolutivo – EVO, que con sus principios de entrega evolutiva[31]:

- ✓ **Aprendizaje:** La Entrega Evolutiva es un ciclo de aprendizaje. Se aprende de la realidad y la experiencia; se aprende lo que funciona y lo que no.
- ✓ **Temprano:** Aprender lo suficiente para cambiar lo que necesita cambiarse antes que sea tarde.
- ✓ **Pequeño:** Pequeños pasos acarrean pequeños riesgos. Manteniendo el ciclo de entrega breve, es poco lo que se pierde cuando algo anda mal.
- ✓ **Más simple:** Lo complejo se hace más simple y más fácil de manejar cuando se lo descompone en pequeñas partes.

Hace posible que el diseño de un compilador para el lenguaje que se planea desarrollar no resulte tan complejo, al lograr reducirlo a pequeños proyectos que se van mostrando a medida que se va desarrollando el sistema completo y reduciendo el riesgo de encontrarse con grandes errores después de un largo tiempo de desarrollo. Por otro lado, se reduce el tiempo para la construcción de un producto puesto que se elimina el tiempo en informes debido a que cada prototipo es un informe, se hacen varias estimaciones de tiempo por prototipo, evitando errores en la estimación del proyecto general, dando un cumplimiento a la fecha por los desarrolladores.

Entre las otras metodologías tomadas en cuenta para este desarrollo, pero que no fueron elegidas están:

RUP: El Proceso Unificado de Desarrollo de Software no fue tomado como metodología porque aunque es una metodología ágil, se hace compleja y lenta al tener que hacer todas las etapas del desarrollo por medio de diagramas.

Cascada: No fue tomado por no ser flexible a los cambios a la hora de modificar requerimientos iniciales y su incapacidad para identificar riesgos.

[31] http://www.gilb.com

En el prototipado evolutivo, las iteraciones siguen un modelo de flujo, tienen un diseño preestablecido y son una secuencia de construcciones definidas desde el principio; los pasos evolutivos se definen primero de una manera genérica y luego se van refinando en cada ciclo, adoptando un carácter cada vez más formal. Las iteraciones no se realizan sólo para corregir los errores de código mediante refinamiento convencional, sino en función de ir mejorando el prototipo con la adición de nuevas funcionalidades.

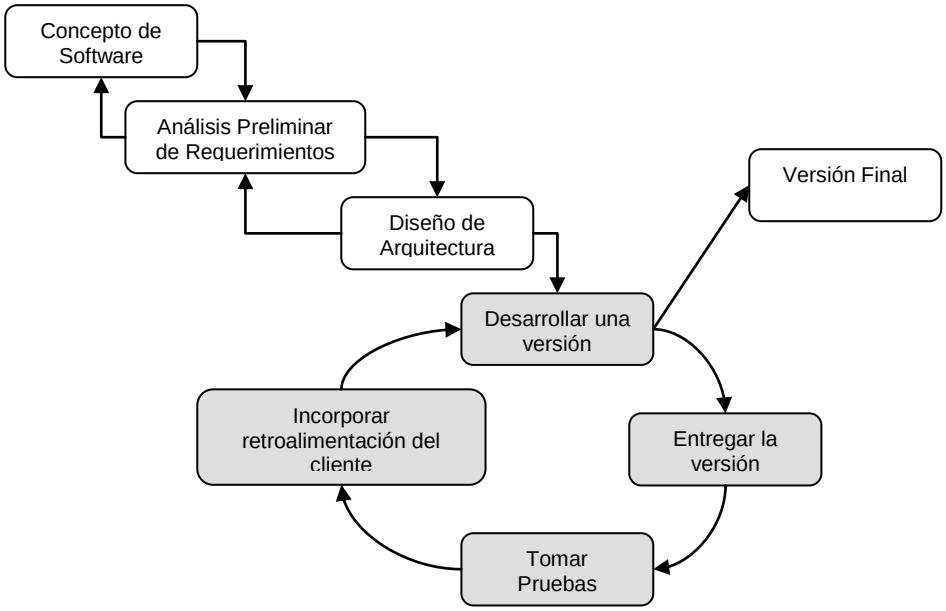


Figura 9. Modelo de Prototipado Evolutivo a seguir [32]

[32] Tomado de http://www.microsoft.com/spanish/msdn/arquitectura/roadmap_arq/heterodox.asp

3. DISEÑO Y DESARROLLO DE LA SOLUCIÓN

3.1 ESPECIFICACIÓN DE REQUERIMIENTOS

3.1.1 Ámbito del sistema

Está dirigido especialmente a la población infantil entre los 7 y los 12 años, sin embargo, no se excluyen su uso por jóvenes, adultos y adultos mayores que deseen entrar al campo de la programación y la robótica.

3.1.2 Requerimientos

Función	Editor de texto con estilos de fuente.
Atributos	- Se distinguirán las palabras reservadas, comandos, variables, cadenas, bucles, etc. - El usuario puede modificar los atributos del editor de texto (color de fondo, fuente, color de fuente). - El editor debe abrir, guardar y crear un nuevo proyecto. - El proyecto consta de un único archivo de texto. Opcional: - El editor debe manejar auto-tabulación, para que sea más entendible el código fuente - El editor debe mostrar en diferente fuente las palabras reservadas del lenguaje a medida que se escribe.

Función	Analizador léxico
Atributos	- Módulo que se encarga de reconocer y clasificar los tokens del código fuente. - Una vez separados todos los tokens del código fuente, debe enviarlos al analizador sintáctico para su correspondiente revisión. - Al encontrar un error se debe retornar el número de línea en el editor de texto, la posición del primer carácter en la línea que produjo el error y el tipo de error encontrado

Función	Analizador sintáctico
Atributos	- Módulo que se encarga de revisar la sintaxis del código fuente del usuario, para luego interpretar las instrucciones. - Luego de revisar la sintaxis, debe comprobar que las variables y

	funciones usadas estén correctamente declaradas y de lo contrario informar las variables que no se declaran

Función	Analizador semántico
Atributos	- Modulo encargado de revisar que el tipo de variables sea compatible en el proceso de una instrucción, y dado el caso que las variables sean diferentes hacer el casting correspondiente.

Función	Evaluador de expresiones aritméticas y lógicas
Atributos	- Dada una serie de tokens numéricos ó lógicos separados por tokens de tipo operador, se debe solucionar y devolver su respuesta de tipo numérico ó lógico, según sea el caso.

Función	Ayuda Interactiva
Atributos	- La ayuda puede ser desplegada en cualquier momento con la tecla F1. - Especificar la forma de operar con el sistema - Especificar la forma como se utilizan los comandos - Especificar la forma de programar teniendo en cuenta las respuestas del objeto móvil - Presentación de ejemplos - La ayuda estará en formato chm o hlp. Opcional: - Al resaltar una palabra y pulsar ayuda, se debe hacer una búsqueda automática de la palabra en mención.

Función	Intérprete de comandos de un texto sintácticamente correcto
Atributos	- El texto a interpretar debe estar completamente corregido. - Para evitar el problema de bucles infinitos, definir un tiempo y/o un número de repetición en el ciclo, que al cumplirse se aborte la operación e informe el ciclo infinito para poder corregirlo. - No se pueden hacer cambios en tiempo de ejecución del intérprete, para hacer cambios en el código, se debe finalizar el interprete Opcional: - Presentar la opción paso a paso, para corrección de errores lógicos, ya que no deben existir semánticos. - Se puede pausar la operación, en cualquier momento mediante las teclas Ctrl + Pausa, y/o pasar a modo paso a paso.

Función	Móvil lógico
Atributos	- Presentar la opción de cambiar el intervalo de tiempo de la

	animación (velocidad), para de esta forma observar cómo sería el comportamiento del móvil en tiempo real. - Sincronizar el intervalo de tiempo del móvil lógico con el móvil físico, para lograr una animación sincronizada con el mundo real. Opcional: - Poder crear un ambiente al móvil, como obstáculos para simular choques. - Presentar una lista de móviles para que el usuario pueda cambiarlo - Se pueden importar nuevos móviles mediante archivos de extensión *.X - Especificar cómo crear nuevos móviles y/o personalizarlos

Función	Móvil físico
Atributos	- Mover adelante y atrás - Girar a la derecha e izquierda sobre un punto - Subir y bajar un lápiz ubicado en el centro del móvil de modo que se deje un rastro al moverse. - Recibirá las instrucciones desde el intérprete por vía inalámbrica. Opcional: - Sensor de distancia de un obstáculo - Sensor de colores - Sensor de obstáculos (avisa cuándo hay un choque)

Función	Programa principal
Atributos	- Presentación de bienvenida - Pantalla diseñada para una resolución de 1024 x 768, para un correcto uso - Pantalla inicial gráfica (acceso por iconos), con accesos: Ayuda Interactiva, Vista de Proyectos, Ejecución de proyectos

Función	Ejecución de proyectos
Atributos	- Mostrará paso a paso el interprete en una ventana de animación - Se animarán proyectos - - Se puede ingresar comandos directamente en la línea de comandos

3.2 ESTRUCTURA DEL LENGUAJE A IMPLEMENTAR

Las instrucciones que se utilizan en el lenguaje a implementar son basadas en el lenguaje LOGO y adaptadas al intérprete desarrollado; éste lenguaje fue bautizado

como Lenguaje de Programación Educativo – LPE puesto que permite por medio de órdenes básicas, incursionar al usuario al mundo de la programación.

Tabla 1. Estructura del Lenguaje a Implementar

COMANDO	DESCRIPCIÓN
AVANZA n *AV n* *ADELANTE n* *AD n*	El móvil se desplaza n unidades hacia adelante
ESCRIBE "xx xxx" *ES*	Escribe en la ventana de textos la palabra situada detrás de las comillas o la frase situada entre los corchetes
BORRAPANTALLA *BP*	Borra los gráficos que se hayan realizado y se coloca la tortuga en el centro de la pantalla (posición inicial)
GIRADERECHA n *GD n*	El móvil gira a la derecha n grados
GIRAIZQUIERDA n *GI n*	El móvil gira a la izquierda n grados
RETROCEDE n *RE n*	El móvil se desplaza n unidades hacia atrás
SUBELAPIZ *SL*	Hace que el móvil no deje rastro cuando avanza a través de la pantalla
BAJALAPIZ *BL*	Hace que el móvil deje rastro cuando avanza a través de la pantalla
CENTRA	Deja el móvil en el centro de la pantalla
PONGROSOR n *PONG n*	Cambia el grosor del lápiz al nivel n (n entre 1 y 10)
PONFONDO n *PONF n*	Cambia el color del fondo a n
PONCOLORLAPIZ n *PONCL n*	Cambia el color de la huella dejada por el lápiz a n
OCULTAMOVIL *OM*	Oculta el móvil
MUESTRAMOVIL *MM*	Muestra el móvil
BORRATEXTO *BT*	Borra el texto que haya en la pantalla de salida de textos
VARIABLE obj	Declara una variable
SI condición *[SINO SI condicion]* *[SINO]* *FINSI*	Esta sentencia significa que, si se cumple la condición, se ejecutan las acciones contenidas en el primer corchete y, en caso contrario, las incluidas en el segundo.
REPITE n *FINREPITE*	Hace que se repita n veces las acciones que se encuentran entre corchetes

PARA var=n HASTA K FINPARA	
MIENTRAS condicion FINMIENTRAS	Ejecuta las instrucciones mientras se cumpla condicion
SAL	Detiene la ejecución de un programa
FUNCION nombreprog sentencias FINFUNCION	Palabras claves para definir una nueva rutina que puede realizar el móvil.

3.3 DIAGRAMAS DE CASOS DE USOS

3.3.1 Usuario

Caso de Uso	Inicio de la aplicación
Actores	Usuario
Tipo	Primario
Descripción	El usuario inicia la aplicación, esta se abre por omisión con un nuevo proyecto abierto. Si el usuario abre la aplicación con la extensión del proyecto, el proyecto se abre y no se crea uno nuevo.

Caso de Uso	Personalizar el editor de texto
Actores	Usuario
Tipo	Secundario
Descripción	El cliente escoge del menú la utilidad correspondiente a personalizar el editor de texto, se despliega una nueva ventana y en ella se puede modificar el tamaño del texto, el color de la fuente de los diferentes tipos de texto existentes (palabras claves, texto común, cadenas y comentarios)

Caso de Uso	Trabajar en un proyecto
Actores	Usuario
Tipo	Primario
Descripción	El usuario corre el programa, ingresa a Vista de Proyectos y solicita crear o abrir un proyecto. El sistema crea/abre un nuevo archivo de texto. El usuario empieza a trabajar en el proyecto y guarda los cambios.
Parámetros	Color y tamaño de fuente para: Palabras claves

	Tipos Comentarios Texto Común Número de espacios para la tabulación. (Me parece que para el caso de la fuente, el tipo de fuente si debe ser la misma, y debería ser Courier New para un perfecto tabulado.)

Caso de Uso	Ejecutar un proyecto
Actores	Usuario
Tipo	Primario
Descripción	El usuario corre el programa, ingresa a Ejecución de Proyectos. El sistema abre el entorno de ejecución. El usuario especifica si desea paso a paso o no.

Caso de Uso	Abrir la ayuda
Actores	Usuario
Tipo	Primario
Descripción	El usuario ingresa al sistema y pide ayuda. El sistema abre el modo de Ayuda. El usuario busca por temas o por palabras claves.

Caso de Uso	Ejecutar una sentencia paso a paso
Actores	Usuario
Tipo	Primario
Descripción	El usuario debe habilitar la ventana de comandos para que pueda ejecutar instrucción por instrucción. En esta ventana no se puede ejecutar ciclos, ni condicionales, solamente sentencias válidas. También puede escribir el nombre del proyecto y sus parámetros y lo ejecuta. El control toma la sentencia y la valida léxicamente. Luego le válida la sintaxis y la semántica. Si no ha encontrado errores ejecuta la sentencia, de lo contrario muestra un mensaje de error en el control debajo de la línea escrita. Si al momento de ejecutar la sentencia la ventana de salida no está activa, ella se abre y se mantiene abierta, para futuras ejecuciones de sentencias.

Caso de Uso	Abrir ventana de salida
Actores	Usuario
Tipo	Primario
Descripción	La ventana de salida es la encargada de mostrar los pasos que se han programado. En un principio se habla de una ventana con animación 2D, pero la última versión implementará animación 3D.

	Cuando el usuario ejecuta un proyecto, esta ventana se abre y muestra la animación solicitada, simulando como sería el movimiento del móvil físico.
Parámetros	Tiempo de espera entre la ejecución de una sentencia y la próxima, o podría ser Velocidad de desplazamiento del móvil lógico. Escenario de fondo Escogencia de un móvil lógico.

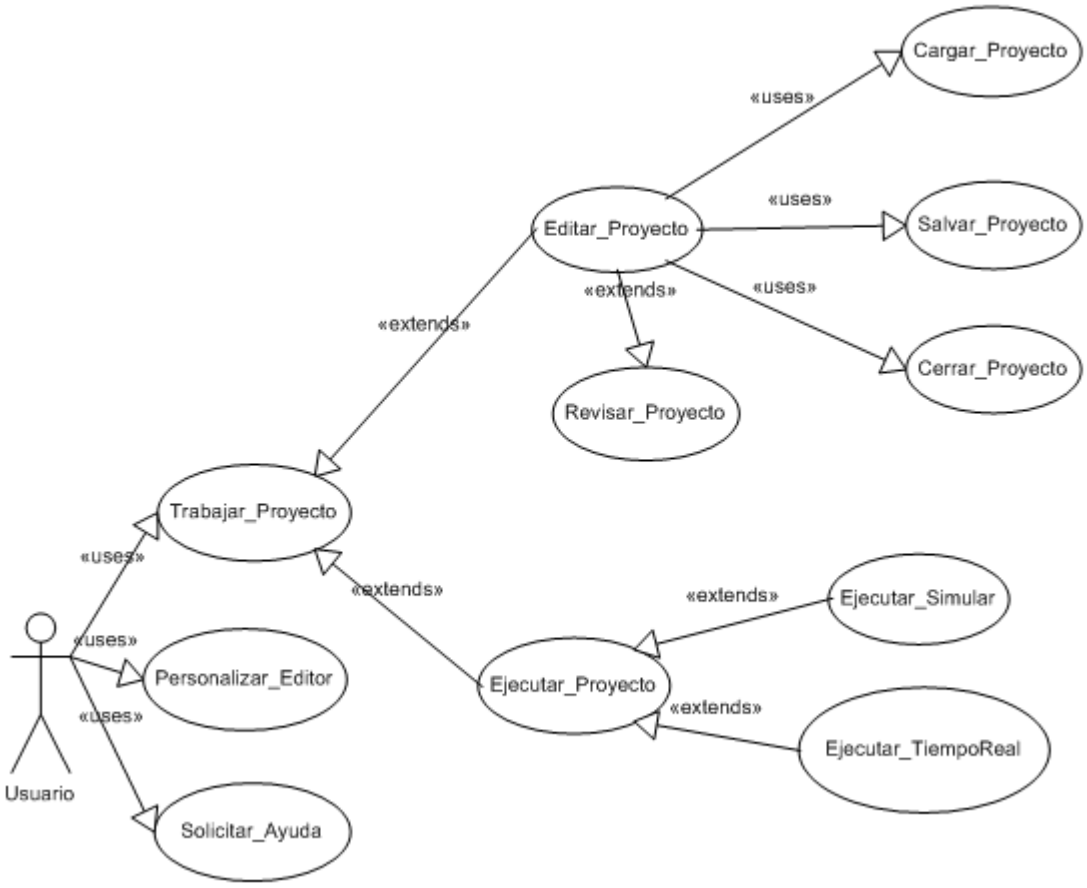


Figura 10. Caso de Uso para el Usuario

3.3.2 Sistema

Caso de Uso	Análisis Léxico
Actores	Sistema
Tipo	Primario
Descripción	Verifica que todos los términos escritos sean conocidos.

Caso de Uso	Análisis sintáctico
Actores	Sistema
Tipo	Primario
Descripción	Verifica que las palabras ingresadas sean empleadas adecuadamente o que no falte una palabra, si al finalizar el proceso encuentra un error de esta categoría muestra el mensaje correspondiente y en ocasiones debe sugerir la solución.

Caso de Uso	Análisis semántico
Actores	Sistema
Tipo	Primario
Descripción	Verifica los cast, e impide ejecutar un programa con conversiones erróneas.

Caso de Uso	Traductor
Actores	Sistema
Tipo	Primario
Descripción	Después de tener un texto válido, lo traduce a un lenguaje intermedio para que ejecute las funciones solicitadas por el usuario.

Figura 11. Caso de Uso para el Sistema

3.3.3 Móvil físico

Caso de Uso	Recibir Señal
Actores	Móvil Físico
Tipo	Primario
Descripción	El móvil debe estar atento a las señales que le envíe el sistema y procesarlas para realizar la operación solicitada

Caso de Uso	Bajar lápiz
Actores	Móvil Físico
Tipo	Primario
Descripción	El móvil bajará el lápiz que tiene en su interior con la ayuda de un motor.

Caso de Uso	Subir lápiz
Actores	Móvil Físico
Tipo	Primario
Descripción	El móvil subirá el lápiz que tiene en su interior con la ayuda de un motor.

Caso de Uso	Avanzar
Actores	Móvil Físico
Tipo	Primario
Descripción	El móvil girará sus ruedas para realizar un movimiento de avance.

Caso de Uso	Retroceder
Actores	Móvil Físico
Tipo	Primario
Descripción	El móvil girará sus ruedas para realizar un movimiento de retroceso.

Caso de Uso	Girar
Actores	Móvil Físico
Tipo	Primario
Descripción	El móvil girará sus ruedas en sentido opuesto una de la otra para realizar un giro sobre su eje.

Caso de Uso	Mostrar Cara
Actores	Móvil Físico
Tipo	Primario
Descripción	El móvil recibirá una señal que le indica el estado de encendido o apagado de cada uno de los leds que componen la cara y encenderá o apagará los leds según como se solicite.

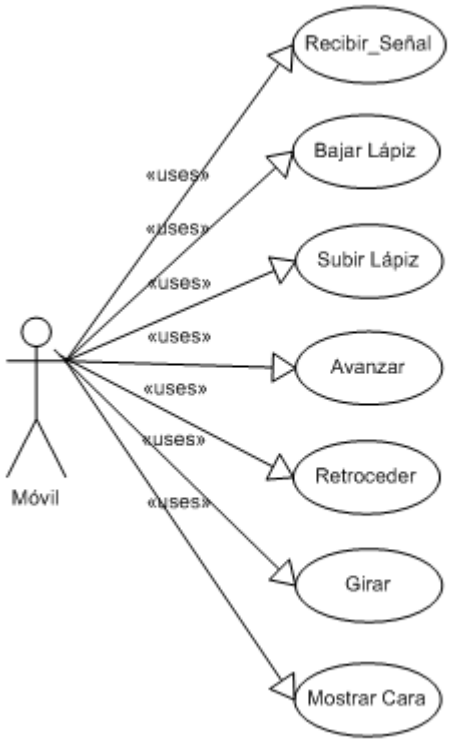


Figura 12. Caso de Uso para el Móvil

3.4 ENTREGAS EVOLUTIVAS

De acuerdo con la metodología elegida, evolutiva, se desarrolló el software del presente proyecto a partir de la presentación de módulos que se fueron reuniendo para dar resultado al entorno de programación final, a continuación se presenta el desarrollo de cada uno de los módulos que llevaron a la elaboración de la aplicación final.

3.4.1 Módulo analizador

Para el desarrollo de esta etapa se tomó como base la gramática del lenguaje LOGO en español, para a partir de ésta generar un dialecto cuya definición se realizó bajo la notación BNF[33], que es fundamental para la delimitación y caracterización de la estructura de un lenguaje.

A la hora de construir el analizador léxico y sintáctico, se vio la necesidad de construir una aplicación que generara el código fuente en Visual Basic .Net 2005 de tal forma que al hacer un cambio en el BNF no fuera necesario la

[33] Ver numeral 4.1

reprogramación del módulo analizador sino que se utilizara dicho generador y se creara nuevamente de forma automática la librería que sería usada desde el sistema final.

3.4.2 Entorno de visualización en 2D

Luego de tener implementado el módulo del analizador de código, se procedió al desarrollo de un entorno de ejecución del lenguaje LPE que permitiera la visualización de los resultados que debía mostrar el móvil lógico.

El entorno elaborado permitía la ejecución de los comandos básicos de LPE y cómo se aprecia a continuación no mostraba tipo alguno de móvil, se consideraba como punto de inicio el centro del panel de dibujo y a partir de ahí se iban dibujando los rastros que dejaría el móvil en el caso de estar el lápiz abajo.

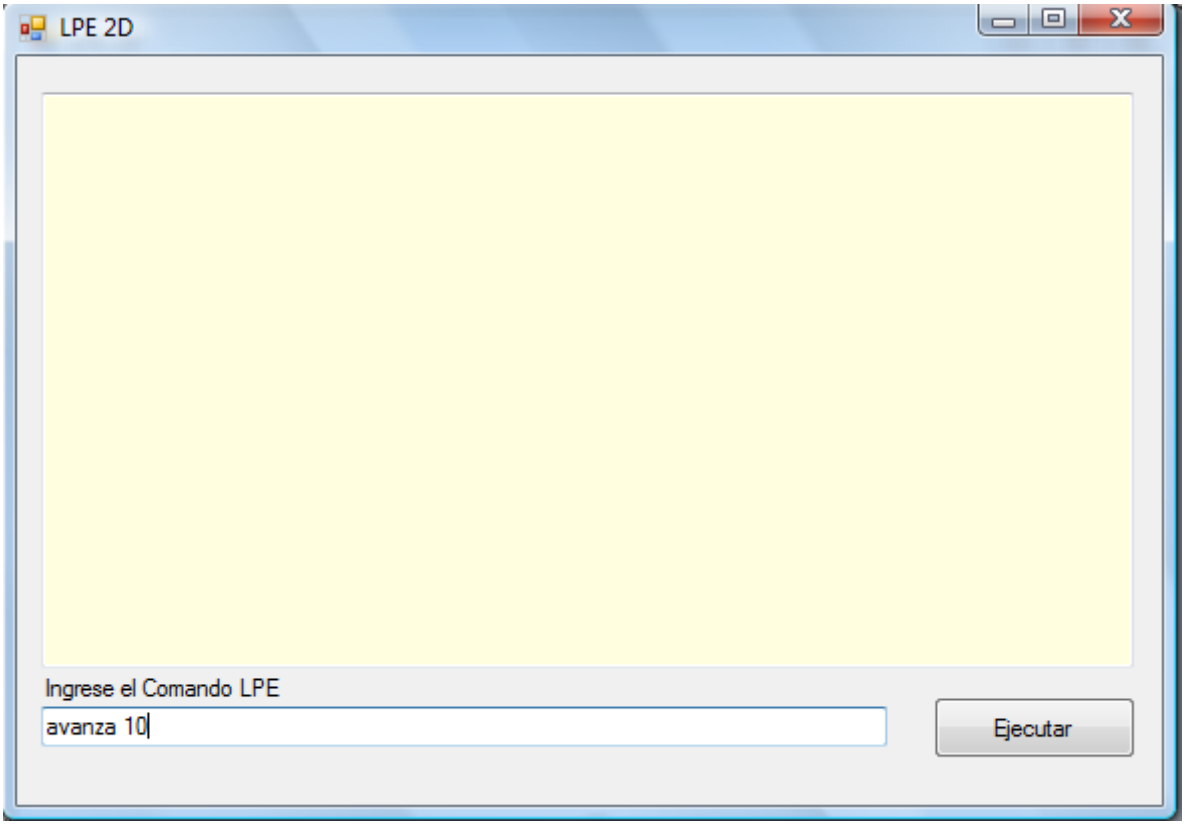


Figura 13. Interfaz LPE en 2D

3.4.3 Entorno de visualización en 3D

Uno de los módulos más importantes es el visualizador en 3D, para este paso, se desarrolló un componente en el SDK de DirectX 10, de modo que recibiera unos parámetros de la aplicación principal y mostrara los pasos solicitados; entre los parámetros y órdenes que reciben están:

- ✓ Desplazarse adelante o hacia atrás
- ✓ Realizar un giro de n grados a la derecha o a la izquierda
- ✓ Dejar un rastro a su paso
- ✓ Grosor del rastro que se deja
- ✓ Color de fondo del escenario
- ✓ Color del rastro del lápiz
- ✓ Modelo del móvil a utilizar

Con el apoyo de los compañeros de Diseño Industrial quienes desarrollaron diferentes modelos en 3D, se ubicó un móvil en el centro del entorno tridimensional, estos fueron desarrollados con la herramienta 3D Studio Max y con el apoyo de un plugin que transforma ficheros *.3ds a *.X de la empresa Pandasoft[34] fue posible hacer la integración de los móviles realizados. Los posibles protagonistas se muestran a continuación:

Figura 14. Móvil en forma de Caracol

Figura 15. Móvil en forma de Rana

[34] Mayor información en http://www.andytather.co.uk/Panda/directxmax.aspx

Figura 16. Móvil en forma de Caracol 2

Figura 17. Móvil en forma de Escarabajo

3.4.4 Comunicación con el móvil físico

El grupo E.R.A. de la Escuela de Ingeniería Eléctrica y Electrónica de la UIS con el apoyo de los estudiantes de Diseño Industrial que diseñaron los modelos en 3D, elaboraron un robot que tiene las características del móvil en forma de escarabajo presentado en la figura 17 y es el encargado de mostrar en la realidad los movimientos que se programen mediante LPE.

Para crear el enlace entre el robot y la aplicación, se desarrolló un módulo que convierte los comandos que es capaz de realizar el móvil físico en señales que son enviadas a través del puerto serial del computador, estas señales las recibe una antena conectada al puerto y luego por vía inalámbrica son transmitidas al robot. Los comandos y su representación en señales por el puerto serial, se muestran en los anexos del presente proyecto.

Entre las características que más llamaron la atención en los niños fue la presencia de una pantalla con base en leds que permiten la edición en pantalla de una figura gestual del robot el cual fue llamado LOUIS, por basarse en la filosofía de LOGO y es una versión creada en la Universidad Industrial de Santander – UIS.

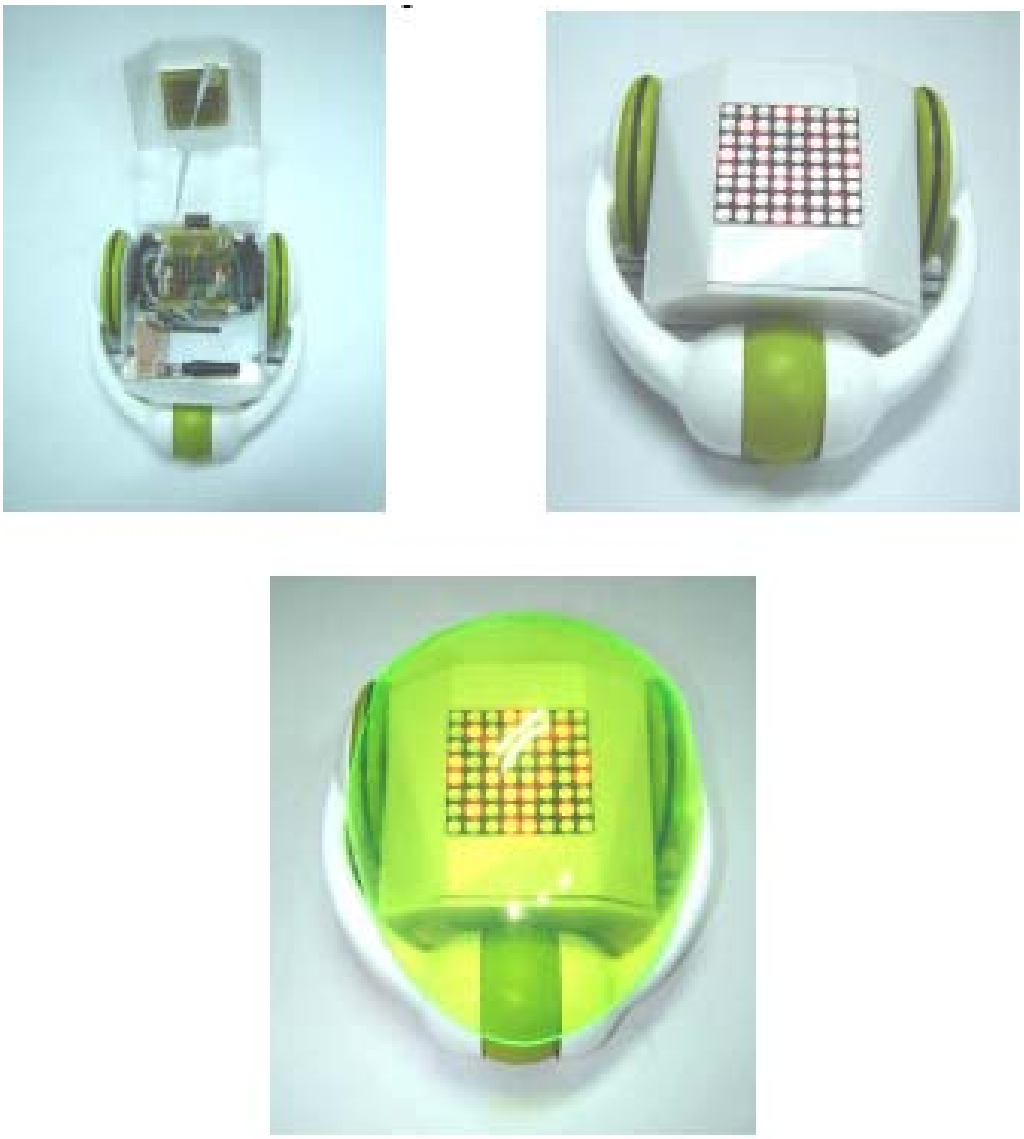


Figura 18. Móvil físico implementado

3.4.5 Interfaz gráfica

La interfaz gráfica de usuario es el medio que posibilita la interacción entre un sistema interactivo y su usuario a través del uso y representación de un lenguaje visual, permite la mostrar la información y acciones disponibles en el programa informático.

Los usuarios a los que está dirigido este lenguaje educativo son principalmente los niños entre 7 y 12 años, y por esto se buscó que la interfaz final fuera llamativa y

fácil de usar. El diseño de la interfaz fue orientado por los compañeros de Diseño Industrial quienes propusieron tres modelos de interfaz y fueron los mismos usuarios quienes las juzgaron y dieron recomendaciones[35].

La implementación de la interfaz se realizó utilizando las herramientas disponibles en Visual Studio .Net 2005 de manera que se ajustaran lo mejor posible al diseño propuesto, consiguiendo integrar los módulos desarrollados en las entregas anteriores y tener una entrega final disponible para el usuario.

[35] Ver capítulo 6

4. RESULTADOS

4.1 DISEÑO DEL LENGUAJE

La definición del lenguaje se realizó utilizando la nomenclatura BNF, a continuación se muestra su programación:

Tabla 2. Definición BNF de LPE

```
' ***** Definición BNF de LPE *****
'
' --- Programa fuente
programaFuente := [NuevaLinea] definicionModulo +
codigoAbierto := [NuevaLinea] [ sentencia delimitador ] *
delimitador := NuevaLinea |
               DosPuntos
sentencia := sentenciaControl |
             comando |
             comandoMovil |
             asignacion
sentenciaControl := declaracion |
                    si |
                    repite |
                    para |
                    mientras |
                    segun
comandoMovil := avanza |
                giraDerecha |
                giraIzquierda |
                retrocede |
                subeLapiz |
                bajaLapiz |
                centro |
                ponGrosor |
                ponFondo |
                ponColorLapiz |
                ocultar |
                mostrar
comando := borraPantalla |
           borraTexto |
           comentario |
           salir |
           salirBucle |
           leer |
           imprimir |
           retorno |
         llamarFuncion
' --- Sentencias
' Asignación
asignacion := variable "=" expresion
variable := identificador [ "(" listaIndices ")" ]
listaIndices := expresion [ Coma listaIndices ]
' --- Sentencias
' Avanzar
avanza := [ "AVANZA" | "AV" | "ADELANTE" | "AD" ] "(" expresion ")"
```

```
' Girar
giraDerecha := [ "GIRADERECHA" | "GD" ] "(" expresion ")"
giraIzquierda := [ "GIRAIZQUIERDA" | "GI" ] "(" expresion ")"
retrocede := [ "RETROCEDE" | "RE" | "ATRAS" | "AT" ] "(" expresion ")"
subeLapiz := "SUBELAPIZ" | "SL"
bajaLapiz := "BAJALAPIZ" | "BL"
centro := "CENTRO"
ponGrosor := [ "PONGROSOR" | "PONG" ] "(" expresion ")"
ponFondo := [ "PONFONDO" | "PONF" ] "(" expresion ")"
ponColorLapiz := [ "PONCOLORLAPIZ" | "PONCL" ] "(" expresion ")"
ocultar := "OCULTAMOVIL" | "OM"
mostrar := "MUESTRAMOVIL" | "MM"
' Borra Pantalla
borraPantalla  := "BORRAPANTALLA" | "BP"
' Borra Texto
borraTexto  := "BORRATEXTO" | "BT"
' Comentario:
comentario :=  Apostrofe AnythingExceptDoubleQuote NuevaLinea
' Declaración
declaracion := tipoDato identificadorVariable [ Coma identificadorVariable ] *
identificadorVariable := identificador [ "(" dimensionArreglo ")" ]
tipoDato := "LOGICO" |
            "ENTERO" |
            "FLOTANTE" |
            "SARTA" |
            "CARACTER"
dimensionArreglo := entero [ Coma dimensionArreglo ]
' si
si := siEncabezado
      [ sinosi ]
      [ sino ]
      finsi
siEncabezado := "SI " expresion NuevaLinea codigoAbierto
sinosi := "SINOSI " expresion NuevaLinea codigoAbierto [ sinosi ]
sino := "SINO " expresion NuevaLinea codigoAbierto
finsi := "FINSI"
' Repite
repite := repiteEncabezado codigoAbierto finRepite
repiteEncabezado := "REPITE "  NuevaLinea
finRepite := "HASTA " expresion
' Mientras
mientras := mientrasEncabezado codigoAbierto finMientras
mientrasEncabezado := "MIENTRAS " expresion NuevaLinea
finMientras := "FINMIENTRAS"
' Para
para := paraEncabezado codigoAbierto finPara
paraEncabezado := "PARA " variable "=" expresion "HASTA " expresion NuevaLinea
finPara := "FINPARA"
' Segun
segun := segunEncabezado segunOpcion+ [ segunOpcionOtro ] finSegun
segunEncabezado := "SEGUN " expresion NuevaLinea
segunOpcion := "CASO " constante codigoAbierto
segunOpcionOtro := "OTROCASO" codigoAbierto
finSegun := "FINSEGUN"
' Declaracion de salida
salir := "SALIR"
salirBucle := "SALIRBUCLE"
' Lectura
leer := Lee listaVariables
listaVariables := variable [ Coma listaVariables ]
```

```
' Impresion en pantalla
imprimir := Escribe lista_expresiones
lista_expresiones := expresion [ Coma lista_expresiones ]
' retorno de un procedimiento o función
retorno := retornoFuncion | retornoProcedimiento
retornoFuncion := Retorne "(" expresion ")"
retornoProcedimiento := Retorne

' --- Expresiones
expresion := termino [ operacion termino]*
operacion := orOp |
          andOp |
          RelOp |
          AddOp |
          MulOp |
          "&" |
          Not termino
termino := numero_sin_signo |
           sarta |
           variable |
           llamarFuncion |
          "(" expresion ")"
llamarFuncion := nombreFuncion "(" [ lista_expresiones ] ")"
nombreFuncion := Abs | Coseno | Largo | Log | Aleatorio | Seno | Mayuscula |
Minuscula | identificador
identificador := Letra LetrasNumerosUnderscores*
orOp := Or
andOp := And
RelOp := "<" | ">" | "<=" | ">=" | "=" | "<>"
AddOp := "+"|"-"
MulOp := "*"|"/"|"\"|"Mod"

' --- Procedimientos y Funciones
definicionModulo := definicionProcedimiento | definicionFuncion
definicionProcedimiento := Procedimiento identificador "(" [ listaParametros ]
")"
                           codigoAbierto FinProcedimiento NuevaLinea
definicionFuncion := Funcion identificador "(" [ listaParametros ] ")" tipoDato
                     codigoAbierto FinFuncion NuevaLinea
listaParametros := definicionFormalParametro [ "," listaParametros ]
definicionFormalParametro := tipoDato identificador

' Datos
constante := sarta | numero
numero := entero | real
numero_sin_signo := entero_sin_signo | real_sin_signo
signo := "+" | "-"
entero := [ signo ] entero_sin_signo
real := [ signo ] real_sin_signo
sarta := Comilla AnythingExceptDoubleQuote Comilla
entero_sin_signo := Num +
real_sin_signo := entero_sin_signo "." entero_sin_signo
Num := 1|2|3|4|5|6|7|8|9|0
NuevaLinea := ( chr10 | chr13 chr10 ) +
DosPuntos := ":"
Coma := ","
Apostrofe := "'"
Comilla := """
Or := "o"
And := "y"
```

```
Not := "no"
Abs := "abs"
LetrasNumerosUnderscores := Letra | "_" | Num
Letra := A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z
```

4.2 GENERADOR DE ANALIZADOR

Se desarrolló una herramienta en Visual Basic .Net 2005 para verificar un BNF y generar su analizador léxico y sintáctico, ésta utilidad toma el BNF del lenguaje y como resultado se obtiene el código fuente en Visual Basic .Net 2005; a partir de esta herramienta se puede generar el analizador para cualquier lenguaje del que se tenga el BNF.

Para el desarrollo de ésta herramienta se utilizaron las librerías de Regex que son una implementación en Microsoft Visual Studio de los clásicos analizadores yacc y lex, para mayor información del uso de regex, el libro de Edward Nigles referenciado en la bibliografía explica su manejo.

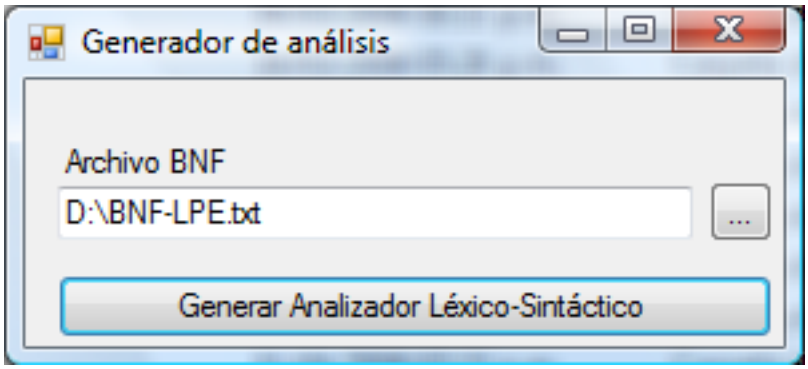


Figura 19. Pantalla inicial del Generador del Analizador

El BNF ingresado corresponde al del lenguaje LPE (ver *Tabla 2*), y al realizar la operación de generación del analizador léxico-sintáctico, el generador presenta como resultado una recopilación de los estados iniciales y finales del autómata que reconoce la gramática del lenguaje LPE y la generación del código en Visual Basic .net 2005 del analizador que se encarga de revisar el código LPE (*ver Anexo 3*), esta salida se muestra en la figura 20.

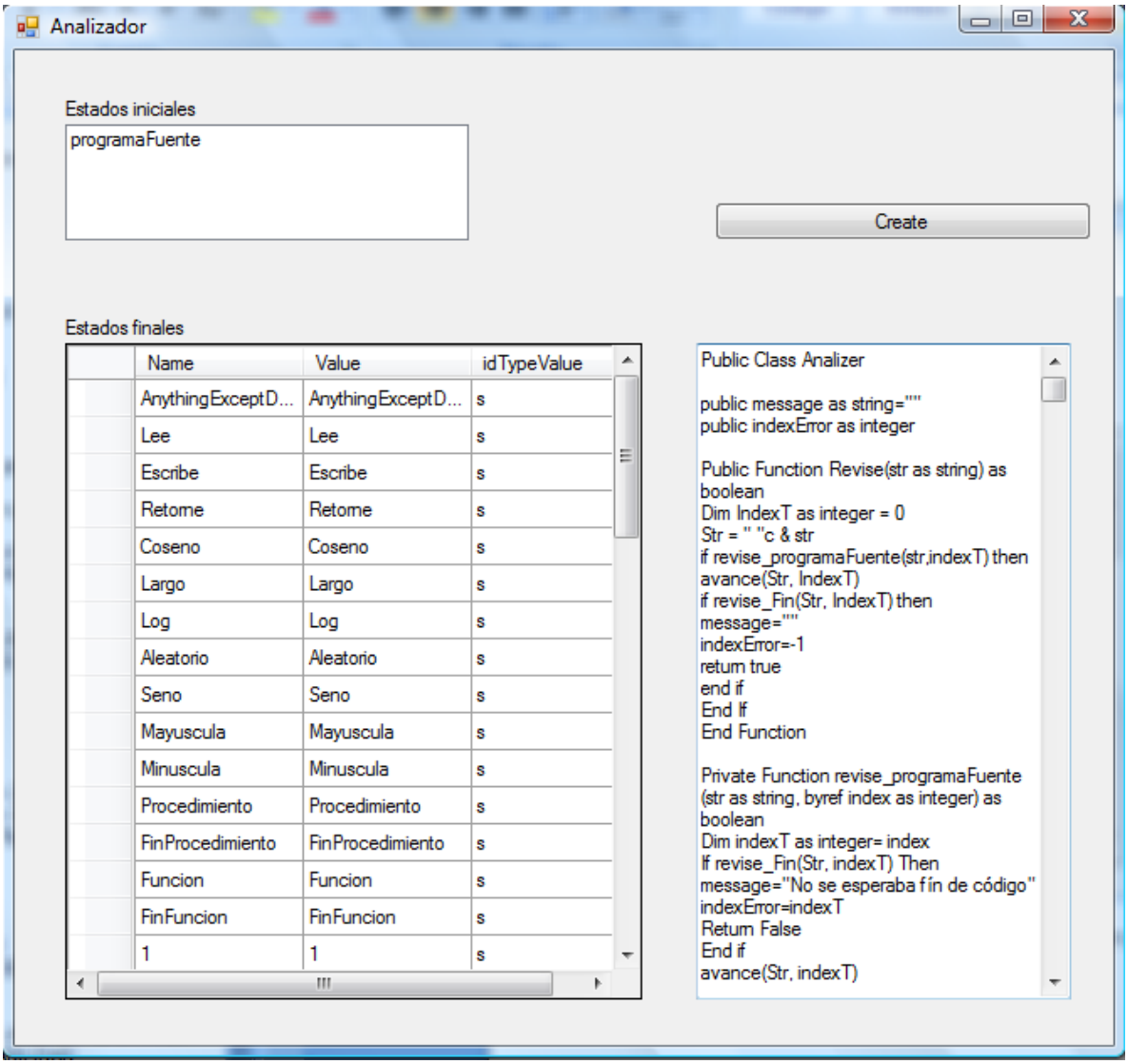


Figura 20. Código fuente para el analizador

Además de la salida presentada en la figura 20, el generador del analizador realiza un análisis del BNF y crea un manual de referencia para el lenguaje LPE compuesto por los símbolos no-terminales que el programa encontró declarados en el BNF y muestra las máquinas de estado finito que componen el autómata que reconoce la gramática del lenguaje.

En las figuras 21, 22 y 23 se muestra la ventana de salida para el análisis realizado por el generador, compuesta por diferentes pestañas, desglosadas así:

1. Resumen con los símbolos no-terminales que el programa encontró declarados en el BNF y que constituyen el manual de referencia del lenguaje[36]. Ver figura 21.

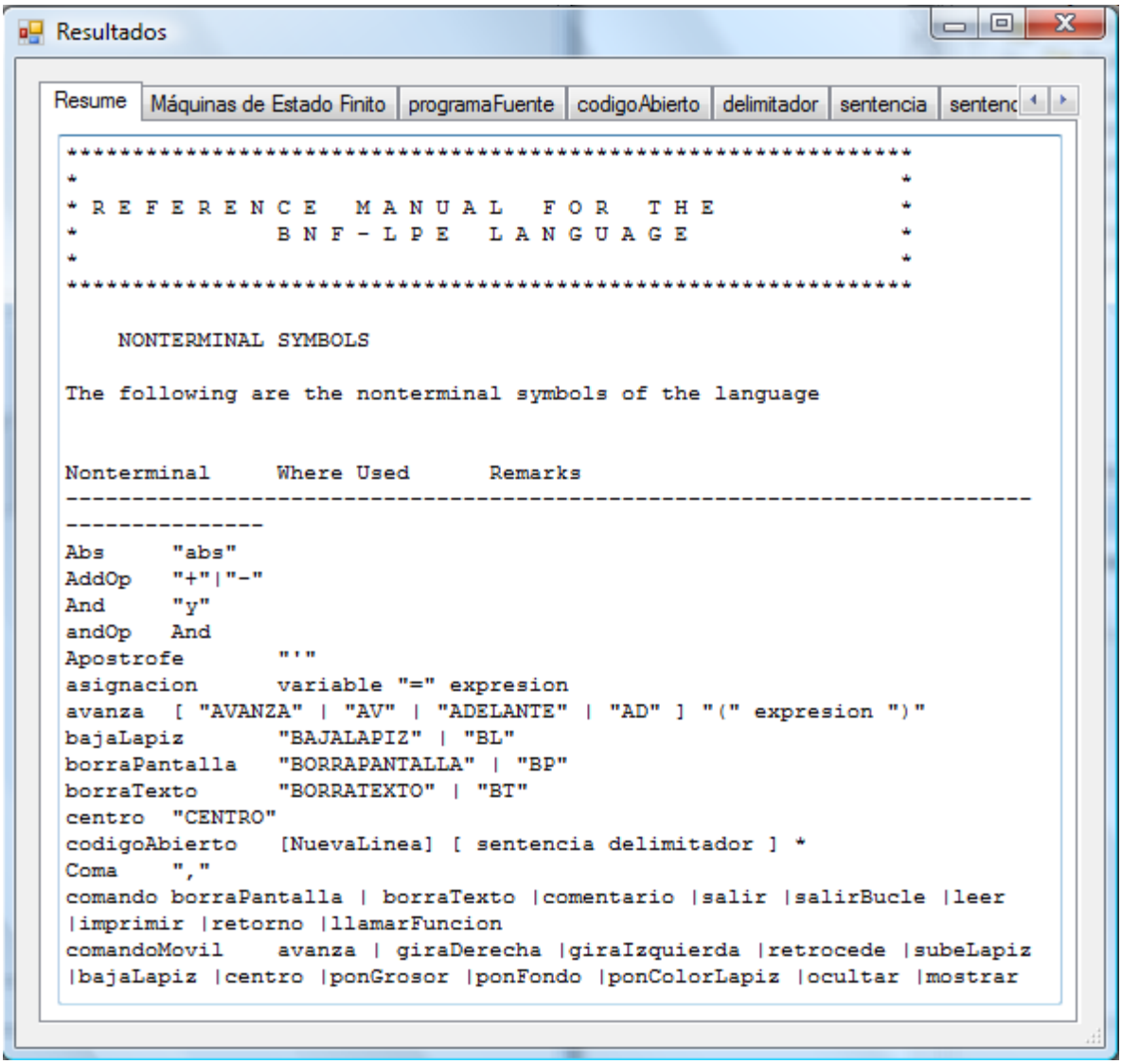


Figura 21. Referencia del Lenguaje LPE

2. En la siguiente pestaña se presentan el total de máquinas de estado finito que constituyen el autómata que reconoce la gramática del lenguaje LPE, que para este caso son 94 máquinas[37]. Ver figura 22.

[36] Ver anexo 4

[37] Ver anexo 2

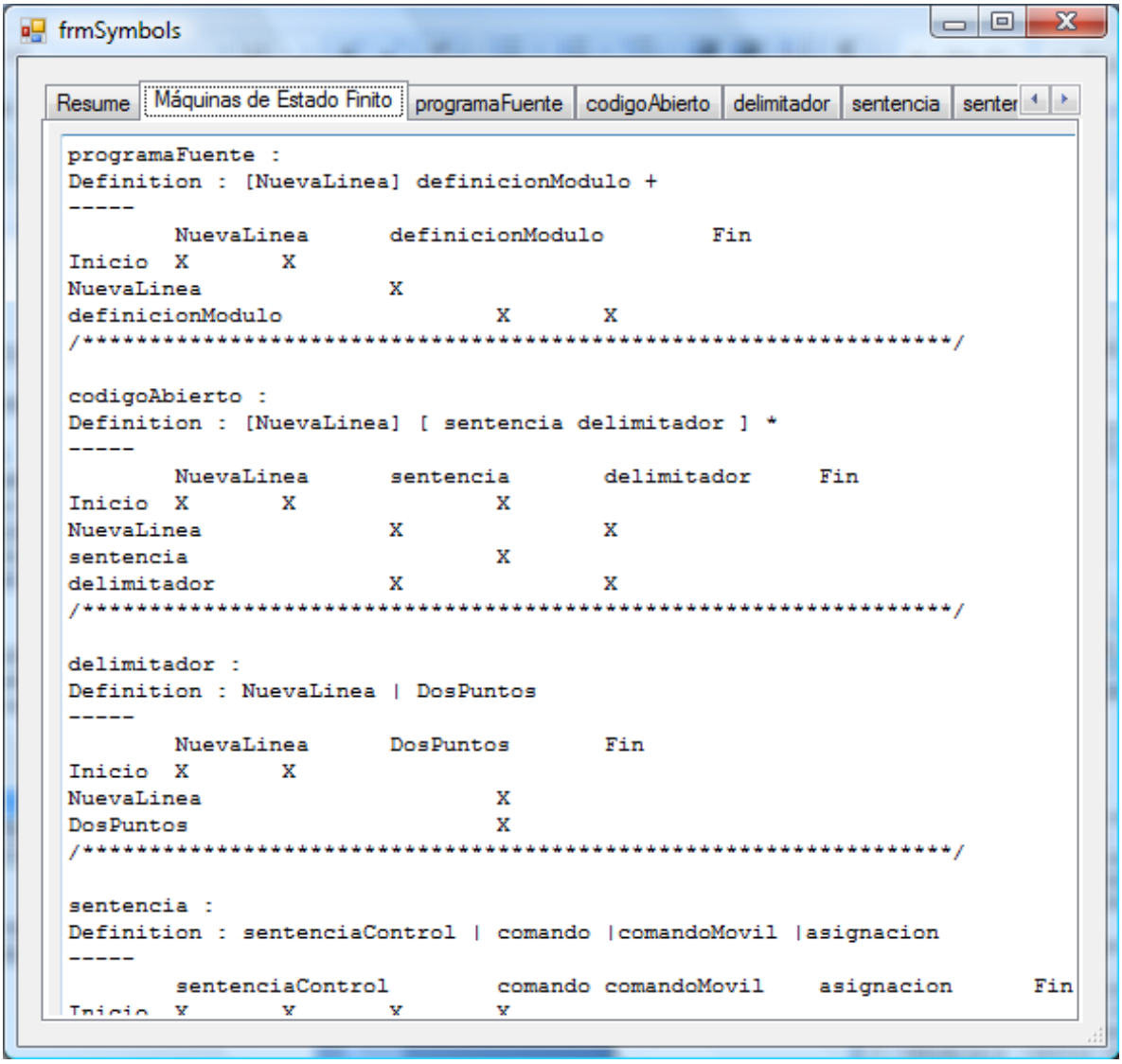


Figura 22. Máquinas de Estado Finito del Lenguaje LPE

3. Las siguientes pestañas desglosan las máquinas de estado finito recolectadas en la segunda pestaña, en total 94. Ver figura 23.

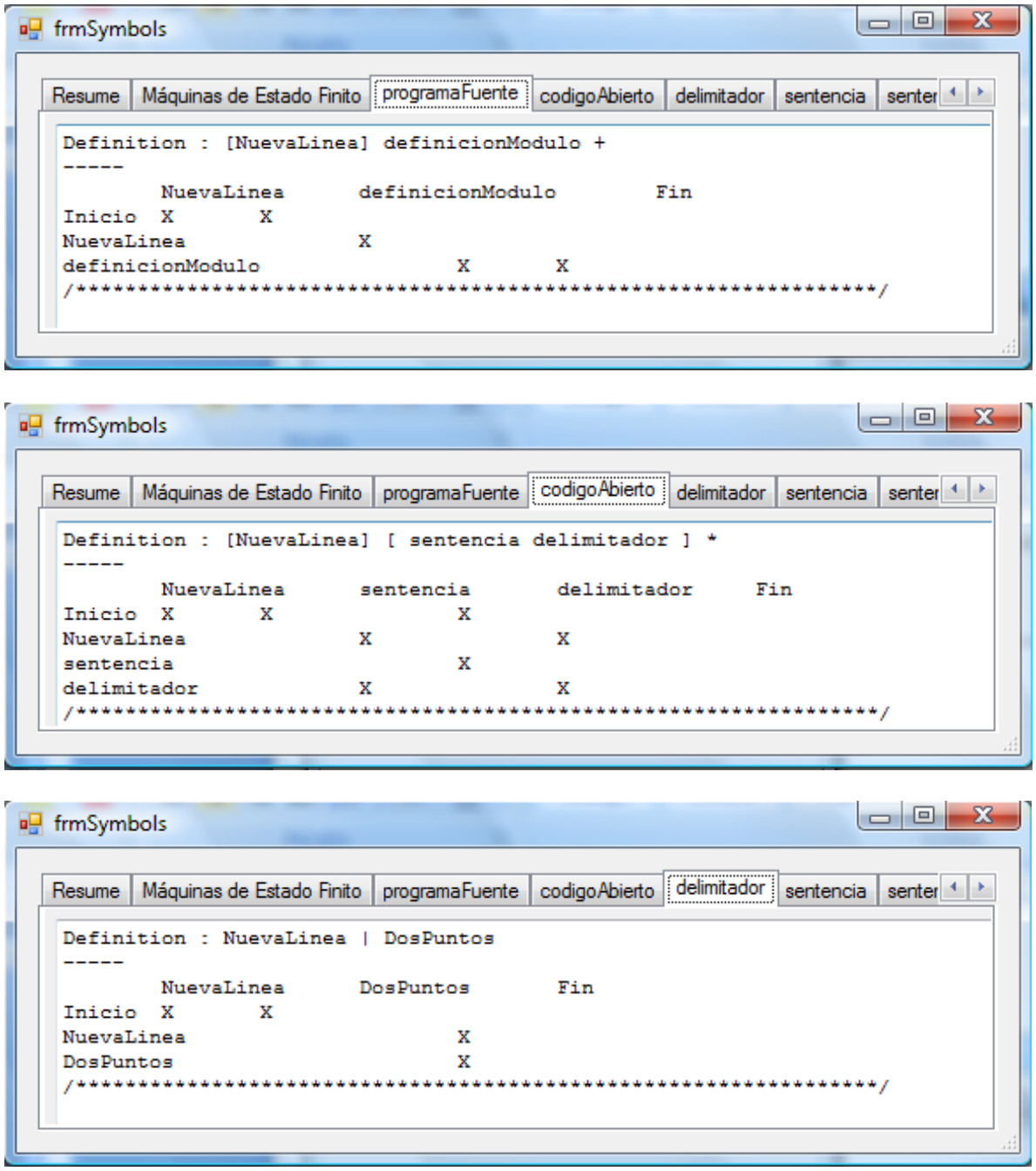


Figura 23. Algunas Máquinas de Estado Finito del Lenguaje LPE

4.3 INTERFAZ DE VISUALIZACIÓN DE LPE

Como resultado de las pruebas realizadas con estudiantes, se desarrolló junto a los compañeros de Diseño Industrial una interfaz con bastante usabilidad y de fácil comprensión especialmente para los niños entre 7 y 12 años.

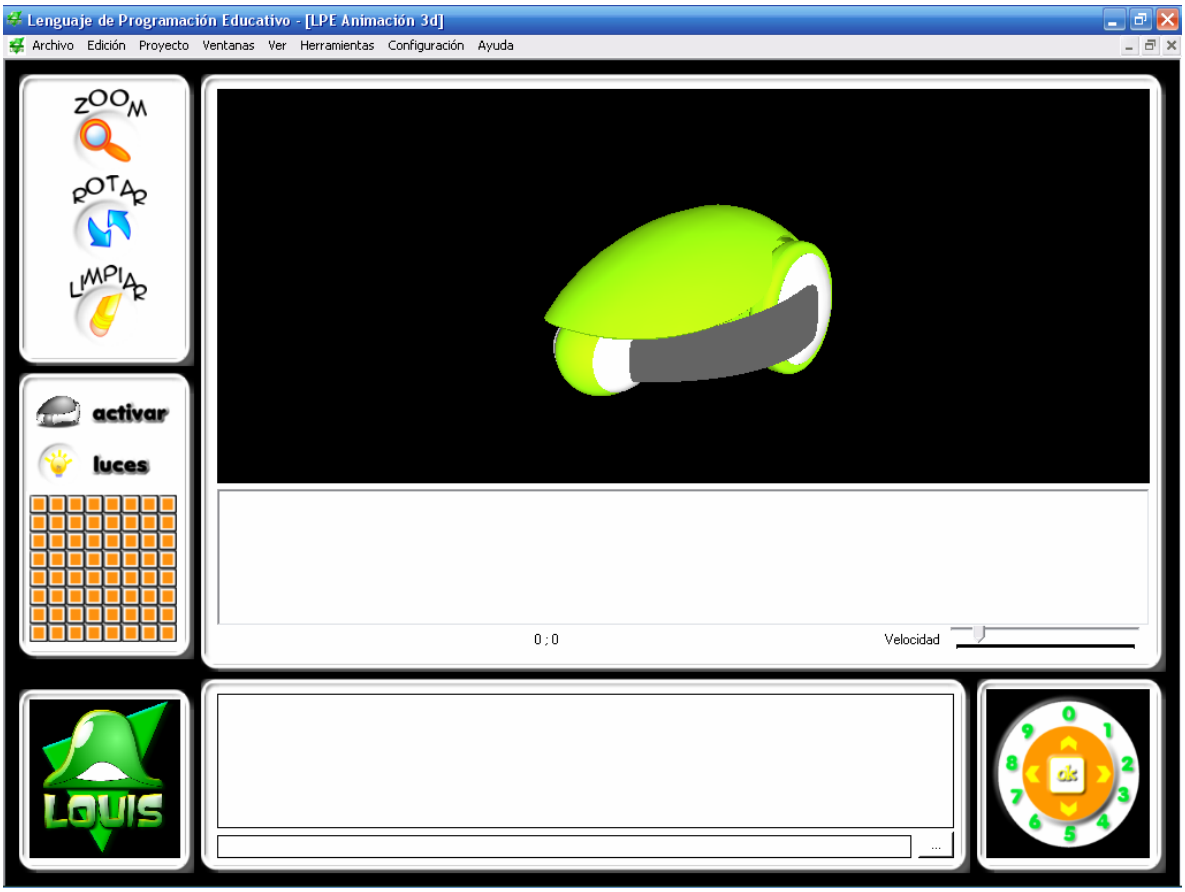


Figura 24. Interfaz final de visualización

4.4 EDITOR DE PROYECTOS

Con el fin de permitir al usuario la programación de código LPE, se desarrolló un editor de texto con diferentes opciones como cambios de color, tipos de fuente, edición, revisión del analizador y ejecución del proyecto.

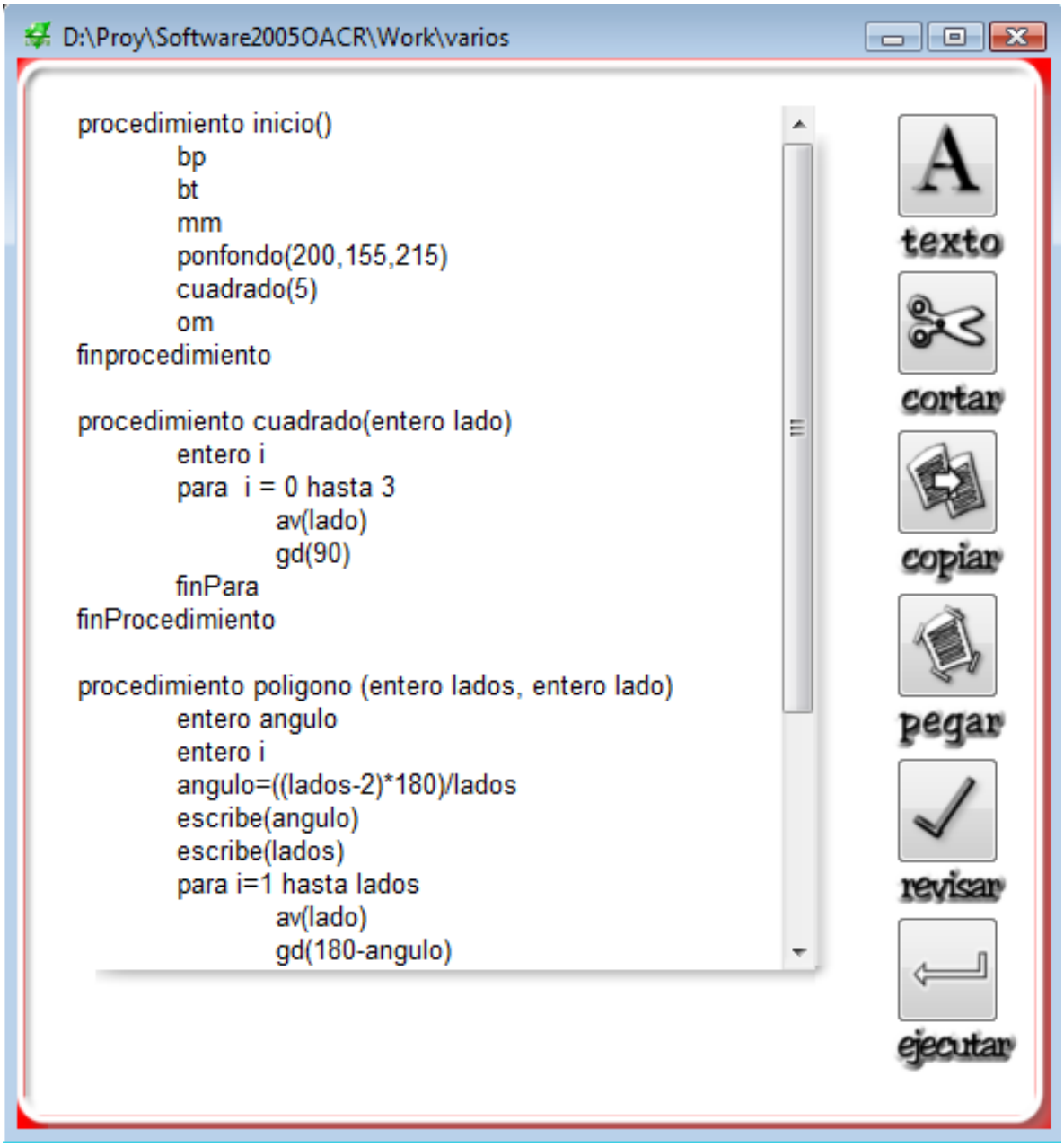


Figura 25. Editor de Proyectos LPE

4.5 ESTRUCTURA GENERAL DEL DESARROLLO

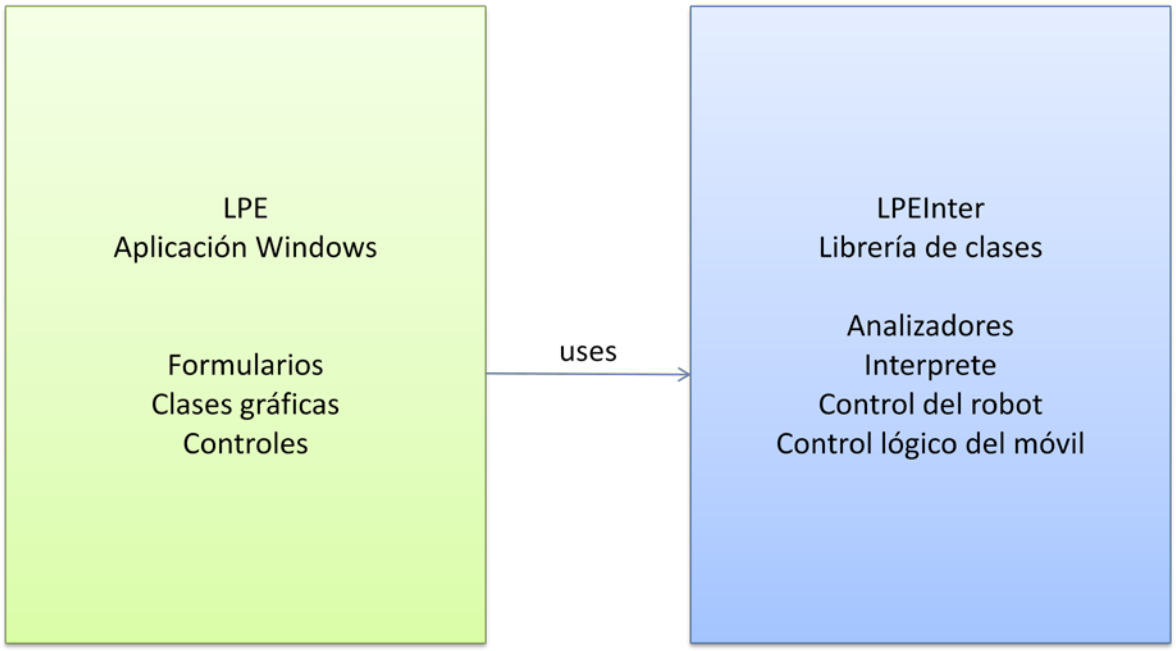


Figura 26. Estructura General del Desarrollo

La solución fue implementada bajo la plataforma .Net, dicha solución se compone de dos proyectos principales que dan la funcionalidad al sistema, y un proyecto de instalación:

- LPE (VB.Net Windows Application)
- LPEInter (VB.Net Dynamic Link Library)
- SetupLPE (Setup Project)

4.5.1 Definición del proyecto LPE

Este proyecto está compuesto por:

- 5 Formularios Windows
- 1 Módulo (Encargado de variables globales)
- 3 Controles
- 5 Clases, encargadas del control gráfico de la aplicación

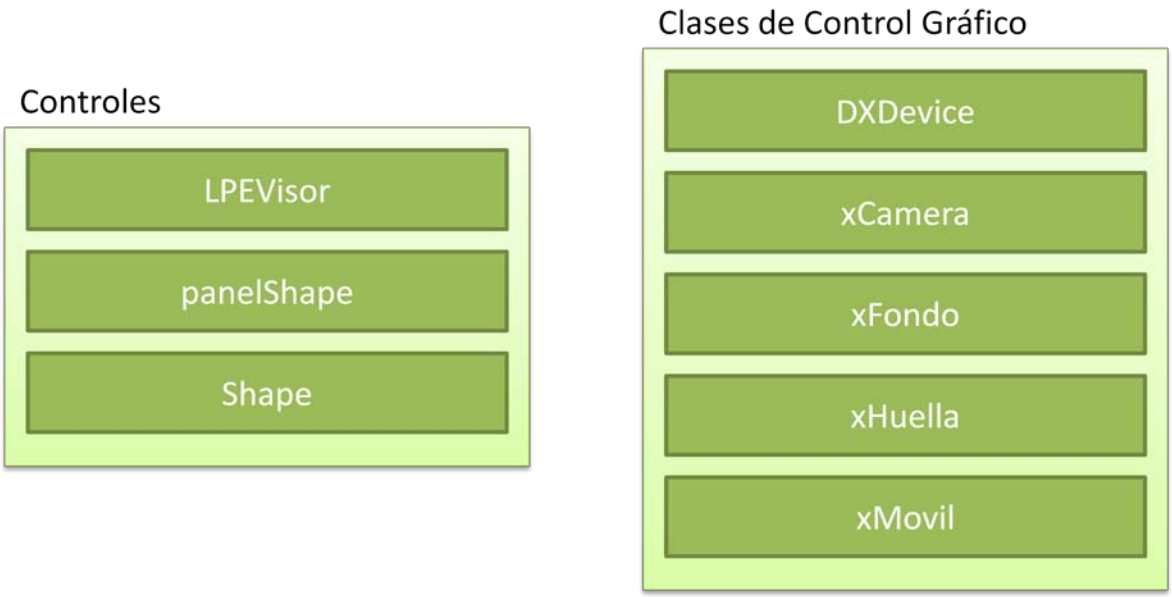


Figura 27. Definición del Proyecto LPE

Los controles creados son los siguientes:

- **LPEVisor.** Control encargado de visualizar las animaciones realizadas en DirectX en un entorno tridimensional. Las principales funciones son:
  - **setZoom.** Establecer el zoom (0 a 100)
  - **getZoom.** Retorna el valor de zoom
  - **Velocidad.** Propiedad que establece y retorna la velocidad de animación.
  - **IniGraphic.** Procedimiento encargado de inicializar el entorno gráfico y disponer de las clases necesarias para trabajar con DIrectX
  - **Pinte.** Procedimiento encargado de coordinar la animación 3D
  - **Execute.** Procedimiento que a partir de un comando anima el móvil lógico.
- **Shape.** Panel con bordes curvos, para mejorar la presencia de los formularios
- **panelShape.** Clase heredada de shape, que vincula la forma curva en un formulario, para mejorar la experiencia visual del aplicativo.

Las clases creadas son:

- **DXDevice.** Encargada de todo el control de la animación 3D en DirectX. Sus funciones principales son:
  - **setupCamera.** Establece la posición inicial de la cámara (observador)
  - **render.** Procedimiento encargado de las luces y entorno del escenario, así como la ubicación de los sprites.

    - **AttempRecovery.** Función encargada de recuperar el hilo con el que se está trabajando la animación, dado que exista un problema.
    - **LoadSprite.** Procedimiento encargado de cargar un sprite en el entorno establecido, sí como de establecer su textura.
    - **SetupLights.** Procedimiento encargado de establecer las luces del entorno
- **xCamera.** Encargada del control de la cámara del escenario creado. Principales funciones:
    - **lookAt.** Propiedad encargada de establecer o retornar el vector (x,y,z) hacia donde la cámara enfoca.
    - **Alpha.** Propiedad encargada de establecer o retornar el ángulo de la cámara en el plano x,z
    - **Betha.** Propiedad encargada de establecer o retornar el ángulo formado entre el eje "y" y el ojo del observador (cámara)
- **xFondo.** Encargada del control del fondo. Se puede implementar un entorno más llamativo en esta clase.
    - **CreateMesh.** Procedimiento encargada de inicializar el entorno.
    - **Render.** Procedimiento encargado de pintar el entorno y de aplicar las transformadas a la matriz de posicionamiento de los sprites.
- **xHuella.** Manipula el pintado del rastro del móvil en el escenario
    - **setPosicion.** Procedimiento encargado de establecer las posiciones (inicio y fin) de cada rastro. Se entiende por rastro a cada línea recta trazada por el móvil lógico.
    - **Render.** Procedimiento encargado de pintar la imagen en el entorno.
- **xMovil.** Manipula el pintado del móvil en el escenario
    - **CreateMesh.** Procedimiento encargada de inicializar el móvil lógico
    - **setPosition.** Establece la posición del objeto móvil en un plano cartesiano.
    - **Render.** Procedimiento encargado de pintar el objeto móvil en el entorno. También verifica el estado del objeto móvil, ya que si está oculto, no se debe mostrar en pantalla

4.5.2 Definición del proyecto LPEInter

Este proyecto está constituido por 8 clases distribuidas de la siguiente manera:

- 3 clases encargadas de los analizadores y del intérprete
- 4 clases encargadas del manejo del móvil lógico
- 1 clase encargada del móvil físico.

Figura 28. Definición del Proyecto LPEInter

A continuación se relacionan las clases creadas, junto con su descripción y principales funciones, procedimientos y/o propiedades implementadas.

- **Analizador.** Encargado del analizador léxico y sintáctico. El código de esta clase es generado automáticamente. Dentro de su funcionalidad está la de dividir el código en tokens y verificar que cumpla con la sintaxis definida
- **Preprocesador.** Analizador semántico y preprocesador de las instrucciones. Verifica que las instrucciones tengan sentido y las ejecuta, generando un nuevo código basado en las instrucciones básicas del móvil (avanzar, retroceder, girar, subir lápiz, bajar lápiz, establecer grosor del lápiz, establecer color de lápiz y escribir), y del entorno (borrar pantalla, centrar, borrar salida de texto, establecer color de fondo, ocultar móvil y mostrar móvil) .
    - **crearVariable.** Crea una variable dentro de la tabla correspondiente a la definición de módulo que pertenece (función o procedimiento) indicando su tipo de dato y su valor inicial.
    - **eliminarVariable.** Elimina una variable de la tabla de declaraciones, cuando la misma ya ha perdido su contexto, por ejemplo, al momento de finalizar la ejecución de una función o procedimiento.

    - **actualizarVariable.** Encargada de actualizar el valor de la variable y de verificar su correspondiente tipo de dato.
    - **evaluarExpresion.** Encargada de evaluar una expresión a partir de la definición producto del analizador léxico. Si hay variables dentro de la expresión, se reemplaza el valor por el correspondiente a la tabla de variables.
    - **AnalizarLinea.** Analiza cada línea de código, esto es el conjunto de tokens de inicio de línea hasta encontrar el fín de Línea. En este punto se direcciona según sea el tipo de instrucción: declaración, asignación, bloque si, bloque según, bloque repite, bloque mientras, bloque para, comando o invocar un procedimiento.
    - **buscarCuerpoInstrucción.** Dada la definición de un bloque, por ejemplo: bloque para, esta función es la encargada de indicar el principio y fin del bloque, para que pueda ser analizado como un subprograma.
    - **dividirCodigo.** Encargado de particionar el código fuente en sus correspondientes funciones y procedimientos.
- **Intérprete.** Manipula la ubicación del móvil lógico, su posición, traslado y rotación.
    - Su propiedad principal es una variable de tipo Movil.
    - ExecuteOpcion. Procedimiento encargado de ejecutar el comando solicitado al móvil.
- **cPoint2.** Clase de apoyo para el manejo de puntos en 2 dimensiones. Implementa la pareja de datos de tipo single para definir el par X,Y en un plano cartesiano
- **cLine.** Clase de apoyo para el manejo de líneas. Esta constituido por dos cPoint2, que indican el inicio y fin de una línea.
- **cRuta.** Almacena un listado de cLine, que conforman el rastro del móvil.
- **Movil.** Manipula el objeto móvil: su posición, orientación y rastro.
    - **orientationRad.** Indica el ángulo de orientación en el plano XY del objeto móvil en radianes, ya que así lo solicita DirectX
    - **ruta.** Propiedad de tipo cRuta, encargada de almacenar la información de la huella dejada por el móvil.
    - **movilActivo.** Propiedad encargada de retornar o establecer si el móvil lógico está o no activo.
    - **Move.** Función que ejecuta el comando de retroceder o avanzar
    - **setPosicion.** Función invocada por move, que calcula la posición final del móvil después del traslado.

    - **rotarDerecha.** Función que ejecuta el comando de giro hacia la derecha
    - **rotarIzquierda.** Función que ejecuta el comando de giro hacia la izquierda
    - **getCurrentPosition.** Función que retorna la posición actual del móvil
    - **getCurrentOrientation.** Función que retorna la orientación actual del móvil en grados
- **MovilFisico.** Enlaza la animación 3d con los comandos enviados al puerto serial con las instrucciones correspondientes de movimiento y control de lápiz.
    - **enviarComandos.** Procedimiento encargado de enviar al puerto serial, el correspondiente comando para que el robot realice el movimiento solicitado. El comando se envía como una línea de caracteres a un puerto serial definido, con la verificación previa de si está activo o no la ejecución de comandos en el robot.
    - **Avanzar, retroceder, girarDerecha, girarIzquierda, bajarLapiz, subirLapiz, enviarCara y parar.** Son las funciones encargadas de generar la línea de caracteres que será enviada por el puerto serial.

5. PRUEBAS

5.1 PRUEBA DE USABILIDAD PARA LA INTERFAZ GRÁFICA

Según la norma ISO 9241:11 de 1.993 la usabilidad es “la facilidad de uso de una aplicación informática”.

Existen cuatro aspectos fundamentales[38] a la hora de definir o cuantificar el término usabilidad, cada uno de ellos está relacionado con el usuario, la tarea y el contexto:

1. Utilidad: la utilidad es la capacidad de una herramienta de ayudar a cumplir tareas específicas. Aunque esta afirmación parece obvia, es importante observar que una herramienta que es muy usable para una tarea, puede ser muy poco usable para otra, aún incluso si se trata de una tarea similar pero no idéntica.
2. Facilidad de uso: la facilidad de uso está en relación directa con la eficiencia o efectividad, medida como velocidad o cantidad de posibles errores. Una herramienta muy fácil de usar permitirá a su usuario efectuar más operaciones por unidad de tiempo (o menor tiempo para la misma operación) y disminuirá la probabilidad de que ocurran errores.
3. Facilidad de aprendizaje: la facilidad de aprendizaje es una medida del tiempo requerido para trabajar con cierto grado de eficiencia en el uso de la herramienta, y alcanzar un cierto grado de retención de estos conocimientos luego de cierto tiempo de no usar la herramienta o sistema.
4. Apreciación: es una medida de percepciones, opiniones, sentimientos y actitudes generadas en el Usuario por la herramienta o sistema; una medida, si se quiere, de su seducción o elegancia.

A partir de estos cuatro parámetros se puede evaluar cualquier producto para analizar la usabilidad, entonces aparecen distintos métodos para llevar a cabo este proceso, dentro de los cuales están: La evaluación heurística y los test con usuarios.

El desarrollo del test de usabilidad, permite conocer con mayor detenimiento si lo planteado por el diseñador es comprendido de forma adecuada por el usuario final, y en la metodología del diseño centrado en el usuario son base fundamental del buen desarrollo de la misma.

---

[38] Tomado de internet http://planeta.gaiasur.com.ar/infoteca/test-de-usabilidad-de-un-sitio.html. Workshop: cómo hacer un test de usabilidad de un sitio. Planificación, selección de usuarios, pruebas, reporte y análisis. Copyright © 1999-2000 Eduardo Mercovich, Gaiasur.

Ahora en este caso el diseño de la interfaz gráfica para el software, es enfocado para niños, y el test debe evaluar la accesibilidad y la comprensión del desarrollo del programa en los niños de 7 a 12 años de edad.

También existen diferentes tipos de test, cada uno con el fin de obtener información específica acerca de diseño. En nuestro caso se puede recurrir a dos tipos de métodos:

- Método de Instrucción previa (Teaching Method). En una fase previa se permite a los participantes interaccionar con el sistema para adquirir cierta soltura en su manejo. Después, habrán de ayudar a un usuario inexperto a realizar las tareas que se le encomienden.
- Método del Descubrimiento Conjunto (Co-discovery Method). Es una variante del test de usabilidad en la que dos participantes intentan realizar las tareas juntos mientras están siendo observados. Tal circunstancia se aproxima a la situación real del contexto de uso y aporta más datos.

Además hay factores que se deben tener en cuenta a la hora de aplicar cualquier técnica[39]:

Protocolo de expresión del usuario: en el momento de realizar las pruebas el usuario expresa verbalmente las sensaciones y percepciones del producto evaluado, esta puede ser protocolo de pensamiento manifestado, que es cuando el usuario va relatando cada una de las sensaciones que le genera el producto, ó puede ser por protocolo de preguntas, donde además el usuario formula preguntas acerca del prototipo evaluado.

Antes de la prueba se crea un guión de preguntas de cómo se van a realizar y cuanto tiempo se demora en cada uno, es importante explicarles a los niños el motivo de la misma, reiterar que se va a evaluar el diseño y no se va a evaluar la evolución de ellos en la prueba.

En el momento del análisis se debe observar los movimientos del niño y cada uno de sus actitudes frente a las diferentes alternativas. El niño debe entender que el evaluador es silencioso y que no lo puede ayudar en la consecución de las tareas impuestas.

En el comienzo de la prueba se le muestra al niño el programa y se le pregunta qué cree que es y para qué cree que es. Las opiniones sobre los colores y la letra que presentan los niños no son relevantes en este caso de la evaluación. Al ser niños hay que estar atentos a los gestos y frases que digan acerca de la interacción. Luego se hace la evaluación de facilidad de uso donde se le asignan tareas al niño para que encuentre las herramientas adecuadas para cada

[39] nosolubilidad.com – todos los derechos reservados, 2003 - 2006

operación. Al realizar cada una de las tareas el evaluador no debe aprobar o desaprobar las respuestas de los niños pues esto influye en la fiabilidad del test.

Cuando un niño no encuentre la herramienta adecuada para la operación se debe cambiar de tarea después de un tiempo prudencial, y no decirle si es correcto o no, pues ello puede influenciar a otros niños. Tampoco es adecuado preguntar cual interfaz le gusta más pues el usuario no es diseñador, y esta prueba debe percibir las acciones tan claramente por el diseñador que se puede observar que funciona y que no.

5.2 ELABORACIÓN DE TEST DE USABILIDAD

La siguiente prueba de usabilidad fue realizada como parte del desarrollo del proyecto de grado “Diseño y construcción de un robot móvil no autónomo, programable por medio de un software basado en la filosofía de logo” por los compañeros de Diseño Industrial.

Objetivo

Analizar la usabilidad en las alternativas diseñadas para niños de 7 a 12 años de edad.

Objetivos específicos

- Observar la comprensión en los niños de cómo sus acciones afecta a la salida del sistema.
- Visualizar si las metáforas utilizadas proporcionan una interfaz intuitiva.
- Medir que tareas se realizan en menos tiempo.
- Analizar la comprensión de la iconografía.
- Identificar los posibles errores para la consecución de una tarea.
- Analizar la facilidad de corrección de errores.
- Analizar los atributos visuales y auditivos que proporcionen mayor interés en los niños por el programa.

Procedimiento

En el colegio Beth Shalom de Bucaramanga en las horas de la mañana, se toma una muestra con 5 niños de 7 a 12 años, a los cuales de acuerdo a la metodología planteada, se les procede a hacer un test comparativo con tres alternativas diseñadas. El test se realiza por medio del desarrollo de una serie de tareas, con las que se trata de medir el grado de comprensión del software.

Figura 29. Niños interactuando con el Software

Figura 30. Niños interactuando con el Móvil Físico

Resultados

Luego de la evaluación a los niños del colegio se sacaron los siguientes gráficos que demuestran la eficiencia en las tres alternativas, por cada una de las preguntas y después el total, así entonces se mide si se cumple o no, la tarea impuesta por el evaluador:

ALTERNATIVA A

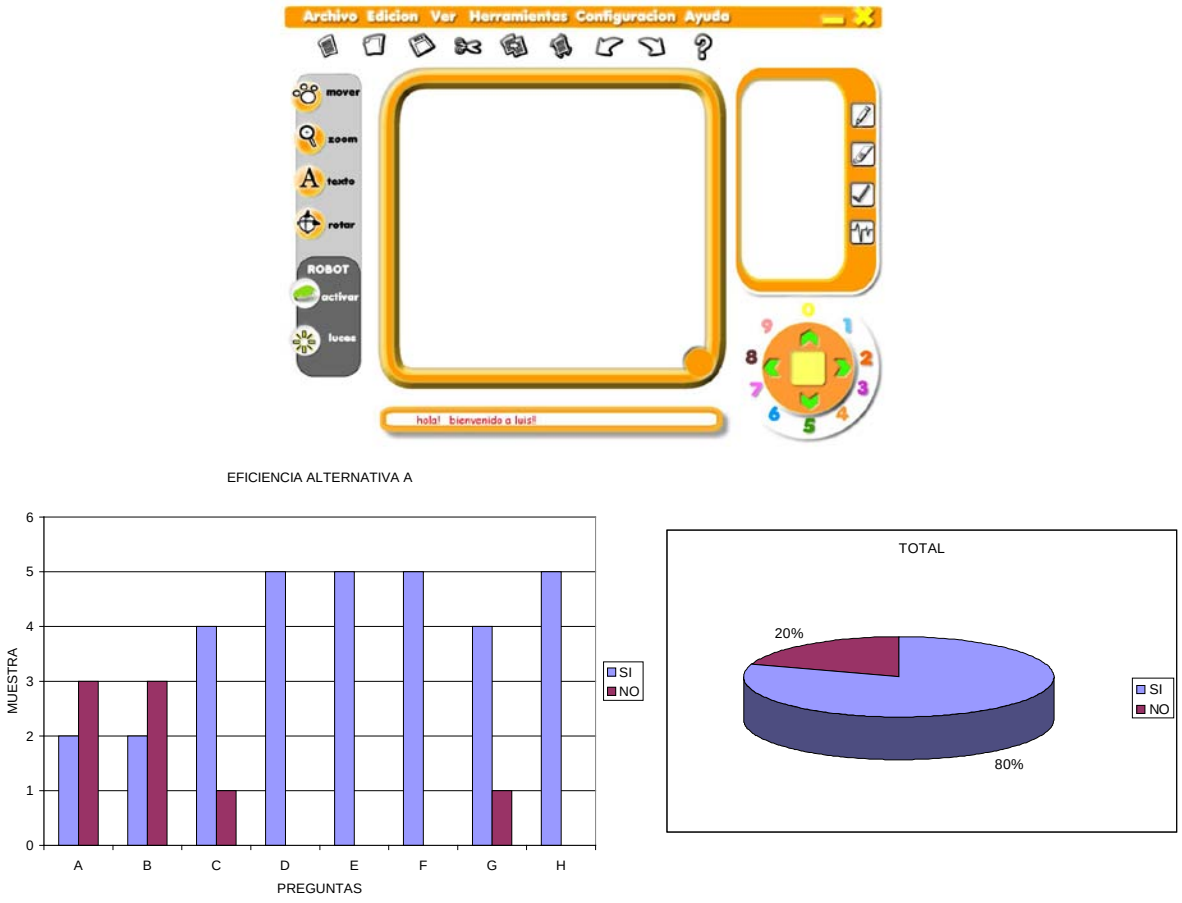
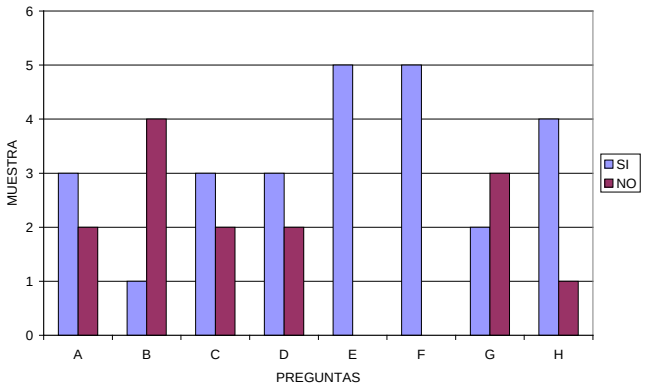


Figura 31. Resultados Alternativa A

ALTERNATIVA B

EFECIENCIA ALTERNATIVA B



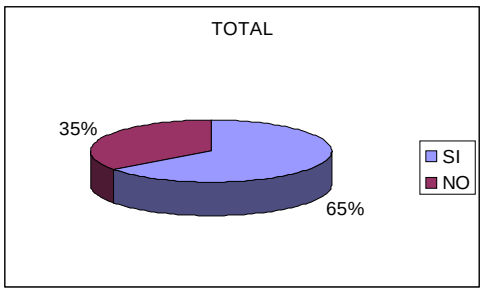


Figura 32. Resultados Alternativa B

ALTERNATIVA C

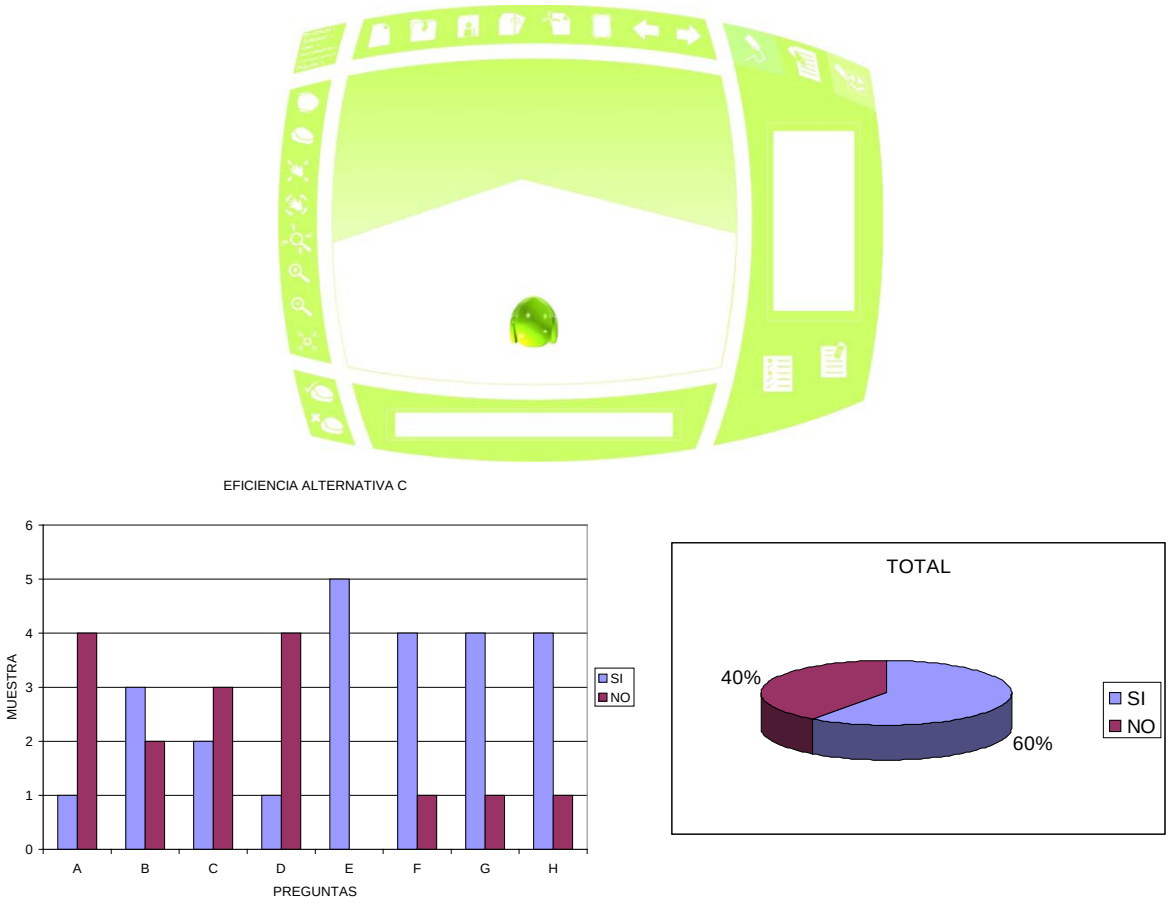


Figura 33. Resultados Alternativa C

Según esta tabulación se observa que la alternativa de mejor usabilidad es la alternativa A, pues es la que posibilita mayor cantidad de aciertos por parte de los niños y, en consecuencia, en la que menos se equivocaron los niños para conseguir la respuesta al ejercicio planteado en cada pregunta.

Se procede a hacer un análisis detallado de la alternativa seleccionada para conocer las rutas más frecuentes que utilizaron los niños y por que las escogieron.

Para la pregunta 1 y 2: cómo escoger desde qué vista dibujar y cómo crear un documento nuevo, existieron problemas pues era difícil explicarles a los niños el concepto de perspectiva o vista y el de un documento nuevo, pues varios optaron por borrar todo y así volver a empezar, sin embargo recorriendo la pantalla y leyendo cada aviso de herramienta (tool tips), encontraron iconos de perspectiva superior y nuevo. Lo que realmente al final se consiguió, pero con mayor cantidad de tiempo (aproximadamente 15 a 30 segundos por cada uno).

En la pregunta 3: donde darle las órdenes al robot para que dibuje, la ruta más utilizada fue la del botón lápiz, o haciendo clic en la ventana de comandos.

La pregunta 4: para cambiar el tipo de letra, fue una de las más rápidas de solucionar, pues los niños buscaban el icono de la letra A, lo que hace notar que ya están familiarizados con él. La mayoría tardó entre 2 a 5 segundos para reconocerlo.

Para corregir errores, que era la pregunta 5, los niños decidieron usar el teclado, es decir que cuando escriben y se equivocan, buscan la tecla borrar en el teclado. Otros encontraron el icono de borrador y sólo uno utilizó la herramienta deshacer.

Las luces del robot, pregunta 6 del test de usabilidad, fue desarrollada por todos por medio del icono de luces y se encontró en promedio de 2 segundos.

Para acercarse al escenario (zoom), 4 niños utilizaron el icono de lupa ubicado al lado izquierdo de la pantalla y lo hicieron entre 3 a 5 segundos aproximadamente.

La última pregunta, la número 8, cuestionaba cómo girar el escenario, fue de las preguntas de mayor ambigüedad pues encontraban el botón de rotar y el botón mover y utilizaban cualquiera de los dos, sin embargo es cuestión de concepto y con el conocimiento del software se soluciona.

6. CONCLUSIONES

La creación de una aplicación que genera el código del analizador a partir del BNF, facilitó la programación del intérprete LPE y permitió validar la gramática del lenguaje diseñado, por tanto se comprobó que la gramática de un lenguaje representada en un BNF es una muy buena manera de presentar un lenguaje de programación.

La aplicación generada interpreta el lenguaje de programación LPE, siendo capaz de resolver expresiones numérico-lógicas, además, permite la creación de rutinas con parámetros de entrada/salida y sentencias de condición y bucles.

El modelo de programación de la filosofía LOGO permite que los usuarios desarrollen su creatividad, imaginación e inteligencia cuando ingresan al campo de la informática, lo que se evidencia con el lenguaje propuesto LPE.

El móvil físico utilizado en este proyecto es la evidencia de los buenos resultados que se obtienen al realizar trabajos interdisciplinarios, como el presente, en el que se reunieron las carreras de Ingeniería de Sistemas, Ingeniería Electrónica y Diseño Industrial.

La herramienta software elaborada cuenta con un entorno gráfico en 3D que el estudiante puede relacionar con los juegos de video actuales y se sentirá más atraído porque a diferencia de un juego podrá apreciar que puede hacer mover el individuo a través de comandos de fácil aprendizaje y no con un mando como normalmente lo haría.

La comunicación con el móvil físico se implementó en una librería que utiliza el puerto serial del PC y codificando los comandos de movimiento, en sartas de caracteres que el robot puede decodificar internamente.

El software desarrollado además de permitir controlar el móvil es una herramienta de carácter educativo que formula una infinidad de posibilidades en el desarrollo de técnicas educativas diferentes, promoviendo el conocimiento de manera intuitiva y lógica.

La implementación del lenguaje LPE en las entidades educativas del país es la base para que se retome el modelo de enseñanza propuesto hace más de 20 años por LOGO y se infunda en el estudiante la sed de conocimiento y no un aprendizaje de memoria.

7. RECOMENDACIONES

Para la implementación del proyecto en las aulas educativas colombianas es necesario crear una metodología de enseñanza que posibilite la libertad de descubrir conocimiento por medio de ejercicios prácticos acordes a los objetivos de la pedagogía según el nivel de educación del usuario.

Tanto el móvil lógico como el físico podrían contar con un sensor de distancia y una vía de comunicación de vuelta con el intérprete para desarrollar aplicaciones en LPE que induzcan al estudiante a crear algoritmos con Inteligencia Artificial aplicada como es el caso de resolver caminos.

El editor de proyectos debería contar con un analizador estático que analice el código a medida que se escriba y reconozca los lexemas con diferentes colores de acuerdo a su clasificación.

La interfaz de comunicación utilizada en el proyecto fue el puerto serial, pero dados los avances en los equipos y el rápido cambio en las tecnologías, se aconseja cambiar ésta vía por el puerto USB o por Bluetooth.

8. BIBLIOGRAFÍA

AGUILLÓN VESGA Edgar Santiago y RUEDA TRIANA Diana Marcela Diseño y Construcción de un Robot Móvil no Auntónomo, Programable por medio de un Software Basado en la Filosofía de LOGO [Libro]. - [s.l.] : Universidad Industrial de Santander, 2007.

AHO Alfred V., SETHI Ravi y ULLMAN Jeffrey Compilers: Principles, Techniques, and Tools [Libro]. - China : Addison Wesley, Pearson Education, Inc, 1986.

BARBOSA Patricio DirectX 10 [En línea] // NEOTEO. - 15 de Mayo de 2006. - http://www.neoteo.com/directx-10.neo.

CORREDOR MONTAGUT Martha Vitalia Intérprete de un lenguaje en español que ejecuta los comandos del LOGO [Libro]. - [s.l.] : Universidad Industrial de Santander, 1985.

FONSECA Cleotilde y SCHAFFER Marilyn ¿Por qué LOGO? Una respuesta de Costa Rica [Publicación periódica] // Boletín de Informática Educativa. - [s.l.] : Universidad Nacional de Colombia, 1990. - 1 : Vol. 3.

GILB Kai Evolutionary Project Management & Product Development [Libro]. - [s.l.] : http://www.gilb.com/Download/EvoProjectMan.pdf, 2006.

GILB Tom Software Metrics [Libro]. - [s.l.] : Chartwell-Bratt, 1976.

GONZÁLEZ Michael y GUTIERREZ Javier Introducción a la Tecnología de Compiladores [En línea]. - 17 de Enero de 2008. - http://www.ctr.unican.es/asignaturas/lan/compila-2en1.pdf.

Grupo de Desarrollo en Allegro GDA Programación 3D - DirectX [En línea]. - Universidad Tecnológica de Pereira, 2006. - http://gda.utp.edu.co:9673/gda/documentacion/programacion_3d/directx.

LEWIS II P. M., ROSENKRANTZ D. J. y STEARNS R. E. Compiler Design Theory [Libro]. - U.S.A. : Addison-Wesley, 1978.

NIGLES Edward G. Build Your Own .NET Language and Compiler [Libro]. - U.S.A. : Apress, 2004.

REYNOSO Carlos Métodos Heterodoxos en Desarrollo de Software [En línea]. - Microsoft Corporation, 26 de Junio de 2006. - http://www.microsoft.com/spanish/msdn/arquitectura/roadmap_arq/heterodox.mspx .

URRAZA Juan de Microsoft DirectX [En línea]. - 2003. - http://www.jeuazarru.com/docs/DirectX.pdf.

ANEXOS

1. LENGUAJE DE COMUNICACIÓN CON EL ROBOT

Por medio del puerto serial se envían al robot los comandos a ejecutar, estos son cadenas de caracteres que repiten cada carácter tres veces para evitar interferencias de ruido y verificar que han llegado bien los datos y para dar por terminada la cadena, se envía una letra "*a*", a continuación se presentan los comandos con su equivalente en cadena.

Tabla 3. Lenguaje de Comunicación con el Robot

Comando	Cadena
Avanza	1000
Retrocede	2000
Giraderecha	3000
Giraizquierda	4000
Bajalapiz	5
Subirlapiz	6
Cara	7 (más 64 caracteres 0 ó 1)
Para	8

En los primeros cuatro comandos los tres ceros a la derecha significan el número de pasos que debe recorrer el robot o los grados que debe girar en caso de ser un comando de giro, por ejemplo, para girar 90° a la izquierda, se enviaría al robot la cadena: *444000999000a*. Como se puede apreciar se repite 3 veces cada carácter de la cadena a excepción del carácter de fin.

Para el caso de los comandos de movimiento del lápiz y de parada, un ejemplo de ordenar al robot bajar el lápiz es: 555a.

La señal de dibujo de la cara del robot se envía con 64 caracteres (también se repite cada uno 3 veces) de 0 ó 1 que indican el estado de encendido de cada uno de los leds, por ejemplo si se desea representar el gesto de la siguiente figura:

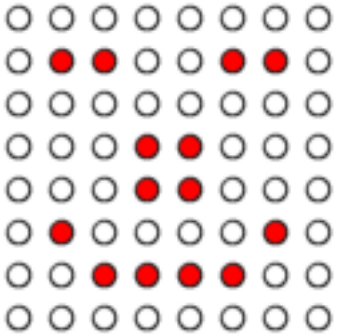


Figura 34. Pantalla Gestual del Robot

Se debe enviar la cadena de los 64 leds precedida por el carácter 7 y cada uno de los caracteres repetido tres veces, la cadena sin repetir sería la siguiente:

7000000000110011000000000000110000001100001000010001111000000000

2. MÁQUINAS DE ESTADO FINITO

Tabla 4. Máquinas de Estado Finito

```
programaFuente :
Definition : [NuevaLinea] definicionModulo +
-----
                NuevaLinea      definicionModulo Fin
Inicio          X               X
NuevaLinea                      X
definicionModulo                X                X
/*************************************************************/

codigoAbierto :
Definition : [NuevaLinea] [ sentencia delimitador ] *
-----
                NuevaLinea      sentencia       delimitador     Fin
Inicio          X               X                               X
NuevaLinea                      X                               X
sentencia                                       X
delimitador                     X                               X
/*************************************************************/

delimitador :
Definition : NuevaLinea | DosPuntos
-----
                NuevaLinea      DosPuntos       Fin
Inicio          X               X
NuevaLinea                                      X
DosPuntos                                       X
/*************************************************************/

sentencia :
Definition : sentenciaControl | comando |comandoMovil |asignacion
-----
                sentenciaControl  comando    comandoMovil    asignacion  Fin
Inicio          X                 X          X               X
sentenciaControl                                                         X
comando                                                                  X
comandoMovil                                                             X
asignacion                                                               X
/*************************************************************/
```

```
sentenciaControl :
Definition : declaracion | si |repite |para |mientras |segun
-----
                declaracion     si              repite          para            mientras        segun   Fin
Inicio          X               X               X               X               X               X
declaracion                                                                                      X
si                                                                                               X
repite                                                                                           X
para                                                                                             X
mientras                                                                                         X
segun                                                                                            X
/*****************************************************************/

comandoMovil :
Definition : avanza | giraDerecha |giraIzquierda |retrocede |subeLapiz |bajaLapiz |centro |ponGrosor |ponFondo
|ponColorLapiz |ocultar |mostrar
-----
            avanza  giraDerecha giraIzquierda  retrocede  subeLapiz  bajaLapiz  centro  ponGrosor  ponFondo
            ponColorLapiz       ocultar        mostrar    Fin
Inicio      X       X           X              X          X          X          X       X          X        X   X
            X
avanza
                    X
giraDerecha
                    X
giraIzquierda
                    X
retrocede
                    X
subeLapiz
                    X
bajaLapiz
                    X
centro
                    X
ponGrosor
                              X
ponFondo
                    X
ponColorLapiz
                    X
```

```
ocultar
                    X
mostrar
                    X
/*************************************************************/

comando :
Definition : borraPantalla | borraTexto |comentario |salir |salirBucle |leer |imprimir |retorno |llamarFuncion
-----
              borraPantalla  borraTexto comentario  salir  salirBucle  leer  imprimir  retorno  llamarFuncion  Fin
Inicio        X              X          X           X      X           X     X         X        X
borraPantalla                                                                                                   X
borraTexto                                                                                                      X
comentario                                                                                                      X
salir                                                                                                           X
salirBucle                                                                                                      X
leer                                                                                                            X
imprimir                                                                                                        X
retorno                                                                                                         X
llamarFuncion                                                                                                   X
/*************************************************************/

asignacion :
Definition : variable "=" expresion
-----
                variable        "="             expresion       Fin
Inicio          X
variable                        X
"="                                             X
expresion                                                       X
/*************************************************************/

variable :
Definition : identificador [ "(" listaIndices ")" ]
-----
                identificador   "("             listaIndices    ")"             Fin
Inicio          X
identificador                   X                                               X
"("                                             X
listaIndices                                                    X
")"                                                                             X
/*************************************************************/
```

```
listaIndices :
Definition : expresion [ Coma listaIndices ]
-----
                expresion       Coma            listaIndices    Fin
Inicio          X
expresion                       X                               X
Coma                                            X
listaIndices                                                    X
/*************************************************************/

avanza :
Definition : [ "AVANZA" | "AV" | "ADELANTE" | "AD" ] "(" expresion ")"
-----
           "AVANZA"   "AV"   "ADELANTE"  "AD"   "("     expresion  ")"    Fin
Inicio     X          X      X           X      X
"AVANZA"                                        X
"AV"                                            X
"ADELANTE"                                      X
"AD"                                            X
"("                                                     X
expresion                                                       X
")"                                                                     X
/*************************************************************/

giraDerecha :
Definition : [ "GIRADERECHA" | "GD" ] "(" expresion ")"
-----
                "GIRADERECHA"   "GD"            "("             expresion       ")"             Fin
Inicio          X               X               X
"GIRADERECHA"                                   X
"GD"                                            X
"("                                                             X
expresion                                                                       X
")"                                                                                             X
/*************************************************************/

giraIzquierda :
Definition : [ "GIRAIZQUIERDA" | "GI" ] "(" expresion ")"
-----
                "GIRAIZQUIERDA" "GI"            "("             expresion       ")"             Fin
Inicio          X               X               X
```

```
"GIRAIZQUIERDA"                                 X
"GI"                                            X
"("                                                         X
expresion                                                                   X
")"                                                                                     X
/*************************************************************/

retrocede :
Definition : [ "RETROCEDE" | "RE" | "ATRAS" | "AT" ] "(" expresion ")"
-----
                "RETROCEDE"     "RE"        "ATRAS"     "AT"        "("       expresion       ")"     Fin
Inicio          X               X           X           X           X
"RETROCEDE"                                                         X
"RE"                                                                X
"ATRAS"                                                             X
"AT"                                                                X
"("                                                                         X
expresion                                                                               X
")"                                                                                             X
/*************************************************************/

subeLapiz :
Definition : "SUBELAPIZ" | "SL"
-----
                "SUBELAPIZ"     "SL"            Fin
Inicio          X               X
"SUBELAPIZ"                                     X
"SL"                                            X
/*************************************************************/

bajaLapiz :
Definition : "BAJALAPIZ" | "BL"
-----
                "BAJALAPIZ"     "BL"            Fin
Inicio          X               X
"BAJALAPIZ"                                     X
"BL"                                            X
/*************************************************************/

centro :
Definition : "CENTRO"
-----
```

```
                "CENTRO"        Fin
Inicio          X
"CENTRO"                        X
/*************************************************************/

ponGrosor :
Definition : [ "PONGROSOR" | "PONG" ] "(" expresion ")"
-----
                "PONGROSOR"     "PONG"          "("             expresion       ")"             Fin
Inicio          X               X               X
"PONGROSOR"                                     X
"PONG"                                          X
"("                                                             X
expresion                                                                       X
")"                                                                                             X
/*************************************************************/

ponFondo :
Definition : [ "PONFONDO" | "PONF" ] "(" expresion ")"
-----
                "PONFONDO"      "PONF"          "("             expresion       ")"             Fin
Inicio          X               X               X
"PONFONDO"                                      X
"PONF"                                          X
"("                                                             X
expresion                                                                       X
")"                                                                                             X
/*************************************************************/

ponColorLapiz :
Definition : [ "PONCOLORLAPIZ" | "PONCL" ] "(" expresion ")"
-----
                "PONCOLORLAPIZ" "PONCL"         "("             expresion       ")"             Fin
Inicio          X               X               X
"PONCOLORLAPIZ"                                 X
"PONCL"                                         X
"("                                                             X
expresion                                                                       X
")"                                                                                             X
/*************************************************************/

ocultar :
```

```
Definition : "OCULTAMOVIL" | "OM"
-----
                "OCULTAMOVIL"   "OM"           Fin
Inicio          X               X
"OCULTAMOVIL"                                  X
"OM"                                           X
/*************************************************************/

mostrar :
Definition : "MUESTRAMOVIL" | "MM"
-----
                "MUESTRAMOVIL"  "MM"           Fin
Inicio          X               X
"MUESTRAMOVIL"                                 X
"MM"                                           X
/*************************************************************/

borraPantalla :
Definition : "BORRAPANTALLA" | "BP"
-----
                "BORRAPANTALLA" "BP"           Fin
Inicio          X               X
"BORRAPANTALLA"                                X
"BP"                                           X
/*************************************************************/

borraTexto :
Definition : "BORRATEXTO" | "BT"
-----
                "BORRATEXTO"    "BT"           Fin
Inicio          X               X
"BORRATEXTO"                                   X
"BT"                                           X
/*************************************************************/

comentario :
Definition : Apostrofe AnythingExceptDoubleQuote NuevaLinea
-----
                                Apostrofe      AnythingExceptDoubleQuote      NuevaLinea     Fin
Inicio                          X
Apostrofe                                      X
AnythingExceptDoubleQuote                                    X
```

```
NuevaLinea                                                                  X
/*************************************************************/

declaracion :
Definition : tipoDato identificadorVariable [ Coma identificadorVariable ] *
-----
                      tipoDato    identificadorVariable   Coma     identificadorVariable    Fin
Inicio                X
tipoDato                          X
identificadorVariable                                     X                                 X
Coma                                                                X
identificadorVariable                                     X                                 X
/*************************************************************/

identificadorVariable :
Definition : identificador [ "(" dimensionArreglo ")" ]
-----
                 identificador   "("            dimensionArreglo          ")"           Fin
Inicio           X
identificador                    X                                                     X
"("                                             X
dimensionArreglo                                                                  X
")"                                                                               X
/*************************************************************/

tipoDato :
Definition : "LOGICO" | "ENTERO" |"FLOTANTE" |"SARTA" |"CARACTER"
-----
                 "LOGICO"        "ENTERO"        "FLOTANTE"      "SARTA"         "CARACTER"      Fin
Inicio           X               X               X               X               X
"LOGICO"                                                                                        X
"ENTERO"                                                                                        X
"FLOTANTE"                                                                                      X
"SARTA"                                                                                         X
"CARACTER"                                                                                      X
/*************************************************************/

dimensionArreglo :
Definition : entero [ Coma dimensionArreglo ]
-----
                 entero          Coma            dimensionArreglo    Fin
Inicio           X
```

```
entero                          X                                   X
Coma                                            X
dimensionArreglo                                                    X
/*************************************************************/

si :
Definition : siEncabezado [ sinosi ][ sino ]finsi
-----
                siEncabezado    sinosi          sino            finsi           Fin
Inicio          X
siEncabezado                    X               X               X
sinosi                                          X               X
sino                                                            X
finsi                                                                           X
/*************************************************************/

siEncabezado :
Definition : "SI " expresion NuevaLinea codigoAbierto
-----
                "SI "           expresion       NuevaLinea      codigoAbierto   Fin
Inicio          X
"SI "                           X
expresion                                       X
NuevaLinea                                                      X
codigoAbierto                                                                   X
/*************************************************************/

sinosi :
Definition : "SINOSI " expresion NuevaLinea codigoAbierto [ sinosi ]
-----
                "SINOSI "       expresion       NuevaLinea      codigoAbierto   sinosi          Fin
Inicio          X
"SINOSI "                       X
expresion                                       X
NuevaLinea                                                      X
codigoAbierto                                                                   X               X
sinosi                                                                                          X
/*************************************************************/

sino :
Definition : "SINO " expresion NuevaLinea codigoAbierto
-----
```

```
                  "SINO "         expresion       NuevaLinea      codigoAbierto   Fin
Inicio            X
"SINO "                           X
expresion                                         X
NuevaLinea                                                        X
codigoAbierto                                                                     X
/*****************************************************************/

finsi :
Definition : "FINSI"
-----
                  "FINSI"         Fin
Inicio            X
"FINSI"                           X
/*****************************************************************/

repite :
Definition : repiteEncabezado codigoAbierto finRepite
-----
                  repiteEncabezado codigoAbierto  finRepite       Fin
Inicio            X
repiteEncabezado                   X
codigoAbierto                                     X
finRepite                                                         X
/*****************************************************************/

repiteEncabezado :
Definition : "REPITE "  NuevaLinea
-----
                  "REPITE "       NuevaLinea      Fin
Inicio            X
"REPITE "                         X
NuevaLinea                                        X
/*****************************************************************/

finRepite :
Definition : "HASTA " expresion
-----
                  "HASTA "        expresion       Fin
Inicio            X
"HASTA "                          X
expresion                                         X
```

```
/*************************************************************/

mientras :
Definition : mientrasEncabezado codigoAbierto finMientras
-----
                   mientrasEncabezado  codigoAbierto   finMientras     Fin
Inicio             X
mientrasEncabezado                     X
codigoAbierto                                        X
finMientras                                                          X
/*************************************************************/

mientrasEncabezado :
Definition : "MIENTRAS " expresion NuevaLinea
-----
                  "MIENTRAS "     expresion       NuevaLinea      Fin
Inicio            X
"MIENTRAS "                       X
expresion                                         X
NuevaLinea                                                        X
/*************************************************************/

finMientras :
Definition : "FINMIENTRAS"
-----
                  "FINMIENTRAS"   Fin
Inicio            X
"FINMIENTRAS"                     X
/*************************************************************/

para :
Definition : paraEncabezado codigoAbierto finPara
-----
                  paraEncabezado  codigoAbierto   finPara         Fin
Inicio            X
paraEncabezado                    X
codigoAbierto                                     X
finPara                                                           X
/*************************************************************/

paraEncabezado :
Definition : "PARA " variable "=" expresion "HASTA " expresion NuevaLinea
```

```
-----
            "PARA "    variable    "="     expresion    "HASTA "    expresion    NuevaLinea      Fin
Inicio      X
"PARA "                X
variable                           X
"="                                       X
expresion                                                X
"HASTA "                                                              X
expresion                                                                          X
NuevaLinea                                                                                    X
/*****************************************************************/

finPara :
Definition : "FINPARA"
-----
                  "FINPARA"       Fin
Inicio            X
"FINPARA"                         X
/*****************************************************************/

segun :
Definition : segunEncabezado segunOpcion+ [ segunOpcionOtro ] finSegun
-----
                  segunEncabezado segunOpcion     segunOpcionOtro finSegun        Fin
Inicio            X
segunEncabezado                   X
segunOpcion                       X               X               X
segunOpcionOtro                                                   X
finSegun                                                                          X
/*****************************************************************/

segunEncabezado :
Definition : "SEGUN " expresion NuevaLinea
-----
                  "SEGUN "        expresion       NuevaLinea      Fin
Inicio            X
"SEGUN "                          X
expresion                                         X
NuevaLinea                                                        X
/*****************************************************************/

segunOpcion :
```

```
Definition : "CASO " constante codigoAbierto
-----
                "CASO "         constante       codigoAbierto   Fin
Inicio          X
"CASO "                         X
constante                                       X
codigoAbierto                                                    X
/*************************************************************/

segunOpcionOtro :
Definition : "OTROCASO" codigoAbierto
-----
                "OTROCASO"      codigoAbierto   Fin
Inicio          X
"OTROCASO"                      X
codigoAbierto                                   X
/*************************************************************/

finSegun :
Definition : "FINSEGUN"
-----
                "FINSEGUN"      Fin
Inicio          X
"FINSEGUN"                      X
/*************************************************************/

salir :
Definition : "SALIR"
-----
                "SALIR"         Fin
Inicio          X
"SALIR"                         X
/*************************************************************/

salirBucle :
Definition : "SALIRBUCLE"
-----
                "SALIRBUCLE"    Fin
Inicio          X
"SALIRBUCLE"                    X
/*************************************************************/
```

```
leer :
Definition : Lee listaVariables
-----
                  Lee             listaVariables  Fin
Inicio            X
Lee                               X
listaVariables                                    X
/*************************************************************/

listaVariables :
Definition : variable [ Coma listaVariables ]
-----
                  variable        Coma            listaVariables  Fin
Inicio            X
variable                          X                               X
Coma                                              X
listaVariables                                                    X
/*************************************************************/

imprimir :
Definition : Escribe lista_expresiones
-----
                  Escribe         lista_expresiones   Fin
Inicio            X
Escribe                           X
lista_expresiones                                 X
/*************************************************************/

lista_expresiones :
Definition : expresion [ Coma lista_expresiones ]
-----
                  expresion       Coma            lista_expresiones   Fin
Inicio            X
expresion                         X                                   X
Coma                                              X
lista_expresiones                                                     X
/*************************************************************/

retorno :
Definition : retornoFuncion | retornoProcedimiento
-----
                        retornoFuncion       retornoProcedimiento      Fin
```

```
Inicio                  X                   X
retornoFuncion                                                  X
retornoProcedimiento                                            X
/*************************************************************/

retornoFuncion :
Definition : Retorne "(" expresion ")"
-----
                Retorne         "("             expresion       ")"             Fin
Inicio          X
Retorne                         X
"("                                             X
expresion                                                       X
")"                                                                             X
/*************************************************************/

retornoProcedimiento :
Definition : Retorne
-----
                Retorne         Fin
Inicio          X
Retorne                         X
/*************************************************************/

expresion :
Definition : termino [ operacion termino]*
-----
                termino         operacion       termino         Fin
Inicio          X
termino                         X                               X
operacion                                       X
termino                         X                               X
/*************************************************************/

operacion :
Definition : orOp | andOp |RelOp |AddOp |MulOp |"&" |Not termino
-----
          orOp        andOp       RelOp       AddOp       MulOp       "&"       Not       termino       Fin
Inicio    X           X           X           X           X           X         X
orOp                                                                                                      X
andOp                                                                                                     X
RelOp                                                                                                     X
```

```
AddOp                                                                                                   X
MulOp                                                                                                   X
"&"                                                                                                     X
Not                                                                                 X
termino                                                                                                 X
/*************************************************************/

termino :
Definition : numero_sin_signo | sarta |variable |llamarFuncion |"(" expresion ")"
-----
                numero_sin_signo  sarta    variable    llamarFuncion    "("     expresion     ")"     Fin
Inicio          X                 X        X           X                X
numero_sin_signo                                                                                        X
sarta                                                                                                   X
variable                                                                                                X
llamarFuncion                                                                                           X
"("                                                                             X
expresion                                                                                   X
")"                                                                                                     X
/*************************************************************/

llamarFuncion :
Definition : nombreFuncion "(" [ lista_expresiones ] ")"
-----
                nombreFuncion   "("            lista_expresiones         ")"         Fin
Inicio          X
nombreFuncion                   X
"("                                            X                       X
lista_expresiones                                                      X
")"                                                                                 X
/*************************************************************/

nombreFuncion :
Definition : Abs | Coseno | Largo | Log | Aleatorio | Seno | Mayuscula | Minuscula | identificador
-----
              Abs    Coseno    Largo    Log    Aleatorio   Seno    Mayuscula   Minuscula   identificador  Fin
Inicio        X      X         X        X      X           X       X           X           X
Abs                                                                                                     X
Coseno                                                                                                  X
Largo                                                                                                   X
Log                                                                                                     X
Aleatorio                                                                                               X
```

```
Seno                                                                                                                X
Mayuscula                                                                                                           X
Minuscula                                                                                                           X
identificador                                                                                                       X
/*************************************************************/

identificador :
Definition : Letra LetrasNumerosUnderscores*
-----
                                Letra           LetrasNumerosUnderscores        Fin
Inicio                          X
Letra                                           X                               X
LetrasNumerosUnderscores                        X                               X
/*************************************************************/

orOp :
Definition : Or
-----
                  Or            Fin
Inicio            X
Or                              X
/*************************************************************/

andOp :
Definition : And
-----
                  And           Fin
Inicio            X
And                             X
/*************************************************************/

RelOp :
Definition : "<" | ">" | "<=" | ">=" | "=" | "<>"
-----
                  "<"           ">"           "<="          ">="          "="           "<>"          Fin
Inicio            X             X             X             X             X             X
"<"                                                                                                 X
">"                                                                                                 X
"<="                                                                                                X
">="                                                                                                X
"="                                                                                                 X
"<>"                                                                                                X
```

```
/*************************************************************/

AddOp :
Definition : "+"|"-"
-----
                "+"             "-"             Fin
Inicio          X               X
"+"                                             X
"-"                                             X
/*************************************************************/

MulOp :
Definition : "*"|"/"|"\"|"Mod"
-----
                "*"             "/"             "\"             "Mod"           Fin
Inicio          X               X               X               X
"*"                                                                             X
"/"                                                                             X
"\"                                                                             X
"Mod"                                                                           X
/*************************************************************/

definicionModulo :
Definition : definicionProcedimiento | definicionFuncion
-----
                        definicionProcedimiento   definicionFuncion       Fin
Inicio                  X                         X
definicionProcedimiento                                                   X
definicionFuncion                                                         X
/*************************************************************/

definicionProcedimiento :
Definition : Procedimiento identificador "(" [ listaParametros ] ")" codigoAbierto FinProcedimiento NuevaLinea
-----
                Procedimiento  identificador  "(" listaParametros ")" codigoAbierto  FinProcedimiento NuevaLinea Fin

Inicio          X
Procedimiento                 X
identificador                               X
"("                                             X               X
listaParametros                                                 X
")"                                                                 X
```

```
codigoAbierto                                                                X
FinProcedimiento                                                                    X
NuevaLinea                                                                                 X
/*************************************************************/

definicionFuncion :
Definition : Funcion identificador "(" [ listaParametros ] ")" tipoDato codigoAbierto FinFuncion NuevaLinea
-----
               Funcion identificador  "("  listaParametros ")"  tipoDato codigoAbierto  FinFuncion  NuevaLinea Fin

Inicio         X
Funcion               X
identificador                   X
"("                                  X              X
listaParametros                                     X
")"                                                      X
tipoDato                                                                 X
codigoAbierto                                                                      X
FinFuncion                                                                                 X
NuevaLinea                                                                                           X
/*************************************************************/

listaParametros :
Definition : definicionFormalParametro [ "," listaParametros ]
-----
                         definicionFormalParametro   ","     listaParametros    Fin
Inicio                   X
definicionFormalParametro                            X                         X
","                                                         X
listaParametros                                                               X
/*************************************************************/

definicionFormalParametro :
Definition : tipoDato identificador
-----
                 tipoDato       identificador   Fin
Inicio           X
tipoDato                        X
identificador                                  X
/*************************************************************/

constante :
```

```
Definition : sarta | numero
-----
                sarta           numero          Fin
Inicio          X               X
sarta                                           X
numero                                          X
/*************************************************************/

numero :
Definition : entero | real
-----
                entero          real            Fin
Inicio          X               X
entero                                          X
real                                            X
/*************************************************************/

numero_sin_signo :
Definition : entero_sin_signo | real_sin_signo
-----
                entero_sin_signo  real_sin_signo   Fin
Inicio          X                 X
entero_sin_signo                                   X
real_sin_signo                                     X
/*************************************************************/

signo :
Definition : "+" | "-"
-----
                "+"             "-"             Fin
Inicio          X               X
"+"                                             X
"-"                                             X
/*************************************************************/

entero :
Definition : [ signo ] entero_sin_signo
-----
                signo       entero_sin_signo        Fin
Inicio          X           X
signo                       X
entero_sin_signo                                    X
```

```
/*************************************************************/

real :
Definition : [ signo ] real_sin_signo
-----
                  signo       real_sin_signo      Fin
Inicio            X           X
signo                         X
real_sin_signo                                    X
/*************************************************************/

sarta :
Definition : Comilla AnythingExceptDoubleQuote Comilla
-----
                          Comilla       AnythingExceptDoubleQuote      Comilla      Fin
Inicio                    X
Comilla                                 X
AnythingExceptDoubleQuote                                            X
Comilla                                                                           X
/*************************************************************/

entero_sin_signo :
Definition : Num +
-----
                  Num             Fin
Inicio            X
Num               X               X
/*************************************************************/

real_sin_signo :
Definition : entero_sin_signo "." entero_sin_signo
-----
                  entero_sin_signo    "."      entero_sin_signo   Fin
Inicio            X
entero_sin_signo                      X
"."                                             X
entero_sin_signo                                                 X
/*************************************************************/

Num :
Definition : 1|2|3|4|5|6|7|8|9|0
-----
```

```
                  1       2       3       4       5       6       7       8       9       0       Fin
Inicio            X       X       X       X       X       X       X       X       X       X
1                                                                                                 X
2                                                                                                 X
3                                                                                                 X
4                                                                                                 X
5                                                                                                 X
6                                                                                                 X
7                                                                                                 X
8                                                                                                 X
9                                                                                                 X
0                                                                                                 X
/*****************************************************************/

NuevaLinea :
Definition : ( chr10 | chr13 chr10 ) +
-----
                  chr10           chr13           chr10a          Fin
Inicio            X               X
chr10             X               X                               X
chr13                                             X
chr10a            X               X                               X
/*****************************************************************/

DosPuntos :
Definition : ":"
-----
                  ":"             Fin
Inicio            X
":"                               X
/*****************************************************************/

Coma :
Definition : ","
-----
                  ","             Fin
Inicio            X
","                               X
/*****************************************************************/

Apostrofe :
Definition : "'"
```

```
-----
                "'"             Fin
Inicio          X
"'"                             X
/*************************************************************/

Comilla :
Definition : """
-----
                ""              "               Fin
Inicio          X
""                              X
"                                               X
/*************************************************************/

Or :
Definition : "o"
-----
                "o"             Fin
Inicio          X
"o"                             X
/*************************************************************/

And :
Definition : "y"
-----
                "y"             Fin
Inicio          X
"y"                             X
/*************************************************************/

Not :
Definition : "no"
-----
                "no"            Fin
Inicio          X
"no"                            X
/*************************************************************/

Abs :
Definition : "abs"
-----
```

```
                  "abs"           Fin
Inicio            X
"abs"                             X
/*************************************************************/

LetrasNumerosUnderscores :
Definition : Letra | "_" | Num
-----
                  Letra           "_"             Num             Fin
Inicio            X               X               X
Letra                                                             X
"_"                                                               X
Num                                                               X
/*************************************************************/

Letra :
Definition : A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z
-----
       A   B   C   D   E   F   G   H   I   J   K   L   M   N   O   P   Q   R   S   T   U   V   W   X  Y   Z  Fin

Inicio X   X   X   X   X   X   X   X   X   X   X   X   X   X   X   X   X   X   X   X   X   X   X   X  X   X
A                                                                                                          X
B                                                                                                          X
C                                                                                                          X
D                                                                                                          X
E                                                                                                          X
F                                                                                                          X
G                                                                                                          X
H                                                                                                          X
I                                                                                                          X
J                                                                                                          X
K                                                                                                          X
L                                                                                                          X
M                                                                                                          X
N                                                                                                          X
O                                                                                                          X
P                                                                                                          X
Q                                                                                                          X
R                                                                                                          X
S                                                                                                          X
T                                                                                                          X
U                                                                                                          X
```

```
V                                                                                                                    X
W                                                                                                                    X
X                                                                                                                    X
Y                                                                                                                    X
Z                                                                                                                    X
/*********************************************************************/
```

3. CÓDIGO FUENTE DEL ANALIZADOR

Dada la extensión del código fuente del analizador generado, alrededor de siete mil líneas de código, este se presenta anexo como un archivo en PDF con el nombre de *Analizador.pdf*.

4. MANUAL DE REFERENCIA DEL LENGUAJE LPE

A continuación se presenta el manual de referencia para el lenguaje LPE, que fue generado por la herramienta de generación del analizador.

Tabla 5. Manual de Referencia del Lenguaje LPE

```
Nonterminal  Where Used   Remarks
-------------------------------------------------------------------------------
-------
Abs    "abs"
AddOp  "+"|"-"
And    "y"
andOp  And
Apostrofe    "'"
asignacion   variable "=" expresion
avanza [ "AVANZA" | "AV" | "ADELANTE" | "AD" ] "(" expresion ")"
bajaLapiz    "BAJALAPIZ" | "BL"
borraPantalla"BORRAPANTALLA" | "BP"
borraTexto   "BORRATEXTO" | "BT"
centro "CENTRO"
codigoAbierto[NuevaLinea] [ sentencia delimitador ] *
Coma   ","
comando      borraPantalla  |  borraTexto  |comentario  |salir  |salirBucle  |leer
|imprimir |retorno |llamarFuncion
comandoMovil avanza | giraDerecha |giraIzquierda |retrocede |subeLapiz |bajaLapiz
|centro |ponGrosor |ponFondo |ponColorLapiz |ocultar |mostrar
comentario   Apostrofe AnythingExceptDoubleQuote NuevaLinea
Comilla      """
constante    sarta | numero
declaracion  tipoDato identificadorVariable [ Coma identificadorVariable ] *
definicionFormalParametro tipoDato identificador
definicionFuncion   Funcion  identificador  "("  [  listaParametros  ]  ")"  tipoDato
codigoAbierto FinFuncion NuevaLinea
definicionModulo    definicionProcedimiento | definicionFuncion
definicionProcedimiento   Procedimiento identificador "(" [ listaParametros ] ")"
codigoAbierto FinProcedimiento NuevaLinea
delimitador  NuevaLinea | DosPuntos
dimensionArreglo    entero [ Coma dimensionArreglo ]
DosPuntos    ":"
entero [ signo ] entero_sin_signo
entero_sin_signo    Num +
expresion    termino [ operacion termino]*
finMientras  "FINMIENTRAS"
finPara      "FINPARA"
finRepite    "HASTA " expresion
finSegun     "FINSEGUN"
finsi  "FINSI"
giraDerecha  [ "GIRADERECHA" | "GD" ] "(" expresion ")"
```

```
giraIzquierda[ "GIRAIZQUIERDA" | "GI" ] "(" expresion ")"
identificadorLetra LetrasNumerosUnderscores*
identificadorVariable     identificador [ "(" dimensionArreglo ")" ]
imprimir     Escribe lista_expresiones
leer   Lee listaVariables
Letra  A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z
LetrasNumerosUnderscores  Letra | "_" | Num
lista_expresiones   expresion [ Coma lista_expresiones ]
listaIndices expresion [ Coma listaIndices ]
listaParametros     definicionFormalParametro [ "," listaParametros ]
listaVariables      variable [ Coma listaVariables ]
llamarFuncionnombreFuncion "(" [ lista_expresiones ] ")"
mientras     mientrasEncabezado codigoAbierto finMientras
mientrasEncabezado  "MIENTRAS " expresion NuevaLinea
mostrar      "MUESTRAMOVIL" | "MM"
MulOp  "*"|"/"|"\"|"Mod"
nombreFuncionAbs  |  Coseno  |  Largo  |  Log  |  Aleatorio  |  Seno  |  Mayuscula  |
Minuscula | identificador
Not    "no"
NuevaLinea   ( chr10 | chr13 chr10 ) +
Num    1|2|3|4|5|6|7|8|9|0
numero entero | real
numero_sin_signo    entero_sin_signo | real_sin_signo
ocultar      "OCULTAMOVIL" | "OM"
operacion    orOp | andOp |RelOp |AddOp |MulOp |"&" |Not termino
Or     "o"
orOp   Or
para   paraEncabezado codigoAbierto finPara
paraEncabezado      "PARA " variable "=" expresion "HASTA " expresion NuevaLinea
ponColorLapiz[ "PONCOLORLAPIZ" | "PONCL" ] "(" expresion ")"
ponFondo     [ "PONFONDO" | "PONF" ] "(" expresion ")"
ponGrosor    [ "PONGROSOR" | "PONG" ] "(" expresion ")"
programaFuente      [NuevaLinea] definicionModulo +
real   [ signo ] real_sin_signo
real_sin_signo      entero_sin_signo "." entero_sin_signo
RelOp  "<" | ">" | "<=" | ">=" | "=" | "<>"
repite repiteEncabezado codigoAbierto finRepite
repiteEncabezado    "REPITE "  NuevaLinea
retorno      retornoFuncion | retornoProcedimiento
retornoFuncion      Retorne "(" expresion ")"
retornoProcedimiento      Retorne
retrocede    [ "RETROCEDE" | "RE" | "ATRAS" | "AT" ] "(" expresion ")"
salir  "SALIR"
salirBucle   "SALIRBUCLE"
sarta  Comilla AnythingExceptDoubleQuote Comilla
segun  segunEncabezado segunOpcion+ [ segunOpcionOtro ] finSegun
segunEncabezado     "SEGUN " expresion NuevaLinea
segunOpcion  "CASO " constante codigoAbierto
segunOpcionOtro     "OTROCASO" codigoAbierto
sentencia    sentenciaControl | comando |comandoMovil |asignacion
sentenciaControl    declaracion | si |repite |para |mientras |segun
si     siEncabezado [ sinosi ][ sino ]finsi
siEncabezado "SI " expresion NuevaLinea codigoAbierto
signo  "+" | "-"
sino   "SINO " expresion NuevaLinea codigoAbierto
sinosi "SINOSI " expresion NuevaLinea codigoAbierto [ sinosi ]
subeLapiz    "SUBELAPIZ" | "SL"
termino      numero_sin_signo | sarta |variable |llamarFuncion |"(" expresion ")"
tipoDato     "LOGICO" | "ENTERO" |"FLOTANTE" |"SARTA" |"CARACTER"
variable     identificador [ "(" listaIndices ")" ]
```

5. MANUAL DE USUARIO

Presentado adjunto a este documento en formato PDF con el nombre de archivo *Manual de Usuario.pdf.*

6. ESTRUCTURA UML DEL DESARROLLO

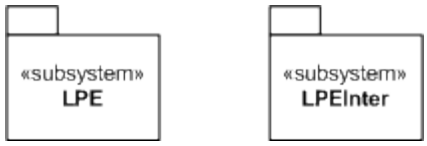


Figura 35. Subsistemas

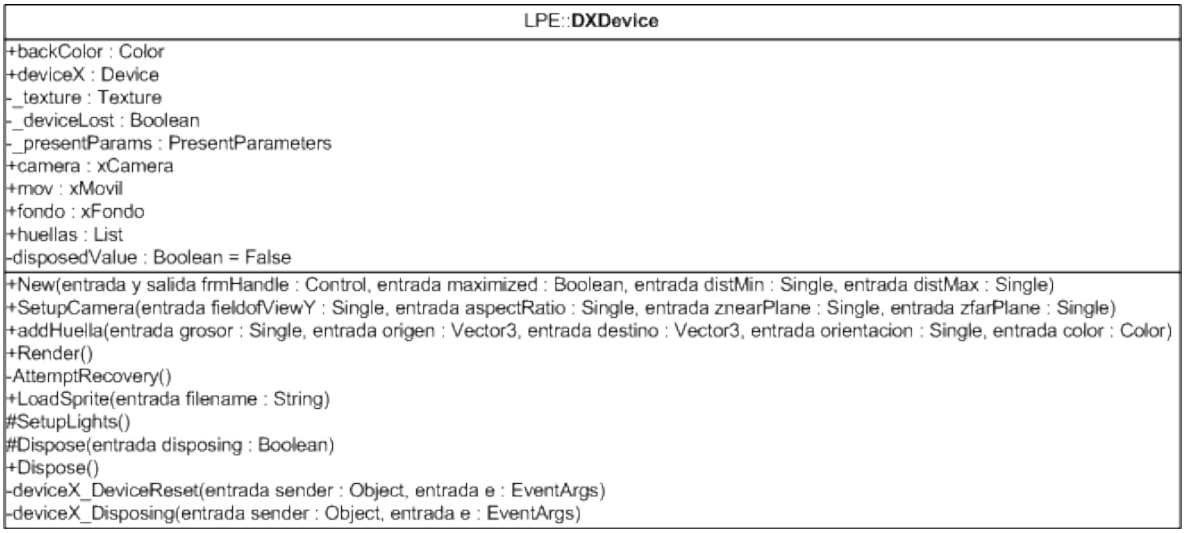


Figura 36. LPE::DXDevice

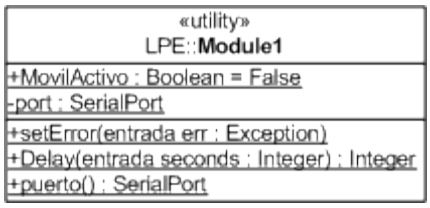


Figura 37. LPE::Module1

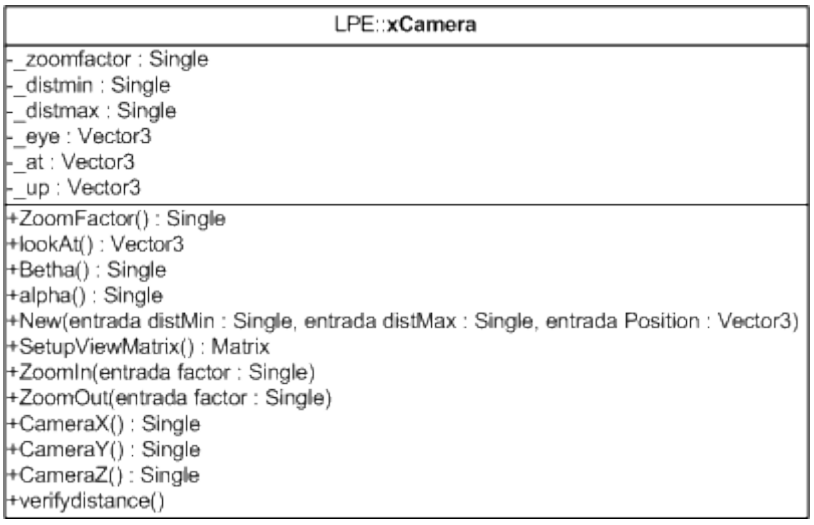


Figura 38. LPE::xCamera

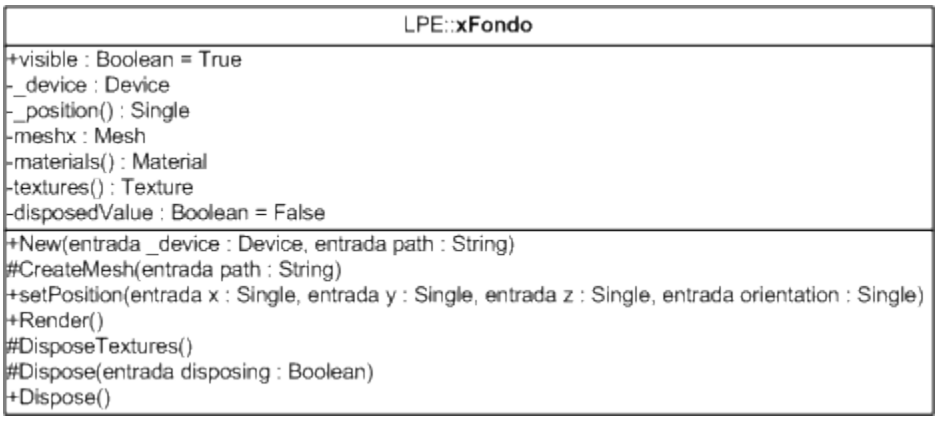


Figura 39. LPE::xFondo

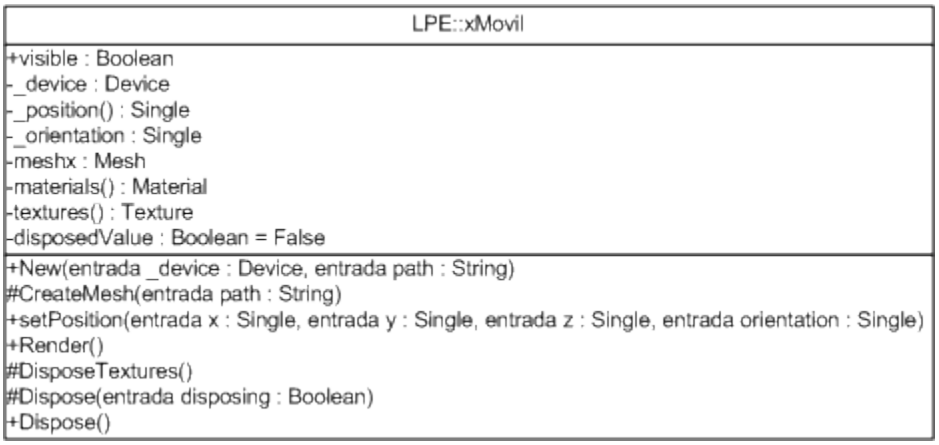


Figura 40. LPE::xMovil

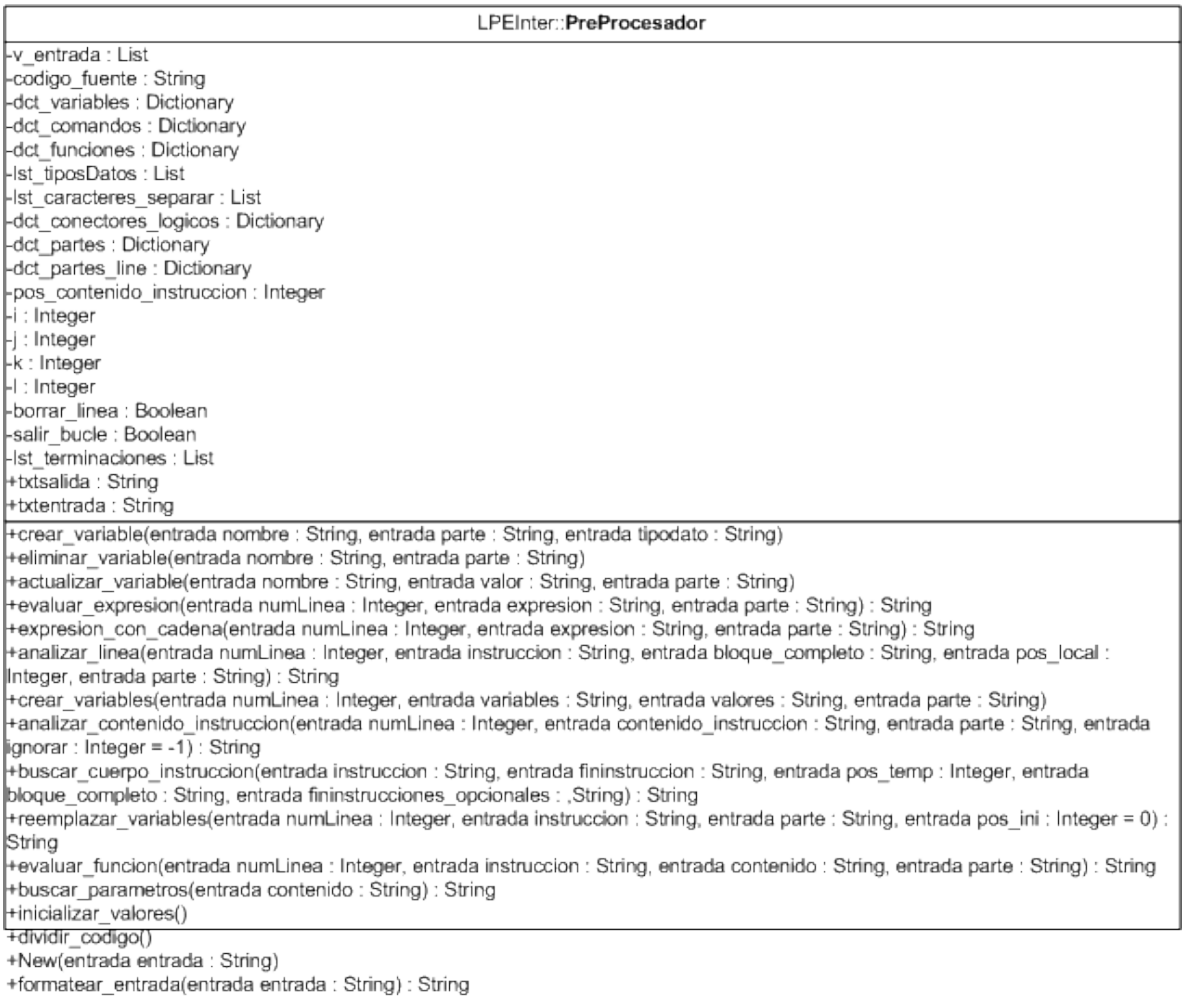


Figura 41. LPEInter::PreProcesador

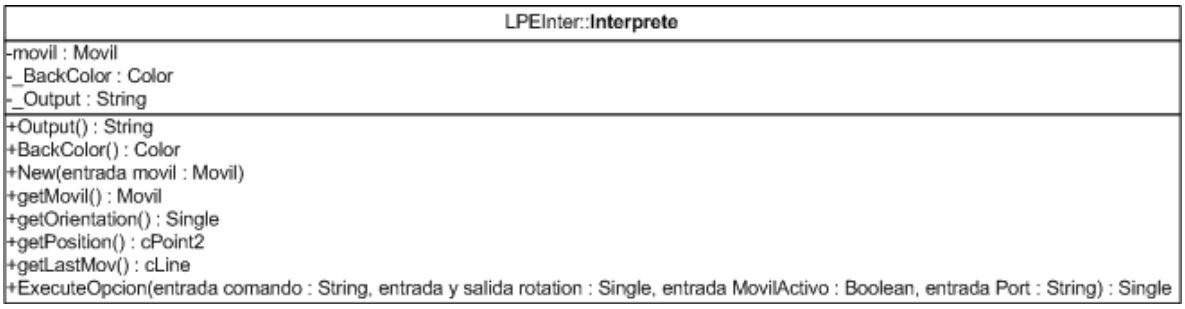


Figura 42. LPEInter::Interprete

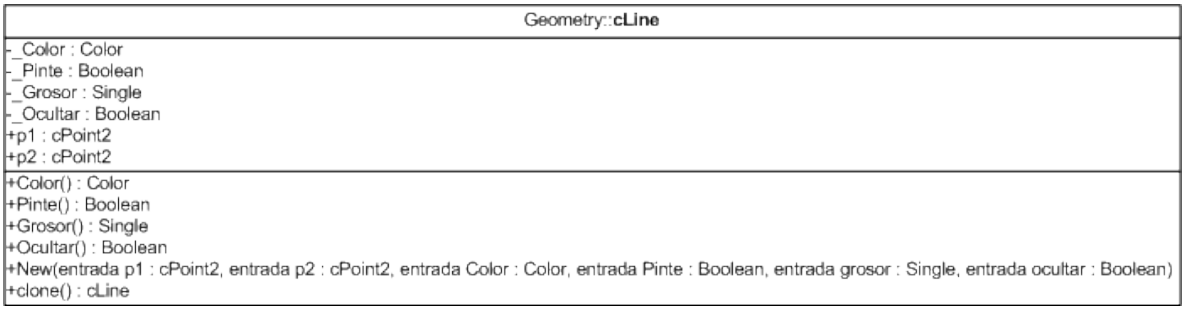


Figura 43. Geometry::cLine

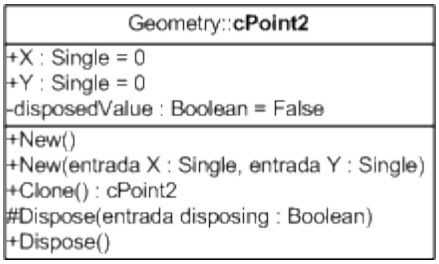


Figura 44. Geometry::cPoint2

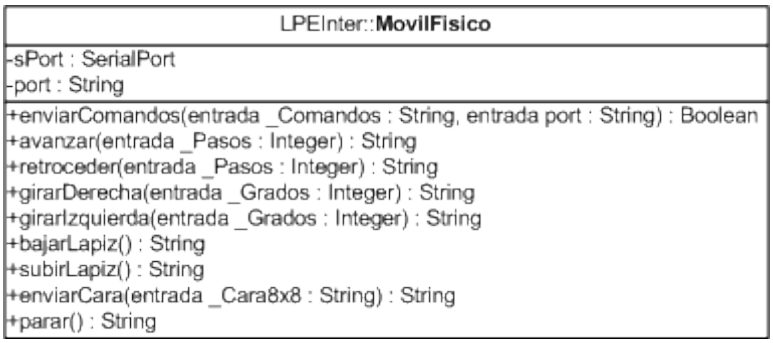


Figura 45. LPEInter::MovilFisico

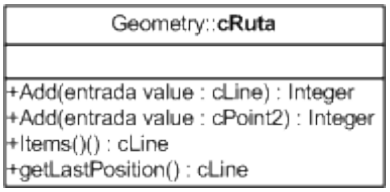


Figura 46. Geometry::cRuta

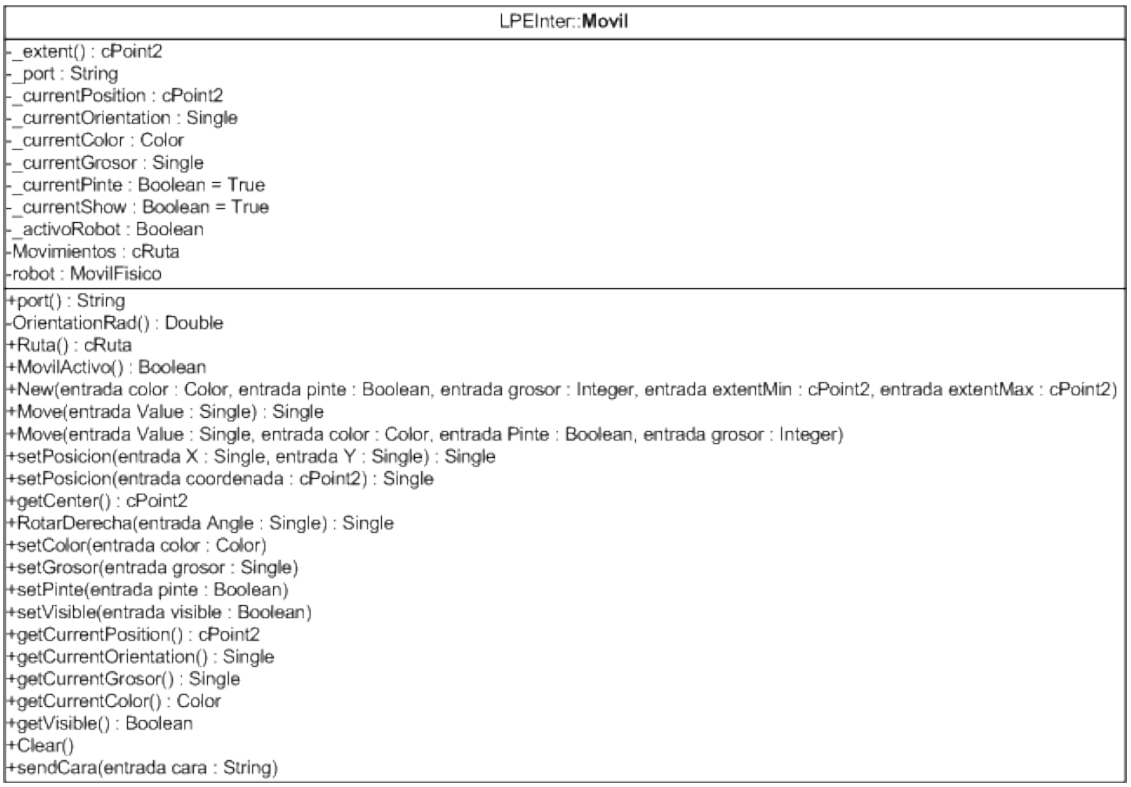


Figura 47. LPEInter::Movil

YEHISON FABIAN BECERRA RODRÍGUEZ
OSCAR ALBERTO CARRILLO ROZO
2008

LOUIS

LPE – LENGUAJE DE PROGRAMACIÓN EDUCATIVA

Manual de Usuario | Escuela de Ingeniería de Sistemas e Informática

TABLA DE CONTENIDO

INTRODUCCION

El software que se desarrolló se fundamentó en la filosofía del lenguaje LOGO, dado que hasta ahora ha sido un lenguaje utilizado con éxito en la educación. Sin embargo, los comandos y las instrucciones cambiaron teniendo en cuenta algunos aspectos del LOGO que han logrado motivar a niños y jóvenes en el uso del computador, como es el uso de un entorno gráfico.

Como resultado del proyecto se buscó ofrecer un software que no sólo pretende familiarizar al usuario con los lenguajes de programación, sino ofrecer un entorno que permita mejorar la ubicación espacial (especialmente en niños), desarrollar la creatividad y afianzar los conocimientos geométricos.

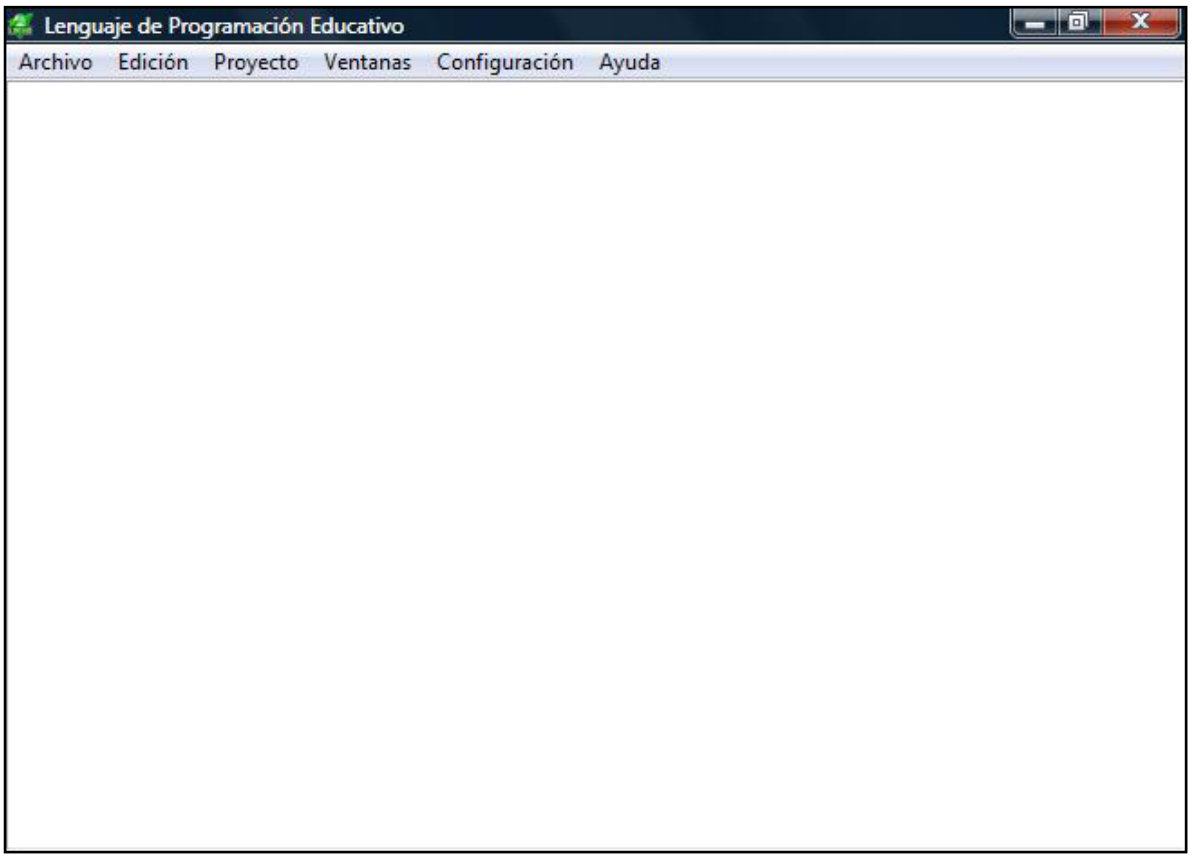
El entorno gráfico se mejoró y estuvo orientado por un proyecto de grado de la Escuela de Diseño Industrial[1] de la Universidad Industrial de Santander, porque se busca que el usuario sienta que está jugando y no programando, pues con los trabajos en LOGO se han obtenido muy buenos resultados ya que se aprende jugando. Adicionalmente con las gráficas se busca que el objeto "móvil", pueda ser cambiado por el usuario, así como el espacio en donde éste transita.

En el presente manual, se encuentra la descripción de las principales ventanas del software desarrollado para el análisis y ejecución de programas en lenguaje LPE que permite el direccionamiento de un robot móvil, así como la estructura del lenguaje implementado y unos ejemplos prácticos para iniciarse en el uso del entorno de desarrollo.

[1] AGUILLÓN VESGA Edgar Santiago y RUEDA TRIANA Diana Marcela DISEÑO Y CONSTRUCCIÓN DE UN ROBOT MÓVIL NO AUTÓNOMO, PROGRAMABLE POR MEDIO DE UN SOFTWARE BASADO EN LA FILOSOFÍA DE LOGO. [Libro]. - 2007.

ENTORNO DE TRABAJO

La ubicación de ventanas e iconos están dispuestos según el grado de utilización y de importancia que cada modulo proporciona, facilitando al usuario el manejo de los procesos.



Barra de título

Es la barra horizontal situada en la parte superior de una ventana del software y que muestra el titulo del archivo que se está ejecutando, además contiene a la izquierda un control (icono referente al programa), que al pulsarlo despliega un menú que permite restaurar, mover, modificar el tamaño, Minimizar, Maximizar, o cerrar la ventana, y a la derecha los botones “Maximizar”, “Minimizar” y “Cerrar”.

Barra de menús

Es la barra horizontal que contiene y despliega los nombres de los menús disponibles en menús desplegables para cada una de las aplicaciones del software. Está situada bajo la barra de titulo.

Niveles principales de la barra de menú.

Archivo

Al desplegar este menú, aparecen las opciones que permiten crear nuevos archivos, abrirlos, guardarlos o salir del entorno de programación, está compuesto por los siguientes subniveles:

Nuevo: Esta opción permite iniciar una nueva sesión de trabajo (Editor de proyectos), con un cuadro de texto en blanco y sin configurar,

Abrir: al seleccionar esta opción activa una ventana de dialogo que permite ubicar en los dispositivos de almacenamiento un archivo LPE para su edición y ejecución.

Guardar, Guardar Como y Guardar todos: cualquiera de estas opciones lleva a una ventana de diálogo la cual da la posibilidad de guardar uno o todos los proyectos en cualquier ubicación.

Salir: Cierra la sesión actual en el entorno de programación y en caso de tener proyectos activos pide confirmación de guardar los cambios al código fuente que se esté revisando o no.

Edición

Al desplegar las opciones del menú Edición aparecerán las siguientes opciones:

Estas opciones estarán activadas dependiendo del área activa en que se esté trabajando (editor de proyectos o ventana de ejecución) y de las operaciones que se hayan realizado sobre ella, los posibles comandos son:

Deshacer: Anula la última operación realizada en la edición de un código fuente.

Rehacer: Valida la última operación realizada.

Copiar: Guarda en el portapapeles del sistema el texto que se encuentre seleccionado en ese momento. Sólo se activara cuando haya un bloque seleccionado.

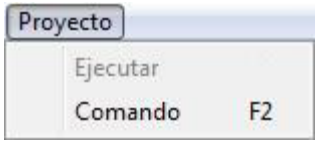
Cortar: Elimina el texto seleccionado y lo introduce en el área del portapapeles de Windows. Solo estará activa cuando haya un bloque seleccionado.

Pegar: Pondrá el texto del portapapeles del sistema en la ventana activa. En el área de Trabajo solo se podrá introducir texto.

Seleccionar Todo: Selecciona todo el contenido del editor de proyectos, para poder realizar alguna operación con el portapapeles.

Proyecto

Al desplegar la opción de Proyecto aparecerán las siguientes opciones.



Ejecutar: Muestra en el entorno de móvil en 3D las acciones planeadas en el código del usuario.

Comando: esta opción permitirá al usuario abrir múltiples ventanas del entorno en 3D y ejecutar las acciones del móvil de una en una.

Ventana

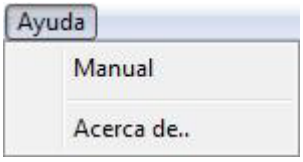
Se despliegan las diferentes ventanas que están desplegadas, bien sean del entorno 3D o del editor de proyectos, permitiendo así acceder de forma rápida a ellas.

Configuración

Permite acceder al menú de opciones para la personalización tanto del entorno 3D como del editor de proyectos:

Ayuda

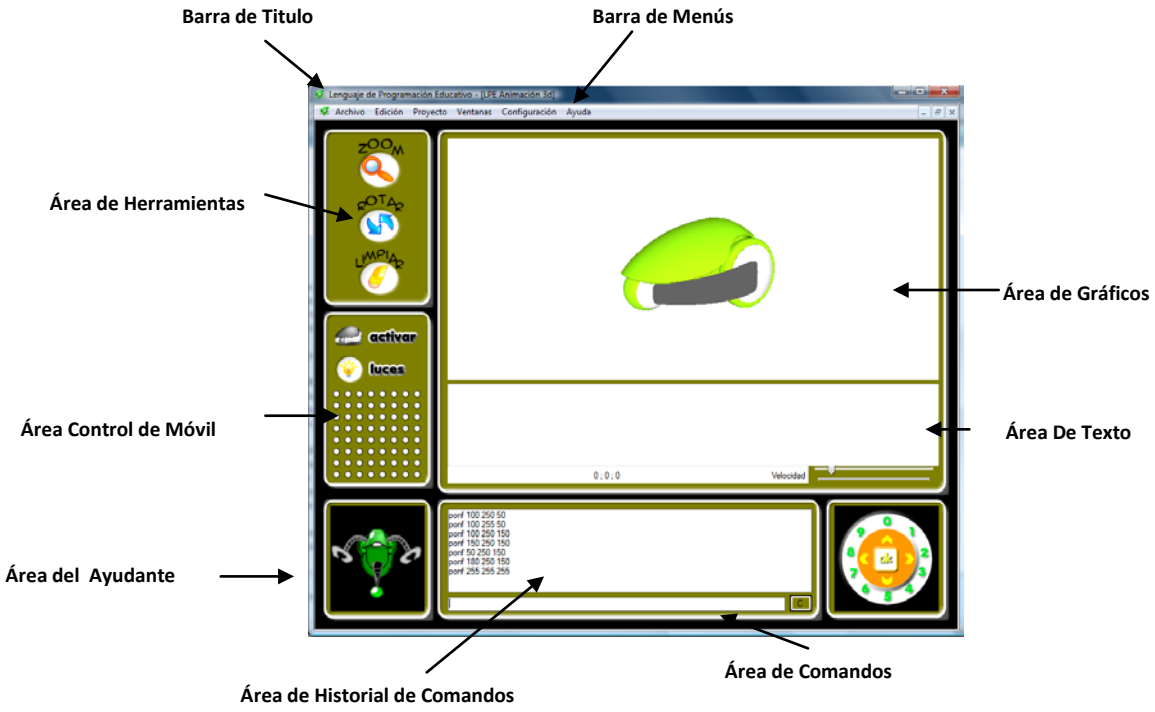
Permite acceder a la guía de información del programa.



Manual: esta opción de menú abrirá la guía, desde donde podrá acceder a cualquier tema acerca del entorno y el lenguaje de programación educativa.

Acerca de…: esta opción abrirá una ventana de dialogo donde se informará de la versión del software con la que se está trabajando y sus autores.

ENTORNO DE EJECUCIÓN



El entorno de ejecución tiene distintas áreas; cuando se muestra ésta ventana se presentan las siete áreas, denominadas área de "GRÁFICOS", área de "COMANDOS", área de "TEXTOS", área de "HERRAMIENTAS", área de "CONTROL DEL MÓVIL FÍSICO", área de "AYUDANTE" y el área de "HISTORIAL".

Área de "COMANDOS"

En esta ventana se escriben los comandos o instrucciones a ejecutar inmediatamente. Se usa fundamentalmente para ver el efecto de un determinado comando, o para modificar el aspecto de la pantalla desde el teclado.

Área de "HISTORIAL DE COMANDOS"

Consiste en un área que permite observar las órdenes que el usuario ingrese en la ventana de comandos.

Área de "TEXTOS"

En esta área aparecen los resultados de tipo texto que se produzcan en un comando. Normalmente, nunca se escribe directamente en esta área.

Área de "GRAFICOS"

En esta ventana aparecerán los resultados de tipo gráfico que se produzcan al ejecutar un comando. Es el entorno del móvil LOUIS, nombre que se le ha dado dentro de este proyecto a la figura que se encarga de hacer los dibujos. En la mayor parte de las aplicaciones, LOUIS se reduce a un objeto en el centro de la pantalla. Al igual que en la ventana de texto, no se escribe ni se dibuja directamente en esta ventana.

Área de "HERRAMIENTAS"

Presenta los botones gráficos para mejorar la visualización del área de gráficos.

a. *Zoom*: Activa una barra de desplazamiento que permite controlar de forma rápida el distanciamiento o acercamiento del móvil. La forma rápida de aplicar esta opción es con un clic derecho sostenido y desplazar hacia arriba o abajo según el acercamiento deseado.

b. *Rotar*: Activa barras de desplazamiento a los extremos del visualizador 3D, proporcionando así diferentes ángulos de visión del móvil. Forma rápida de aplicar esta opción es dar clic izquierdo sostenido y rotar el ángulo deseado.

c. *Limpiar***:** Esta opción permite limpiar los trazos en el visualizador 3D y centrar el móvil.

Área de "CONTROL DEL MÓVIL FÍSICO"

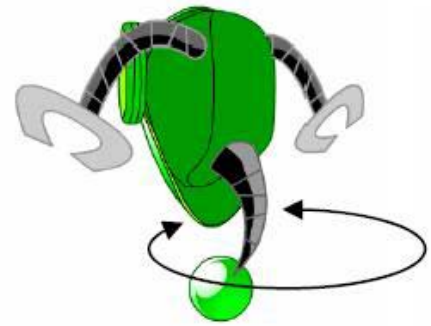
Permite activar y desactivar el envío de señales a través del puerto serial de tal manera que el móvil físico pueda ejecutarlas. Además se presenta una pantalla gestual que permite dibujar por medio de puntos la cara que presentará el robot al momento de activarse.

Área de "AYUDANTE"

Muestra el personaje animado apoya el manejo de las herramientas del software y además actúa como elemento dinámico que atrae la atención de los usuarios. Este personaje actuará

dependiendo a la herramienta a utilizar es decir al implementar la opción “ZOOM” el ayudante cambiará su aspecto del que está por defecto igualmente para los demás casos como “ROTAR” Y “MOVER”.

ZOOM



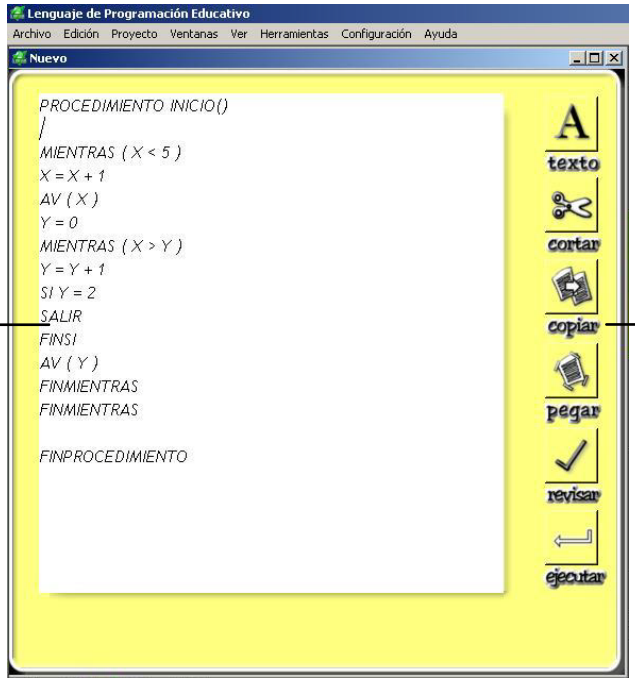
ROTAR

LIMPIAR

EDITOR DE PROYECTOS

Esta ventana de trabajo tiene como objetivo la creación de bloques de sentencias, de código o comandos para la ejecución de un proyecto o trayectoria que se desee hacer con el móvil tanto físico como lógico. Esta a su vez permite la opción de abrir proyectos existentes para su edición o ejecución.

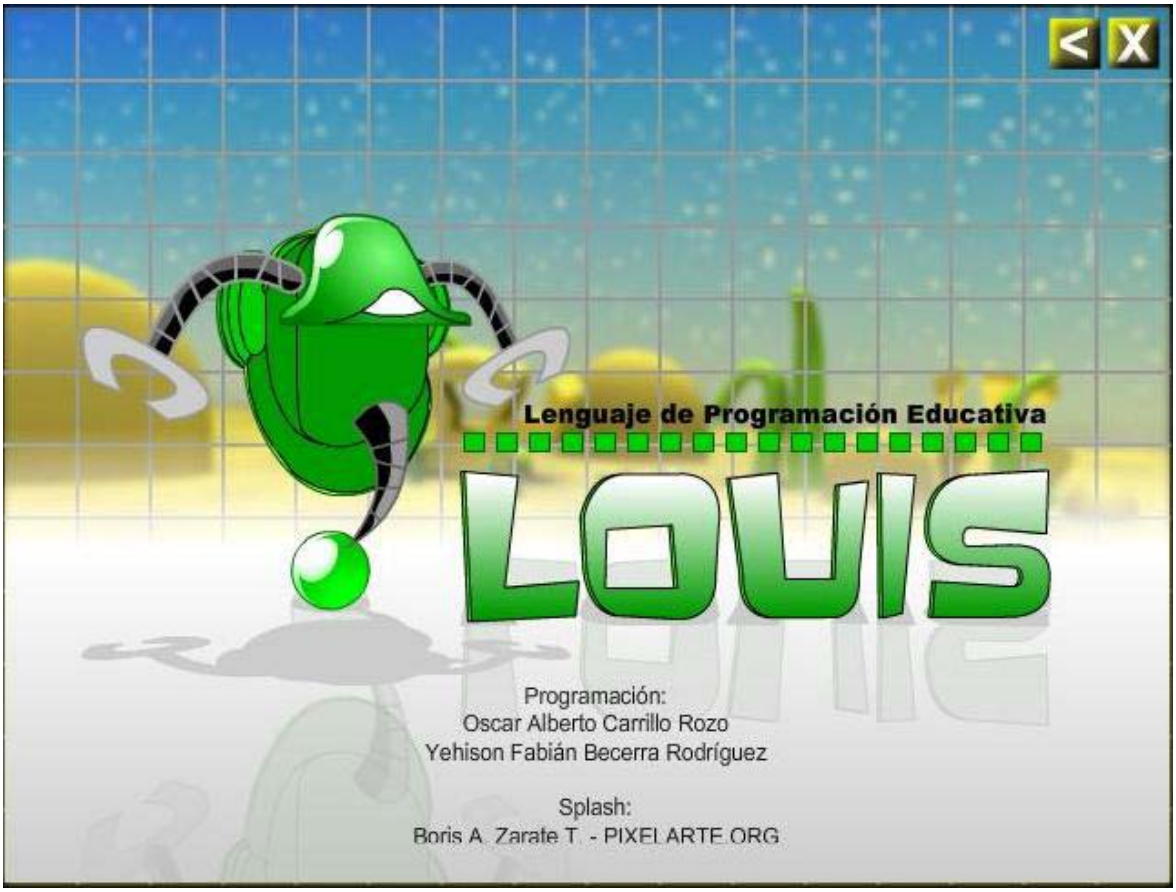
La ventana de editor de proyectos proporciona al usuario un editor de texto y su respetiva barra de herramientas de configuración de las cuales ya se mencionaron previamente en el menú Edición. En la barra de herramientas de esta ventana se encuentran dos botones nuevos que son "REVISAR" y "EJECUTAR".



a) *Revisar*. Esta opción proporciona al usuario un análisis del código mostrando los errores que se encuentren en la creación de un proyecto.
b) *Ejecutar*. Abre un nuevo entorno de ejecución para la visualización de las funciones que se indican en el código y que son realizadas por el móvil, cada editor de proyectos cuenta con su entorno de ejecución.

VENTANA “Acerca de…”

Al ejecutar éste menú, se activa una animación (splash) donde se presenta al usuario el nombre y el personaje representativo del programa que a su vez es el ayudante. Y revelando la información de sus realizadores.



ESTRUCTURA DEL LENGUAJE

COMANDO	DESCRIPCIÓN
AVANZA n *AV n* *ADELANTE n* *AD n*	El móvil se desplaza n unidades hacia adelante
ESCRIBE "xx xxx" *ES*	Escribe en la ventana de textos la palabra situada detrás de las comillas o la frase situada entre los corchetes
BORRAPANTALLA *BP*	Borra los gráficos que se hayan realizado y se coloca la tortuga en el centro de la pantalla (posición inicial)
GIRADERECHA n *GD n*	El móvil gira a la derecha n grados
GIRAIZQUIERDA n *GI n*	El móvil gira a la izquierda n grados
RETROCEDE n *RE n*	El móvil se desplaza n unidades hacia atrás
SUBELAPIZ *SL*	Hace que el móvil no deje rastro cuando avanza a través de la pantalla
BAJALAPIZ *BL*	Hace que el móvil deje un rastro cuando avanza a través de la pantalla
CENTRO	Desplaza el móvil hacia el centro de la pantalla
PONGROSOR n *PONG n*	Cambia el grosor del lápiz al nivel n (n entre 1 y 10)
PONFONDO (r, g, b) *PONF (r, g, b)*	Cambia el color del fondo al indicado por los niveles rojo, verde y azul que se indiquen en los parámetros r, g y b respectivamente.
PONCOLORLAPIZ (r, g, b) *PONCL (r, g, b)*	Cambia el color de la huella dejada por el lápiz al indicado por los niveles rojo, verde y azul que se indiquen en los parámetros r, g y b respectivamente
OCULTAMOVIL *OM*	Oculta el móvil
MUESTRAMOVIL *MM*	Muestra el móvil
BORRATEXTO *BT*	Borra el texto que haya en la pantalla de salida de textos
VARIABLE obj	Declara una variable que puede ser de tipo lógico, entero, flotante, sarta y carácter.
SI condición *[SINO SI condicion]* *[SINO]* *FINSI*	Esta sentencia significa que, si se cumple la condición, se ejecutan las acciones contenidas en el primer corchete y, en caso contrario, las incluidas en el segundo.

REPITE n *FINREPITE* *PARA var=n HASTA K* *FINPARA*	Hace que se repita n veces las acciones que se encuentran entre corchetes
MIENTRAS condicion *FINMIENTRAS*	Ejecuta las instrucciones mientras se cumpla condición
SAL	Detiene la ejecución de un programa
PROCEDIMIENTO a *Sentencias* *FINPROCEDIMIENTO*	Palabras claves para definir un nuevo procedimiento de nombre a.
FUNCION nombreprog *sentencias* *FINFUNCION*	Palabras claves para definir una nueva rutina que puede realizar el móvil.

EJEMPLOS

Cuadrado

El siguiente procedimiento construye un cuadrado de longitud de lado “lado”.

```
procedimiento cuadrado(entero lado)
	entero i
	para  i = 0 hasta 3
		av(lado)
		gd(90)
	finPara
finProcedimiento
```

Casa

Invocando el siguiente procedimiento se puede dibujar una casa cuyo alto es el valor “tam” y llama al procedimiento cuadrado

```
Procedimiento Casa(entero tam)
	cuadrado(tam)
	sl
	av(tam)
	bl
	gd(30)
	av(tam/2)
	gd (60)
	av (tam/2)
	gd (60)
	av (tam/2)
	sl
	gd (30)
	av(tam)
	gd (90)
	av(tam*3/8)
	gd(90)
	bl
	av(tam/2)
	gi(90)
	av(tam/4)
	gi(90)
	av(tam/2)
	sl
	gd(90)
	av(tam*3/8)
	gd (90)
	av (tam*5/8)
	gd (90)
	av (tam/8)
	bl
	av (tam/8)
	gi(90)
```

```
       av (tam/8)
       gi(90)
       av (tam/8)
       gi(90)
       av (tam/8)
       gi(90)
finProcedimiento
```

Polígono

El siguiente procedimiento dibuja un polígono regular con “lados” número de lados y “lado” largo de cada lado.

```
procedimiento poligono (entero lados, entero lado)
       entero angulo
       entero i
       angulo=((lados-2)*180)/lados
       escribe(angulo)
       escribe(lados)
       lados=lados-2
       para i=1 hasta lados
              av(lado)
              gd(180-angulo)
       finpara
finprocedimiento
```

Flor

```
procedimiento inicio()
       bp
       om
       pong(4)
       Para i=1 hasta 6
              arcod(10,10)
              gd(90)
              arcod(10,10)
              gd(30)
       finPara
finProcedimiento

procedimiento arcod (entero lados, entero lado)
       entero angulo
       entero i
       lados=lados*4
       angulo=((lados-2)*180)/lados
       lados=lados/4
       para i=1 hasta lados
              av(lado)
              gd(180-angulo)
       finpara
finprocedimiento
```

Sol

En el siguiente ejemplo se muestra cómo a partir del llamado a otros procedimientos es posible dibujar figuras más complejas:

```
procedimiento inicio()
	bp
	om
	pong(4)
	Para i=1 hasta 16
		rama(10,5)
		gd(165)
		rama(10,5)
		gd(45)
	finPara
finProcedimiento

procedimiento rama(entero lados,entero lado)
	arcod(lados,lado)
	arcoi(lados,lado)
finProcedimiento

procedimiento arcod (entero lados, entero lado)
	entero angulo
	entero i
	lados=lados*4
	angulo=((lados-2)*180)/lados
	lados=lados/4
	para i=1 hasta lados
		av(lado)
		gd(180-angulo)
	finpara
finprocedimiento

procedimiento arcoi (entero lados, entero lado)
	entero angulo
	entero i
	lados=lados*4
	angulo=((lados-2)*180)/lados
	lados=lados/4
	para i=1 hasta lados
		av(lado)
		gi(180-angulo)
	finpara
finprocedimiento
```

```
Public Class Analizer

    Public message As String = ""
    Public indexError As Integer

    Public Function Revise(ByVal str As String) As Boolean
        Dim IndexT As Integer = 0
        Str = " "c & str
        If revise_programaFuente(str, indexT) Then
            avance(Str, IndexT)
            If revise_Fin(Str, IndexT) Then
                message = ""
                indexError = -1
                Return True
            End If
        End If
    End Function

    Private Function revise_programaFuente(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        If revise_Fin(Str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(Str, indexT)
        'Verifica el estado NuevaLinea
        If revise_programaFuente_0(Str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado definicionModulo
        If revise_programaFuente_1(Str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_NuevaLinea(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(Str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        'Verifica el estado chr10
        If revise_NuevaLinea_2(Str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado chr13
        If revise_NuevaLinea_3(Str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_definicionModulo(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        If revise_Fin(Str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(Str, indexT)
        'Verifica el estado definicionProcedimiento
        If revise_definicionModulo_5(Str, indexT) Then
```

```
            index = indexT
            Return True
        End If
        'Verifica el estado definicionFuncion
        If revise_definicionModulo_6(Str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_codigoAbierto(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        If revise_Fin(Str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(Str, indexT)
        'Verifica el estado NuevaLinea
        If revise_codigoAbierto_7(Str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado sentencia
        If revise_codigoAbierto_8(Str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado Fin
        If revise_codigoAbierto_10(Str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_sentencia(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(Str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(Str, indexT)
        'Verifica el estado sentenciaControl
        If revise_sentencia_11(Str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado comando
        If revise_sentencia_12(Str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado comandoMovil
        If revise_sentencia_13(Str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado asignacion
        If revise_sentencia_14(Str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_delimitador(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
```

```
        If revise_Fin(Str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(Str, indexT)
        'Verifica el estado NuevaLinea
        If revise_delimitador_15(Str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado DosPuntos
        If revise_delimitador_16(Str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_DosPuntos(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(Str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        'Verifica el estado ":"
        If revise_DosPuntos_17(Str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_sentenciaControl(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        If revise_Fin(Str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(Str, indexT)
        'Verifica el estado declaracion
        If revise_sentenciaControl_18(Str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado si
        If revise_sentenciaControl_19(Str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado repite
        If revise_sentenciaControl_20(Str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado para
        If revise_sentenciaControl_21(Str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado mientras
        If revise_sentenciaControl_22(Str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado segun
        If revise_sentenciaControl_23(Str, indexT) Then
            index = indexT
```

```
            Return True
        End If
    End Function

    Private Function revise_comando(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(Str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(Str, indexT)
        'Verifica el estado borraPantalla
        If revise_comando_24(Str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado borraTexto
        If revise_comando_25(Str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado comentario
        If revise_comando_26(Str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado salir
        If revise_comando_27(Str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado salirBucle
        If revise_comando_28(Str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado leer
        If revise_comando_29(Str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado imprimir
        If revise_comando_30(Str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado retorno
        If revise_comando_31(Str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado llamarFuncion
        If revise_comando_32(Str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_comandoMovil(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        If revise_Fin(Str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(Str, indexT)
        'Verifica el estado avanza
```

```
        If revise_comandoMovil_33(Str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado giraDerecha
        If revise_comandoMovil_34(Str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado giraIzquierda
        If revise_comandoMovil_35(Str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado retrocede
        If revise_comandoMovil_36(Str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado subeLapiz
        If revise_comandoMovil_37(Str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado bajaLapiz
        If revise_comandoMovil_38(Str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado centro
        If revise_comandoMovil_39(Str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado ponGrosor
        If revise_comandoMovil_40(Str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado ponFondo
        If revise_comandoMovil_41(Str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado ponColorLapiz
        If revise_comandoMovil_42(Str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado ocultar
        If revise_comandoMovil_43(Str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado mostrar
        If revise_comandoMovil_44(Str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_asignacion(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(Str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(Str, indexT)
```

```
        'Verifica el estado variable
        If revise_asignacion_45(Str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_declaracion(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        If revise_Fin(Str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(Str, indexT)
        'Verifica el estado tipoDato
        If revise_declaracion_48(Str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_si(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(Str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(Str, indexT)
        'Verifica el estado siEncabezado
        If revise_si_52(Str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_repite(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(Str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(Str, indexT)
        'Verifica el estado repiteEncabezado
        If revise_repite_56(Str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_para(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(Str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(Str, indexT)
        'Verifica el estado paraEncabezado
        If revise_para_59(Str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_mientras(ByVal str As String, ByRef index As Integer) As Boolean
```

```
        Dim indexT As Integer = index
        If revise_Fin(Str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(Str, indexT)
        'Verifica el estado mientrasEncabezado
        If revise_mientras_62(Str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_segun(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(Str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(Str, indexT)
        'Verifica el estado segunEncabezado
        If revise_segun_65(Str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_avanza(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(Str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(Str, indexT)
        'Verifica el estado "AVANZA"
        If revise_avanza_69(Str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado "("
        If revise_avanza_70(Str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado "AV"
        If revise_avanza_73(Str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado "ADELANTE"
        If revise_avanza_74(Str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado "AD"
        If revise_avanza_75(Str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_giraDerecha(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        If revise_Fin(Str, indexT) Then
            message = "No se esperaba fin de código"
```

```
            indexError = indexT
            Return False
        End If
        avance(str, indexT)
        'Verifica el estado "GIRADERECHA"
        If revise_giraDerecha_76(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado "GD"
        If revise_giraDerecha_80(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado "("
        If revise_giraDerecha_77(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_giraIzquierda(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(str, indexT)
        'Verifica el estado "GIRAIZQUIERDA"
        If revise_giraIzquierda_81(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado "GI"
        If revise_giraIzquierda_85(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado "("
        If revise_giraIzquierda_82(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_retrocede(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(str, indexT)
        'Verifica el estado "RETROCEDE"
        If revise_retrocede_86(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado "("
        If revise_retrocede_87(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado "RE"
        If revise_retrocede_90(str, indexT) Then
            index = indexT
            Return True
```

```
        End If
        'Verifica el estado "ATRAS"
        If revise_retrocede_91(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado "AT"
        If revise_retrocede_92(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_subeLapiz(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(str, indexT)
        'Verifica el estado "SUBELAPIZ"
        If revise_subeLapiz_93(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado "SL"
        If revise_subeLapiz_94(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_bajaLapiz(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(str, indexT)
        'Verifica el estado "BAJALAPIZ"
        If revise_bajaLapiz_95(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado "BL"
        If revise_bajaLapiz_96(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_centro(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(str, indexT)
        'Verifica el estado "CENTRO"
        If revise_centro_97(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_ponGrosor(ByVal str As String, ByRef index As Integer) As Boolean
```

```
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(str, indexT)
        'Verifica el estado "PONGROSOR"
        If revise_ponGrosor_98(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado "PONG"
        If revise_ponGrosor_102(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado "("
        If revise_ponGrosor_99(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_ponFondo(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(str, indexT)
        'Verifica el estado "PONFONDO"
        If revise_ponFondo_103(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado "PONF"
        If revise_ponFondo_107(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado "("
        If revise_ponFondo_104(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_ponColorLapiz(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(str, indexT)
        'Verifica el estado "PONCOLORLAPIZ"
        If revise_ponColorLapiz_108(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado "PONCL"
        If revise_ponColorLapiz_112(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado "("
```

```
        If revise_ponColorLapiz_109(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_ocultar(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(str, indexT)
        'Verifica el estado "OCULTAMOVIL"
        If revise_ocultar_113(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado "OM"
        If revise_ocultar_114(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_mostrar(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(str, indexT)
        'Verifica el estado "MUESTRAMOVIL"
        If revise_mostrar_115(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado "MM"
        If revise_mostrar_116(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_borraPantalla(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(str, indexT)
        'Verifica el estado "BORRAPANTALLA"
        If revise_borraPantalla_117(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado "BP"
        If revise_borraPantalla_118(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_borraTexto(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
```

```
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(str, indexT)
        'Verifica el estado "BORRATEXTO"
        If revise_borraTexto_119(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado "BT"
        If revise_borraTexto_120(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_comentario(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(str, indexT)
        'Verifica el estado Apostrofe
        If revise_comentario_121(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_salir(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(str, indexT)
        'Verifica el estado "SALIR"
        If revise_salir_124(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_salirBucle(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(str, indexT)
        'Verifica el estado "SALIRBUCLE"
        If revise_salirBucle_125(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_leer(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
```

```
        End If
        avance(str, indexT)
        'Verifica el estado Lee
        If revise_leer_126(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_imprimir(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(str, indexT)
        'Verifica el estado Escribe
        If revise_imprimir_128(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_retorno(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(str, indexT)
        'Verifica el estado retornoFuncion
        If revise_retorno_130(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado retornoProcedimiento
        If revise_retorno_131(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_llamarFuncion(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(str, indexT)
        'Verifica el estado nombreFuncion
        If revise_llamarFuncion_132(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_variable(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(str, indexT)
        'Verifica el estado identificador
```

```
        If revise_variable_136(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_expresion(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(str, indexT)
        'Verifica el estado termino
        If revise_expresion_140(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_identificador(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(str, indexT)
        'Verifica el estado Letra
        If revise_identificador_143(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_listaIndices(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(str, indexT)
        'Verifica el estado expresion
        If revise_listaIndices_145(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_Coma(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        'Verifica el estado ","
        If revise_Coma_148(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_Apostrofe(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
```

```
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        'Verifica el estado "'"
        If revise_Apostrofe_149(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_AnythingExceptDoubleQuote(ByVal str As String, ByRef index As
    Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        If str(indexT) <> Chr(13) And str(index) <> Chr(10) And str(indexT) <> "" Then
            index = indexT + 1
            Return True
        End If
    End Function

    Private Function revise_tipoDato(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(str, indexT)
        'Verifica el estado "LOGICO"
        If revise_tipoDato_150(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado "ENTERO"
        If revise_tipoDato_151(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado "FLOTANTE"
        If revise_tipoDato_152(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado "SARTA"
        If revise_tipoDato_153(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado "CARACTER"
        If revise_tipoDato_154(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_identificadorVariable(ByVal str As String, ByRef index As Integer)
     As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
```

```
        avance(str, indexT)
        'Verifica el estado identificador
        If revise_identificadorVariable_155(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_dimensionArreglo(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(str, indexT)
        'Verifica el estado entero
        If revise_dimensionArreglo_159(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_entero(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(str, indexT)
        'Verifica el estado signo
        If revise_entero_162(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado entero_sin_signo
        If revise_entero_163(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_siEncabezado(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(str, indexT)
        'Verifica el estado "SI "
        If revise_siEncabezado_164(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_sinosi(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(str, indexT)
        'Verifica el estado "SINOSI "
```

```
        If revise_sinosi_168(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_sino(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(str, indexT)
        'Verifica el estado "SINO "
        If revise_sino_173(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_finsi(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(str, indexT)
        'Verifica el estado "FINSI"
        If revise_finsi_177(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_repiteEncabezado(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(str, indexT)
        'Verifica el estado "REPITE "
        If revise_repiteEncabezado_178(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_finRepite(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(str, indexT)
        'Verifica el estado "HASTA "
        If revise_finRepite_180(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_mientrasEncabezado(ByVal str As String, ByRef index As Integer) As↙
     Boolean
```

```
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(str, indexT)
        'Verifica el estado "MIENTRAS "
        If revise_mientrasEncabezado_182(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_finMientras(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(str, indexT)
        'Verifica el estado "FINMIENTRAS"
        If revise_finMientras_185(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_paraEncabezado(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(str, indexT)
        'Verifica el estado "PARA "
        If revise_paraEncabezado_186(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_finPara(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(str, indexT)
        'Verifica el estado "FINPARA"
        If revise_finPara_193(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_segunEncabezado(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
```

```
        avance(str, indexT)
        'Verifica el estado "SEGUN "
        If revise_segunEncabezado_194(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_segunOpcion(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(str, indexT)
        'Verifica el estado "CASO "
        If revise_segunOpcion_197(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_segunOpcionOtro(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(str, indexT)
        'Verifica el estado "OTROCASO"
        If revise_segunOpcionOtro_200(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_finSegun(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(str, indexT)
        'Verifica el estado "FINSEGUN"
        If revise_finSegun_202(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_constante(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(str, indexT)
        'Verifica el estado sarta
        If revise_constante_203(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado numero
```

```
        If revise_constante_204(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_Lee(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        'Terminal
        If indexT + 3 <= str.Length AndAlso str.Substring(indexT, 3).ToLower = "lee" Then
            indexT += 3
            index = indexT
            Return True
        Else
        End If
    End Function

    Private Function revise_listaVariables(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(str, indexT)
        'Verifica el estado variable
        If revise_listaVariables_205(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_Escribe(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        'Terminal
        If indexT + 7 <= str.Length AndAlso str.Substring(indexT, 7).ToLower = "escribe" Then
            indexT += 7
            index = indexT
            Return True
        Else
        End If
    End Function

    Private Function revise_lista_expresiones(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(str, indexT)
        'Verifica el estado expresion
        If revise_lista_expresiones_208(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function
```

```
    Private Function revise_retornoFuncion(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(str, indexT)
        'Verifica el estado Retorne
        If revise_retornoFuncion_211(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_retornoProcedimiento(ByVal str As String, ByRef index As Integer)
    As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(str, indexT)
        'Verifica el estado Retorne
        If revise_retornoProcedimiento_215(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_Retorne(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        'Terminal
        If indexT + 7 <= str.Length AndAlso str.Substring(indexT, 7).ToLower = "retorne" Then
            indexT += 7
            index = indexT
            Return True
        Else
        End If
    End Function

    Private Function revise_termino(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(str, indexT)
        'Verifica el estado numero_sin_signo
        If revise_termino_216(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado sarta
        If revise_termino_217(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado variable
        If revise_termino_218(str, indexT) Then
```

```
            index = indexT
            Return True
        End If
        'Verifica el estado llamarFuncion
        If revise_termino_219(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado "("
        If revise_termino_220(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_operacion(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(str, indexT)
        'Verifica el estado orOp
        If revise_operacion_223(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado andOp
        If revise_operacion_224(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado RelOp
        If revise_operacion_225(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado AddOp
        If revise_operacion_226(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado MulOp
        If revise_operacion_227(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado "&"
        If revise_operacion_228(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado Not
        If revise_operacion_229(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_orOp(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(str, indexT)
        'Verifica el estado Or
```

```
        If revise_orOp_231(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_andOp(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(str, indexT)
        'Verifica el estado And
        If revise_andOp_232(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_RelOp(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        'Verifica el estado "<"
        If revise_RelOp_233(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado ">"
        If revise_RelOp_234(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado "<="
        If revise_RelOp_235(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado ">="
        If revise_RelOp_236(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado "="
        If revise_RelOp_237(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado "<>"
        If revise_RelOp_238(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_AddOp(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        'Verifica el estado "+"
        If revise_AddOp_239(str, indexT) Then
```

```
            index = indexT
            Return True
        End If
        'Verifica el estado "-"
        If revise_AddOp_240(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_MulOp(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        'Verifica el estado "*"
        If revise_MulOp_241(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado "/"
        If revise_MulOp_242(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado "\"
        If revise_MulOp_243(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado "Mod"
        If revise_MulOp_244(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_Not(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        'Verifica el estado "no"
        If revise_Not_245(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_numero_sin_signo(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(str, indexT)
        'Verifica el estado entero_sin_signo
        If revise_numero_sin_signo_246(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado real_sin_signo
        If revise_numero_sin_signo_247(str, indexT) Then
```

```
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_sarta(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(str, indexT)
        'Verifica el estado Comilla
        If revise_sarta_248(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_nombreFuncion(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(str, indexT)
        'Verifica el estado Abs
        If revise_nombreFuncion_251(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado Coseno
        If revise_nombreFuncion_252(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado Largo
        If revise_nombreFuncion_253(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado Log
        If revise_nombreFuncion_254(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado Aleatorio
        If revise_nombreFuncion_255(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado Seno
        If revise_nombreFuncion_256(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado Mayuscula
        If revise_nombreFuncion_257(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado Minuscula
        If revise_nombreFuncion_258(str, indexT) Then
            index = indexT
            Return True
        End If
```

```
        'Verifica el estado identificador
        If revise_nombreFuncion_259(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_Abs(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        'Verifica el estado "abs"
        If revise_Abs_260(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_Coseno(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        'Terminal
        If indexT + 6 <= str.Length AndAlso str.Substring(indexT, 6).ToLower = "coseno" Then
            indexT += 6
            index = indexT
            Return True
        Else
        End If
    End Function

    Private Function revise_Largo(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        'Terminal
        If indexT + 5 <= str.Length AndAlso str.Substring(indexT, 5).ToLower = "largo" Then
            indexT += 5
            index = indexT
            Return True
        Else
        End If
    End Function

    Private Function revise_Log(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        'Terminal
        If indexT + 3 <= str.Length AndAlso str.Substring(indexT, 3).ToLower = "log" Then
            indexT += 3
            index = indexT
            Return True
        Else
        End If
    End Function
```

```
    Private Function revise_Aleatorio(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        'Terminal
        If indexT + 9 <= str.Length AndAlso str.Substring(indexT, 9).ToLower = "aleatorio"
    Then
            indexT += 9
            index = indexT
            Return True
        Else
        End If
    End Function

    Private Function revise_Seno(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        'Terminal
        If indexT + 4 <= str.Length AndAlso str.Substring(indexT, 4).ToLower = "seno" Then
            indexT += 4
            index = indexT
            Return True
        Else
        End If
    End Function

    Private Function revise_Mayuscula(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        'Terminal
        If indexT + 9 <= str.Length AndAlso str.Substring(indexT, 9).ToLower = "mayuscula"
    Then
            indexT += 9
            index = indexT
            Return True
        Else
        End If
    End Function

    Private Function revise_Minuscula(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        'Terminal
        If indexT + 9 <= str.Length AndAlso str.Substring(indexT, 9).ToLower = "minuscula"
    Then
            indexT += 9
            index = indexT
            Return True
        Else
        End If
    End Function

    Private Function revise_Letra(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
```

```
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        'Verifica el estado A
        If revise_Letra_261(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado B
        If revise_Letra_262(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado C
        If revise_Letra_263(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado D
        If revise_Letra_264(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado E
        If revise_Letra_265(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado F
        If revise_Letra_266(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado G
        If revise_Letra_267(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado H
        If revise_Letra_268(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado I
        If revise_Letra_269(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado J
        If revise_Letra_270(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado K
        If revise_Letra_271(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado L
        If revise_Letra_272(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado M
        If revise_Letra_273(str, indexT) Then
            index = indexT
            Return True
```

```
        End If
        'Verifica el estado N
        If revise_Letra_274(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado O
        If revise_Letra_275(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado P
        If revise_Letra_276(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado Q
        If revise_Letra_277(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado R
        If revise_Letra_278(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado S
        If revise_Letra_279(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado T
        If revise_Letra_280(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado U
        If revise_Letra_281(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado V
        If revise_Letra_282(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado W
        If revise_Letra_283(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado X
        If revise_Letra_284(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado Y
        If revise_Letra_285(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado Z
        If revise_Letra_286(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_LetrasNumerosUnderscores(ByVal str As String, ByRef index As      ↙
```

```
    Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        'Verifica el estado Letra
        If revise_LetrasNumerosUnderscores_287(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado "_"
        If revise_LetrasNumerosUnderscores_288(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado Num
        If revise_LetrasNumerosUnderscores_289(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_Or(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        'Verifica el estado "o"
        If revise_Or_290(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_And(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        'Verifica el estado "y"
        If revise_And_291(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_definicionProcedimiento(ByVal str As String, ByRef index As ↙
    Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(str, indexT)
        'Verifica el estado Procedimiento
        If revise_definicionProcedimiento_292(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_definicionFuncion(ByVal str As String, ByRef index As Integer) As ↙
```

```
    Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(str, indexT)
        'Verifica el estado Funcion
        If revise_definicionFuncion_300(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_Procedimiento(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        'Terminal
        If indexT + 13 <= str.Length AndAlso str.Substring(indexT, 13).ToLower =
    "procedimiento" Then
            indexT += 13
            index = indexT
            Return True
        Else
        End If
    End Function

    Private Function revise_listaParametros(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(str, indexT)
        'Verifica el estado definicionFormalParametro
        If revise_listaParametros_309(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_FinProcedimiento(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        'Terminal
        If indexT + 16 <= str.Length AndAlso str.Substring(indexT, 16).ToLower =
    "finprocedimiento" Then
            indexT += 16
            index = indexT
            Return True
        Else
        End If
    End Function

    Private Function revise_Funcion(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
```

```
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        'Terminal
        If indexT + 7 <= str.Length AndAlso str.Substring(indexT, 7).ToLower = "funcion" Then
            indexT += 7
            index = indexT
            Return True
        Else
        End If
    End Function

    Private Function revise_FinFuncion(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        'Terminal
        If indexT + 10 <= str.Length AndAlso str.Substring(indexT, 10).ToLower = "finfuncion" ↙
    Then
            indexT += 10
            index = indexT
            Return True
        Else
        End If
    End Function

    Private Function revise_definicionFormalParametro(ByVal str As String, ByRef index As      ↙
    Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(str, indexT)
        'Verifica el estado tipoDato
        If revise_definicionFormalParametro_312(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_numero(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(str, indexT)
        'Verifica el estado entero
        If revise_numero_314(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado real
        If revise_numero_315(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_real(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
```

```
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(str, indexT)
        'Verifica el estado signo
        If revise_real_316(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado real_sin_signo
        If revise_real_317(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_entero_sin_signo(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(str, indexT)
        'Verifica el estado Num
        If revise_entero_sin_signo_318(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_real_sin_signo(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(str, indexT)
        'Verifica el estado entero_sin_signo
        If revise_real_sin_signo_319(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_signo(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(str, indexT)
        'Verifica el estado "+"
        If revise_signo_322(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado "-"
        If revise_signo_323(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function
```

```
    Private Function revise_Comilla(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        'Verifica el estado ""
        If revise_Comilla_324(str, indexT) Then
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_Num(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        'Verifica el estado 1
        If revise_Num_326(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado 2
        If revise_Num_327(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado 3
        If revise_Num_328(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado 4
        If revise_Num_329(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado 5
        If revise_Num_330(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado 6
        If revise_Num_331(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado 7
        If revise_Num_332(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado 8
        If revise_Num_333(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado 9
        If revise_Num_334(str, indexT) Then
            index = indexT
            Return True
        End If
        'Verifica el estado 0
        If revise_Num_335(str, indexT) Then
```

```
            index = indexT
            Return True
        End If
    End Function

    Private Function revise_1(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        'Terminal
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "1" Then
            indexT += 1
            index = indexT
            Return True
        Else
        End If
    End Function

    Private Function revise_2(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        'Terminal
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "2" Then
            indexT += 1
            index = indexT
            Return True
        Else
        End If
    End Function

    Private Function revise_3(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        'Terminal
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "3" Then
            indexT += 1
            index = indexT
            Return True
        Else
        End If
    End Function

    Private Function revise_4(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        'Terminal
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "4" Then
            indexT += 1
            index = indexT
            Return True
        Else
        End If
    End Function
```

```
    Private Function revise_5(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        'Terminal
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "5" Then
            indexT += 1
            index = indexT
            Return True
        Else
        End If
    End Function

    Private Function revise_6(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        'Terminal
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "6" Then
            indexT += 1
            index = indexT
            Return True
        Else
        End If
    End Function

    Private Function revise_7(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        'Terminal
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "7" Then
            indexT += 1
            index = indexT
            Return True
        Else
        End If
    End Function

    Private Function revise_8(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        'Terminal
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "8" Then
            indexT += 1
            index = indexT
            Return True
        Else
        End If
    End Function

    Private Function revise_9(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
```

```
            Return False
        End If
        'Terminal
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "9" Then
            indexT += 1
            index = indexT
            Return True
        Else
        End If
    End Function

    Private Function revise_0(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        'Terminal
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "0" Then
            indexT += 1
            index = indexT
            Return True
        Else
        End If
    End Function

    Private Function revise_chr10(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(str, indexT)
        'Terminal
        If str(indexT) = Chr(10) Then
            indexT += 1
            index = indexT
            Return True
        Else
            message = "Error inesperado"
            indexError = indexT
        End If
    End Function

    Private Function revise_chr13(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        avance(str, indexT)
        'Terminal
        If str(indexT) = Chr(13) Then
            indexT += 1
            index = indexT
            Return True
        Else
            message = "Error inesperado"
            indexError = indexT
        End If
    End Function

    Private Function revise_A(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
```

```
            indexError = indexT
            Return False
        End If
        'Terminal
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "a" Then
            indexT += 1
            index = indexT
            Return True
        Else
        End If
    End Function

    Private Function revise_B(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        'Terminal
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "b" Then
            indexT += 1
            index = indexT
            Return True
        Else
        End If
    End Function

    Private Function revise_C(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        'Terminal
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "c" Then
            indexT += 1
            index = indexT
            Return True
        Else
        End If
    End Function

    Private Function revise_D(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        'Terminal
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "d" Then
            indexT += 1
            index = indexT
            Return True
        Else
        End If
    End Function

    Private Function revise_E(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        'Terminal
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "e" Then
```

```
            indexT += 1
            index = indexT
            Return True
        Else
        End If
    End Function

    Private Function revise_F(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        'Terminal
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "f" Then
            indexT += 1
            index = indexT
            Return True
        Else
        End If
    End Function

    Private Function revise_G(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        'Terminal
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "g" Then
            indexT += 1
            index = indexT
            Return True
        Else
        End If
    End Function

    Private Function revise_H(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        'Terminal
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "h" Then
            indexT += 1
            index = indexT
            Return True
        Else
        End If
    End Function

    Private Function revise_I(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        'Terminal
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "i" Then
            indexT += 1
            index = indexT
            Return True
        Else
        End If
```

```
    End Function

    Private Function revise_J(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        'Terminal
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "j" Then
            indexT += 1
            index = indexT
            Return True
        Else
        End If
    End Function

    Private Function revise_K(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        'Terminal
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "k" Then
            indexT += 1
            index = indexT
            Return True
        Else
        End If
    End Function

    Private Function revise_L(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        'Terminal
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "l" Then
            indexT += 1
            index = indexT
            Return True
        Else
        End If
    End Function

    Private Function revise_M(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        'Terminal
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "m" Then
            indexT += 1
            index = indexT
            Return True
        Else
        End If
    End Function

    Private Function revise_N(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
```

```
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        'Terminal
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "n" Then
            indexT += 1
            index = indexT
            Return True
        Else
        End If
    End Function

    Private Function revise_O(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        'Terminal
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "o" Then
            indexT += 1
            index = indexT
            Return True
        Else
        End If
    End Function

    Private Function revise_P(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        'Terminal
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "p" Then
            indexT += 1
            index = indexT
            Return True
        Else
        End If
    End Function

    Private Function revise_Q(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        'Terminal
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "q" Then
            indexT += 1
            index = indexT
            Return True
        Else
        End If
    End Function

    Private Function revise_R(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        'Terminal
```

```
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "r" Then
            indexT += 1
            index = indexT
            Return True
        Else
        End If
    End Function

    Private Function revise_S(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        'Terminal
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "s" Then
            indexT += 1
            index = indexT
            Return True
        Else
        End If
    End Function

    Private Function revise_T(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        'Terminal
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "t" Then
            indexT += 1
            index = indexT
            Return True
        Else
        End If
    End Function

    Private Function revise_U(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        'Terminal
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "u" Then
            indexT += 1
            index = indexT
            Return True
        Else
        End If
    End Function

    Private Function revise_V(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        'Terminal
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "v" Then
            indexT += 1
            index = indexT
            Return True
        Else
```

```
        End If
    End Function

    Private Function revise_W(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        'Terminal
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "w" Then
            indexT += 1
            index = indexT
            Return True
        Else
        End If
    End Function

    Private Function revise_X(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        'Terminal
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "x" Then
            indexT += 1
            index = indexT
            Return True
        Else
        End If
    End Function

    Private Function revise_Y(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        'Terminal
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "y" Then
            indexT += 1
            index = indexT
            Return True
        Else
        End If
    End Function

    Private Function revise_Z(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
            message = "No se esperaba fin de código"
            indexError = indexT
            Return False
        End If
        'Terminal
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "z" Then
            indexT += 1
            index = indexT
            Return True
        Else
        End If
    End Function

    Private Function revise_programaFuente_0(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
```

```
        Dim indexT As Integer = index
        If revise_NuevaLinea(str, indexT) Then
            'Verifica el estado definicionModulo
            If revise_programaFuente_1(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_programaFuente_1(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        If revise_definicionModulo(str, indexT) Then
            'Verifica el estado definicionModulo
            If revise_programaFuente_1(Str, indexT) Then
                index = indexT
                Return True
            End If
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_NuevaLinea_2(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        If revise_chr10(str, indexT) Then
            'Verifica el estado chr10
            If revise_NuevaLinea_2(Str, indexT) Then
                index = indexT
                Return True
            End If
            'Verifica el estado chr13
            If revise_NuevaLinea_3(Str, indexT) Then
                index = indexT
                Return True
            End If
            index = indexT
            Return True
        ElseIf indexError <= indexT Then
            message = "Se esperaba 'chr10'"
            indexError = indexT
        End If
    End Function

    Private Function revise_NuevaLinea_3(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        If revise_chr13(str, indexT) Then
            'Verifica el estado chr10
            If revise_NuevaLinea_4(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_NuevaLinea_4(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        If revise_chr10(str, indexT) Then
            'Verifica el estado chr10
            If revise_NuevaLinea_2(Str, indexT) Then
                index = indexT
                Return True
```

```
            End If
            'Verifica el estado chr13
            If revise_NuevaLinea_3(Str, indexT) Then
                index = indexT
                Return True
            End If
            index = indexT
            Return True
        ElseIf indexError <= indexT Then
            message = "Se esperaba 'chr10'"
            indexError = indexT
        End If
    End Function

    Private Function revise_definicionModulo_5(ByVal str As String, ByRef index As Integer) As
     Boolean
        Dim indexT As Integer = index
        If revise_definicionProcedimiento(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_definicionModulo_6(ByVal str As String, ByRef index As Integer) As
     Boolean
        Dim indexT As Integer = index
        If revise_definicionFuncion(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_codigoAbierto_7(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        If revise_NuevaLinea(str, indexT) Then
            'Verifica el estado sentencia
            If revise_codigoAbierto_8(Str, indexT) Then
                index = indexT
                Return True
            End If
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_codigoAbierto_8(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        If revise_sentencia(str, indexT) Then
            'Verifica el estado delimitador
            If revise_codigoAbierto_9(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_codigoAbierto_9(ByVal str As String, ByRef index As Integer) As
    Boolean
```

```
        Dim indexT As Integer = index
        If revise_delimitador(str, indexT) Then
            'Verifica el estado sentencia
            If revise_codigoAbierto_8(Str, indexT) Then
                index = indexT
                Return True
            End If
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_codigoAbierto_10(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        If revise_Fin(str, indexT) Then
        End If
    End Function

    Private Function revise_sentencia_11(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        If revise_sentenciaControl(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_sentencia_12(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        If revise_comando(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_sentencia_13(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        If revise_comandoMovil(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_sentencia_14(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        If revise_asignacion(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function
```

```
    Private Function revise_delimitador_15(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        If revise_NuevaLinea(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_delimitador_16(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        If revise_DosPuntos(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_DosPuntos_17(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = ":" Then
            indexT += 1
            index = indexT
            Return True
        ElseIf indexError <= indexT Then
            message = "Se esperaba ':'"
            indexError = indexT
        End If
    End Function

    Private Function revise_sentenciaControl_18(ByVal str As String, ByRef index As Integer)
    As Boolean
        Dim indexT As Integer = index
        If revise_declaracion(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_sentenciaControl_19(ByVal str As String, ByRef index As Integer)
    As Boolean
        Dim indexT As Integer = index
        If revise_si(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_sentenciaControl_20(ByVal str As String, ByRef index As Integer)
    As Boolean
        Dim indexT As Integer = index
        If revise_repite(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
```

```
            Return True
        End If
    End If
End Function

Private Function revise_sentenciaControl_21(ByVal str As String, ByRef index As Integer) ↙
As Boolean
    Dim indexT As Integer = index
    If revise_para(str, indexT) Then
        If revise_finToken(Str, indexT) Then
            avance(Str, indexT)
            index = indexT
            Return True
        End If
    End If
End Function

Private Function revise_sentenciaControl_22(ByVal str As String, ByRef index As Integer) ↙
As Boolean
    Dim indexT As Integer = index
    If revise_mientras(str, indexT) Then
        If revise_finToken(Str, indexT) Then
            avance(Str, indexT)
            index = indexT
            Return True
        End If
    End If
End Function

Private Function revise_sentenciaControl_23(ByVal str As String, ByRef index As Integer) ↙
As Boolean
    Dim indexT As Integer = index
    If revise_segun(str, indexT) Then
        If revise_finToken(Str, indexT) Then
            avance(Str, indexT)
            index = indexT
            Return True
        End If
    End If
End Function

Private Function revise_comando_24(ByVal str As String, ByRef index As Integer) As Boolean
    Dim indexT As Integer = index
    If revise_borraPantalla(str, indexT) Then
        If revise_finToken(Str, indexT) Then
            avance(Str, indexT)
            index = indexT
            Return True
        End If
    End If
End Function

Private Function revise_comando_25(ByVal str As String, ByRef index As Integer) As Boolean
    Dim indexT As Integer = index
    If revise_borraTexto(str, indexT) Then
        If revise_finToken(Str, indexT) Then
            avance(Str, indexT)
            index = indexT
            Return True
        End If
    End If
End Function

Private Function revise_comando_26(ByVal str As String, ByRef index As Integer) As Boolean
    Dim indexT As Integer = index
    If revise_comentario(str, indexT) Then
        If revise_finToken(Str, indexT) Then
            avance(Str, indexT)
            index = indexT
```

```
                Return True
            End If
        End If
    End Function

    Private Function revise_comando_27(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_salir(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_comando_28(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_salirBucle(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_comando_29(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_leer(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_comando_30(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_imprimir(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_comando_31(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_retorno(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_comando_32(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_llamarFuncion(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
```

```
    End Function

    Private Function revise_comandoMovil_33(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        If revise_avanza(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_comandoMovil_34(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        If revise_giraDerecha(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_comandoMovil_35(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        If revise_giraIzquierda(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_comandoMovil_36(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        If revise_retrocede(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_comandoMovil_37(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        If revise_subeLapiz(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_comandoMovil_38(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        If revise_bajaLapiz(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
```

```
                Return True
            End If
        End If
    End Function

    Private Function revise_comandoMovil_39(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        If revise_centro(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_comandoMovil_40(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        If revise_ponGrosor(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_comandoMovil_41(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        If revise_ponFondo(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_comandoMovil_42(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        If revise_ponColorLapiz(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_comandoMovil_43(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        If revise_ocultar(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_comandoMovil_44(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        If revise_mostrar(str, indexT) Then
```

```
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_asignacion_45(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        If revise_variable(str, indexT) Then
            'Verifica el estado "="
            If revise_asignacion_46(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_asignacion_46(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "=" Then
            indexT += 1
            'Verifica el estado expresion
            If revise_asignacion_47(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_asignacion_47(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        If revise_expresion(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_declaracion_48(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        If revise_tipoDato(str, indexT) Then
            'Verifica el estado identificadorVariable
            If revise_declaracion_49(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_declaracion_49(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        If revise_identificadorVariable(str, indexT) Then
            'Verifica el estado Coma
            If revise_declaracion_50(Str, indexT) Then
                index = indexT
                Return True
            End If
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
```

```
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_declaracion_50(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        If revise_Coma(str, indexT) Then
            'Verifica el estado identificadorVariable
            If revise_declaracion_51(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_declaracion_51(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        If revise_identificadorVariable(str, indexT) Then
            'Verifica el estado Coma
            If revise_declaracion_50(Str, indexT) Then
                index = indexT
                Return True
            End If
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_si_52(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_siEncabezado(str, indexT) Then
            'Verifica el estado sinosi
            If revise_si_53(Str, indexT) Then
                index = indexT
                Return True
            End If
            'Verifica el estado sino
            If revise_si_54(Str, indexT) Then
                index = indexT
                Return True
            End If
            'Verifica el estado finsi
            If revise_si_55(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_si_53(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_sinosi(str, indexT) Then
            'Verifica el estado sino
            If revise_si_54(Str, indexT) Then
                index = indexT
                Return True
            End If
            'Verifica el estado finsi
            If revise_si_55(Str, indexT) Then
                index = indexT
                Return True
            End If
```

```
        End If
    End Function

    Private Function revise_si_54(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_sino(str, indexT) Then
            'Verifica el estado finsi
            If revise_si_55(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_si_55(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_finsi(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_repite_56(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_repiteEncabezado(str, indexT) Then
            'Verifica el estado codigoAbierto
            If revise_repite_57(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_repite_57(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_codigoAbierto(str, indexT) Then
            'Verifica el estado finRepite
            If revise_repite_58(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_repite_58(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_finRepite(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_para_59(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_paraEncabezado(str, indexT) Then
            'Verifica el estado codigoAbierto
            If revise_para_60(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function
```

```
    Private Function revise_para_60(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_codigoAbierto(str, indexT) Then
            'Verifica el estado finPara
            If revise_para_61(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_para_61(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_finPara(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_mientras_62(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        If revise_mientrasEncabezado(str, indexT) Then
            'Verifica el estado codigoAbierto
            If revise_mientras_63(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_mientras_63(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        If revise_codigoAbierto(str, indexT) Then
            'Verifica el estado finMientras
            If revise_mientras_64(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_mientras_64(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        If revise_finMientras(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_segun_65(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_segunEncabezado(str, indexT) Then
            'Verifica el estado segunOpcion
            If revise_segun_66(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function
```

```
    Private Function revise_segun_66(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_segunOpcion(str, indexT) Then
            'Verifica el estado segunOpcion
            If revise_segun_66(Str, indexT) Then
                index = indexT
                Return True
            End If
            'Verifica el estado segunOpcionOtro
            If revise_segun_67(Str, indexT) Then
                index = indexT
                Return True
            End If
            'Verifica el estado finSegun
            If revise_segun_68(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_segun_67(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_segunOpcionOtro(str, indexT) Then
            'Verifica el estado finSegun
            If revise_segun_68(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_segun_68(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_finSegun(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_avanza_69(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 6 <= str.Length AndAlso str.Substring(indexT, 6).ToLower = "avanza" Then
            indexT += 6
            'Verifica el estado "("
            If revise_avanza_70(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_avanza_70(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "(" Then
            indexT += 1
            'Verifica el estado expresion
            If revise_avanza_71(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function
```

```
    Private Function revise_avanza_71(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_expresion(str, indexT) Then
            'Verifica el estado ")"
            If revise_avanza_72(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_avanza_72(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = ")" Then
            indexT += 1
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_avanza_73(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 2 <= str.Length AndAlso str.Substring(indexT, 2).ToLower = "av" Then
            indexT += 2
            'Verifica el estado "("
            If revise_avanza_70(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_avanza_74(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 8 <= str.Length AndAlso str.Substring(indexT, 8).ToLower = "adelante" Then
            indexT += 8
            'Verifica el estado "("
            If revise_avanza_70(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_avanza_75(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 2 <= str.Length AndAlso str.Substring(indexT, 2).ToLower = "ad" Then
            indexT += 2
            'Verifica el estado "("
            If revise_avanza_70(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_giraDerecha_76(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 11 <= str.Length AndAlso str.Substring(indexT, 11).ToLower = "giraderecha"↙
     Then
```

```
            indexT += 11
            'Verifica el estado "("
            If revise_giraDerecha_77(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_giraDerecha_77(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "(" Then
            indexT += 1
            'Verifica el estado expresion
            If revise_giraDerecha_78(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_giraDerecha_78(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        If revise_expresion(str, indexT) Then
            'Verifica el estado ")"
            If revise_giraDerecha_79(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_giraDerecha_79(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = ")" Then
            indexT += 1
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_giraDerecha_80(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 2 <= str.Length AndAlso str.Substring(indexT, 2).ToLower = "gd" Then
            indexT += 2
            'Verifica el estado "("
            If revise_giraDerecha_77(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_giraIzquierda_81(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 13 <= str.Length AndAlso str.Substring(indexT, 13).ToLower =
    "giraizquierda" Then
```

```
            indexT += 13
            'Verifica el estado "("
            If revise_giraIzquierda_82(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_giraIzquierda_82(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "(" Then
            indexT += 1
            'Verifica el estado expresion
            If revise_giraIzquierda_83(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_giraIzquierda_83(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        If revise_expresion(str, indexT) Then
            'Verifica el estado ")"
            If revise_giraIzquierda_84(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_giraIzquierda_84(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = ")" Then
            indexT += 1
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_giraIzquierda_85(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 2 <= str.Length AndAlso str.Substring(indexT, 2).ToLower = "gi" Then
            indexT += 2
            'Verifica el estado "("
            If revise_giraIzquierda_82(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_retrocede_86(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 9 <= str.Length AndAlso str.Substring(indexT, 9).ToLower = "retrocede"
    Then
```

```
            indexT += 9
            'Verifica el estado "("
            If revise_retrocede_87(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_retrocede_87(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "(" Then
            indexT += 1
            'Verifica el estado expresion
            If revise_retrocede_88(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_retrocede_88(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        If revise_expresion(str, indexT) Then
            'Verifica el estado ")"
            If revise_retrocede_89(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_retrocede_89(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = ")" Then
            indexT += 1
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_retrocede_90(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 2 <= str.Length AndAlso str.Substring(indexT, 2).ToLower = "re" Then
            indexT += 2
            'Verifica el estado "("
            If revise_retrocede_87(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_retrocede_91(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 5 <= str.Length AndAlso str.Substring(indexT, 5).ToLower = "atras" Then
            indexT += 5
```

```
            'Verifica el estado "("
            If revise_retrocede_87(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_retrocede_92(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 2 <= str.Length AndAlso str.Substring(indexT, 2).ToLower = "at" Then
            indexT += 2
            'Verifica el estado "("
            If revise_retrocede_87(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_subeLapiz_93(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 9 <= str.Length AndAlso str.Substring(indexT, 9).ToLower = "subelapiz"
    Then
            indexT += 9
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_subeLapiz_94(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 2 <= str.Length AndAlso str.Substring(indexT, 2).ToLower = "sl" Then
            indexT += 2
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_bajaLapiz_95(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 9 <= str.Length AndAlso str.Substring(indexT, 9).ToLower = "bajalapiz"
    Then
            indexT += 9
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_bajaLapiz_96(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
```

```
        avance(Str, indexT)
        If indexT + 2 <= str.Length AndAlso str.Substring(indexT, 2).ToLower = "bl" Then
            indexT += 2
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_centro_97(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 6 <= str.Length AndAlso str.Substring(indexT, 6).ToLower = "centro" Then
            indexT += 6
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_ponGrosor_98(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 9 <= str.Length AndAlso str.Substring(indexT, 9).ToLower = "pongrosor"
    Then
            indexT += 9
            'Verifica el estado "("
            If revise_ponGrosor_99(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_ponGrosor_99(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "(" Then
            indexT += 1
            'Verifica el estado expresion
            If revise_ponGrosor_100(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_ponGrosor_100(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        If revise_expresion(str, indexT) Then
            'Verifica el estado ")"
            If revise_ponGrosor_101(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_ponGrosor_101(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
```

```
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = ")" Then
            indexT += 1
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_ponGrosor_102(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 4 <= str.Length AndAlso str.Substring(indexT, 4).ToLower = "pong" Then
            indexT += 4
            'Verifica el estado "("
            If revise_ponGrosor_99(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_ponFondo_103(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 8 <= str.Length AndAlso str.Substring(indexT, 8).ToLower = "ponfondo" Then
            indexT += 8
            'Verifica el estado "("
            If revise_ponFondo_104(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_ponFondo_104(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "(" Then
            indexT += 1
            'Verifica el estado expresion
            If revise_ponFondo_105(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_ponFondo_105(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        If revise_expresion(str, indexT) Then
            'Verifica el estado ")"
            If revise_ponFondo_106(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_ponFondo_106(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = ")" Then
```

```
            indexT += 1
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_ponFondo_107(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 4 <= str.Length AndAlso str.Substring(indexT, 4).ToLower = "ponf" Then
            indexT += 4
            'Verifica el estado "("
            If revise_ponFondo_104(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_ponColorLapiz_108(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 13 <= str.Length AndAlso str.Substring(indexT, 13).ToLower = ↙
    "poncolorlapiz" Then
            indexT += 13
            'Verifica el estado "("
            If revise_ponColorLapiz_109(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_ponColorLapiz_109(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "(" Then
            indexT += 1
            'Verifica el estado expresion
            If revise_ponColorLapiz_110(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_ponColorLapiz_110(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        If revise_expresion(str, indexT) Then
            'Verifica el estado ")"
            If revise_ponColorLapiz_111(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_ponColorLapiz_111(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = ")" Then
```

```
            indexT += 1
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_ponColorLapiz_112(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 5 <= str.Length AndAlso str.Substring(indexT, 5).ToLower = "poncl" Then
            indexT += 5
            'Verifica el estado "("
            If revise_ponColorLapiz_109(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_ocultar_113(ByVal str As String, ByRef index As Integer) As       ↙
    Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 11 <= str.Length AndAlso str.Substring(indexT, 11).ToLower = "ocultamovil"↙
     Then
            indexT += 11
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_ocultar_114(ByVal str As String, ByRef index As Integer) As       ↙
    Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 2 <= str.Length AndAlso str.Substring(indexT, 2).ToLower = "om" Then
            indexT += 2
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_mostrar_115(ByVal str As String, ByRef index As Integer) As       ↙
    Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 12 <= str.Length AndAlso str.Substring(indexT, 12).ToLower = "muestramovil↙
    " Then
            indexT += 12
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_mostrar_116(ByVal str As String, ByRef index As Integer) As       ↙
    Boolean
```

```
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 2 <= str.Length AndAlso str.Substring(indexT, 2).ToLower = "mm" Then
            indexT += 2
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_borraPantalla_117(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 13 <= str.Length AndAlso str.Substring(indexT, 13).ToLower =             ↙
    "borrapantalla" Then
            indexT += 13
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_borraPantalla_118(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 2 <= str.Length AndAlso str.Substring(indexT, 2).ToLower = "bp" Then
            indexT += 2
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_borraTexto_119(ByVal str As String, ByRef index As Integer) As    ↙
    Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 10 <= str.Length AndAlso str.Substring(indexT, 10).ToLower = "borratexto" ↙
    Then
            indexT += 10
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_borraTexto_120(ByVal str As String, ByRef index As Integer) As    ↙
    Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 2 <= str.Length AndAlso str.Substring(indexT, 2).ToLower = "bt" Then
            indexT += 2
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function
```

```
    Private Function revise_comentario_121(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Apostrofe(str, indexT) Then
            'Verifica el estado AnythingExceptDoubleQuote
            If revise_comentario_122(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_comentario_122(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_AnythingExceptDoubleQuote(str, indexT) Then
            'Verifica el estado NuevaLinea
            If revise_comentario_123(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_comentario_123(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_NuevaLinea(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_salir_124(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 5 <= str.Length AndAlso str.Substring(indexT, 5).ToLower = "salir" Then
            indexT += 5
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_salirBucle_125(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 10 <= str.Length AndAlso str.Substring(indexT, 10).ToLower = "salirbucle" Then
            indexT += 10
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_leer_126(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Lee(str, indexT) Then
            'Verifica el estado listaVariables
```

```
            If revise_leer_127(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_leer_127(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_listaVariables(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_imprimir_128(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        If revise_Escribe(str, indexT) Then
            'Verifica el estado lista_expresiones
            If revise_imprimir_129(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_imprimir_129(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        If revise_lista_expresiones(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_retorno_130(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        If revise_retornoFuncion(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_retorno_131(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        If revise_retornoProcedimiento(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_llamarFuncion_132(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
```

```
        If revise_nombreFuncion(str, indexT) Then
            'Verifica el estado "("
            If revise_llamarFuncion_133(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_llamarFuncion_133(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "(" Then
            indexT += 1
            'Verifica el estado lista_expresiones
            If revise_llamarFuncion_134(Str, indexT) Then
                index = indexT
                Return True
            End If
            'Verifica el estado ")"
            If revise_llamarFuncion_135(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_llamarFuncion_134(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        If revise_lista_expresiones(str, indexT) Then
            'Verifica el estado ")"
            If revise_llamarFuncion_135(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_llamarFuncion_135(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = ")" Then
            indexT += 1
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_variable_136(ByVal str As String, ByRef index As Integer) As      ↙
    Boolean
        Dim indexT As Integer = index
        If revise_identificador(str, indexT) Then
            'Verifica el estado "("
            If revise_variable_137(Str, indexT) Then
                index = indexT
                Return True
            End If
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
```

```
    End Function

    Private Function revise_variable_137(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "(" Then
            indexT += 1
            'Verifica el estado listaIndices
            If revise_variable_138(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_variable_138(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        If revise_listaIndices(str, indexT) Then
            'Verifica el estado ")"
            If revise_variable_139(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_variable_139(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = ")" Then
            indexT += 1
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_expresion_140(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        If revise_termino(str, indexT) Then
            'Verifica el estado operacion
            If revise_expresion_141(Str, indexT) Then
                index = indexT
                Return True
            End If
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_expresion_141(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        If revise_operacion(str, indexT) Then
            'Verifica el estado termino
            If revise_expresion_142(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
```

```
    End Function

    Private Function revise_expresion_142(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        If revise_termino(str, indexT) Then
            'Verifica el estado operacion
            If revise_expresion_141(Str, indexT) Then
                index = indexT
                Return True
            End If
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_identificador_143(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        If revise_Letra(str, indexT) Then
            'Verifica el estado LetrasNumerosUnderscores
            If revise_identificador_144(Str, indexT) Then
                index = indexT
                Return True
            End If
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_identificador_144(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        If revise_LetrasNumerosUnderscores(str, indexT) Then
            'Verifica el estado LetrasNumerosUnderscores
            If revise_identificador_144(Str, indexT) Then
                index = indexT
                Return True
            End If
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_listaIndices_145(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        If revise_expresion(str, indexT) Then
            'Verifica el estado Coma
            If revise_listaIndices_146(Str, indexT) Then
                index = indexT
                Return True
            End If
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function
```

```
    Private Function revise_listaIndices_146(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Coma(str, indexT) Then
            'Verifica el estado listaIndices
            If revise_listaIndices_147(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_listaIndices_147(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_listaIndices(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_Coma_148(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "," Then
            indexT += 1
            index = indexT
            Return True
        ElseIf indexError <= indexT Then
            message = "Se esperaba ','"
            indexError = indexT
        End If
    End Function

    Private Function revise_Apostrofe_149(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "'" Then
            indexT += 1
            index = indexT
            Return True
        ElseIf indexError <= indexT Then
            message = "Se esperaba ''''"
            indexError = indexT
        End If
    End Function

    Private Function revise_tipoDato_150(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 6 <= str.Length AndAlso str.Substring(indexT, 6).ToLower = "logico" Then
            indexT += 6
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_tipoDato_151(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 6 <= str.Length AndAlso str.Substring(indexT, 6).ToLower = "entero" Then
```

```
            indexT += 6
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_tipoDato_152(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 8 <= str.Length AndAlso str.Substring(indexT, 8).ToLower = "flotante" Then
            indexT += 8
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_tipoDato_153(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 5 <= str.Length AndAlso str.Substring(indexT, 5).ToLower = "sarta" Then
            indexT += 5
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_tipoDato_154(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 8 <= str.Length AndAlso str.Substring(indexT, 8).ToLower = "caracter" Then
            indexT += 8
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_identificadorVariable_155(ByVal str As String, ByRef index As
    Integer) As Boolean
        Dim indexT As Integer = index
        If revise_identificador(str, indexT) Then
            'Verifica el estado "("
            If revise_identificadorVariable_156(Str, indexT) Then
                index = indexT
                Return True
            End If
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_identificadorVariable_156(ByVal str As String, ByRef index As
```

```
    Integer) As Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "(" Then
            indexT += 1
            'Verifica el estado dimensionArreglo
            If revise_identificadorVariable_157(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_identificadorVariable_157(ByVal str As String, ByRef index As
    Integer) As Boolean
        Dim indexT As Integer = index
        If revise_dimensionArreglo(str, indexT) Then
            'Verifica el estado ")"
            If revise_identificadorVariable_158(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_identificadorVariable_158(ByVal str As String, ByRef index As
    Integer) As Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = ")" Then
            indexT += 1
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_dimensionArreglo_159(ByVal str As String, ByRef index As Integer)
    As Boolean
        Dim indexT As Integer = index
        If revise_entero(str, indexT) Then
            'Verifica el estado Coma
            If revise_dimensionArreglo_160(Str, indexT) Then
                index = indexT
                Return True
            End If
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_dimensionArreglo_160(ByVal str As String, ByRef index As Integer)
    As Boolean
        Dim indexT As Integer = index
        If revise_Coma(str, indexT) Then
            'Verifica el estado dimensionArreglo
            If revise_dimensionArreglo_161(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_dimensionArreglo_161(ByVal str As String, ByRef index As Integer)
```

```
    As Boolean
        Dim indexT As Integer = index
        If revise_dimensionArreglo(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_entero_162(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_signo(str, indexT) Then
            'Verifica el estado entero_sin_signo
            If revise_entero_163(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_entero_163(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_entero_sin_signo(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_siEncabezado_164(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 3 <= str.Length AndAlso str.Substring(indexT, 3).ToLower = "si " Then
            indexT += 3
            'Verifica el estado expresion
            If revise_siEncabezado_165(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_siEncabezado_165(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        If revise_expresion(str, indexT) Then
            'Verifica el estado NuevaLinea
            If revise_siEncabezado_166(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_siEncabezado_166(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        If revise_NuevaLinea(str, indexT) Then
            'Verifica el estado codigoAbierto
            If revise_siEncabezado_167(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
```

```
    End Function

    Private Function revise_siEncabezado_167(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        If revise_codigoAbierto(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_sinosi_168(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 7 <= str.Length AndAlso str.Substring(indexT, 7).ToLower = "sinosi " Then
            indexT += 7
            'Verifica el estado expresion
            If revise_sinosi_169(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_sinosi_169(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_expresion(str, indexT) Then
            'Verifica el estado NuevaLinea
            If revise_sinosi_170(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_sinosi_170(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_NuevaLinea(str, indexT) Then
            'Verifica el estado codigoAbierto
            If revise_sinosi_171(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_sinosi_171(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_codigoAbierto(str, indexT) Then
            'Verifica el estado sinosi
            If revise_sinosi_172(Str, indexT) Then
                index = indexT
                Return True
            End If
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_sinosi_172(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_sinosi(str, indexT) Then
            If revise_finToken(Str, indexT) Then
```

```
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_sino_173(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 5 <= str.Length AndAlso str.Substring(indexT, 5).ToLower = "sino " Then
            indexT += 5
            'Verifica el estado expresion
            If revise_sino_174(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_sino_174(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_expresion(str, indexT) Then
            'Verifica el estado NuevaLinea
            If revise_sino_175(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_sino_175(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_NuevaLinea(str, indexT) Then
            'Verifica el estado codigoAbierto
            If revise_sino_176(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_sino_176(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_codigoAbierto(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_finsi_177(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 5 <= str.Length AndAlso str.Substring(indexT, 5).ToLower = "finsi" Then
            indexT += 5
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_repiteEncabezado_178(ByVal str As String, ByRef index As Integer) ↙
    As Boolean
        Dim indexT As Integer = index
```

```
        avance(Str, indexT)
        If indexT + 7 <= str.Length AndAlso str.Substring(indexT, 7).ToLower = "repite " Then
            indexT += 7
            'Verifica el estado NuevaLinea
            If revise_repiteEncabezado_179(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_repiteEncabezado_179(ByVal str As String, ByRef index As Integer) ↙
    As Boolean
        Dim indexT As Integer = index
        If revise_NuevaLinea(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_finRepite_180(ByVal str As String, ByRef index As Integer) As      ↙
    Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 6 <= str.Length AndAlso str.Substring(indexT, 6).ToLower = "hasta " Then
            indexT += 6
            'Verifica el estado expresion
            If revise_finRepite_181(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_finRepite_181(ByVal str As String, ByRef index As Integer) As      ↙
    Boolean
        Dim indexT As Integer = index
        If revise_expresion(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_mientrasEncabezado_182(ByVal str As String, ByRef index As         ↙
    Integer) As Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 9 <= str.Length AndAlso str.Substring(indexT, 9).ToLower = "mientras "     ↙
    Then
            indexT += 9
            'Verifica el estado expresion
            If revise_mientrasEncabezado_183(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_mientrasEncabezado_183(ByVal str As String, ByRef index As         ↙
    Integer) As Boolean
        Dim indexT As Integer = index
        If revise_expresion(str, indexT) Then
            'Verifica el estado NuevaLinea
```

```
            If revise_mientrasEncabezado_184(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_mientrasEncabezado_184(ByVal str As String, ByRef index As
    Integer) As Boolean
        Dim indexT As Integer = index
        If revise_NuevaLinea(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_finMientras_185(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 11 <= str.Length AndAlso str.Substring(indexT, 11).ToLower = "finmientras"
     Then
            indexT += 11
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_paraEncabezado_186(ByVal str As String, ByRef index As Integer) As
     Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 5 <= str.Length AndAlso str.Substring(indexT, 5).ToLower = "para " Then
            indexT += 5
            'Verifica el estado variable
            If revise_paraEncabezado_187(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_paraEncabezado_187(ByVal str As String, ByRef index As Integer) As
     Boolean
        Dim indexT As Integer = index
        If revise_variable(str, indexT) Then
            'Verifica el estado "="
            If revise_paraEncabezado_188(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_paraEncabezado_188(ByVal str As String, ByRef index As Integer) As
     Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "=" Then
            indexT += 1
            'Verifica el estado expresion
            If revise_paraEncabezado_189(Str, indexT) Then
                index = indexT
```

```
            Return True
        End If
    End If
End Function

Private Function revise_paraEncabezado_189(ByVal str As String, ByRef index As Integer) As
 Boolean
    Dim indexT As Integer = index
    If revise_expresion(str, indexT) Then
        'Verifica el estado "HASTA "
        If revise_paraEncabezado_190(Str, indexT) Then
            index = indexT
            Return True
        End If
    End If
End Function

Private Function revise_paraEncabezado_190(ByVal str As String, ByRef index As Integer) As
 Boolean
    Dim indexT As Integer = index
    avance(Str, indexT)
    If indexT + 6 <= str.Length AndAlso str.Substring(indexT, 6).ToLower = "hasta " Then
        indexT += 6
        'Verifica el estado expresion
        If revise_paraEncabezado_191(Str, indexT) Then
            index = indexT
            Return True
        End If
    End If
End Function

Private Function revise_paraEncabezado_191(ByVal str As String, ByRef index As Integer) As
 Boolean
    Dim indexT As Integer = index
    If revise_expresion(str, indexT) Then
        'Verifica el estado NuevaLinea
        If revise_paraEncabezado_192(Str, indexT) Then
            index = indexT
            Return True
        End If
    End If
End Function

Private Function revise_paraEncabezado_192(ByVal str As String, ByRef index As Integer) As
 Boolean
    Dim indexT As Integer = index
    If revise_NuevaLinea(str, indexT) Then
        If revise_finToken(Str, indexT) Then
            avance(Str, indexT)
            index = indexT
            Return True
        End If
    End If
End Function

Private Function revise_finPara_193(ByVal str As String, ByRef index As Integer) As
Boolean
    Dim indexT As Integer = index
    avance(Str, indexT)
    If indexT + 7 <= str.Length AndAlso str.Substring(indexT, 7).ToLower = "finpara" Then
        indexT += 7
        If revise_finToken(Str, indexT) Then
            avance(Str, indexT)
            index = indexT
            Return True
        End If
    End If
End Function
```

```
    Private Function revise_segunEncabezado_194(ByVal str As String, ByRef index As Integer)
    As Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 6 <= str.Length AndAlso str.Substring(indexT, 6).ToLower = "segun " Then
            indexT += 6
            'Verifica el estado expresion
            If revise_segunEncabezado_195(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_segunEncabezado_195(ByVal str As String, ByRef index As Integer)
    As Boolean
        Dim indexT As Integer = index
        If revise_expresion(str, indexT) Then
            'Verifica el estado NuevaLinea
            If revise_segunEncabezado_196(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_segunEncabezado_196(ByVal str As String, ByRef index As Integer)
    As Boolean
        Dim indexT As Integer = index
        If revise_NuevaLinea(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_segunOpcion_197(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 5 <= str.Length AndAlso str.Substring(indexT, 5).ToLower = "caso " Then
            indexT += 5
            'Verifica el estado constante
            If revise_segunOpcion_198(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_segunOpcion_198(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        If revise_constante(str, indexT) Then
            'Verifica el estado codigoAbierto
            If revise_segunOpcion_199(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_segunOpcion_199(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        If revise_codigoAbierto(str, indexT) Then
            If revise_finToken(Str, indexT) Then
```

```
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_segunOpcionOtro_200(ByVal str As String, ByRef index As Integer) ↙
    As Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 8 <= str.Length AndAlso str.Substring(indexT, 8).ToLower = "otrocaso" Then
            indexT += 8
            'Verifica el estado codigoAbierto
            If revise_segunOpcionOtro_201(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_segunOpcionOtro_201(ByVal str As String, ByRef index As Integer) ↙
    As Boolean
        Dim indexT As Integer = index
        If revise_codigoAbierto(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_finSegun_202(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 8 <= str.Length AndAlso str.Substring(indexT, 8).ToLower = "finsegun" Then
            indexT += 8
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_constante_203(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        If revise_sarta(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_constante_204(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        If revise_numero(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
```

```
    End Function

    Private Function revise_listaVariables_205(ByVal str As String, ByRef index As Integer) As
     Boolean
        Dim indexT As Integer = index
        If revise_variable(str, indexT) Then
            'Verifica el estado Coma
            If revise_listaVariables_206(Str, indexT) Then
                index = indexT
                Return True
            End If
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_listaVariables_206(ByVal str As String, ByRef index As Integer) As
     Boolean
        Dim indexT As Integer = index
        If revise_Coma(str, indexT) Then
            'Verifica el estado listaVariables
            If revise_listaVariables_207(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_listaVariables_207(ByVal str As String, ByRef index As Integer) As
     Boolean
        Dim indexT As Integer = index
        If revise_listaVariables(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_lista_expresiones_208(ByVal str As String, ByRef index As Integer)
     As Boolean
        Dim indexT As Integer = index
        If revise_expresion(str, indexT) Then
            'Verifica el estado Coma
            If revise_lista_expresiones_209(Str, indexT) Then
                index = indexT
                Return True
            End If
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_lista_expresiones_209(ByVal str As String, ByRef index As Integer)
     As Boolean
        Dim indexT As Integer = index
        If revise_Coma(str, indexT) Then
            'Verifica el estado lista_expresiones
            If revise_lista_expresiones_210(Str, indexT) Then
                index = indexT
                Return True
            End If
```

```
        End If
    End Function

    Private Function revise_lista_expresiones_210(ByVal str As String, ByRef index As Integer)↙
     As Boolean
        Dim indexT As Integer = index
        If revise_lista_expresiones(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_retornoFuncion_211(ByVal str As String, ByRef index As Integer) As↙
     Boolean
        Dim indexT As Integer = index
        If revise_Retorne(str, indexT) Then
            'Verifica el estado "("
            If revise_retornoFuncion_212(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_retornoFuncion_212(ByVal str As String, ByRef index As Integer) As↙
     Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "(" Then
            indexT += 1
            'Verifica el estado expresion
            If revise_retornoFuncion_213(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_retornoFuncion_213(ByVal str As String, ByRef index As Integer) As↙
     Boolean
        Dim indexT As Integer = index
        If revise_expresion(str, indexT) Then
            'Verifica el estado ")"
            If revise_retornoFuncion_214(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_retornoFuncion_214(ByVal str As String, ByRef index As Integer) As↙
     Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = ")" Then
            indexT += 1
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_retornoProcedimiento_215(ByVal str As String, ByRef index As      ↙
    Integer) As Boolean
```

```
        Dim indexT As Integer = index
        If revise_Retorne(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_termino_216(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        If revise_numero_sin_signo(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_termino_217(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        If revise_sarta(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_termino_218(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        If revise_variable(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_termino_219(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        If revise_llamarFuncion(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_termino_220(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "(" Then
            indexT += 1
            'Verifica el estado expresion
            If revise_termino_221(Str, indexT) Then
                index = indexT
                Return True
            End If
```

```
        End If
    End Function

    Private Function revise_termino_221(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        If revise_expresion(str, indexT) Then
            'Verifica el estado ")"
            If revise_termino_222(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_termino_222(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = ")" Then
            indexT += 1
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_operacion_223(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        If revise_orOp(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_operacion_224(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        If revise_andOp(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_operacion_225(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        If revise_RelOp(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_operacion_226(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        If revise_AddOp(str, indexT) Then
```

```
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_operacion_227(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        If revise_MulOp(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_operacion_228(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "&" Then
            indexT += 1
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_operacion_229(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        If revise_Not(str, indexT) Then
            'Verifica el estado termino
            If revise_operacion_230(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_operacion_230(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        If revise_termino(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_orOp_231(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Or(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function
```

```
    Private Function revise_andOp_232(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_And(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_RelOp_233(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "<" Then
            indexT += 1
            index = indexT
            Return True
        ElseIf indexError <= indexT Then
            message = "Se esperaba '<'"
            indexError = indexT
        End If
    End Function

    Private Function revise_RelOp_234(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = ">" Then
            indexT += 1
            index = indexT
            Return True
        ElseIf indexError <= indexT Then
            message = "Se esperaba '>'"
            indexError = indexT
        End If
    End Function

    Private Function revise_RelOp_235(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If indexT + 2 <= str.Length AndAlso str.Substring(indexT, 2).ToLower = "<=" Then
            indexT += 2
            index = indexT
            Return True
        ElseIf indexError <= indexT Then
            message = "Se esperaba '<='"
            indexError = indexT
        End If
    End Function

    Private Function revise_RelOp_236(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If indexT + 2 <= str.Length AndAlso str.Substring(indexT, 2).ToLower = ">=" Then
            indexT += 2
            index = indexT
            Return True
        ElseIf indexError <= indexT Then
            message = "Se esperaba '>='"
            indexError = indexT
        End If
    End Function

    Private Function revise_RelOp_237(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "=" Then
            indexT += 1
            index = indexT
            Return True
        ElseIf indexError <= indexT Then
            message = "Se esperaba '='"
            indexError = indexT
        End If
```

```
    End Function

    Private Function revise_RelOp_238(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If indexT + 2 <= str.Length AndAlso str.Substring(indexT, 2).ToLower = "<>" Then
            indexT += 2
            index = indexT
            Return True
        ElseIf indexError <= indexT Then
            message = "Se esperaba '<>'"
            indexError = indexT
        End If
    End Function

    Private Function revise_AddOp_239(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "+" Then
            indexT += 1
            index = indexT
            Return True
        ElseIf indexError <= indexT Then
            message = "Se esperaba '+'"
            indexError = indexT
        End If
    End Function

    Private Function revise_AddOp_240(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "-" Then
            indexT += 1
            index = indexT
            Return True
        ElseIf indexError <= indexT Then
            message = "Se esperaba '-'"
            indexError = indexT
        End If
    End Function

    Private Function revise_MulOp_241(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "*" Then
            indexT += 1
            index = indexT
            Return True
        ElseIf indexError <= indexT Then
            message = "Se esperaba '*'"
            indexError = indexT
        End If
    End Function

    Private Function revise_MulOp_242(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "/" Then
            indexT += 1
            index = indexT
            Return True
        ElseIf indexError <= indexT Then
            message = "Se esperaba '/'"
            indexError = indexT
        End If
    End Function

    Private Function revise_MulOp_243(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "\" Then
            indexT += 1
            index = indexT
            Return True
        ElseIf indexError <= indexT Then
```

```
            message = "Se esperaba '\'"
            indexError = indexT
        End If
    End Function

    Private Function revise_MulOp_244(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If indexT + 3 <= str.Length AndAlso str.Substring(indexT, 3).ToLower = "mod" Then
            indexT += 3
            index = indexT
            Return True
        ElseIf indexError <= indexT Then
            message = "Se esperaba 'Mod'"
            indexError = indexT
        End If
    End Function

    Private Function revise_Not_245(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If indexT + 2 <= str.Length AndAlso str.Substring(indexT, 2).ToLower = "no" Then
            indexT += 2
            index = indexT
            Return True
        ElseIf indexError <= indexT Then
            message = "Se esperaba 'no'"
            indexError = indexT
        End If
    End Function

    Private Function revise_numero_sin_signo_246(ByVal str As String, ByRef index As Integer) ↙
    As Boolean
        Dim indexT As Integer = index
        If revise_entero_sin_signo(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_numero_sin_signo_247(ByVal str As String, ByRef index As Integer) ↙
    As Boolean
        Dim indexT As Integer = index
        If revise_real_sin_signo(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_sarta_248(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Comilla(str, indexT) Then
            'Verifica el estado AnythingExceptDoubleQuote
            If revise_sarta_249(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_sarta_249(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_AnythingExceptDoubleQuote(str, indexT) Then
            'Verifica el estado Comilla
            If revise_sarta_250(Str, indexT) Then
```

```
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_sarta_250(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Comilla(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_nombreFuncion_251(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        If revise_Abs(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_nombreFuncion_252(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        If revise_Coseno(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_nombreFuncion_253(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        If revise_Largo(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_nombreFuncion_254(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        If revise_Log(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_nombreFuncion_255(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        If revise_Aleatorio(str, indexT) Then
```

```
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_nombreFuncion_256(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        If revise_Seno(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_nombreFuncion_257(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        If revise_Mayuscula(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_nombreFuncion_258(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        If revise_Minuscula(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_nombreFuncion_259(ByVal str As String, ByRef index As Integer) As ↙
    Boolean
        Dim indexT As Integer = index
        If revise_identificador(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_Abs_260(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If indexT + 3 <= str.Length AndAlso str.Substring(indexT, 3).ToLower = "abs" Then
            indexT += 3
            index = indexT
            Return True
        ElseIf indexError <= indexT Then
            message = "Se esperaba 'abs'"
            indexError = indexT
        End If
    End Function

    Private Function revise_Letra_261(ByVal str As String, ByRef index As Integer) As Boolean
```

```
        Dim indexT As Integer = index
        If revise_A(str, indexT) Then
            index = indexT
            Return True
        ElseIf indexError <= indexT Then
            message = "Se esperaba 'A'"
            indexError = indexT
        End If
    End Function

    Private Function revise_Letra_262(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_B(str, indexT) Then
            index = indexT
            Return True
        ElseIf indexError <= indexT Then
            message = "Se esperaba 'B'"
            indexError = indexT
        End If
    End Function

    Private Function revise_Letra_263(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_C(str, indexT) Then
            index = indexT
            Return True
        ElseIf indexError <= indexT Then
            message = "Se esperaba 'C'"
            indexError = indexT
        End If
    End Function

    Private Function revise_Letra_264(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_D(str, indexT) Then
            index = indexT
            Return True
        ElseIf indexError <= indexT Then
            message = "Se esperaba 'D'"
            indexError = indexT
        End If
    End Function

    Private Function revise_Letra_265(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_E(str, indexT) Then
            index = indexT
            Return True
        ElseIf indexError <= indexT Then
            message = "Se esperaba 'E'"
            indexError = indexT
        End If
    End Function

    Private Function revise_Letra_266(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_F(str, indexT) Then
            index = indexT
            Return True
        ElseIf indexError <= indexT Then
            message = "Se esperaba 'F'"
            indexError = indexT
        End If
    End Function

    Private Function revise_Letra_267(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_G(str, indexT) Then
            index = indexT
```

```
            Return True
        ElseIf indexError <= indexT Then
            message = "Se esperaba 'G'"
            indexError = indexT
        End If
    End Function

    Private Function revise_Letra_268(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_H(str, indexT) Then
            index = indexT
            Return True
        ElseIf indexError <= indexT Then
            message = "Se esperaba 'H'"
            indexError = indexT
        End If
    End Function

    Private Function revise_Letra_269(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_I(str, indexT) Then
            index = indexT
            Return True
        ElseIf indexError <= indexT Then
            message = "Se esperaba 'I'"
            indexError = indexT
        End If
    End Function

    Private Function revise_Letra_270(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_J(str, indexT) Then
            index = indexT
            Return True
        ElseIf indexError <= indexT Then
            message = "Se esperaba 'J'"
            indexError = indexT
        End If
    End Function

    Private Function revise_Letra_271(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_K(str, indexT) Then
            index = indexT
            Return True
        ElseIf indexError <= indexT Then
            message = "Se esperaba 'K'"
            indexError = indexT
        End If
    End Function

    Private Function revise_Letra_272(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_L(str, indexT) Then
            index = indexT
            Return True
        ElseIf indexError <= indexT Then
            message = "Se esperaba 'L'"
            indexError = indexT
        End If
    End Function

    Private Function revise_Letra_273(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_M(str, indexT) Then
            index = indexT
            Return True
        ElseIf indexError <= indexT Then
            message = "Se esperaba 'M'"
```

```
            indexError = indexT
        End If
    End Function

    Private Function revise_Letra_274(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_N(str, indexT) Then
            index = indexT
            Return True
        ElseIf indexError <= indexT Then
            message = "Se esperaba 'N'"
            indexError = indexT
        End If
    End Function

    Private Function revise_Letra_275(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_O(str, indexT) Then
            index = indexT
            Return True
        ElseIf indexError <= indexT Then
            message = "Se esperaba 'O'"
            indexError = indexT
        End If
    End Function

    Private Function revise_Letra_276(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_P(str, indexT) Then
            index = indexT
            Return True
        ElseIf indexError <= indexT Then
            message = "Se esperaba 'P'"
            indexError = indexT
        End If
    End Function

    Private Function revise_Letra_277(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Q(str, indexT) Then
            index = indexT
            Return True
        ElseIf indexError <= indexT Then
            message = "Se esperaba 'Q'"
            indexError = indexT
        End If
    End Function

    Private Function revise_Letra_278(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_R(str, indexT) Then
            index = indexT
            Return True
        ElseIf indexError <= indexT Then
            message = "Se esperaba 'R'"
            indexError = indexT
        End If
    End Function

    Private Function revise_Letra_279(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_S(str, indexT) Then
            index = indexT
            Return True
        ElseIf indexError <= indexT Then
            message = "Se esperaba 'S'"
            indexError = indexT
        End If
    End Function
```

```
    Private Function revise_Letra_280(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_T(str, indexT) Then
            index = indexT
            Return True
        ElseIf indexError <= indexT Then
            message = "Se esperaba 'T'"
            indexError = indexT
        End If
    End Function

    Private Function revise_Letra_281(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_U(str, indexT) Then
            index = indexT
            Return True
        ElseIf indexError <= indexT Then
            message = "Se esperaba 'U'"
            indexError = indexT
        End If
    End Function

    Private Function revise_Letra_282(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_V(str, indexT) Then
            index = indexT
            Return True
        ElseIf indexError <= indexT Then
            message = "Se esperaba 'V'"
            indexError = indexT
        End If
    End Function

    Private Function revise_Letra_283(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_W(str, indexT) Then
            index = indexT
            Return True
        ElseIf indexError <= indexT Then
            message = "Se esperaba 'W'"
            indexError = indexT
        End If
    End Function

    Private Function revise_Letra_284(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_X(str, indexT) Then
            index = indexT
            Return True
        ElseIf indexError <= indexT Then
            message = "Se esperaba 'X'"
            indexError = indexT
        End If
    End Function

    Private Function revise_Letra_285(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Y(str, indexT) Then
            index = indexT
            Return True
        ElseIf indexError <= indexT Then
            message = "Se esperaba 'Y'"
            indexError = indexT
        End If
    End Function

    Private Function revise_Letra_286(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
```

```
        If revise_Z(str, indexT) Then
            index = indexT
            Return True
        ElseIf indexError <= indexT Then
            message = "Se esperaba 'Z'"
            indexError = indexT
        End If
    End Function

    Private Function revise_LetrasNumerosUnderscores_287(ByVal str As String, ByRef index As ↙
    Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Letra(str, indexT) Then
            index = indexT
            Return True
        ElseIf indexError <= indexT Then
            message = "Se esperaba 'Letra'"
            indexError = indexT
        End If
    End Function

    Private Function revise_LetrasNumerosUnderscores_288(ByVal str As String, ByRef index As ↙
    Integer) As Boolean
        Dim indexT As Integer = index
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "_" Then
            indexT += 1
            index = indexT
            Return True
        ElseIf indexError <= indexT Then
            message = "Se esperaba '_'"
            indexError = indexT
        End If
    End Function

    Private Function revise_LetrasNumerosUnderscores_289(ByVal str As String, ByRef index As ↙
    Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Num(str, indexT) Then
            index = indexT
            Return True
        ElseIf indexError <= indexT Then
            message = "Se esperaba 'Num'"
            indexError = indexT
        End If
    End Function

    Private Function revise_Or_290(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "o" Then
            indexT += 1
            index = indexT
            Return True
        ElseIf indexError <= indexT Then
            message = "Se esperaba 'o'"
            indexError = indexT
        End If
    End Function

    Private Function revise_And_291(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "y" Then
            indexT += 1
            index = indexT
            Return True
        ElseIf indexError <= indexT Then
            message = "Se esperaba 'y'"
            indexError = indexT
        End If
    End Function
```

```
    Private Function revise_definicionProcedimiento_292(ByVal str As String, ByRef index As ↙
    Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Procedimiento(str, indexT) Then
            'Verifica el estado identificador
            If revise_definicionProcedimiento_293(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_definicionProcedimiento_293(ByVal str As String, ByRef index As ↙
    Integer) As Boolean
        Dim indexT As Integer = index
        If revise_identificador(str, indexT) Then
            'Verifica el estado "("
            If revise_definicionProcedimiento_294(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_definicionProcedimiento_294(ByVal str As String, ByRef index As ↙
    Integer) As Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "(" Then
            indexT += 1
            'Verifica el estado listaParametros
            If revise_definicionProcedimiento_295(Str, indexT) Then
                index = indexT
                Return True
            End If
            'Verifica el estado ")"
            If revise_definicionProcedimiento_296(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_definicionProcedimiento_295(ByVal str As String, ByRef index As ↙
    Integer) As Boolean
        Dim indexT As Integer = index
        If revise_listaParametros(str, indexT) Then
            'Verifica el estado ")"
            If revise_definicionProcedimiento_296(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_definicionProcedimiento_296(ByVal str As String, ByRef index As ↙
    Integer) As Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = ")" Then
            indexT += 1
            'Verifica el estado codigoAbierto
            If revise_definicionProcedimiento_297(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function
```

```
    Private Function revise_definicionProcedimiento_297(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_codigoAbierto(str, indexT) Then
            'Verifica el estado FinProcedimiento
            If revise_definicionProcedimiento_298(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_definicionProcedimiento_298(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_FinProcedimiento(str, indexT) Then
            'Verifica el estado NuevaLinea
            If revise_definicionProcedimiento_299(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_definicionProcedimiento_299(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_NuevaLinea(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_definicionFuncion_300(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_Funcion(str, indexT) Then
            'Verifica el estado identificador
            If revise_definicionFuncion_301(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_definicionFuncion_301(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_identificador(str, indexT) Then
            'Verifica el estado "("
            If revise_definicionFuncion_302(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_definicionFuncion_302(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "(" Then
            indexT += 1
            'Verifica el estado listaParametros
            If revise_definicionFuncion_303(Str, indexT) Then
```

```
                index = indexT
                Return True
            End If
            'Verifica el estado ")"
            If revise_definicionFuncion_304(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_definicionFuncion_303(ByVal str As String, ByRef index As Integer)↙
     As Boolean
        Dim indexT As Integer = index
        If revise_listaParametros(str, indexT) Then
            'Verifica el estado ")"
            If revise_definicionFuncion_304(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_definicionFuncion_304(ByVal str As String, ByRef index As Integer)↙
     As Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = ")" Then
            indexT += 1
            'Verifica el estado tipoDato
            If revise_definicionFuncion_305(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_definicionFuncion_305(ByVal str As String, ByRef index As Integer)↙
     As Boolean
        Dim indexT As Integer = index
        If revise_tipoDato(str, indexT) Then
            'Verifica el estado codigoAbierto
            If revise_definicionFuncion_306(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_definicionFuncion_306(ByVal str As String, ByRef index As Integer)↙
     As Boolean
        Dim indexT As Integer = index
        If revise_codigoAbierto(str, indexT) Then
            'Verifica el estado FinFuncion
            If revise_definicionFuncion_307(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_definicionFuncion_307(ByVal str As String, ByRef index As Integer)↙
     As Boolean
        Dim indexT As Integer = index
        If revise_FinFuncion(str, indexT) Then
            'Verifica el estado NuevaLinea
            If revise_definicionFuncion_308(Str, indexT) Then
                index = indexT
                Return True
```

```
            End If
        End If
    End Function

    Private Function revise_definicionFuncion_308(ByVal str As String, ByRef index As Integer)
     As Boolean
        Dim indexT As Integer = index
        If revise_NuevaLinea(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_listaParametros_309(ByVal str As String, ByRef index As Integer)
    As Boolean
        Dim indexT As Integer = index
        If revise_definicionFormalParametro(str, indexT) Then
            'Verifica el estado ","
            If revise_listaParametros_310(Str, indexT) Then
                index = indexT
                Return True
            End If
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_listaParametros_310(ByVal str As String, ByRef index As Integer)
    As Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "," Then
            indexT += 1
            'Verifica el estado listaParametros
            If revise_listaParametros_311(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_listaParametros_311(ByVal str As String, ByRef index As Integer)
    As Boolean
        Dim indexT As Integer = index
        If revise_listaParametros(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_definicionFormalParametro_312(ByVal str As String, ByRef index As
    Integer) As Boolean
        Dim indexT As Integer = index
        If revise_tipoDato(str, indexT) Then
            'Verifica el estado identificador
            If revise_definicionFormalParametro_313(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
```

```
    End Function

    Private Function revise_definicionFormalParametro_313(ByVal str As String, ByRef index As ↙
    Integer) As Boolean
        Dim indexT As Integer = index
        If revise_identificador(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_numero_314(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_entero(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_numero_315(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_real(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_real_316(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_signo(str, indexT) Then
            'Verifica el estado real_sin_signo
            If revise_real_317(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_real_317(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_real_sin_signo(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_entero_sin_signo_318(ByVal str As String, ByRef index As Integer) ↙
    As Boolean
        Dim indexT As Integer = index
        If revise_Num(str, indexT) Then
            'Verifica el estado Num
            If revise_entero_sin_signo_318(Str, indexT) Then
                index = indexT
                Return True
            End If
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
```

```
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_real_sin_signo_319(ByVal str As String, ByRef index As Integer) As↙
     Boolean
        Dim indexT As Integer = index
        If revise_entero_sin_signo(str, indexT) Then
            'Verifica el estado "."
            If revise_real_sin_signo_320(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_real_sin_signo_320(ByVal str As String, ByRef index As Integer) As↙
     Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "." Then
            indexT += 1
            'Verifica el estado entero_sin_signo
            If revise_real_sin_signo_321(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_real_sin_signo_321(ByVal str As String, ByRef index As Integer) As↙
     Boolean
        Dim indexT As Integer = index
        If revise_entero_sin_signo(str, indexT) Then
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_signo_322(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "+" Then
            indexT += 1
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_signo_323(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        avance(Str, indexT)
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = "-" Then
            indexT += 1
            If revise_finToken(Str, indexT) Then
                avance(Str, indexT)
                index = indexT
                Return True
            End If
        End If
    End Function
```

```
    Private Function revise_Comilla_324(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = """" Then
            indexT += 1
            'Verifica el estado "
            If revise_Comilla_325(Str, indexT) Then
                index = indexT
                Return True
            End If
        End If
    End Function

    Private Function revise_Comilla_325(ByVal str As String, ByRef index As Integer) As
    Boolean
        Dim indexT As Integer = index
        If indexT + 1 <= str.Length AndAlso str.Substring(indexT, 1).ToLower = """" Then
            indexT += 1
            index = indexT
            Return True
        ElseIf indexError <= indexT Then
            message = "Se esperaba ''"
            indexError = indexT
        End If
    End Function

    Private Function revise_Num_326(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_1(str, indexT) Then
            index = indexT
            Return True
        ElseIf indexError <= indexT Then
            message = "Se esperaba '1'"
            indexError = indexT
        End If
    End Function

    Private Function revise_Num_327(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_2(str, indexT) Then
            index = indexT
            Return True
        ElseIf indexError <= indexT Then
            message = "Se esperaba '2'"
            indexError = indexT
        End If
    End Function

    Private Function revise_Num_328(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_3(str, indexT) Then
            index = indexT
            Return True
        ElseIf indexError <= indexT Then
            message = "Se esperaba '3'"
            indexError = indexT
        End If
    End Function

    Private Function revise_Num_329(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_4(str, indexT) Then
            index = indexT
            Return True
        ElseIf indexError <= indexT Then
            message = "Se esperaba '4'"
            indexError = indexT
        End If
```

```
    End Function

    Private Function revise_Num_330(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_5(str, indexT) Then
            index = indexT
            Return True
        ElseIf indexError <= indexT Then
            message = "Se esperaba '5'"
            indexError = indexT
        End If
    End Function

    Private Function revise_Num_331(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_6(str, indexT) Then
            index = indexT
            Return True
        ElseIf indexError <= indexT Then
            message = "Se esperaba '6'"
            indexError = indexT
        End If
    End Function

    Private Function revise_Num_332(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_7(str, indexT) Then
            index = indexT
            Return True
        ElseIf indexError <= indexT Then
            message = "Se esperaba '7'"
            indexError = indexT
        End If
    End Function

    Private Function revise_Num_333(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_8(str, indexT) Then
            index = indexT
            Return True
        ElseIf indexError <= indexT Then
            message = "Se esperaba '8'"
            indexError = indexT
        End If
    End Function

    Private Function revise_Num_334(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_9(str, indexT) Then
            index = indexT
            Return True
        ElseIf indexError <= indexT Then
            message = "Se esperaba '9'"
            indexError = indexT
        End If
    End Function

    Private Function revise_Num_335(ByVal str As String, ByRef index As Integer) As Boolean
        Dim indexT As Integer = index
        If revise_0(str, indexT) Then
            index = indexT
            Return True
        ElseIf indexError <= indexT Then
            message = "Se esperaba '0'"
            indexError = indexT
        End If
    End Function

    Private Function revise_Fin(ByVal str As String, ByRef index As Integer) As Boolean
```

```
        If str.Length <= index Then Return True
    End Function

    Private Sub avance(ByVal str As String, ByRef index As Integer)
        While str.Length > index AndAlso saltarCaracter(str(index))
            index += 1
        End While
    End Sub

    Private Function saltarCaracter(ByVal car As Char) As Boolean
        If car = " "c OrElse Asc(car) = 9 Then Return True
    End Function

    Private Function revise_inicioToken(ByVal str As String, ByVal index As Integer) As
    Boolean
        If index >= Str.Length Then Return False
        If saltarCaracter(Str(index)) Then Return False
        If index = 0 Then Return True
        If Me.saltarCaracter(Str(index - 1)) Then Return True
        If Str(index) = Chr(13) Then Return True
        If Str(index) = Chr(10) Then Return True
        If str(index) = "," Then Return True
        If str(index) = "+" Then Return True
        If str(index) = "-" Then Return True
        If str(index) = "=" Then Return True
        If str(index) = "*" Then Return True
        If str(index) = "/" Then Return True
        If str(index) = "(" Then Return True
        If str(index) = ")" Then Return True
        If Str(index - 1) = Chr(13) Then Return True
        If Str(index - 1) = Chr(10) Then Return True
        If str(index - 1) = "," Then Return True
        If str(index - 1) = "+" Then Return True
        If str(index - 1) = "-" Then Return True
        If str(index - 1) = "=" Then Return True
        If str(index - 1) = "*" Then Return True
        If str(index - 1) = "/" Then Return True
        If str(index - 1) = "(" Then Return True
        If str(index - 1) = ")" Then Return True
    End Function

    Private Function revise_FinToken(ByVal str As String, ByVal index As Integer) As Boolean
        If Me.revise_inicioToken(Str, index) Then Return True
        If index >= Str.Length Then Return True
        If saltarCaracter(Str(index)) Then Return True
    End Function
End Class
```