COMPILADOR DIDÁCTICO

Ing. **SALVADOR NAVARRIA**
Profesor de Programación
Ing. **CRISTINA A. PARRAGA**
Profesora de Programación I
Facultad de Ingeniería - UNIVERSIDAD DE MENDOZA.

RESUMEN

El presente trabajo surgido a través de nuestra experiencia como una necesidad académica, está orientado a dos grupos con distinto nivel de conocimiento en informática y que hemos clasificado en:

1) Alumnos principiantes.
2) Alumnos con conocimientos básicos de programación.

Para el primer grupo se ha preparado un analizador léxico y una simulación gráfica para visualizar los conceptos de variable como una localidad de memoria y el proceso de asignación que se efectúa al ejecutar un programa. En esta parte del análisis también se muestra el proceso interno de un lazo o bucle para que se destaque el beneficio del uso de los mismos.

Para el segundo grupo, este programa permite obtener una idea gráfica de la forma en que un computador realiza el primer proceso de la compilación que caracteriza los lenguajes de alto nivel. En este sector del sistema se ha previsto también un analizador sintáctico, la faz de interpretación y la asignación de memoria. Se sigue así las diversas fases del mecanismo y de un compilador.

El uso de este sistema redunda en una mejor y más rápida comprensión de estos conceptos normalmente difíciles de captar en forma teórica. Desde el punto de vista docente permite un seguimiento paulatino del avance en la comprensión de estos temas por parte del alumnado, detectando fácilmente en qué áreas encuentra mayores dificultades.

INTRODUCCIÓN

Cuando se pretende trabajar en Programación con lenguajes de alto nivel se hace imprescindible tener claro el concepto de intérpretes y compiladores, concepto que para los neófitos suele traer serios problemas de comprensión.

A través de esta dificultad que se nos presenta en nuestra área de trabajo, la educación en informática, surge la idea de representar en forma visible el trabajo interno de un compilador.

En realidad no se pretende construir un compilador sino simplemente esquematizar en forma gráfica y lo más didáctico posible cual es el principio de funcionamiento y como se desarrollan sus diferentes etapas básicas.

Debido a los distintos niveles de conocimiento del grupo hacia el cual está dirigido nuestro trabajo hemos implementado el trabajo en dos fases.

Si bien la división no es estructural dado que el trabajo es básicamente el mismo de acuerdo al enfoque que se le dé puede ser aplicado para enseñar a personas con distintos nveles de conocimiento.

1 - CONSIDERACIONES TEÓRICAS

Como se recordará un compilador es una caja negra que acepta como entrada un programa escrito en algún lenguaje de alto nivel —programa fuente— y produce a la salida un programa equivalente escrito en lenguaje de máquina.

Un compilador real realiza esta tarea generalmente en cinco fases sucesivas, según lo mostrado en la figura I.

Ahora bien, nosotros hacemos hincapié únicamente en las cuatro primeras etapas del compilador real por cuanto la generación del código objeto requiere conocimientos avanzados de computación tales como lenguaje assembler, modos de direccionamientos, etc. que no han sido alcanzados por los estudiantes a quienes está dirigido este trabajo.

Es por ello que queda reducido a un compilador imaginario como el que se presenta en la figura II y que hemos dado en llamar "Compilador Didáctico".

Aquí se ha omitido la última fase.

2 - ESQUEMA GENERAL DEL SISTEMA PREVISTO

La idea general es que el alumno se ubique frente a un monitor e ingrese su programa fuente codificando en algún lenguaje de alto nivel como por ejemplo Pascal, una vez cumplida esta etapa el sistema le permitirá identificar todo el léxico que compone su programa.

Nuestro compilador didáctico en una primera etapa le devuelve por pantalla un listado de lo que la máquina interpreta como palabra reservada,

identificador, función, valores numéricos, etc. y que haya utilizado en su programa. Se muestra las confusiones posibles que puedan producirse en la computadora cuando se introduce algún término en forma incorrecta.

Por ejemplo ante una sentencia con error de sintaxis este compilador le dirá que no la reconoce tal como él pretendía. De esta manera, y según nuestra experiencia docente, detectamos que el alumno se concientiza de la importancia en programación de la correcta sintaxis de un lenguaje particular.

Esta misma fase es útil para explicar el funcionamiento del explorador (SCANER) ya que secuencialmente va detectando los delimitadores naturales (espacios en blanco) y ubicando los elementos básicos en las tablas de símbolos correspondientes a Palabra Reservada, Identificadores, y Literales (comentarios). Figura III.

Una vez que el programa fuente ha sido descompuesto en elementos básicos el compilador debe reconocer y chequear sintácticamente cada fase. La etapa de chequeo trata de mostrar los errores más comunes y provee cierta información sobre la posible causa. Aquí se chequea la existencia en cada frase del signo de asignación que identifica un enunciado aritmético. De la existencia o no del signo dependerá el tratamiento a seguir. Si no existe el signo se interpreta que debe existir en esa línea una palabra reservada y se chequea su sintaxis y estructura particular.

En cambio si existe el signo se presume que es un enunciado matemático y sobre el mismo se analizan la existencia de paréntesis y la jerarquía algebraica de las operaciones, las que muestran en pantalla de acuerdo a la forma y orden sucesivo en que se llevarán a cabo en tiempo de ejecución.

De esta forma el estudiante puede visualizar la jerarquía con que la máquina realiza las operaciones. Partiendo desde el núcleo de la misma se hace la primera representación de la operación, sucesivamente se va ampliando hacia el exterior la ejecución de toda la sentencia.

Para los alumnos avanzados esta representación facilita la visualización de la estructura de un árbol analítico de una expresión algebraica determinada y aunque no se muestre con nodos y ramas se visualiza la jerarquía de las operaciones.

El flujo del sistema para este chequeo se observa en la figura IV.

Es importante que el alumno tome conciencia que su programa y sus datos ocupan una posición en la memoria ubicados en una dirección determinada. Por ello se pensó en destacar en forma gráfica este proceso, que en nuestro trabajo se realiza en forma simulada.

Esquemáticamente verá las direcciones sucesivas de memoria que van ocupando cada una de las sentencias de su programa fuente, luego se hará gráfica la idea de que cada variable simple tiene una locación de memoria que le es propia. Finalmente verá que las variables con subíndice no son más que una sucesión de variables simples en localidades de memoria sucesivas lo que le hace destacable al alumno la importancia del dimensionamiento (Figura V).

Por último a los efectos de lograr que el alumno interprete que cada variable independientemente del valor que tome tiene una locación de memoria determinada y además que expresiones del tipo I = I + 1 no deben entenderse como igualdades matemáticas sino como asignación es que se pensó en la representación de un ciclado y su alcance.

Considerando además que los lazos son ideales para visualizar una sentencia de asignación a través de sus contadores es que se representa la variación y el alcance de la misma en todo el recorrido de un bucle mostrando en forma destacada cada vez que se produce una nueva asignación. El diagrama en bloque de este proceso se representa en la figura VI.

4 - CARACTERÍSTICAS DEL SISTEMA

4.1 - Análisis lexicográfico

Esta es la primera faz del proyecto. Aquí se capturan todos los datos que servirán a lo largo del proceso es decir, el programa fuente que codifica el alumno en un lenguaje de alto nivel. En las instrucciones de entrada se aclara que debe hacerlo línea por línea sin olvidar la señal de final de línea (E O L) implementado con un punto y coma respetando la sintaxis del lenguaje Pascal. Con un eco por pantalla se observa el listado completo del programa fuente.

Esta sección del programa irá explorando línea por línea y dentro de cada línea carácter por carácter siguiendo el flujo mostrado en la figura Vll. Cada espacio en blanco es un delimitador natural y las palabras o símbolos del programa fuente entre estos delimitadores son analizados a posteriori.

Se distingue primero entre valores numéricos y palabras y sobre las palabras se determina si se encuentran en el set de palabras reservadas. De no hallarse, el explorador las considera como identificadores.

Análogo procedimiento sigue con los caracteres especiales considerándolos delimitadores.

Todo aquel símbolo que no esté dentro del set caracteres permitidos para un lenguaje particular será indicado como un error.

De todo este proceso surgen las tablas de palabras reservadas las cuales almacenan además información auxiliar tal como el número de línea en la cual ha sido utilizada.

Asimismo genera la tabla de identificadores las que también poseen información complementaria sobre el número de línea en la cual fue usada y el tipo de variable que le fue dado en las sentencias de declaración. La tabla de literales contiene los comentarios que se distinguen por el uso de los caracteres (*. Se adiciona el número de línea donde fueron escritos a los efectos de ignorarlos en el análisis sintáctico.

4.2 - Reconocimiento sintáctico

En esta parte del sistema cada línea del programa pasa por el análisis de su estructura. Para esto se inicia el proceso detectando si la línea de programa posee un signo de asignación (:= para el lenguaje de Pascal)

De acuerdo a esto el reconocimiento se encamina por dos ramas diferentes.

4.2.1 - Caso de no existencia del signo de asignación

Las sentencias que no presentan dicho signo se presumen que son instrucciones ejecutables no aritméticos por lo tanto debe existir en dicha línea alguna palabra reservada, la que tiene que ser detectada.

Si al explorar la línea no encuentra ninguna palabra reservada se indicará al estudiante que está cometiendo algún error.

En cambio si existe esta palabra, se identifica a que orden se refiere y luego se chequea la estructura propia de ese tipo especial de sentencia.

Entre éstas distinguiremos instrucciones de encabezamiento, de entrada/salida de datos, de bifurcación, de lazos y de fin de programa. En todos los casos si la estructura no está correctamente dispuesta se emite el mensaje de error correspondiente.

4.2.2 - Caso de existencia del signo de asignación

El signo de asignación caracteriza a todo enunciado de tipo matemático pero también se presenta en alguna instrucción de ciclado como por ejemplo la estructura FOR - - - TO - - - - - DO del lenguaje Pascal. Es por esto que una vez detectado dicho signo se generan dos ramas por donde según el análisis tomando los elementos básicos a la izquierda y a la derecha del mismo.

Tomando la parte izquierda se testea la existencia de caracteres especiales. El resultado negativo de esta comparación nos lleva a buscar ese elemento en la tabla de identificadores, no hallarlo implica un error por identificador no definido.

Cuando se halla un carácter especial se compara a ver si es un blanco. De no serlo implicaría que un posible identificador ha sido construido usando un carácter especial, lo que constituye un error sintáctico.

La existencia del banco nos lleva a dividir esta parte izquierda en otras dos partes ya que estamos frente a la posibilidad del tipo de sentencia FOR - TO-DO

De esta forma tomando esta nueva parte izquierda deberá coincidir con una palabra reservada. De no ser así se imprime un mensaje de error, y tomando la nueva parte derecha se verifica si es un identificador definido dentro de la tarea correspondiente.

Retomando la rama que partía desde la parte derecha del signo de asignación deberá verificarse si tiene una estructura matemática correcta, siempre y cuando en la parte izquierda no existieran palabras reservadas.

Una vez ubicadas como expresión aritmética deberá testearse si existen paréntesis dentro de la misma.

De existir se localiza el nivel más interior. En este nivel se buscan los operadores matemáticos, los cuales si existen se representan en pantalla en forma destacada y de acuerdo a las reglas del algebra correspondientes.

El próximo paso es detectar si el elemento básico anterior al paréntesis de apertura es un operador matemático. En caso negativo se verifica si se trata de una función legal, de no serlo se recurre a la tabla de identificadores. Si no está allí emite un mensaje de error.

Este proceso se repite buscando los niveles más exteriores de paréntesis.

En el caso de no detectar paréntesis se localizan los operadores y se visualiza la operación atendiendo a la jerarquía natural.

Si tampoco hay operadores se recorrerá la tabla de identificadores en su búsqueda. No hallarlo implica un error que se imprime por pantalla.

5 - ASIGNACIÓN DE RECURSO MEMORIA

Completado el análisis alfabético y la interpretación sintáctica, deben asignarse los volúmenes de memoria adecuados al programa.

Es interesante que el alumno se dé cuenta que toda esa secuencia de instrucciones que pretende ejecutar ocupan un lugar en la memoria. Sobre todo cuando, como nosotros, se trabaja con microcomputadores donde el recurso de almacén es escaso. La visualización de este espacio, de memoria creará en él conciencia de que deberá *optimizar* su programa.

Por ello en esta sección se simulan dos direcciones de memoria que serán los límites entre los cuales se almacenan las sucesivas sentencias del programa fuente. De hecho el espacio comprendido entre ambos límites tendrá un valor proporcional a la cantidad de líneas que introdujo.

A continuación la rutina de asignación de memoria explora la tabla de identificadores y asignará una posición a cada elemento que representa una variable simple. Como en esta tabla existe información descriptora del tipo de variable se asociarán por una parte las de tipo entero y las booleanas y por otra las reales a los efectos de simular el hecho de que la aritmética opera de un modo diferente cuando se trabaja con reales o con enteros.

Posteriormente en la misma pantalla se trata de destacar el modo de almacenamiento y la filosofía propia del uso de variables con subíndices o arreglos.

Sucesivos cuadritos con el mismo nombre de variable y distinguidos por el subíndice simulan este proceso.

También se necesita espacio para guardar los resultados parciales de las expresiones.

Los alumnos podrán observar con esta consideración la generación de subárboles con los que se resuelven las operaciones intermedias de manera que se dan cuenta de la forma en que se-van descomponiendo las estructuras en expresiones subordinadas más sencillas.

6 - VISUALIZACION DE CICLADOS

La intención de incluir esta sección fue aprovechar los ciclos como la mejor forma de exponer el proceso de asignación dinámica que implica la variación de los contadores de un lazo, típicamente de la forma I = I + 1.

La visualización incluye una representación de la locación de memoria que almacena a la variable contador. La variable de control aparece entonces con su valor numérico correspondiente a medida que se repite el recorrido del bucle.

Paralelamente se muestran las sucesivas sentencias que se ejecutarán repetidamente con intervalos de tiempo que permitan una clara observación.

En este caso es fundamental el uso de pantalla para graficar la secuencia de operación de un contador ya que permite al estudiante seguir paso a paso a través del video, la variación que suple la variable incrementada a medida que se ejecutan las diferentes instrucciones de un bucle.

Aquí hacemos uso de las características resaltantes de la pantalla como por ejemplo el video invertido a fin de llamar la atención del alumno y cumplir con los fines didácticos pretendidos.

7 IMPLEMENTACION

Todo el sistema se ha codificado en lenguaje BASIC. El mismo ha sido implementado utilizando el minicomputador B-22de BURROUGHS.

Se pensó en la utilización del lenguaje BASIC pues el propio programa puede ser analizado por los alumnos de ciclos superiores que se inician en la temática de Software de base y programación de sistemas, quienes además de ya conocer este lenguaje lo pueden implementar en la mayoría de los microcomputadores para su estudio.

Como se desprenderá de todo el análisis expuesto precedentemente este sistema es útil para dos grupos de alumnos: 1) Alumnos principiantes; 2) alumnos con conocimientos de programación.

Para el primero de ellos está destinado todo lo referente al análisis léxico de un programa para lo cual se ha aprovechado la capacidad de graficación de los computadores mostrando al alumno paso a paso y en forma destacada los posibles errores sintácticos y de estructura que comete. Además este manejo interactivo entre el léxico que utiliza el estudiante y la captación de la

computadora de las posibles falencias que se van produciendo durante la ejecución del programa, permiten que el alumno mantenga un diálogo fluido con el computador.

Esto ocasiona que el estudiante asimile los conocimientos mucho más rápidamente que con un análisis teórico.

Además la ventaja de este compilador didáctico es la visualización paso a paso de su programa, situación que no es permitida en un proceso de compilación normal.

Hemos detectado que al hacer uso de este compilador, el estudiante comienza a distinguir perfectamente la definición y uso de los elementos que componen un programa como variables, palabras reservadas, delimitadores, operadores, etc.

Además al aprender a distinguir estas nociones, toma una visión global de la estructura de los lenguajes de programación e independientemente de uno de ellos en particular. Al ampliar así su espectro, el alumno no se encasilla en un determinado lenguaje, sino que simplemente cambiando los términos de codificación puede encarar con éxito el manejo de cualquier otro tipo de lenguaje.

Para los alumnos con conocimientos de programación y dado que todo lenguaje de alto nivel está ligado a un compilador o intérprete, el presente sistema nos ha permitido orientarlos en los términos en que trabaja un compilador.

De esta forma lo que normalmente es una caja negra para un estudiante recién iniciado, para uno con conocimientos avanzados le da la visión general con que se traduce un lenguaje para ser ejecutado.

Asimismo si se sigue el desarrollo de los diagramas mostrados anteriormente se capta los diferentes pasos para producir la compilación.

Y al analizar cada programa que compone nuestro sistema en forma particular, se empieza a tomar contacto con los criterios que utiliza un computador para analizar un programa.

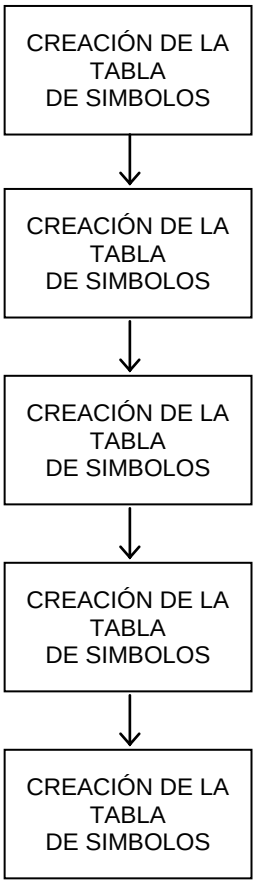
8 - CONSIDERACIONES FINALES

Esta parte del sistema que hemos presentado contempla la utilización por parte del alumno de un programa fuente codificado solamente en PASCAL pero queremos destacar que el sistema está previsto de tal forma que variando solamente las palabras reservadas y algún tipo de estructura particular permitirá analizar distintos lenguajes tales como BASIC o FORTRAN o algún otro de estructura similar.

En cuanto a las conclusiones obtenidas de la experimentación de este sistema podemos decir que el mismo ha redundado en una mejor y más rápida comprensión de los conceptos de programación normalmente complicados para captarse teóricamente.

Desde el punto de vista docente nos ha permitido un seguimiento paulatino del avance de la comprensión de estos temas por parte del alumnado detectando fácilmente en qué áreas encuentra mayores dificultades.

COMPILADOR



COMPILADOR DIDÁCTICO

Fig. II

ANÁLISIS LÉXICO
DIAGRAMA EN BLOQUE

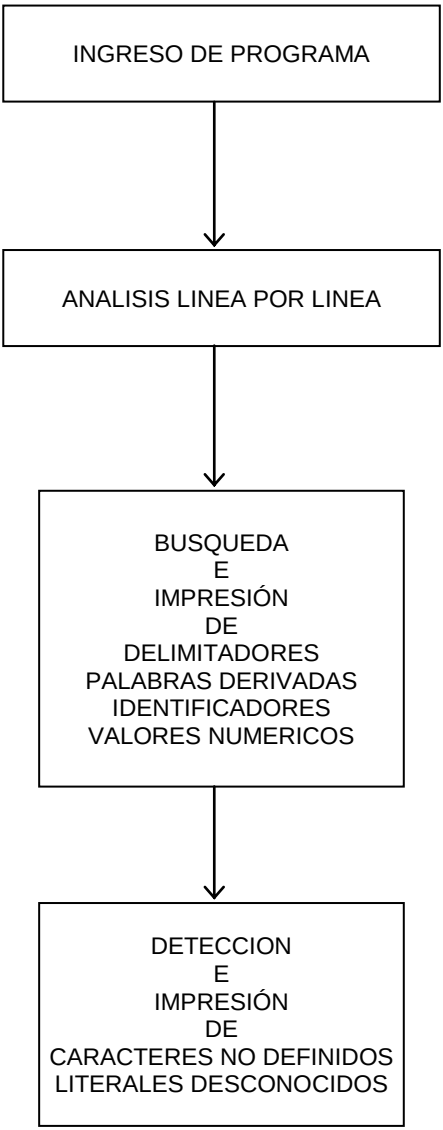


Fig. III

ANÁLISIS SINTÁCTICO

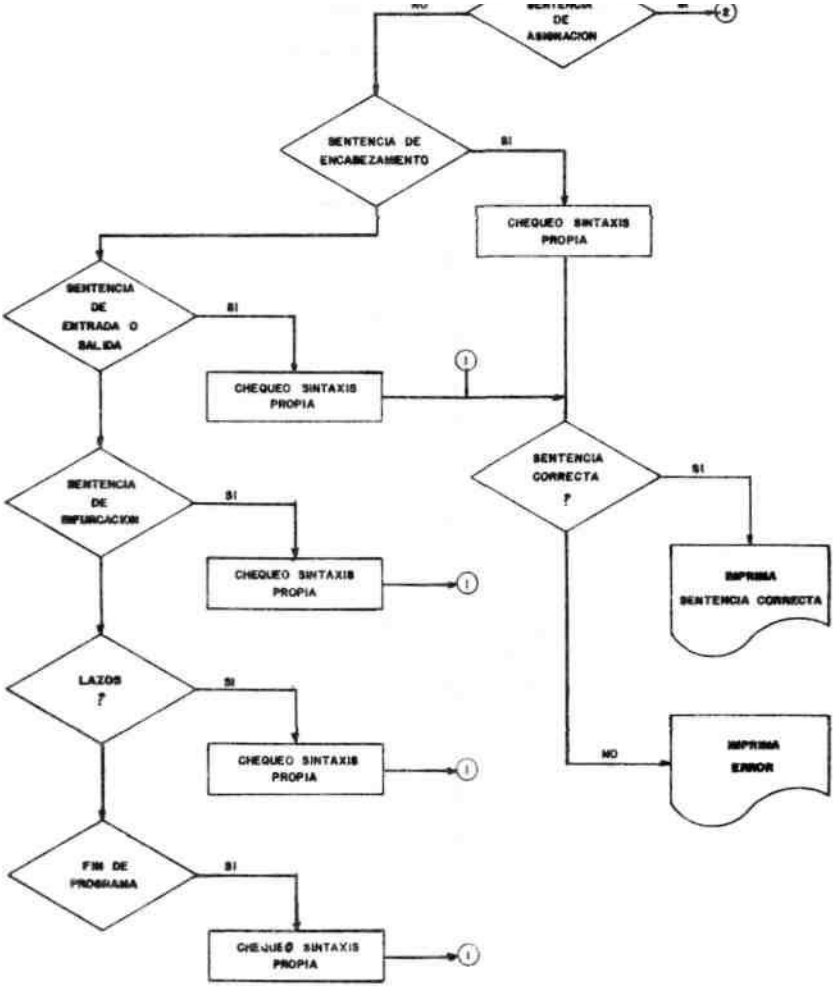


Fig. IV-a

ANÁLISIS SINTÁCTICO

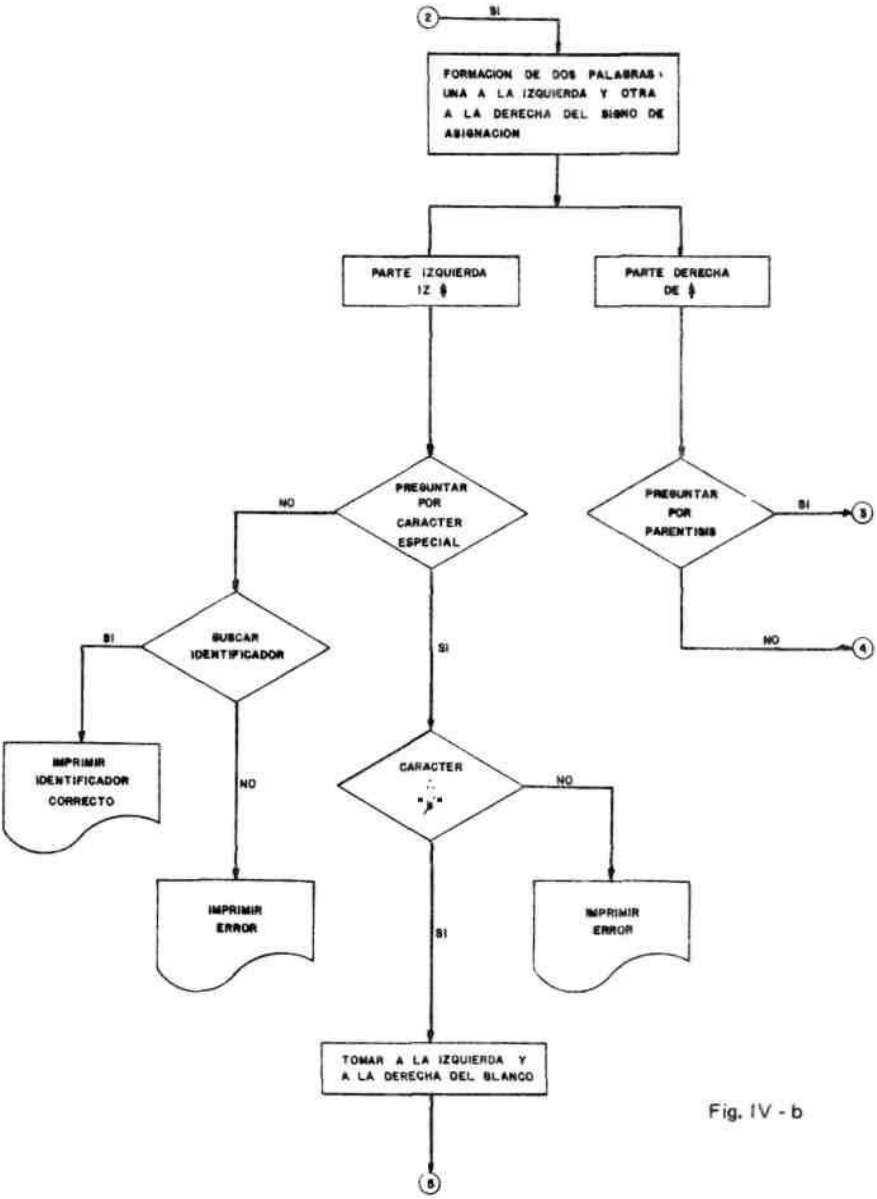


Fig. IV - b

ANÁLISIS SINTÁCTICO

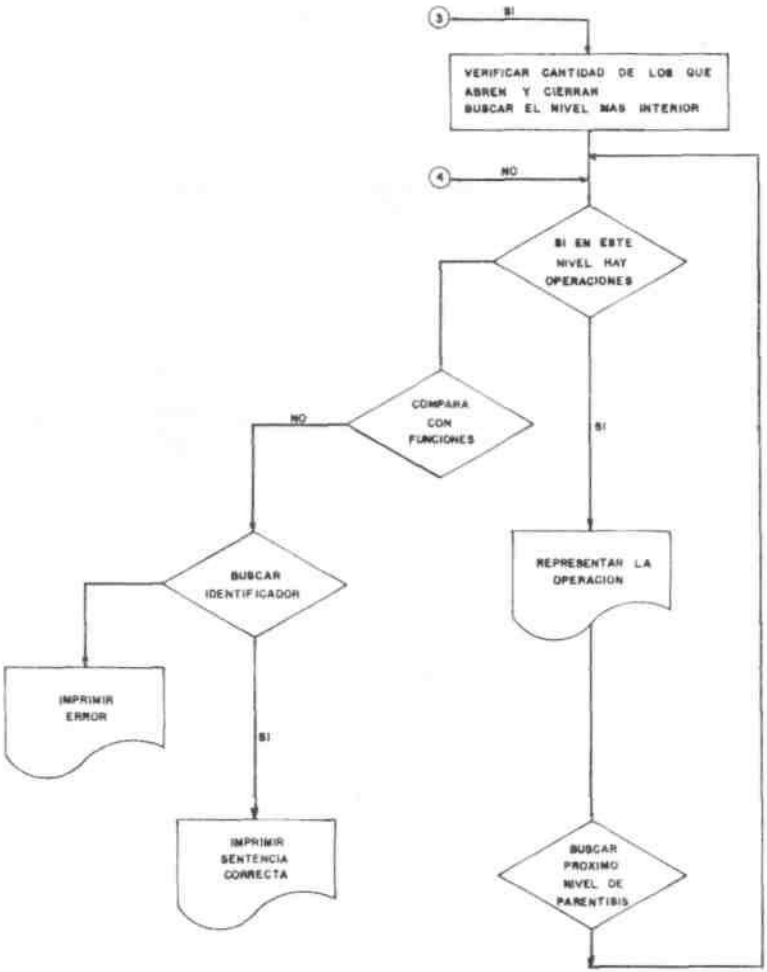


Fig. IV -c

ANÁLISIS SINTÁCTICO

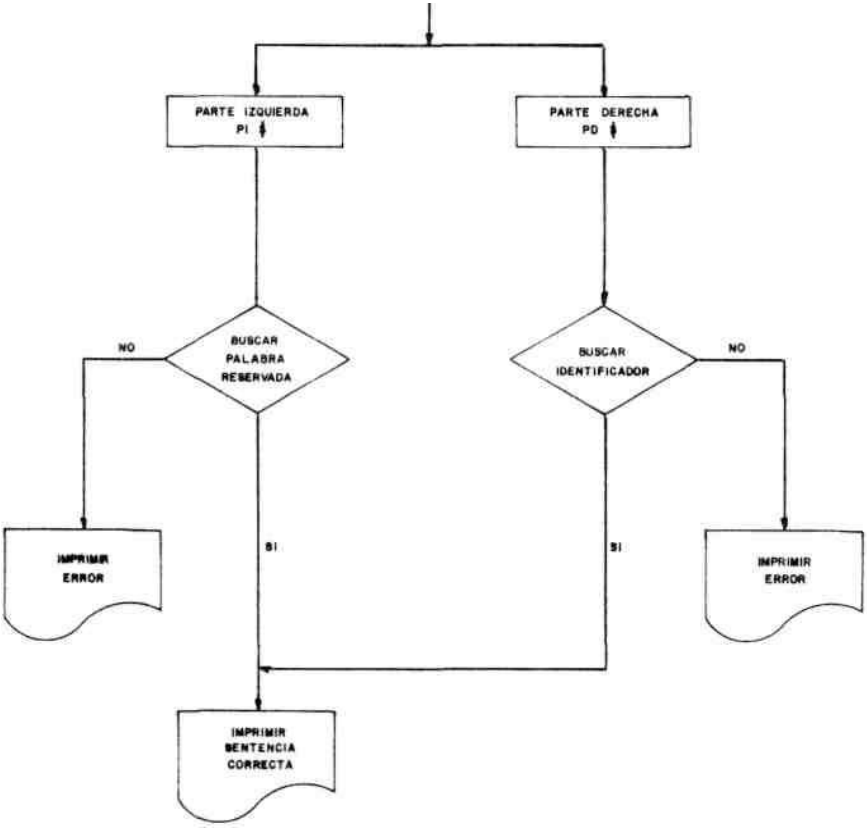


Fig. IV-d

VISUALIZACION DE ASIGNACIÓN DE MEMORIA

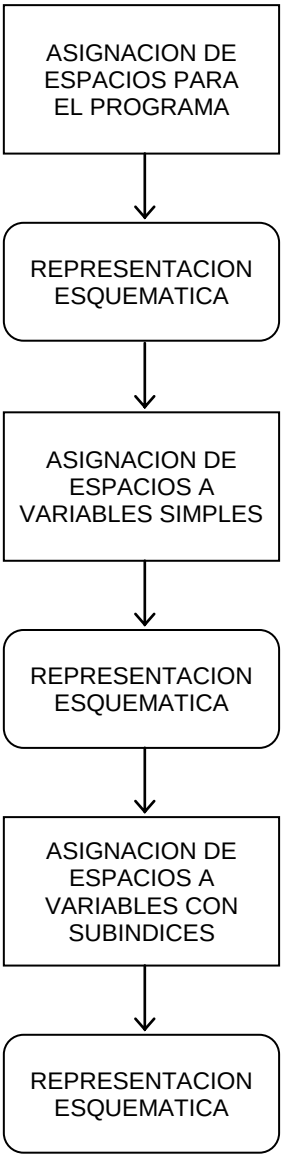


Fig. V

VISUALIZACION DE CICLADO

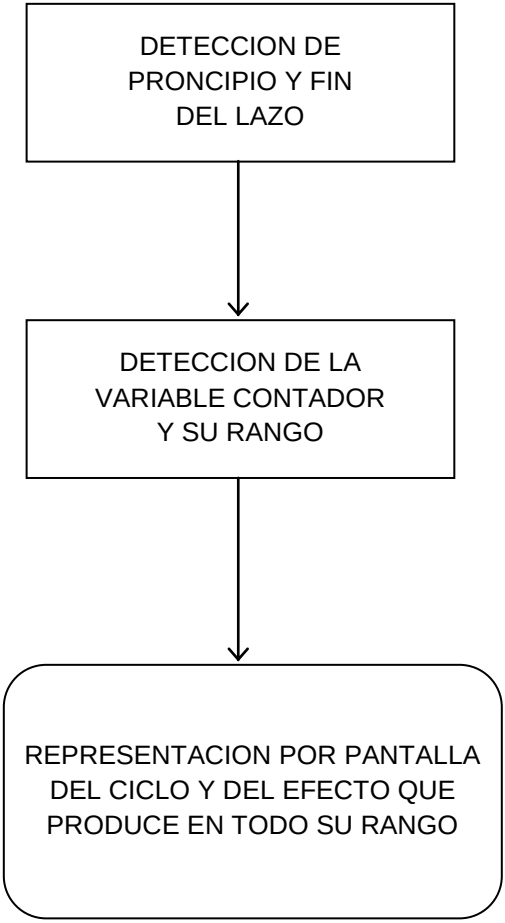


Fig. VI

ANÁLISIS LÉXICO
DIAGRAMA DE FLUJO

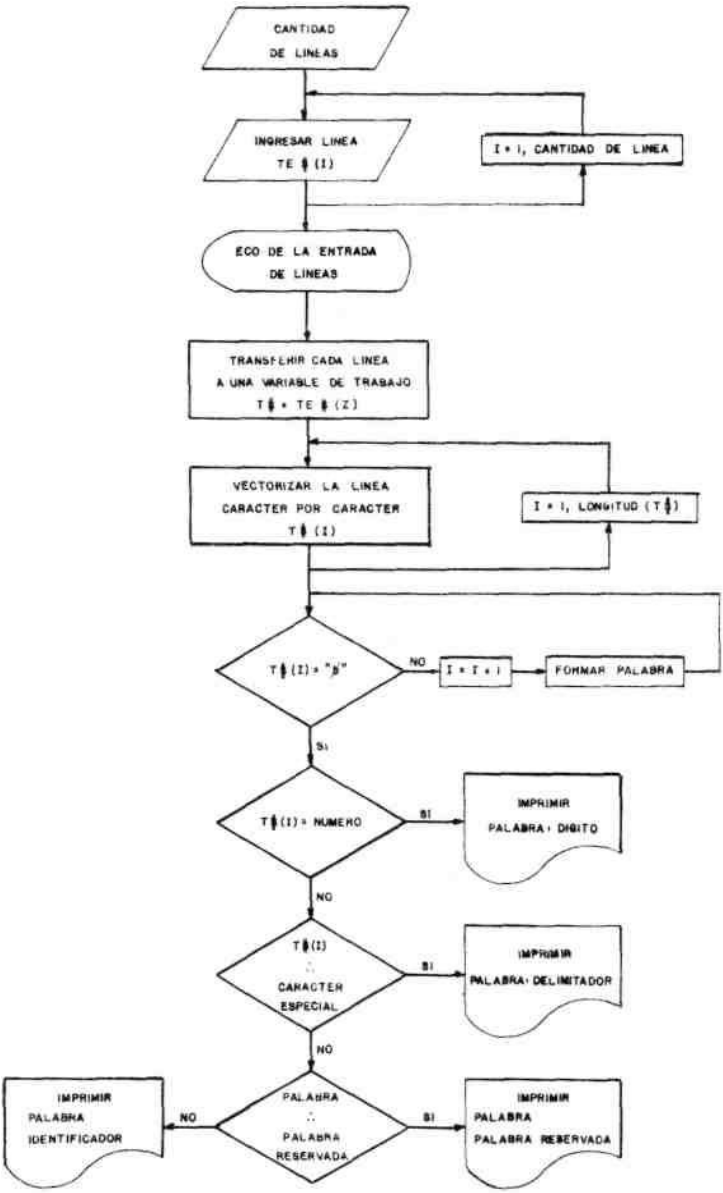


Fig. VII