UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

RECONOCIMIENTO DE FORMAS CON TECNICAS DE INTELIGENCIA ARTIFICIAL

T E S I S

QUE PARA OBTENER EL TITULO DE

INGENIERO EN COMPUTACION

PRESENTAN

ALVARO ESTANDIA GONZALEZ LUNA
RAFAEL GUZMAN MEJIA
JORGE RUIZ GARZA
HECTOR ZAVALA LUNA

México, D.F. **Marzo, 1986**

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

TABLA DE CONTENIDO

VII. CONCLUSIONES

1. INTRODUCCION

La Inteligencia Artificial (IA) es la rama de la Computación preocupada en diseñar sistemas computacionales que contengan características asociadas con la inteligencia humana tratando, en algunos casos, de entender los principios que la hacen posible. A estos sistemas se les denomina "Sistemas Inteligentes" [2 cap. 1 ; 3 vol. 1 pag. 3-11 ; 19 cap. 1].

Como toda disciplina científica, ésta posee varias áreas y técnicas de investigación, cada una con sus propios intereses y terminología.

Estas áreas son:

- Solución de Problemas
- Procesamiento de Lenguaje Natural
- Programación Automática
- Lógica Computacional
- Sistemas expertos
- Robótica
- Visión y Reconocimiento de Patrones

Solución de Problemas. [3 vol. 3 cap XV ; 19 cap. 2 ; 8]

Casi todas las tareas que se procesan en las computadoras se pueden clasificar como problemas. En cada caso se desea que la computadora efectúe las operaciones necesarias para lograr una meta. Por ejemplo, un programa de nómina procesa los datos de un

trabajador como son: sueldo, días trabajados, préstamos, etc., dando como resultado la impresión de los cheques de pago y otros reportes. Este es un caso de un programa que se limita a resolver un problema específico. En contraste a esto, la IA trata de diseñar programas de propósito general, los cuales sean capaces de solucionar diversos problemas. Esta tarea se lleva a cabo desarrollando métodos de solución que se aplican al conocimiento relevante a dicha tarea. Esta forma es similar a la usada por el hombre. Como ejemplo podemos citar al sistema MOLGEN [3 vol. 3 pags. 551-556], el cual asiste a genetistas moleculares en la planeción de experimentos a través de subdividir el problema.

Procesamiento de Lenguaje Natural. [3 caps. IV-V ; 19 cap.8]

Hasta ahora la comunicación con las computadoras se ha efectuado a un nivel muy primitivo, usando códigos o instrucciones rígidas, las cuales poseen una sintaxis precisa y cualquier desviación en ella es rechazada. El uso de computadoras para personal no especializado requiere de un entrenamiento previo, costoso y prolongado. Debido a esto, esta rama trata de lograr una comunicación natural hombre-máquina usando el lenguaje habitual consiguiéndose así una interacción sencilla. Como ejemplo tenemos al sistema HEARSAY II [3 vol. 1 pags. 343-348], diseñado para el entendimiento de lenguaje hablado usando fuentes de conocimiento independientes.

Programación Automática. [3 vol. 2 cap. X]

Uno de los principales problemas en computación es la creación y mantenimiento de programas. En este campo es donde se han

invertido mayores recursos y el resultado no ha sido el deseado. Generalmente después de haber implementado y liberado un sistema, éste aún contiene errores de difícil corrección y de grandes consecuencias.

La meta de la programación automática es el que una computadora genere sus propios programas por medio de especificaciones del problema a solucionar. Dentro de los sistemas existentes tenemos a APPRENTICE [3 vol. 2 pags 343-349], el cual actúa como socio y crítico al programador, quien se encarga del diseño e implantación, mientras APPRENTICE se encarga de la documentación, verificación y corrección del mismo.

Lógica Computacional. [3 vol. 3 caps. XI-XII]

En ciertas ocasiones ha surgido la necesidad de probar que un conjunto de hechos son el resultado de otros previamente demostrados. Así la lógica computacional es el arte de programar una computadora para que sea capaz de hacer deducciones.

Una aplicación obvia de este tipo de sistemas es en Matemáticas, donde después de haber definido un conjunto de axiomas tenga como resultado la deducción de teoremas. Uno de los ejemplos más ilustrativos es el sistema BMTP " Boyer-Moore Theorem Prover", el cual posee una teoría Matemática extensa donde se postulan teoremas que son probados automáticamente [3 vol. 3 pags. 102-113].

Sistemas expertos. [2 cap. 9 ; 3 vol. 2 ; 19 caps. 6 y 8]

Existen algunos trabajos que sólo pueden ser resueltos por expertos que han acumulado una gran cantidad de conocimiento, como serían diagnósticos médicos, consultores legales, diseño electrónico y análisis científico. Los programas que efectúan estos trabajos son muy útiles ya que usualmente poseen conocimiento de humanos muy calificados y especialistas en su tema. Estos programas son llamados sistemas expertos. Su característica más importante es su dependencia e interacción con una gran base de conocimiento siendo el principal problema a resolver la representación del conocimiento en ésta. Como ejemplo de estos sistemas se encuentra MYCIN [2 pag. 241-245 ; 3 vol. 2 pag. 184-192], programa que diagnostica infecciones bacterianas.

Robótica. [1 ; 3 vol. 3 ; 11 ; 19]

En los últimos años se ha incrementado la atención en esta área de investigación. Este campo se ha preocupado desde la optimización de movimientos de brazos de un robot, hasta métodos para planear una secuencia de acciones que lleven a la obtención de una meta específica. A pesar de ello los robots existentes usados en aplicaciones industriales son dispositivos sencillos, programados para efectuar únicamente trabajos repetitivos específicos. La mayoría de ellos son ciegos y sólo unos pocos cuentan con un sistema de visión rudimentaria, por tanto ha surgido la necesidad de dotarlos de una inteligencia capaz de reconocer aquello que están observando. Un sistema que se puede nombrar en este campo es TRANSISTOR WIRE-BONDING SYSTEM (ver punto 2.3), que se encarga de ensamblar diferentes tipos de transistores.

Visión y reconocimiento de patrones. [2; 3 vol. 3; 4; 5 ; 8; 10; 16; 18; 19]

Esta rama de la inteligencia artificial trata de "ver y entender" imágenes, lo que constituye un medio muy importante de comunicación con el mundo exterior. La solución de este problema puede conducir a sistemas de visión que tengan aplicaciones prácticas por sí mismos (análisis de mapas, por ejemplo), así como a mecanismos de visión compactos que se puedan incorporar como parte de un robot.

Generalmente los sistemas reconocedores de patrones consisten de dos partes: la primera se encarga de extraer los elementos que forman una escena previamente obtenida por medio de una o varias cámaras de televisión, digitalizando la imagen para su proceso, y la segunda, encargada de evaluar los objetos contenidos en la imagen a partir de las características obtenidas por la primera fase.

Investigadores en este campo han encontrado grandes dificultades en la extracción de características de objetos en una imagen, ya que algunos presentan dificultades para distinguir sus límites por estar cubiertos uno con otro, por el tipo de material en la superficie y la iluminación. Asimismo, otro de los problemas difíciles de resolver es el relacionado con el tiempo necesario para extraer todos los elementos de una escena.

En cuanto a la evaluación de los objetos, el problema radica

en la elección de métodos y reglas que definan cada uno de sus elementos, haciéndolo diferente a los demás. Esto requiere un conocimiento "a priori" de lo que se trata de reconocer. Esto implica la búsqueda de características de las formas independientes, en lo posible, a su posición, color, tamaño, etc. El segundo obstáculo es la elección o creación de algoritmos eficientes de búsqueda y formas de almacenamiento (representación del conocimiento) óptimas y económicas en cuanto a tiempo y espacio.

El trabajar con imágenes que presenten los problemas descritos anteriormente, dificulta su solución. Por ahora el reconocimiento de formas ha realizado notables avances si se trabaja con escenarios que cumplan con los siguientes criterios:

- Que las características de los elementos presentes en una escena no cambien considerablemente dentro del rango previsto a reconocer, es decir, que estos cambios no vayan más allá de lo que el sistema pueda manejar, por ejemplo cambios drásticos de ángulo de visión, luminosidad, textura, etc. El tipo de aprendizaje que posee el sistema es el llamado Aprendizaje por Observación que se explica en esencia en el capítulo 5.

- Que las características de la imagen estén dominadas por las características de los objetos que la forman, esto es, que la imagen esté formada por elementos previamente conocidos cuya interrelación dentro de la imagen no altere sus características básicas.

Estos criterios implican que analizar un escenario formado por figuras limitadas, como rectángulos y cuadrados, es diferente a decidir si un cuarto que tiene un elemento complejo por ejemplo el teléfono, como le ocurriría a un robot que necesitara diferenciar colores, escalas, coordenadas etc., para distinguir teléfonos, basureros, puertas y máquinas de escribir de una oficina. En este caso se tendría que recurrir a una teoría de visión usando mayores elementos descriptivos.

La creación de programas inteligentes requiere de ciertas bases sobre cómo se logra el conocimiento y cómo se puede incorporar éste a sistemas de cómputo. Para desarrollar sistemas como el que se presenta, es necesario responder las siguientes preguntas a lo largo de este trabajo.

- Qué clase de conocimiento se usó ?
- Cómo se presentó ?
- Cuánto se requirió ?

En Inteligencia Artificial la representación del conocimiento es una combinación de estructuras de datos y procedimientos, que usados correctamente, conducen a un ambiente cognoscitivo. Este trabajo involucra el diseño de varias clases de estructuras para guardar información como también el desarrollo que permita su manejo inteligente para hacer ciertos razonamientos o inferencias. Por tanto, las metas de un sistema de Inteligencia Artificial pueden ser descritas en términos de tareas que lleven a la

adquisición de conocimiento. En este proceso se involucran tres pasos: la obtención de más conocimiento, el manejo de información de las bases de conocimiento y el razonamiento sobre el uso de esta información para la búsqueda de una solución.

1.2 APLICACIONES DE SISTEMAS RECONOCEDORES DE PATRONES Y VISION

A pesar de que el entendimiento del proceso de visión resulte complicado, se encuentran aplicaciones concretas y exitosas en diferentes campos.

Una de las áreas es la del procesamiento e interpretación de imágenes bidimensionales. Aquí se incluyen algunas aplicaciones:

Percepción Remota

- Análisis de imágenes de satélites para monitoreo de recursos naturales. Se puede citar al Interpretador Multibanda de Fotografías Aéreas [3 vol. III pags. 305-308], el cual reconoce regiones que podrian ser: de vegetación, de caminos, sombreadas y homogéneas.

Control de Calidad

- Definiendo modelos de los objetos sin defecto y su relación espacial con respecto a cada producto terminado. Cuando las diferencias entre el modelo y el producto terminado bajo análisis rebasa ciertos límites prestablecidos, éste es rechazado, de otra manera se acepta. Existen tres métodos para lograrlo: el primero compara las imágenes "pixel" a "pixel", el segundo extrae de ellas ciertas características y son

éstas las que compara; mientras que el tercero divide las imágenes en regiones y las compara éstas una a una por región (análisis por ventana).

Médicas

- Análisis Celular Sanguíneo (Young 1960). Este sistema divide glóbulos blancos en categorías como son: neutrófilos, eosinófilos, basófilos, linfocitos y monocitos [2 pags. 205-209].

Otra de las áreas de gran interés es la visión enfocada a robots para la industria, lo cual permitirá incrementar su versatilidad y su flexibilidad. En este campo la información tridimensional resulta indispensable y se necesitan desarrollar dispositivos que resuelvan la detección de profundidad. En la actualidad existen algunos sistemas implementados que obtienen la información "visual" a través de cámaras de televisión y algoritmos apropiados para su procesamiento, para guiar los movimientos de manipuladores (robots):

Aplicaciones:

- Robots armadores de piezas electrónicas, como el sistema TRANSISTOR WIRE-BONDING SYSTEM (consultar sección 2.3.2).

- Robots reconocedores de partes involucradas en procesos en una línea de producción sin necesidad de estar en un orden específico, como el caso del sistema CONSIGHT-1 (ver inciso 2.3.3).

1.3 OBJETIVOS

1.3.1 Objetivo General.

El objetivo general de esta tesis es el diseño e implantación de un sistema inteligente capaz de reconocer y aprender ciertas figuras e imágenes bidimensionales, de forma eficiente y confiable, para utilizarlo en aplicaciones prácticas.

1.3.2 Objetivos Específicos.

- Contribuir con algunas bases teóricas para el desarrollo de sistemas reconocedores específicos, que puedan ser utilizados en robótica para conducir a una automatización inteligente de algunos procesos industriales.

- Utilizar técnicas de diseño estructurado de "Software" para producir un sistema de fácil mantenimiento, altamente modificable, flexible, de uso sencillo, de alta calidad y transportable a otros sistemas de cómputo.

- Contribuir a mantener vivo el interés, dentro de la Ingeniería, por este tema de gran potencialidad y tan poco desarrollado en México. Ya que en la actualidad se ha manifestado un gran interés por la Inteligencia Artificial (IA) únicamente en universidades y como parte de tesis profesionales, pero para que ésta se desarrolle plenamente en nuestro pais, debe haber interacción entre estos grupos con la industria.

1.4 ORGANIZACION DE LA TESIS.

Esta tesis está organizada en siete capítulos muy relacionados con la estructura del sistema desarrollado, el cual se bautizó como SIREI, "SIstema REconocedor de Imágenes". En este primer capítulo se ha incluido una breve introducción del problema a resolver. Se dio una breve reseña de algunas aplicaciones en este campo y los objetivos que se buscan alcanzar con el desarrollo de esta tesis.

En el segundo capítulo se describe al sistema, explicando genéricamente cada una de sus partes y se da una breve reseña de algunos sistemas desarrollados en este campo, comparándolos con respecto a SIREI. Adicionalmente se habla de las estructuras de las fuentes de conocimiento utilizadas.

En el tercero se habla del generador, la primera etapa del sistema, cuya función es formar un escenario por medio de ciertas figuras básicas o creando otras por medio de líneas. Este módulo simula y sustituye la interfaz de la computadora con el medio ambiente, como sería en este caso una cámara de televisión y la electrónica necesaria para incorporar una imagen digitalizada al computador.

El cuarto capítulo trata sobre la función de reconocer una imagen, sin importar si ésta es producto de simulaciones en computadora o extraída de un medio ambiente industrial a través de una cámara de T.V., formada previamente por el módulo generador. Esta acción se lleva a cabo reconociendo formas básicas,

compuestas por líneas, y analizando la relación entre ellas. Una vez terminado este paso, el sistema determina si la figura generada forma parte del conjunto de figuras contenidas en la base de conocimiento.

En el quinto capítulo se describe el módulo de aprendizaje, cuya función es incorporar, en caso de así desearlo el usuario, una figura inexistente en la base de conocimiento. Asimismo, se hace una revisión de los diferentes tipos de aprendizaje empleados por sistemas de inteligencia artificial.

En el sexto capítulo se detallan los programas de utilería creados para facilitar el uso y matenimiento del sistema. Estas rutinas fueron diseñadas buscando una mayor interacción del usuario con el sistema.

En el capítulo séptimo , las conclusiones, se evalúan los algoritmos y el sistema en general, asimismo se proponen extensiones futuras a este trabajo y se analizan los objetivos alcanzados.

En los apéndices se explica y describe el método de diseño usado para la elaboración del sistema, en cuanto a "software", se incluyen los listados fuentes de los programas que forman el sistema, como también los diagramas de estructura de los mismos para mayor entendimiento del flujo de información y control. Por último se cuenta con el manual de usuario de SIREI, el cual tiene como finalidad ilustrar al usuario los pasos que se tienen que seguir para la utilización del sistema.

2. DESCRIPCION DEL SISTEMA

2.1 INTRODUCCION

Para poder entender mejor las principales características que definen al sistema, así como las similitudes y diferencias que presenta en relación a otros sistemas existentes, en el presente capítulo se describen, en la sección 2.2 las diferentes etapas de SIREI, donde se podrán apreciar las principales características que poseen. Posteriormente en la sección 2.3 se discuten algunos sistemas existentes, así como algunas similitudes y diferencias que SIREI y estos sistemas presentan.

2.2 SIREI

El sistema consta de cuatro módulos y dos bases de conocimiento. La figura 2.1 ilustra las interacciones que existen entre ellos. Los módulos permiten la interacción del usuario con el sistema, mientras que en las bases de conocimiento se encuentra toda la información que dispone SIREI para poder efectuar el reconocimiento y el aprendizaje. El generador auxilia al usuario en la creación de la imagen que se desea analizar, la cual se genera en las pantallas gráficas con que cuenta la APPLE IIe (equipo en el que se implementó el sistema). El reconocedor trata de identificar la imagen generada, y el aprendizaje, en caso de necesitarse, permite la adición de nuevas imágenes al sistema. Por su parte las utilerías hacen transparente al usuario algunos procesos, que si bien no forman parte del sistema, le son de gran ayuda durante la utilización del mismo (crean las bases de conocimiento, las depuran y emiten reportes de su contenido).

COMPONENTES DEL SISTEMA

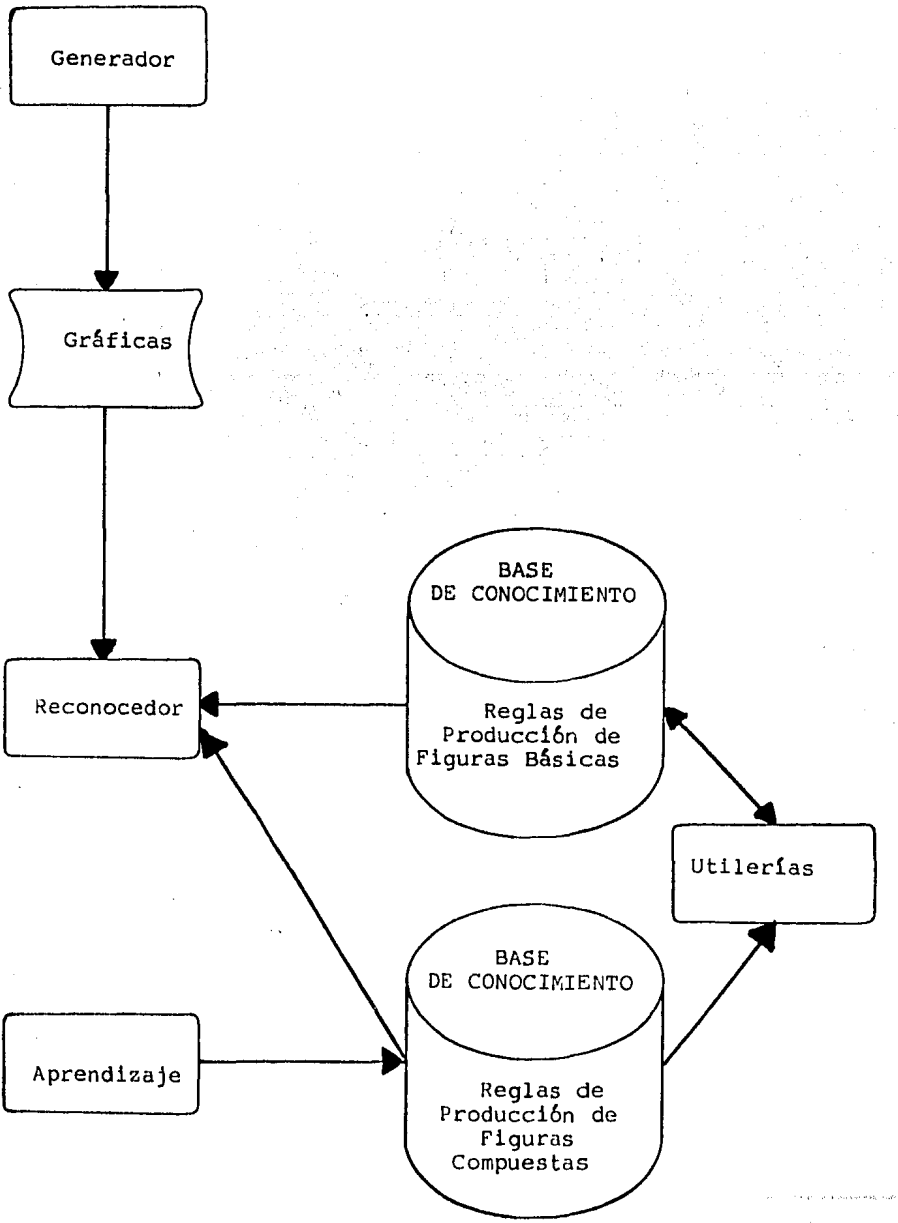


Figura 2.1

A continuación se describen las características generales de cada uno de los módulos de SIREI :

2.2.1 GENERADOR

La finalidad de este módulo es la creación de figuras susceptibles de ser reconocidas por el sistema. Permite al usuario dibujar figuras básicas en la pantalla de la computadora.

Por figuras básicas (FBs) se entienden aquellas figuras que constituyen una sola figura geométrica, tales como el cuadrado y el rectángulo o cualquier otra figura formada por rectas (figuras poligonales). Al combinar varias figuras básicas, el usuario produce lo que se denominó figuras compuestas (FCs), lo cual permite la creación de una gran variedad de figuras a reconocer. Este módulo realiza validaciones sobre las dimensiones de las figuras que el usuario pretende dibujar para que no excedan las dimensiones físicas de la pantalla. SIREI requiere de este módulo pues es el que simula la interfaz entre el mundo real y la imagen digitalizada que se desea reconocer.

La salida de este módulo es una figura compuesta (o básica, si así se desea), en forma de una gráfica que la computadora almacena en las localidades de memoria que tiene expresamente destinadas para tal fin. Conviene resaltar que la gráfica así producida se encuentra ya digitalizada y codificada en la computadora. En un sistema de reconocimiento de formas de aplicación práctica, se requeriría un módulo más, capaz de captar la imagen

del mundo exterior, digitalizarla y preprocesarla, para la eliminación de "ruidos", en forma que pueda ser analizada por una computadora. [3 vol. 3 cap. XIII ; 4, 10, 16].

2.2.2 RECONOCEDOR

Este módulo efectúa el reconocimiento de la figura creada por el usuario mediante el Generador de Figuras.

La labor de reconocimiento se realiza en tres etapas:

- Generación del Medio Ambiente.

Durante esta etapa, el reconocedor carga la Base de Conocimiento de las Figuras Básicas existentes en el sistema, crea la Matriz Gráfica, que es la estructura de la información que posee la imagen a reconocer, y encuentra dentro de ella el primer punto iluminado con el que comenzará a trabajar; es decir, esta etapa le sirve al reconocedor para crear en memoria principal las distintas estructuras de información que utilizará a lo largo del reconocimiento, lo cual optimiza el tiempo de proceso, medido en microsegundos, que es varios miles de veces menor que el de acceso a disco, medido en milisegundos.

- Armado de la RFCG.

La finalidad de esta etapa es la de ir reconociendo las distintas FBs que forman la imagen y la de integrar éstas formando la representación de la figura compuesta generada (RFCG). A grandes rasgos, el procedimiento aquí utilizado

consiste en encontrar puntos iluminados dentro de la matriz gráfica, identificar el vértice que corresponda a cada uno de ellos, ya que no se asegura que los puntos iluminados encontrados sean vértices; después, identificar los lados o sujetos que lo forman para posteriormente escoger dentro del conjunto de figuras básicas definidas, alguna que se apegue a las características recién encontradas, esto es, que posea la información del vértice identificado. Una vez encontrada una figura básica que cumple con lo anterior, el procedimiento continúa con la identificación del resto de dicha figura y al terminar de efectuarla, la guarda dentro de la estructura de la información de la figura compuesta que está siendo formada. El procedimiento se repetirá cuantas veces sea necesario hasta que detecte que dentro de la pantalla no existen más figuras básicas adyacentes a las encontradas.

- Reconocimiento de la FCG

Una vez creada la RFCG, ésta se busca entre todas las estructuras compuestas previamente aprendidas y almacenadas (ver punto 2.2.5). En caso de tener éxito se le informa al usuario de la figura que se trata, en caso contario, el sistema pasará al Módulo de Aprendizaje en espera de que el usuario decida si esta nueva RFCG se aprende o no. La heurística aquí utilizada consiste en recorrer la información de las figuras compuestas, que se encuentra ordenada alfabéticamente en forma ascendente a través de una llave de identificación única, mediante un proceso de búsqueda binaria.

2.2.3 APRENDIZAJE.

Una vez que el sistema ha armado la RFCG y que no ha sido identificada, este módulo consulta al usuario sobre la incorporación de esta figura al conjunto de figuras que el sistema puede reconocer (denominado Base de Conocimiento de Figuras Compuestas o BCFC).

Como SIREI ya ha creado una estructura que representa a la figura compuesta en estudio (RFCG) y además al haber tratado de encontrarla dentro de la BCFC previamente, conoce la posición que dentro de ella debe ocupar, únicamente falta guardarla en el lugar correspondiente. De esto se encarga el Módulo de Aprendizaje, el cual genera el espacio requerido en la posición solicitada, recorriendo la información de aquellas figuras que se encuentren a partir de la posición en la que la nueva figura es insertada.

2.2.4 UTILERIAS.

Este módulo consta de varios programas que permiten el mantenimiento y la consulta del sistema. Aunque estas rutinas no forman realmente parte del sistema, se crearon porque facilitan grandemente algunos otros procesos que el usuario pudiera requerir durante la utilización de SIREI.

Las dos primeras rutinas se encargan de crear (inicializar) las Bases de Conocimiento con que cuenta SIREI para que sean utilizadas posteriormente por los procesos del sistema.

Otra de las rutinas es la encargada de "cargar" nuevas figuras básicas a la base de conocimiento (ver punto 2.2.5), en forma tal, que el sistema sea capaz de reconocer estas nuevas FBs por sí mismas o como componentes de figuras compuestas.

Existe también una rutina que permite la eliminación de las FCs irrelevantes, que las borra de la BCFC.

Las rutinas restantes se emplean para obtener listados de los conjuntos de figuras (básicas y compuestas) que el sistema posee en sus Bases de Conocimiento.

Las dos bases de conocimiento utilizadas por los distintos módulos del sistema son :

2.2.5 BASES DE CONOCIMIENTO [3 vol.1 cap. III].

El conjunto de figuras básicas que el sistema puede reconocer se denomina Base de Conocimiento de Figuras Básicas (BCFB). Este es un conjunto dinámico ya que, mediante el módulo cargador perteneciente a las Utilerías, es posible aumentar elementos del conjunto logrando una independencia de éste con respecto al sistema.

Cada figura básica se representa mediante una regla de producción y una función que se almacenan de acuerdo con ciertas estructuras de datos específicas, las cuales se describen en mayor detalle en el capítulo 4.

Al conjunto de figuras compuestas que el sistema puede reconocer se conoce como Base de Conocimiento de Figuras Compuestas (BCFC). Este también es un conjunto dinámico, pero no se modifica mediante Utilerías, sino que se actualiza e incrementa directamente mediante el Módulo de Aprendizaje.

Cada figura compuesta se representa mediante una regla de producción basada en una relación entre figuras básicas. La manera como este conjunto se almacena es sumamente importante pues un reconocimiento eficiente en cuanto a rapidez y exactitud dependerá en mucho de las rutas de acceso al conjunto de figuras compuestas. Esto se trata con mayor profundidad en el capítulo 5.

Si se desea conocer a detalle las partes que componen a cada módulo, se deben consultar los siguientes cuatro capítulos, que tratan a detalle cada uno de ellos, o bien, los diagramas de estructura que aparecen en el apéndice B.

2.3 SISTEMAS EXISTENTES

En esta sección se describen brevemente algunos sistemas existentes, comparándolos y contrastándolos con SIREI.

2.3.1. Sistema: SEE, Proyecto MAC, tesis doctoral, M.I.T. [8]

Autor : Adolfo Guzmán Arenas

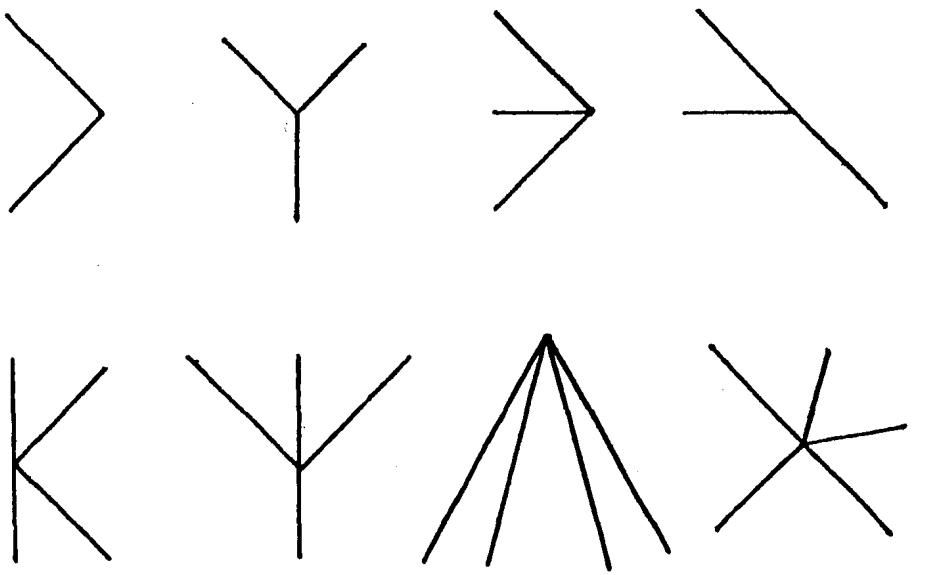
Año : 1968

El sistema SEE analiza escenas formadas por bloques geomé-

tricos sin contar con modelos de objetos almacenados previamente en el sistema.

SEE identifica los bloques aunque alguna o algunas de sus superficies no estén completamente visibles. Su parte principal analiza la imagen en términos de uniones o vértices y regiones, en busca de pistas que indiquen cuál de las dos regiones forma parte del mismo cuerpo. Guzmán define un conjunto de uniones las cuales se consideran significativas para el análisis. Una unión es un punto donde se encuentran dos o más líneas.

Clasificación de las uniones posibles :



El estudio de las regiones alrededor de las uniones sugiere la heurística para decidir qué región forma parte de una cara de un cuerpo o de otro. El programa usa todas las evidencias disponibles para determinar cuáles regiones deben ser unidas para formar un cuerpo. Por medio de las reglas heurísticas de unión se establecen las relaciones entre las regiones y sus ligas.

El trabajo de Guzmán fue uno de los más distinguidos por ser el primero que usó métodos heurísticos extensivos para el entendimiento de imágenes. Demostró que este tipo de problemas pueden ser interpretados por medio de procesos simbólicos en lugar de procedimientos de comparación numérica. Su investigación motivó el estudio al mundo de bloques (desarrollado por Roberts, Falk, Waltz y Shirai entre otros) [3 vol. 3 cap. XIII pags. 125-194). Aunque su sistema reconoce objetos tridimensionales su heurística está rígidamente ligada a imágenes bidimensionales.

SIREI al igual que SEE usa métodos heurísticos y análisis de vértices. SIREI no tiene implementada la tercera dimensión, lo cual forma parte de una extensión futura. Una mejora de SIREI en relación a SEE es que posee dos bases de conocimiento donde tiene almacenadas las figuras sujetas al reconocimiento y tiene además opción a aprender nuevas figuras.

2.3.2. Sistema: "Transistor Wire - Bonding System".

Autor : Kashioka, Ejiri, Sakamoto.

Año : 1976. [3 vol. 3 pags. 301-306]

Este sistema fue el primer robot de producción con uso extensivo de funciones de procesamiento de imágenes. El sistema visual localiza un transistor y automáticamente efectúa las conexiones con alambre de oro entre sus terminales y la terminal correspondiente.

Este robot es capaz de ensamblar 2000 transistores por hora, duplicando la velocidad de las máquinas semiautomáticas, con una precisión del 99 %. Su ventaja es que 5 grupos de 10 máquinas cada uno, pueden compartir una minicomputadora central, utilizando de esta manera un procesador de imágenes por grupo.

La tarea de este sistema de visión es la obtención de las coordenadas precisas de la base y el emisor del dispositivo y el enviar esta información al servomecanismo. Para hacer esto, se analiza un área de 1,100 x 800 micrómetros por medio de una cámara de televisión montada en un microscopio. Esta imagen se digitaliza para formar una matriz de 160 x 120 "pixels". Para localizar la posición y orientación del dispositivo, se buscan dos o tres patrones estándares de 12 x 12 "pixels". La búsqueda se efectúa por medio de electrónica especial que recibe el cuadro correspondiente a la imagen de la cámara de T.V. y el patrón estándar de la minicomputadora. La comparación se efectúa por un barrido de aproximadamente 16.7 milisegundos (ms) de duración. La evaluación de la posición se desarrolla en la computadora simultáneamente al tiempo de borrado de un patrón. Posteriormente se usa lógica adicional para incrementar la precisión. El sistema

tiene la flexibilidad necesaria para manejar varios tipos de transistores. Los patrones de transistores adicionales se generan de forma interactiva.

Como se puede observar, éste es un sistema que se encuentra implementado en una aplicación industrial. El sistema cumple su objetivo pero es poco versátil para poder ensamblar otros dispositivos que no sean transistores como sería el caso de circuitos integrados conteniendo otro tipo de elementos discretos. Se tomó en consideración porque muestra la factibilidad de aplicaciones prácticas de SIREI, con el cual se podría ampliar la gama de dispositivos a ensamblar, ya que el sistema estaría capacitado para aprender a reconocer distintos dispositivos. Las modificaciones necesarias estarían en función de estas nuevas piezas que se desearan identificar. Además se tendrían que analizar las reglas de producción de las figuras básicas, de forma que se pudieran combinar éstas para el reconocimiento de diferentes partes electrónicas compuestas por varios elementos discretos interconectados.

2.3.3. Sistema : Consight-1. [3 vol. 3 pags. 301 a 306]
Autor : Holland, Rossol y Ward
Año : 1979.

Consight-1 es un robot con visión que identifica y toma, aleatoriamente, partes de una banda sin fin en un proceso de producción. Su sistema de visión está diseñado para trabajar en un ambiente "ruidoso" y determina la posición y orientación de

las partes en la banda. Después de la localización de una pieza, la banda se detiene y el robot transfiere esta parte a una localidad predeterminada.

Una característica de este sistema es la facilidad de obtener imágenes de los objetos que no siempre tienen un alto contraste con respecto al fondo de la banda. Para lograr esto, el sistema proyecta un haz intenso de luz a la superficie de la banda. Una cámara con 256 fotoceldas sensa este haz. Cuando el objeto llega a la zona iluminada intercepta la luz dando como resultado que la superficie de la banda aparezca iluminada y el objeto obscuro. Para evitar las sombras se usan dos fuentes de iluminación. Así se obtiene la silueta del objeto. La cámara barre la banda a una tasa constante independiente de la velocidad de esta banda. Una línea de barrido se muestrea para cada incremento del viaje de la banda, midiendo la posición y la velocidad de la banda. El muestreo continuo de líneas barridas produce una imagen bidimensional. Por tanto el sistema no almacena toda la imagen sino que la procesa línea a línea. Cada objeto que pasa por la sección de visión actualiza las estadísticas de cada componente. Estas incluyen posición, color, conteo de "pixels", sumas de coordenadas en (x,y), y sumas de los productos de (x,y). Cuando el objeto pasó completamente esta región se usan estas estadísticas para calcular numéricamente sus descriptores.

Este sistema es de gran importancia porque nos describe la forma de implantar la visión a SIREI como una ampliación futura

al mismo. Como se sabe, éste no cuenta con un sistema de visión y este paso se simula por medio del módulo de generación. Consight-1 posee esta característica, la cual incorporada a SIREI, le daría la posibilidad de convertirse en un sistema de robótica más completo. Al utilizar una cámara de TV y acoplarle un digitalizador, éste le pasaría la imagen digitalizada a SIREI en vez del generador; con ello, el sistema sería capaz de reconocer imágenes del mundo real.

Con estos sistemas anteriormente descritos se dio una breve reseña de cómo se está atacando el problema de reconocimiento de formas y qué similitudes poseen éstos con SIREI. Además se plantea como a través de extensiones a SIREI se podría obtener un comportamiento más versátil que el presentado por ellos.

Como se vio, en la solución del reconocimiento intervienen varios factores. El primero de ellos es el de visión, cómo se resuelve el problema de proporcionar una imagen confiable, libre de "ruido" y conteniendo las características necesarias para su proceso. Este factor no fue considerado para SIREI, ya que requiere de equipo de costo muy alto, por lo que se simuló esta etapa dando como resultado una imagen óptima.

El segundo factor es la extracción de elementos de una imagen y su reconocimiento. Algunos sistemas mostrados usan métodos estadísticos para clasificar patrones. En estos métodos se comparan características de la figura en análisis con un conjunto de características almacenadas. A estos métodos se les critica

fuertemente por su falta de poder descriptivo, ya que sólo se encargan de clasificar patrones y no de describir sus características [3 vol. 3 cap. XIII].

Por otra parte existen métodos sintácticos que aplicados al análisis de escenas constituyen una herramienta valiosa para la solución de este problema. Se dice que constituyen el lenguaje de la teoría de visión. Los patrones están considerados como enunciados en un lenguaje por una gramática formal.

Esta aproximación sintáctica asume una gramática de la imagen para constituir la representación de los objetos y su interrelación en la escena. A este método se le conoce como reglas de producción y se explicará ampliamente en el capítulo 4, ya que es el adoptado en el sistema [3 vol. 1 pags. 190 a 199 ; 5 ; 6]

En los siguientes capítulos se describen detalladamente cada uno de los módulos de SIREI.

3. GENERADOR

3.1 INTRODUCCION [9, 13].

La función de este módulo es permitir al usuario la creación de una imagen compuesta de figuras geométricas simples denominadas básicas, como son un cuadrado, un rectángulo o cualquier otra figura cerrada formada por líneas rectas.

La imagen se crea en la pantalla de la computadora, lo cual simula la interfaz entre SIREI y el mundo exterior. El tamaño de la pantalla de la computadora Apple IIe [15] es de 280 por 192 "pixels", donde las coordenadas del punto inferior izquierdo son (0,0) y las del punto superior derecho son (279,191). Esto implica que las dimensiones máximas de una figura son de 280 "pixels" en sentido horizontal y de 192 "pixels" en sentido vertical. El módulo generador realiza verificaciones para no permitir que algún punto de una figura quede fuera de la pantalla. Existen además ciertos requerimientos para que una figura pueda ser reconocida por SIREI, los cuales se describen en el capítulo cuatro.

Cuando el programa principal de SIREI invoca al generador, se despliega un menú con seis posibles opciones de la siguiente manera :

0. Fin
1. Inicializa gráficas
2. Cuadrado
3. Rectángulo
4. Generación libre
5. Modo de graficación

Para seleccionar cada uno de los módulos, sólo basta teclear su número correspondiente. A continuación se explica a detalle cada módulo del generador.

3.2 INICIALIZA GRAFICAS.

Esta opción debe escogerse cada vez que se desee comenzar a crear una nueva imagen. Su finalidad es la de borrar la información existente dentro de la memoria de la computadora asignada para alojar la gráfica, lo que a su vez se traduce en la inicialización de la pantalla.

3.3 MACRODEFINICIONES

Se nombró "macrodefiniciones" a las opciones 2 y 3 del menú del generador ya que ambas permiten la generación directa de cuadrados y rectángulos sin tenerlos que definir lado por lado, esto le da al usuario una gran facilidad en la creación de imágenes compuestas de combinaciones de estas figuras.

Para la generación del cuadrado es necesario que el usuario indique la posición del vértice inferior izquierdo del mismo, así como el tamaño de uno de sus lados dando el valor en número de "pixels". Definidos los datos anteriores, la pantalla cambiará al modo de graficación mostrando una cruz en la posición que ocupará el vértice inferior izquierdo de la figura, en caso de no ser ésta la posición deseada, ésta se puede variar oprimiendo las siguientes teclas:

E - avanzar un pixel hacia arriba

X - avanzar un pixel hacia abajo

S - avanzar un pixel a la izquierda

D - avanzar un pixel a la derecha

RETURN - dibujar la figura

En lo que se refiere al rectángulo, del usuario se requiere la posición del vértice inferior izquierdo y las dimensiones de la base y altura, aplicándose los mismos pasos necesarios para el cuadrado en cuanto a modificación de posición del vértice inferior izquierdo en el modo de graficación.

Las macrodefiniciones agilizan la creación de gráficas y fueron diseñadas para dar una mayor comodidad y facilidad en el uso del generador.

3.4 CREACION LIBRE DE FIGURAS.

El sistema permite al usuario crear cualquier tipo de figuras a base de líneas rectas, presentando así una gran versatilidad para generar una amplia gama de imágenes.

Para la generación libre de figuras se requiere, como primer paso, que se indiquen las coordenadas de un punto. Una vez definido éste, se pasa al modo de graficación donde al igual que en el cuadrado y rectángulo aparecerá una cruz en el punto inicial. Esta posición se puede modificar con los movimientos descritos en el punto anterior. En el momento de oprimir RETURN, queda defini-

da la primera posición y resta el mover la cruz al segundo punto deseado y oprimir RETURN para que la recta sea dibujada en la pantalla. A continuación se define otro punto y el proceso se continúa hasta que la figura deseada se complete.

De esta manera no se restringe la generación de figuras a cuadrados y rectángulos exclusivamente, ya que SIREI es un sistema diseñado para manejar gran variedad de figuras. Para esto, se puede usar la opción de generación libre de figuras o definir el procedimiento que dibuja esta figura e incorporarla posteriormente al menú como una macrodefinición más. Lo único necesario para que estas nuevas figuras puedan ser reconocidas por SIREI, es que la representación de las mismas (Reglas de Producción de Figuras) se incluya por medio del Cargador de Figuras Básicas al sistema.

3.5 MODO DE GRAFICACION.

Durante el proceso de generación de figuras, la pantalla pasa alternativamente por el modo texto y el modo de graficación. El modo texto se emplea para presentar el menú y pedir los datos de coordenadas y dimensiones, mientras que en el modo de graficación se presenta la cruz y se dibujan las figuras deseadas. El sistema realiza los cambios entre estos modos automáticamente, dejando siempre el modo texto para presentar el menú y permitir al usuario seguir dibujando. La opción 5, Modo de Graficación, permite al usuario observar la FC que está siendo generada, con lo cual puede ver si cumple o no con las características deseadas. Para continuar con la etapa de generación basta con teclear RETURN para que se regrese al menú de opciones del generador.

Después de haber generado una imagen empleando las facilidades de este módulo, se procede a su reconocimiento según se explica en el capítulo siguiente.

Cabe señalar que esta información podría ser introducida a SIREI por otros medios como podría ser un digitalizador conectado a una cámara de TV.

4. RECONOCEDOR

4.1 INTRODUCCION

La finalidad de este módulo es la de reconocer Figuras Compuestas (FCs), generadas mediante el módulo descrito en el capítulo anterior. El algoritmo utilizado emplea un reconocimiento por etapas, esto es, primero se reconocen una a una las Figuras Básicas (FBs) que forman la imagen a reconocer, de manera sistemática siguiendo un orden preestablecido, y una vez identificadas éstas, tomando en cuenta su posición relativa en la imagen, se pasa a identificar la FC que forman. En caso de que ésta no exista en la base de conocimiento del sistema, el módulo así lo indica y si el usuario lo desea, la aprende formando parte de su acervo de FCs.

Las entradas a este módulo son :

- La información de la imagen que se desea reconocer.
- La información de todas aquellas FBs existentes hasta ese momento.
- La información de las FCs aprendidas.

Después de ejecutarse los distintos procesos del módulo, se obtiene como salida lo siguiente :

- La Representación de la Figura Compuesta Generada (RFCG), mediante una cadena de caracteres.
- La representación de la FC que, dentro de las aprendidas, corresponde a la anterior (si éste es el caso), la cual también es una cadena de caracteres.

- El aviso de que la FCG no se encuentra dentro de la información aprendida y la posición que su representación debe tener dentro de dicha información, para que posteriormente, si el usuario lo desea, pueda aprenderse.

La representación de estas FCs se va modificando a lo largo del proceso de reconocimiento, hasta que al final de éste, se tiene su representación en una estructura de datos predefinida para contener la información de las FCs aprendidas, lo cual permita compararla con ellas y determinar si se reconoce o no.

En este capítulo, para facilitar la descripción de este módulo, se muestra la figura 4.1, que contiene las tres distintas etapas que lo forman, así como la relación que guardan entre sí. Estas etapas son: generación del medio ambiente, armado de la RFCG y reconocimiento de la FCG. En la primera etapa se crean las estructuras de datos que se utilizan durante el armado de la representación de la FCG y además localiza el primer punto iluminado de la pantalla que sirve de base para el reconocimiento de la figura; la segunda etapa reconoce una a una las figuras que forman la imagen y al mismo tiempo genera la RFCG; en la tercera etapa, se busca dentro de la estructura de datos de las FCs aprendidas la representación recién generada, en espera de poder reconocerla.

FIGURA 4.1

MODULO RECONOCEDOR

Una vez explicados a grandes rasgos la finalidad, los procesos y la información que maneja el módulo, se pasará a definir detalladamente cada una de las etapas. En la sección 4.2 se habla de las distintas estructuras de datos que se utilizarán a lo largo del módulo; en la 4.3 se explican a detalle las tres etapas que lo constituyen, mientras que en la 4.4 se mencionan las distintas restricciones que existen, tanto en la información que puede manejar el sistema, como en la implementación del mismo.

4.2 REPRESENTACION DE LA IMAGEN

Una de las partes fundamentales dentro del sistema, y en especial dentro de este módulo, es la representación interna del contenido de la imagen desplegada en la pantalla. Esta será procesada paulatinamente por las diferentes etapas del módulo, hasta llegar al resultado deseado (la RFCG). La finalidad de esta sección es explicar las transformaciones que la representación de la imagen va sufriendo desde que entra al módulo hasta que sale de él.

Para poder reconocer las distintas FBs (figuras geométricas simples) que se encuentran dentro de la pantalla, se tiene que accesar continuamente la información de la imagen de la pantalla. La manera en que ésta información es manejada por el sistema operativo es compleja. Al principio del módulo se vacía dicha información en una matriz booleana, que es utilizada para consultar la información contenida en la pantalla durante todos los procesos del reconocedor.

Conforme avanza el proceso de reconocimiento, se va formando la RFCG. En ella se va agregando la información correspondiente a cada una de las FBs reconocidas, así como la relación de posición que guardan entre sí. Esta RFCG es la representación de la información de la imagen que se desea reconocer. Para poder ir armándola, el reconocedor utiliza la información de las FBs contenida dentro de la base de conocimiento denominada "Base de Conocimiento de las Figuras Básicas" (BCFB), y así, identificando una a una las FBs que conforman la imagen se irá formando la RFCG.

Cuando la construcción de la RFCG se ha completado, ésta se compara con las representaciones de las FCs que han sido aprendidas anteriormente, tratando de determinar su existencia en el acervo de SIREI (lo que viene siendo el reconocimiento). La información de las FCs aprendidas está contenida en la base de conocimiento denominada "Base de Conocimiento de Figuras Compuestas" (BCFC).

A continuación se hará una descripción precisa de cada una de las estructuras de datos utilizadas para manejar la información y de aquéllas usadas durante la comparación que determina el reconocimiento.

4.2.1 ORGANIZACION DE LA INFORMACION.

4.2.1.1 PANTALLA.

Es la primera estructura que guarda la información de la FC a reconocer y es creada por el sistema operativo con la información proporcionada por el usuario durante la etapa

de generación. Se encuentra definida en la memoria de la computadora comenzando en la localidad 2000H para la página cero y 4000H para la uno. La manera como es accesada por el sistema operativo es bastante compleja, lo que determina que se redefina su información dentro de la matriz gráfica (MG) -siguiente estructura a definir- ya que la parte del módulo que reconoce las FBs, continuamente hace referencia a esta información.

4.2.1.2 MATRIZ GRAFICA

El proceso de almacenar la información de la pantalla en la MG representa la interfaz entre el módulo generador y el reconocedor. Este consulta la información contenida en esta estructura para la identificación de las FBs que forman la imagen a reconocer.

Dentro del diseño del sistema fue necesario crear una rutina que efectúe un mapeo de las localidades de la memoria que contienen la información de la gráfica y pase su contenido a una estructura que facilite su manejo. La Apple IIe (sistema utilizado para la implementación de SIREI) almacena la información de las gráficas desplegadas en la pantalla en localidades predeterminadas de la memoria, y aunque es posible accesar los contenidos de dichas localidades desde un programa escrito en Pascal -por lo que parecería lógico tomar de ahí toda la información de las gráficas-, en la práctica, por lo complejo del manejo de dichas localidades

por parte del sistema operativo y por la gran interacción que el reconocedor requiere con dicha información, se optó por crear una matriz booleana que contenga la información del estado de los "pixels" de la pantalla y que, además, la tenga ordenada de manera que facilite su consulta, con lo que se reduce considerablemente el tiempo de procesamiento de la etapa de armado de la RFCG ya que no es necesario el mapeo que se requiere cuando la información se toma directamente de la pantalla.

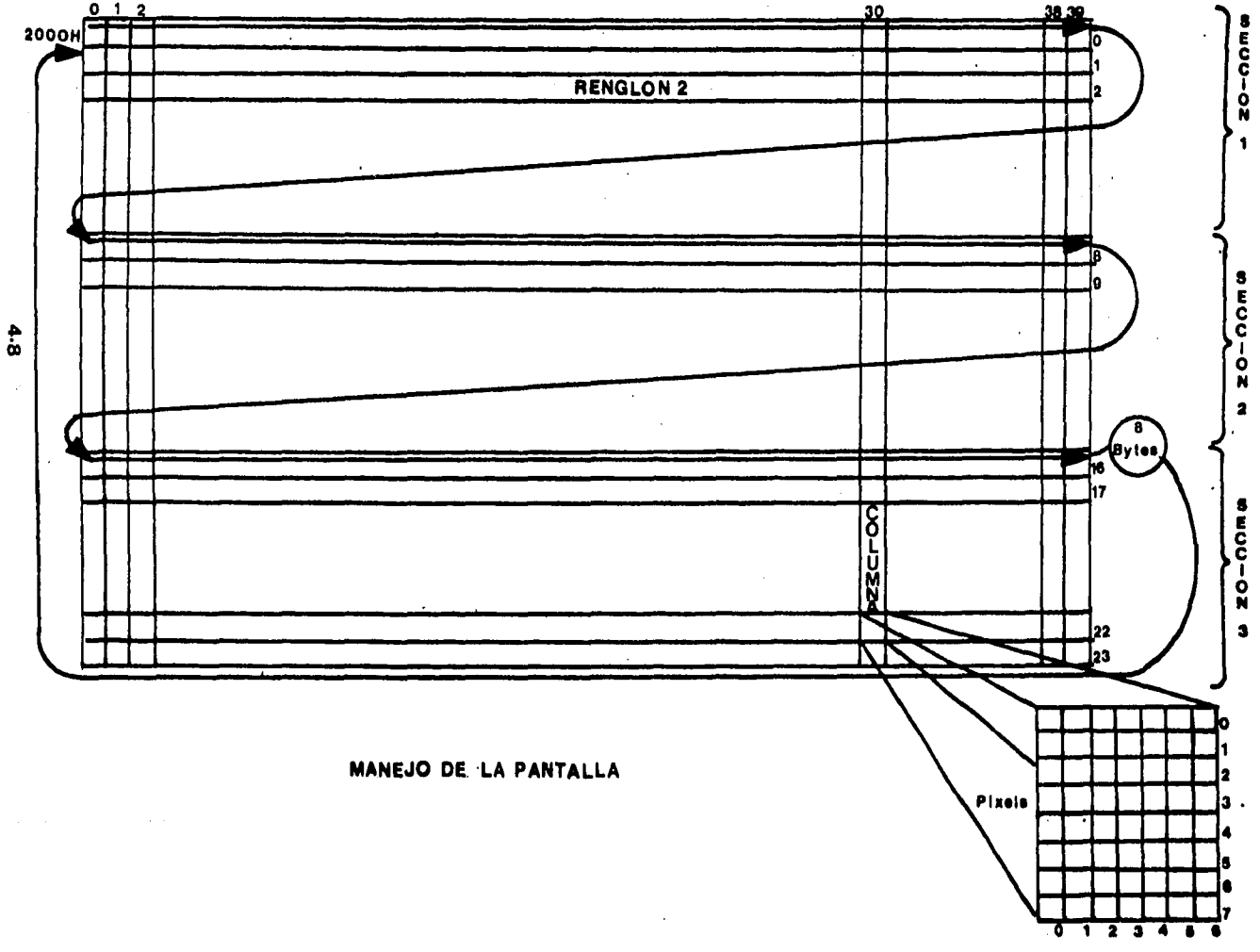
La MG tiene las mismas dimensiones de la pantalla, es decir, de 192 renglones por 280 columnas. Se emplea un sólo bit para representar cada "pixel" por lo cual el tamaño de dicha matriz es igual a :

192	x	280	=	53,760 bits ó
53,760	/	8	=	6,720 Bytes

es decir, menos de 7KB de memoria.

En la figura 4.2 se muestra gráficamente la manera como el sistema operativo guarda esta información en la memoria. El mecanismo empleado es el siguiente: la pantalla se divide en tres secciones de ocho renglones cada una, donde cada renglón consta de ocho líneas de 280 "pixels" (esto nos da 192 líneas de "pixels"). El sistema operativo toma la primera línea del primer renglón de cada una de las tres secciones y la guarda en forma consecutiva ocupando 128

FIGURA 4.2



MANEJO DE LA PANTALLA

bytes (cada 7 "pixels" son almacenados en un byte, por lo que una línea de 280 "pixels" ocupa 280 / 7 = 40 bytes, y al guardar las tres líneas mencionadas se ocupan 120 bytes; los 8 bytes restantes son usados por el sistema operativo). A continuación toma la primera línea del segundo renglón de cada sección y la almacena en los siguientes 128 bytes. Este proceso se repite hasta completar los ocho renglones de cada sección. En este punto la primera línea de los ocho renglones de las tres secciones ha sido almacenada (es decir, se han guardado 24 líneas de 280 "pixels" en 1024 bytes). Este proceso se repite para las siete líneas restantes. Consultar la referencia [15] para mayor detalle.

Aunque la MG ocupa cerca de 7 KB extra de memoria, su uso se justifica al considerar el gran ahorro en tiempo de proceso que se elimina al evitar la continua utilización de una función de mapeo, así como el ahorro en líneas de programación.

4.2.1.3 INFORMACION DE LAS FIGURAS BASICAS.

Como se mencionó anteriormente, las FBs son los elementos primarios del reconocedor y son éstas las que al combinarse dan como resultado las distintas FCs que se tratarán de identificar. El propósito de esta sección es el definir las distintas características que poseen las FBs, la información que contienen y la manera como se representan en la computadora.

La información que define a cada FB al igual que sus características está contenida dentro de dos grandes grupos de información : las Reglas de Producción (RPs) y las Funciones (FNs) [3 vol. 3 pags. 287 a 291, 5, 6].

Una RP es un mecanismo mediante el cual se definen las características de la forma de la FB. Las RPs de las distintas FBs se definen de la siguiente manera :

<LADO IZQ> ::= <LADO DER>

donde "LADO IZQ" es el identificador de la FB y "LADO DER" es un conjunto de "MODIFICADORES","ITEMS" y "OPERADORES".

Los "ITEMS" son los elementos primarios (o lados) de las FBs. Dentro de SIREI los posibles items son :

- Línea Vertical, denotada como <V>.
- Línea Horizontal, denotada como <H>.
- Línea Diagonal Derecha, denotada como <D>.
- Línea Diagonal Izquierda, denotada como <I>.

Cada item tiene una "cabeza" y una "cola" que indican el sentido en el que debe recorrerse (de la cola a la cabeza). Para el item <V>, el sentido de recorrido es de abajo hacia arriba, es decir, la cola está en el extremo inferior de la línea y la cabeza en el superior. Para el item <H> el

sentido de recorrido es de izquierda a derecha. Para las diagonales (items <D> e <I>) el sentido de recorrido es del extremo inferior al superior.

Un "MODIFICADOR" es un elemento que altera las características del recorrido de los items. SIREI posee dos modificadores: el modificador de cambio de sentido de recorrido denotado como "@" y el modificador nulo denotado como " ".

El modificador "@" invierte el sentido de recorrido, es decir, pone la cabeza como la cola y viceversa, mientras que el operador nulo deja sin alterar el sentido de recorrido del item.

La combinación <MODIFICADOR><ITEM> se denomina <SUJETO>.

En la tabla 4.1 se observan los distintos modificadores e items que posee SIREI.

Un "OPERADOR" es un elemento que sirve para relacionar dos sujetos. Por el momento el único operador definido en SIREI es la "concatenación", denotada como "+".

De acuerdo con lo expuesto anteriormente, una RP tiene la siguiente estructura:

<FB> ::= <SUJ><OP><SUJ><OP>...<SUJ><OP>

TABLA 4.1

Sujetos de SIREI

ITEM	REPR.	MOD.	SUJETO	REPR.
V		" "	V	
		"@"	@V	
H		" "	H	
		"@"	@H	
D		" "	D	
		"@"	@D	
I		" "	I	
		"@"	@I	

Toda RP es cíclica, esto es, presupone que el punto final del recorrido es el mismo que el punto inicial.

Otra característica importante de las RPs de las FBs es su sentido de recorrido; es decir, que para recorrer cualquier FB se seguirá siempre el mismo sentido, el cual es contrario a las manecillas del reloj. Como no existen restricciones en cuanto a cuál es el punto inicial en el recorrido de una FB, las RPs son circulares, esto es, el recorrido de ellas puede iniciarse en cualquiera de sus sujetos.

Aunque SIREI fue diseñado para manejar cualquier FB que se le defina mediante el Cargador de FBs (consultar el

capítulo 6), al inicio del sistema, SIREI tiene definidas como FBs al cuadrado y al rectángulo, cuyas RPs son :

CUADRADO::= H+ V+@H+@V

RECTANGULO::= H+ V+@H+@V

Como se observa, la RP no discrimina totalmente a una FB de otra ya que como el caso del cuadrado y del rectángulo, puede haber más de dos FBs con la misma RP, lo cual indica que es necesario un segundo criterio que además de definir a mayor detalle la FB, sirva para discriminar. a distintas FBs que posean igual RP. Este criterio adicional se denominó Función (FN) y lo que hace es validar que algunos sujetos de la RP posean igual longitud o, para otros casos, que las dimensiones entre dos sujetos sean diferentes, esto es que, para poder diferenciar las FBs cuando hay más de una con igual RP, se valida que tanto para los sujetos que se encuentrán dentro de la parte de igualdades de la FN como para los que se encuentran dentro de la parte de desigualdades se cumpla esta característica.

La FN asignada a las dos FBs iniciales es :

Para el cuadrado :

Long SUJ1 (H) = Long SUJ2 (V)

Long SUJ1 (H) = Long SUJ3 (@H)

Long SUJ1 (H) = Long SUJ4 (@V)

Para el rectángulo :

Long SUJ1 (H) = Long SUJ3 (@H)

Long SUJ2 (V) = Long SUJ4 (@V)

Long SUJ1 (H) <>Long SUJ2 (V)

de lo cual se puede observar que con este segundo criterio de diferenciación ya es posible identificar a cada una de las FBs que posean igual RP. Además, esta información describe algunas otras características de las FBs.

Como características de la FN se tiene que :

- Se deben considerar los sujetos tal como se definió la RP (esto es el sujeto inicial de la RP para la FN siempre es el primer sujeto definido al crear la regla).

- Las relaciones de igualdad y/o desigualdad entre sujetos de la RP siempre son por pares de lados.

- Para que la FN se acepte como correcta se deben haber cumplido todas las igualdades y desigualdades especificadas dentro de ésta.

El reconocedor de SIREI continuamente hace referencia a esta información durante la etapa de armado de la RFCG por lo que aunque cuente con la BCFB (residente en disco), al

principio del módulo se carga a memoria esta información para agilizar esta etapa del reconocedor. En la figura 4.3 se muestra la estructura de la información en la memoria. De ahí se observa que existe un vector de entrada, el cual permite el acceso a la información de las FBs.

Cada uno de los elementos del vector es un apuntador a una de las reglas de producción de las FBs, la cual posee dos tipos de registros: registro sujeto, el cual posee un sujeto y un apuntador al siguiente operador de la regla; y un registro operador, que contiene, tanto un operador, como un apuntador al siguiente sujeto de la regla.

Las principales características de las RPs son :

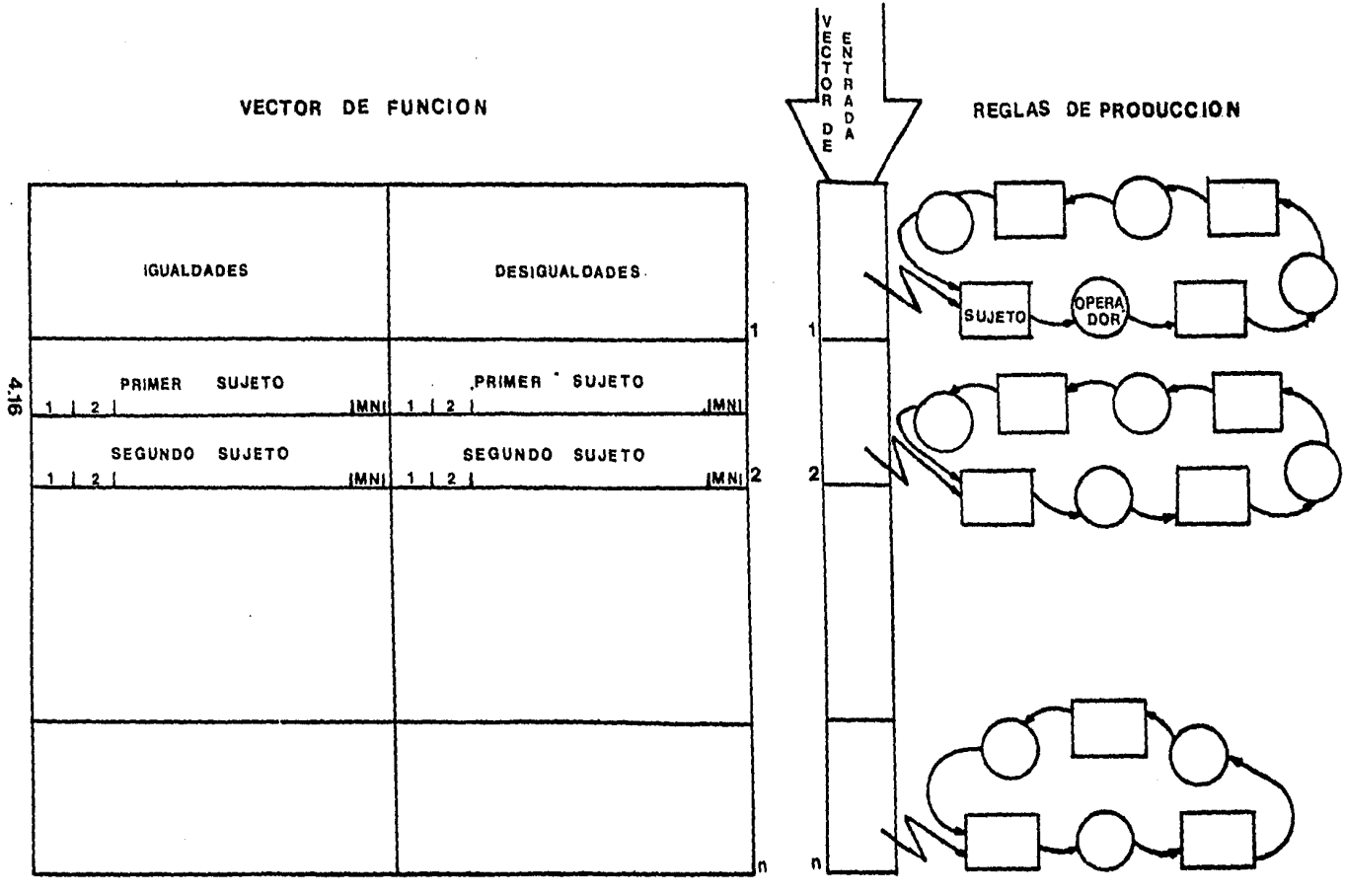
- Todas las RPs son circulares, con un sentido único de recorrido (contrario al de las manecillas del reloj).

- Aunque poseen una sóla entrada, una vez dentro de la RP se puede efectuar el recorrido de ésta partiendo de cualquiera de sus sujetos.

- Al hacer la implementación mediante apuntadores, el tamaño de las reglas es dinámico y depende únicamente del número de sujetos que posea la FB en cuestión.

FIGURA 4.3

INFORMACION DE LAS FIGURAS BASICAS EN MEMORIA



- Como se definió un sentido de recorrido único, las colas que forman a las RP poseen una sóla liga, la cual obliga el sentido de recorrido de éstas.

Como se mencionó anteriormente y como puede observarse en la misma figura 4.3, además de las RPs cada FB tiene una FN la cual está definida dentro del vector de función de igual dimensión al vector de entrada ya que a cada uno de los elementos de uno de ellos corresponde un elemento del otro. El contenido de este vector de función puede a su vez dividirse en dos grupos de información. En el primer grupo se encuentran, en parejas ordenadas, todos aquellos sujetos de la RP de la FB que deben poseer igual longitud, mientras que en el segundo grupo de parejas ordenadas, se encuentran todos aquellos sujetos que deben poseer longitudes diferentes.

4.2.1.4 REPRESENTACION DE LA FIGURA COMPUESTA GENERADA.

Esta estructura contiene la representación de todas las FBs que fueron reconocidas durante la primera parte del módulo, así como las relaciones existentes entre ellas. Su armado es por partes, esto es, por cada FB reconocida, y queda completamente definida cuando se finaliza el barrido de la MG que posee la información de la pantalla.

La representación de una figura compuesta se realiza mediante una regla de producción que posee la siguiente estructura:

<FC> ::= <FB>[OP...OP]<FB>[OP...OP]...<*>

Donde

<FC> - figura compuesta

<FB> - figura básica

<OP> - operador

<*> - operador asterisco

<FB> es una letra de la "A" a la "Z", que corresponde al lado izquierdo (identificador) de alguna regla de producción de una figura básica definida.

<OP> puede ser uno de los dos siguientes operadores de figuras compuestas:

"/" que es el operador "fin de sujeto".

"*" que es el operador "fin de regla de producción de figura básica".

Con el objeto de clarificar el uso de esta representación, en la figura 4.4 de la siguiente página se encuentra un ejemplo de una figura compuesta formada por dos rectángulos y un cuadrado, así como su regla de producción.

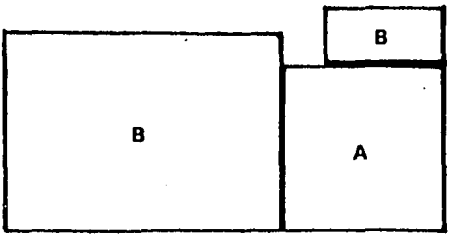


FIGURA 4.4

Ejemplo de una Figura Compuesta

La regla de producción correspondiente es:

<FC> ::= B/A//B***

donde :

"B" significa que la primera figura básica (un rectángulo).

"/" indica que en el primer sujeto (lado base) de B no hay figuras básicas adyacentes.

"A" nos indica que un cuadrado se encuentra adyacente al siguiente (segundo) lado de "B", recorriendo la figura en sentido contrario a las manecillas del reloj.

"/" indica que en el primer lado de "A" no hay figuras básicas adyacentes.

"/" indica que en el segundo lado de "A" no hay figuras básicas adyacentes.

"B" significa que hay un rectángulo adyacente al siguiente (tercer) lado de "A".

"*" indica que no hay figuras adyacentes a "B".

"*" indica que ya no hay más figuras adyacentes a "A".

"*" indica que ya no hay más figuras adyacentes al primer rectángulo (la primera "B"), por lo que indica también el fin de la figura compuesta.

La construcción de la RFCG se efectúa a través de una cadena de caracteres de dimensión máxima igual a 32, la cual contiene a los identificadores de las FBs que la forman, así como los caracteres que definen a los operadores de relación entre las mismas. Las principales razones por las que se escogió este tamaño para la implementación son :

- Esta dimensión de las RPs permite el utilizar FCs formadas de entre seis y quince FBs, lo cual permite una amplia gama de figuras a considerar.

- La tasa de transmisión de entrada-salida de la Apple IIe es de 512 bytes / acceso, por lo que es conveniente que este número fuera el múltiplo del tamaño de las RPs de las FCs, puesto que facilita este proceso de transmisión.

Además, en caso de necesitarse RPs de dimensión mayor, es sumamente sencillo aumentar el tamaño definido, pues bastará asignar a la constante que define esta dimensión el valor deseado y compilar los módulos respectivos (Reconocedor, Aprendizaje y Utilerías).

Al finalizar este proceso, la RFCG se busca dentro de la BCFC para saber si ya es conocida. En caso de no serlo, el usuario puede darla de alta por medio del módulo de aprendizaje.

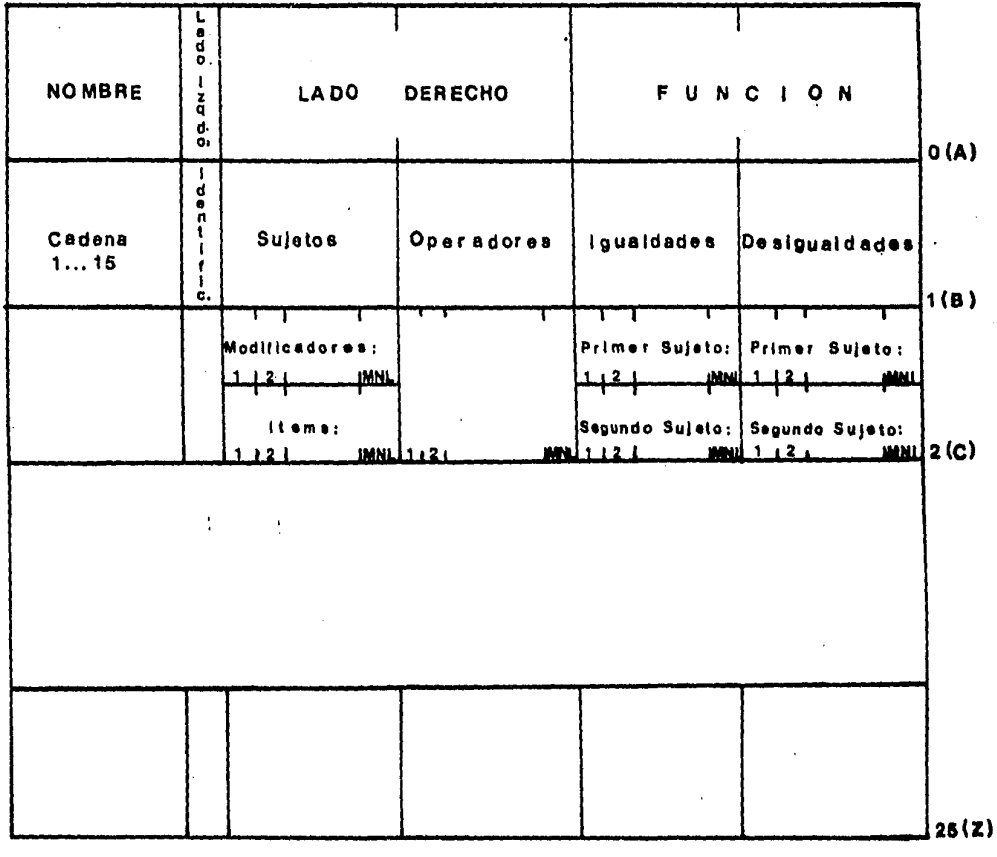
4.2.2 BASES DE CONOCIMIENTO

4.2.2.1 BASE DE CONOCIMIENTO DE LAS FIGURAS BASICAS

Esta BCFB posee a todas aquellas FBs que pueden ser identificadas por SIREI. En la figura 4.5 se puede apreciar gráficamente la estructura de esta BCFB, la cual reside en disco bajo el nombre de "BAS_MAST.DATA". Esta base de conocimiento se encuentra dentro de un archivo de acceso aleatorio que posee la siguiente estructura: el primer campo denominado "nombre" contiene precisamente el nombre con que se conoce a la FB en cuestión, el cual puede ser una cadena de hasta 15 caracteres. El siguiente campo es el identificador de dicha figura, también denominado "lado izquierdo de la regla de producción", el cual consta de un caracter alfabético (de la "A" a la "Z"). Los siguientes dos campos corresponden al lado derecho de la regla; el primero de ellos contiene a los sujetos que ésta posee (modificadores y items), mientras que el segundo posee a los operadores que relacionan a dichos sujetos. Los dos últimos campos corresponden a la función de la FB; el primero contiene todas las igualdades y el segundo las desigualdades propias de la misma.

FIGURA 4.5

BASE DE CONOCIMIENTO DE FIGURAS BASICAS



La alimentación a ésta BCFB la efectúa el programa cargador de FBs que es una de las utilerías del sistema (consultar el capitulo 6). En un principio se alimenta esta base de conocimiento con la información del cuadrado y del rectángulo.

Como puede observarse, en esta BCFB se encuentra toda la información que define a las FBs (para ver cuál es esta información consultar la sección 4.2.1.3). De aquí es de donde la etapa de generación del medio ambiente del módulo la toma para cargarla en memoria principal.

4.2.2.2 BASE DE CONOCIMIENTO DE FIGURAS COMPUESTAS

La base de conocimiento de figuras compuestas consiste en un archivo secuencial de acceso aleatorio, donde cada registro almacena la representación de una figura compuesta. Estas representaciones se almacenan siguiendo un orden alfabético ascendente. Al ser este archivo de acceso aleatorio permite efectuar búsquedas binarias, que es la manera como el Reconocedor decide si la representación de una figura compuesta se encuentra en esta base de conocimiento o no. Cada una de estas figuras compuestas está representada por una cadena de caracteres que forma una regla de producción según se expuso en el punto 4.2.1.4.

La BCFC reside en disco bajo el nombre de "A_FC.DATA". Su alimentación se efectúa a través del módulo de aprendi-

zaje bajo la autorización del usuario. En un principio, la BCFC se encuentra vacía. Cada una de las representaciones de las FCs aprendidas posee la misma estructura que la definida para la RFCG, ya que esto es necesario si se desea compararlas con ella para poder decidir si ya fue aprendida o no. En la figura 4.6 se muestra gráficamente esta estructura.

Cabe mencionar que el primer registro de esta lista residente en disco posee el número de FCs aprendidas denotado por NAFC, ya que esta estructura es dinámica.

4.2.3 ESTRUCTURAS AUXILIARES.

Además de la información de las FBs y de las FCs, SIREI requiere de información adicional que le permita generar la RFCG, la identificación de ésta y su aprendizaje en caso necesario. Para ello, se crean -durante el proceso de reconocimiento de las FBs- dos estructuras de información adicionales, las cuales son descritas a continuación:

4.2.3.1 INFORMACION TEMPORAL.

Esta información se almacena en una estructura que contiene los sujetos de la RP correspondiente a la FB que está siendo analizada, así como los puntos donde comienzan cada uno de ellos. Se omite la información de dónde termina cada sujeto pues dicho punto es el mismo que el punto inicial del siguiente sujeto. Esto puede observarse en la figura 4.7. En la misma, la información que contien puede agruparse en 2 grandes conjuntos: en el primero se encuen

FIGURA 4.6

BASE DE CONOCIMIENTO DE F.C.

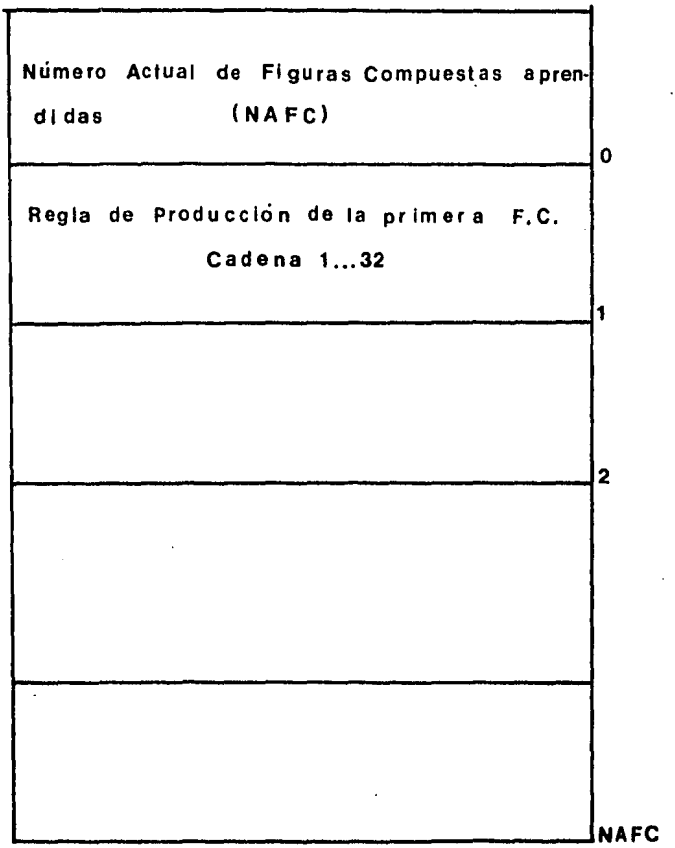
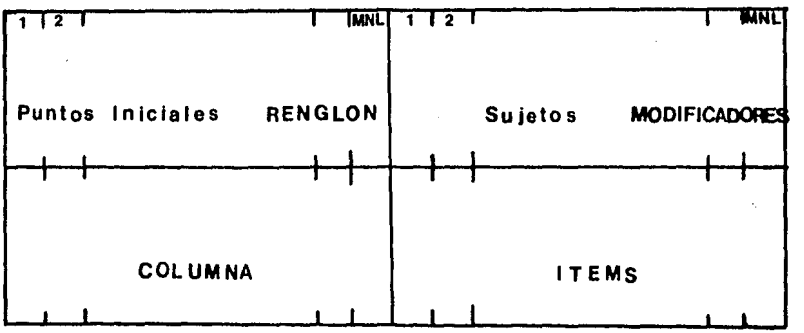


FIGURA 4.7

INFORMACION TEMPORAL :



tran almacenados los puntos iniciales de cada uno de los sujetos que forman la RP de la FB actual (la posición del renglón y la de la columna que forman a dichos puntos), mientras que en el segundo están estos sujetos (sus modificadores y items). Esta estructura es necesaria porque en ella se encuentra la posición dentro de la pantalla de la FB actual y es fundamental para poder efectuar el barrido de las inmediaciones de ella en espera de encontrar más puntos iluminados (consultar la siguiente sección donde se explica este proceso).

Una vez que se ha terminado de reconocer la FB, esta información está completa.

Por cada FB encontrada se tiene una de estas estructuras, la cual permanece activa hasta el fin del barrido de las inmediaciones de dicha FB, de ahí que se le haya denominado temporal. Como esta estructura contiene la posición de la FB actual dentro de la pantalla, el módulo reconocedor la consulta para poder accesar los puntos adyacentes a los que forman cada uno de sus sujetos y considerar aquellos que estén iluminados en el reconocimiento de nuevas FBs adyacentes. Al finalizar la consulta de estos puntos, esta información ya no es necesaria, por lo que puede eliminarse, debido a las características de este proceso (recursividad), esto ocurre automáticamente.

4.2.3.2 PARAMETROS PARA EL APRENDIZAJE

Cada vez que la parte de identificación de la FCG indique que la figura en análisis no ha sido aprendida anteriormente, SIREI permite que se aprenda, esto es que el usuario decida si se agrega su representación a la BCFC. Para agilizar el proceso de aprendizaje, el módulo reconocedor le pasa al módulo de aprendizaje, la RFCG a aprender y la posición que ésta debe ocupar dentro de la BCFC, con lo cual este último únicamente genera el espacio en el lugar indicado y almacena la representación de la FC. El proceso así llevado, es sumamente rápido pues no requiere la búsqueda del lugar que le corresponde a la RFCG dentro de la BCFC. Esta información, la RFCG y la posición que a ésta le corresponde dentro de la BCFC, se pasa de un módulo al otro a través de un archivo de parámetros denominado " A_REGPAR.DATA ".

4.3 ETAPAS DEL RECONOCEDOR.

El módulo reconocedor es el más importante y complejo de SIREI ya que en él es donde se analiza la imagen generada por el usuario y consultando las bases de conocimiento que se poseen se puede identificar la RFCG, indicando si ésta ya ha sido aprendida con anterioridad.

Para poder realizar esto, el reconocedor toma como información de entrada la imagen generada que se encuentra en la pantalla y las bases de conocimiento existentes. Esta imagen se

almacena en memoria por el sistema operativo de tal forma que hace difícil su manejo, como ya se describió anteriormente, por lo cual se reordena la información en nuevas estructuras de manejo más ágil y sencillo, y después, se identifican las FBs contenidas en la pantalla lo que da como resultado la RFCG. Adicionalmente, se indica si ésta existe o no en la BCFC y su posición correspondiente dentro de la misma.

Las distintas etapas en las que se puede dividir al módulo son tres : generación del medio ambiente, armado de la RFCG y reconocimiento de la FCG. En los siguientes incisos se detallará cada una de estas etapas, poniendo principal interés en su finalidad, la manera como dividen el trabajo, la información que manejan y los resultados en ellas obtenidos.

4.3.1 GENERACION DEL MEDIO AMBIENTE.

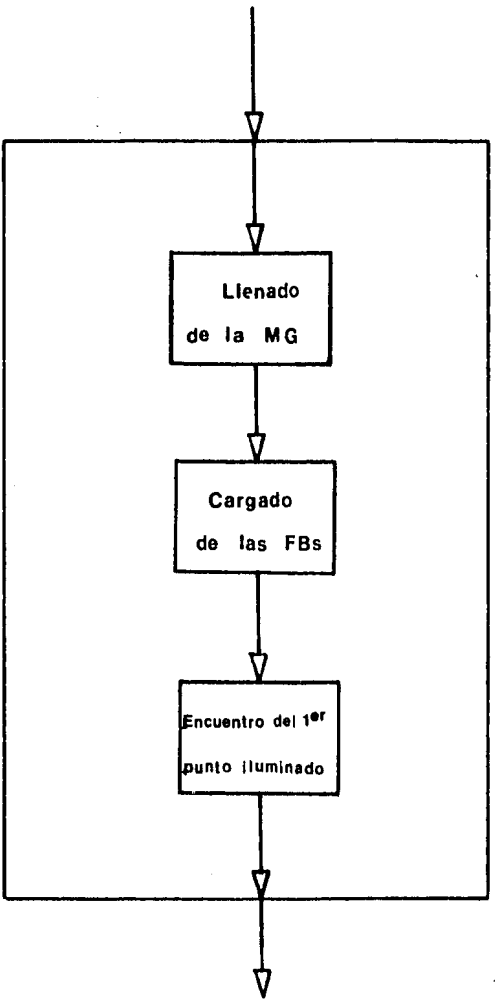
La finalidad de esta etapa es la creación de las estructuras necesarias para el procesamiento de la imagen generada en espera de crear la RFCG así como su identificación. Esta etapa cuenta con tres procesos principales: generación de la matriz gráfica, cargado de la información de las FBs y posicionamiento en el primer punto iluminado de la MG, los cuales se detallan a continuación y aparecen gráficamente descritos en la figura 4.8.

4.3.1.1 GENERACION DE LA MATRIZ GRAFICA.

Este proceso accesa la información de la pantalla, contenida en la memoria principal de la computadora a partir de la localidad 2000H, y la vacía en la MG, la cual será

FIGURA 4.8

ETAPA: Generación del medio ambiente



consultada por los demás procesos del módulo, para poder crear la RFCG. Esta MG es una matriz de 192 renglones por 280 columnas de elementos booleanos (falso o verdadero), cada uno de los cuales corresponde a un "pixel" de la pantalla.

El procedimiento empleado consiste en tomar cada una de las líneas de "pixels" y colocarlas dentro de la MG, cambiando cada "pixel" por el valor booleano correspondiente - verdadero si está prendido o falso si está apagado - (consultar sección 4.2).

4.3.1.2 CARGADO DE LA INFORMACION DE LAS FIGURAS BASICAS.

Durante la ejecución de esta rutina, el módulo reconocedor de SIREI accesa la información de las BCFB y la guarda en la memoria principal dentro de una estructura definida para este fin (consultar la figura 4.3 anteriormente descrita). Con esto, la identificación de las FBs es más rápida ya que tendrá en memoria la información, eliminando la necesidad de recurrir a la BCFB residente en disco. Este cargado de la BCFB a memoria es conveniente y agiliza el acceso a esta información si el número de FBs no es excesivo. Se considera que hasta para diez FBs esta implementación es conveniente, ya que para esta cantidad la memoria utilizada es aproximadamente un Kbyte. Si se desea implementar el sistema en algún otro equipo, el número de FBs adecuado dependerá de las características que éste posea.

Este proceso se implementó de la siguiente manera: primero se lleva a cabo la inicialización, tanto del vector de apuntadores de entrada, como del vector de la función; segundo, se almacena en memoria principal la información de las FBs existentes en la BCFB una a una, esto es, primero se crea la RP de la FB actual; segundo se carga dentro del vector correspondiente el apuntador a dicha regla y finalmente se carga la FN de la FB dentro del vector respectivo. La segunda parte del proceso, el almacenamiento en memoria principal de las FBs existentes en la BCFB se repite tantas veces como FBs estén contenidas en la BCFB.

4.3.1.3 POSICIONAMIENTO EN EL PRIMER PUNTO ILUMINADO DE LA MATRIZ GRAFICA.

Una vez que la MG ha sido generada y que se tiene en memoria la información de las FBs existentes, la etapa de generación del medio ambiente barre la MG en espera de encontrar un punto iluminado. Este punto será el de partida para la etapa de armado de la FCG.

Este último proceso de esta etapa del reconocedor se implementó de la siguiente manera. El proceso barre por renglones la MG, comenzando por el renglón inferior (0), y finalizando en el superior (191). Dentro de cada renglón el barrido se efectúa de izquierda a derecha, es decir, se inicia con la columna cero y se finaliza con la 279.

Aunque no se asegura que este recorrido es el más rápido para encontrar un punto iluminado de la pantalla, se escogió porque sí asegura que el primer punto iluminado que encuentra equivale al vértice inferior de la primera FB, que será considerada como la base de la FCG; pues el método asegura que no existe hacia abajo del sujeto de este vértice de la FB algún otro punto iluminado. Con este dato, la información de la imagen a procesar, contenida en la MG, y la información de las FBs disponibles, SIREI pasa a generar la RFCG.

4.3.2 ARMADO DE LA RFCG.

En esta etapa, se irán identificando una a una las FBs existentes en la MG y al mismo tiempo se irán colocando dentro de la RFCG, hasta finalizar el barrido de la MG, con lo cual quedará armada esta RFCG. La manera como ésta se genera se muestra en el diagrama de flujo de la figura 4.9 y su explicación a detalle se describe a continuación.

4.3.2.1 IDENTIFICACION DEL TIPO DE VERTICE.

Este proceso se encarga de definir los sujetos que llegan al punto definido como vértice, ya que de ellos partirá el recorrido de la FB que se desea identificar. Los posibles vértices se muestran en la tabla 4.2 , la cual muestra en cada renglón, los dos sujetos que forman a cada una de ellos.

FIGURA 4.9

ETAPA: ARMADO DE LA R F C G

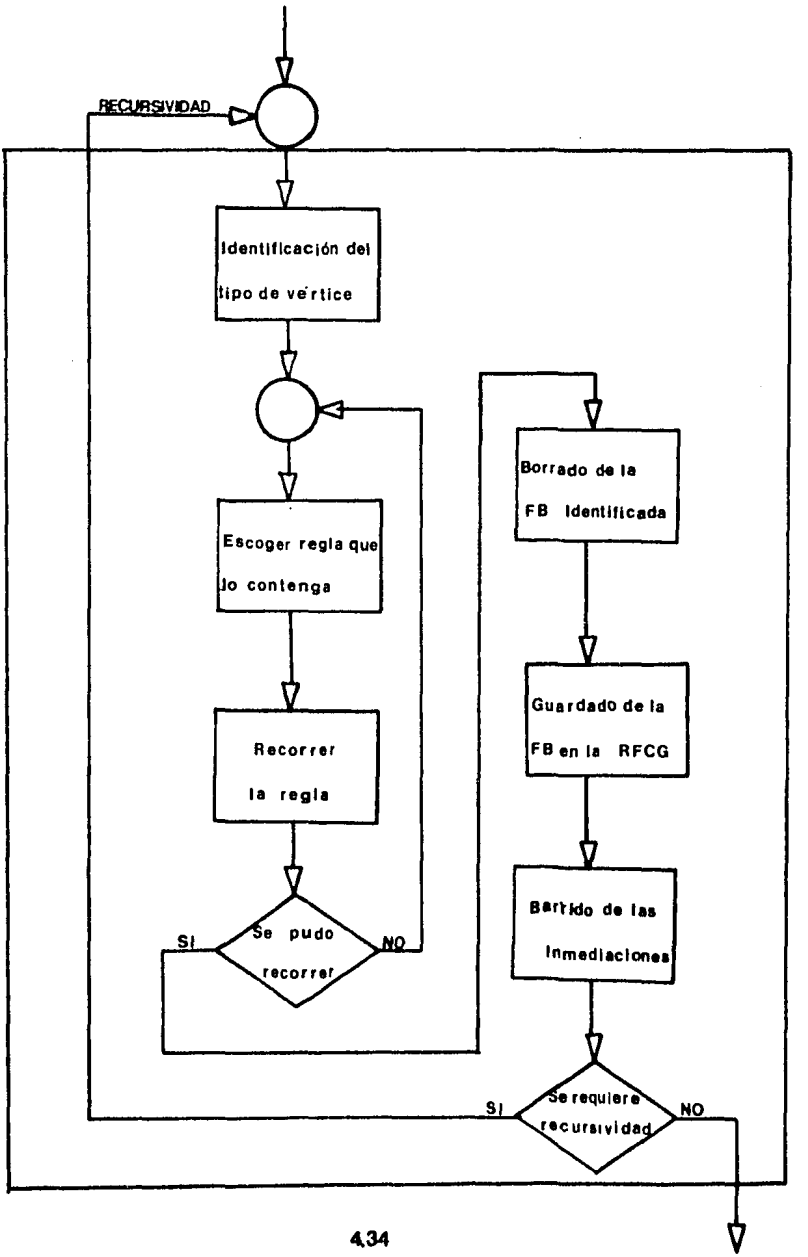


TABLA 4.2

Tipo de vértices existentes

VERTICE SUJETO1	VERTICE SUJETO2
H	V
H	@V
H	D
H	@D
H	I
H	@I
@H	V
@H	@V

VERTICE SUJETO1	VERTICE SUJETO2
@H	D
@H	@D
@H	I
@H	@I
V	D
V	@D
V	I
V	@I

VERTICE SUJETO1	VERTICE SUJETO2
@V	D
@V	@D
@V	I
@V	@I
D	I
D	@I
@D	I
@D	@I

La manera como se implementó este proceso es la siguiente:

Partiendo del punto definido como vértice se trata de seguir una trayectoria horizontal (hacia la derecha o hacia la izquierda del vértice). Posteriormente se trata de encontrar una de las trayectorias verticales (hacia arriba o hacia abajo del vértice). Si después de esto ya se encontraron los dos sujetos que definen al vértice, el proceso finalizará, pero si esto no ha ocurrido, el módulo reconocedor finalizará indicando que por errores en la generación de la imagen no se puede continuar con el reconocimiento.

4.3.2.2 ESCOGER UNA REGLA QUE CONTENGA AL VERTICE IDENTIFICADO.

Una vez que se han encontrado los dos sujetos que forman al vértice en cuestión, se debe encontrar el sentido correcto de recorrido de la regla. Lógicamente, un vértice debe unir la cabeza de un sujeto con la cola de otro. Además esta unión debe coincidir con el correcto sentido dde recorrido de la RP (que como ya se ha dicho es contrario a las manecillas del reloj). Esto se efectúa cambiando el modificador del sujeto que no sigue el sentido de recorrido correcto. En la tabla 4.3 se especifican los posibles vértices (Columna A) y esos mismos ya modificados (Columna B).

Después de haber realizado esta modificación en uno de los sujetos del vértice, se busca dentro de la información de las FBs en memoria una RP que lo contenga para tratar de asignarla a la FB que se trata de identificar.

4.3.2.3 RECORRIDO DE LA REGLA ESCOGIDA.

Una vez que se ha optado por una RP específica, el proceso de identificación de dicha FB se vuelve anticipativo, pues al tener la RP se sabe cuál será el siguiente sujeto que se debe encontrar; esto permite saber qué y en dónde buscarlo.

Una vez definidos los dos sujetos que forman el nuevo vértice encontrado, se escoje de entre el conjunto de FBs

TABLA 4.3

Asignación de Vértices

COLUMNA 2			COLUMNA 3	
SUJETO1	SUJETO2	VERTICE	SUJETO1	SUJETO2
H	V		H	@V
H	@V		@H	@V
H	D		H	@D
H	@D		@H	@D
H	I		H	@I
H	@I		@H	@I
@H	V		H	V
@H	@V		@H	V
@H	D		H	D
@H	@D		@H	D
@H	I		H	I
@H	@I		@H	I
V	D		@V	D
V	@D		V	D
V	I		V	@I
V	@I		@V	@I
@V	D		@V	@D
@V	@D		V	@D
@V	I		V	I
@V	@I		@V	I
D	I		D	@I
D	@I		@D	@I
@D	I		D	I
@D	@I		@D	I

disponibles, una que posea dichos sujetos. La RP de esta FB tratará de recorrerse, en espera que la FB analizada cumpla con ella. Si esto no llega a realizarse, se regresa a escoger otra RP que posea al nuevo vértice.

Una vez que se identifica la RP correspondiente, esto es, que coinciden la representación de la FB y la información contenida en la RP, se procede a evaluar la FN asociada a dicha RP esperando que ésta tambien se cumpla. Para ello, se van tomando, del vector de función, parejas de sujetos que deban poseer igual longitud y se prueba esto; después, se hace lo mismo para la parte de las desigualdades. En caso de que todo esto se cumpla, se declara como terminada la identificación de la FB en estudio asignándole a ésta las características de la clase de FB a que corresponde.

Si en alguno de los pasos anteriores se observaron discrepancias entre la información de la FB que se desea asignar a la FB en estudio y ésta, el proceso debe regresar nuevamente a la parte que escoge la RP de una FB que contenga los sujetos identificados. Este procedimiento se debe repetir tantas veces como sea necesario hasta que la información de una de las FBs en memoria corresponda con las características de la FB en estudio, con lo cual se acepta como correcta la identificación de la FB, o hasta que ya no existan más FBs que puedan tratar de asignársele a la FB en estudio, terminando para este último caso el procesamiento del reconocedor indicando que se generó una FB no definida para el sistema y por tanto no puede ser reconocida.

4.3.2.4 BORRADO DE LA FIGURA BASICA IDENTIFICADA.

Una vez que la FB ha sido identificada, el siguiente paso que efectúa el módulo en esta etapa, es el borrado de la figura encontrada, lo que evita que posteriormente se consideren los puntos que la definen como parte de otras FBs a identificar. Se puede borrar la información de la MG porque dentro de la información temporal se encuentra la referente, tanto a la FB de que se trata, como la posición de ésta dentro de la pantalla.

Este proceso sigue la misma trayectoria, dentro de la imagen, que el proceso de identificación de la FB.

4.3.2.5 ACTUALIZACION DE LA RFCG.

Después de haber borrado de la información de la MG los puntos correspondientes a la FB recién identificada, se actualiza la RFCG para que en ella aparezca la información referente a dicha FB. Para efectuarlo se trae de la SCFB el identificador de la FB y se coloca al final de la cadena que define a la RP de la RFCG.

4.3.2.6 BARRIDO DE LAS INMEDIACIONES DE LA FB IDENTIFICADA.

El último proceso de esta etapa del módulo se encarga de barrer los puntos adyacentes a la FB -basándose en la información temporal- en busca de nuevos puntos iluminados que correspondan a puntos de nuevas FB a identificar. Cada vez que se encuentre un punto se deberá constatar que se

trata de un vértice, y en caso de no serlo se seguirá el sujeto al que pertenece dicho punto hasta encontrar el vértice correspondiente.

Cuando un nuevo vértice ha sido identificado, se hace un "autollamado" a esta etapa del módulo ARMA-FCG (la cual es recursiva) para que un nuevo proceso de identificación de la nueva FB se efectúe.

Cada vez que el proceso de barrido de las inmediaciones finalice de analizar las localidades adyacentes a uno de los sujetos de la RP de la FB se colocará al final de la RP de la FCG un operador de fin de sujeto ("/").

Algo similar se efectúa cada vez que se finalice de analizar las localidades adyacentes a una RP de FB, ya que se colocará al final de la RFCG el operador de fin de RP ("*").

Hay que tomar en cuenta que para denotar el fin de figura, el proceso inserta inicialmente tantos operadores (" / ") como sujetos restantes (lados) y sin figuras adyacentes tenga. Siendo esta información redundante y confusa, los operadores fin de sujeto se sustituyen por el operador (" * ") fin de RP (figura). Con esto, se logra una mejor claridad en la RFCG como también un ahorro de espacio requerido para su representación en memoria.

Para clarificar como se lleva a efecto este proceso, se presentan los siguientes ejemplos :

Supóngase que se tiene la figura 10 a reconocer:

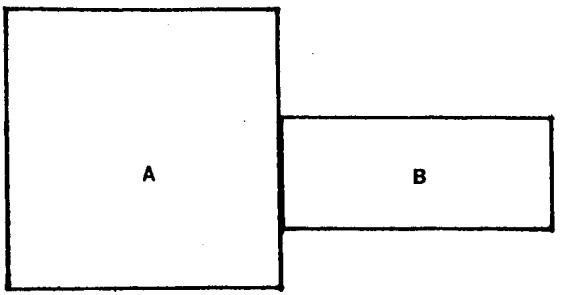


Figura 10

Una vez finalizado el analisis de las inmediaciones de la figura B, la RFCG del ejemplo, es la siguiente :

RFCG ::= A/B////

en este momento, los últimos cuatro operadores (" / ") son sustituidos por un operador (" * ") de la siguiente forma:

RFCG ::= A/B*

lo cual indica que el análisis de las inmediaciones de B ha finalizado. En este momento el proceso de reconocimiento regresa a la figura A para la búsqueda de más figuras adyacentes a

ella. En la etapa final del mismo, la RFCG se encuentra como sigue:

RFCG ::= A/B*///

de igual forma, los últimos tres operadores (" / ") se sustituyen por el operador (" * ") dando como resultado la siguiente RFCG final :

RFCG ::= A/B**

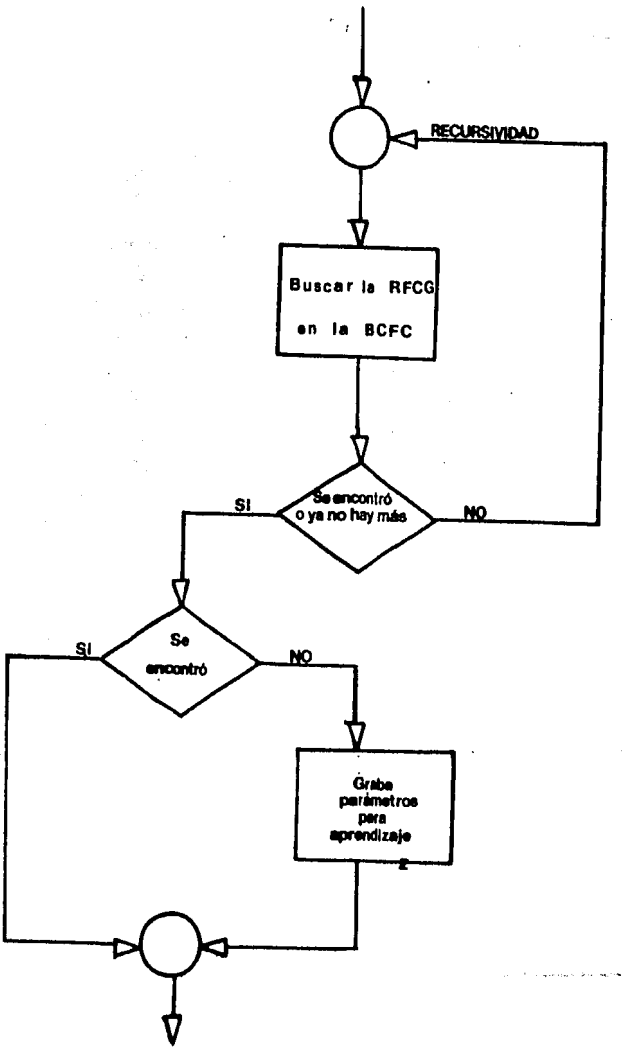
La etapa de armado de la RFCG finalizará cuando se haya terminado se ejecutar el proceso de barrido de las inmediaciones de la FB base (FB del primer nivel), quedando armada por completo la RFCG.

4.3.3 RECONOCIMIENTO DE LA FIGURA COMPUESTA GENERADA.

Durante esta última etapa del módulo reconocedor, SIREI busca dentro de la BCFC la RP que define a la FCG para decir cuál es o en caso contrario informar al usuario que no existe, dándole la opción de poder aprenderla; en caso afirmativo, se guarda en el lugar que le corresponde. Para efectuar esto, la etapa utiliza un procedimiento denominado identificación de la FCG el cual se describe a continuación y se muestra gráficamente en la figura 4.11.

FIGURA 4.11

ETAPA: IDENTIFICACION DE LA RFCG



4.3.3.1 IDENTIFICACION DE LA FCG.

Las dos entradas a este proceso son : la RFCG y la BCFC. La manera como el proceso opera es la siguiente.

Primero, encuentra el tamaño de la BCFC y posteriormente ve si la RFCG es mayor a la última RP de la BCFC (hay que recordar que esta base de conocimiento ésta ordenada alfabeticamente de manera ascendente). En caso que esto se cumpla, quiere decir que la RFCG no ha sido aprendida con anterioridad por lo que inmediatamente el proceso lo indicará, al igual que la posición que le corresponderá en caso de ser aprendida (la siguiente a la última RP de la BCFC).

Si la RFCG fue menor, el proceso de identificación llama a una rutina recursiva de búsqueda binaria, que se encarga de llamarse a sí misma hasta encontrar la RP buscada o indicar que no existe y la posición que le correspondería en caso de aprenderse.

Con esto finaliza el procesamiento que realiza el módulo reconocedor e indica a SIREI la RFCG reconocida. En caso de no existir ésta en la BCFC, SIREI obtiene el lugar en el que debe colocarse al momento de efectuar su aprendizaje.

4.4 RESTRICCIONES DEL MODULO.

Ya se sabe que aunque teóricamente se trabaje con un cierto universo de información, en la práctica éste se ve afectado

por algunos problemas. Para SIREI, su universo es el conjunto de todas las figuras bidimensionales posibles formadas por líneas rectas que describan trayectorias cerradas. En cuanto a la implementación del sistema, este universo se forma por horizontales y verticales únicamente, pero posee una lógica que permite la inclusión de las diagonales (consúltese sección 4.4.2).

A continuación se mencionan los dos tipos de restricciones que tiene SIREI : restricciones a la información y restricciones del equipo utilizado para su implementación, así como el efecto sobre su universo.

4.4.1 RESTRICCIONES DE LA INFORMACION.

Las principales restricciones que sobre la información son las siguientes :

- Las FBs que forman la imagen deberán estar adyacentes entre sí (basta con que tengan un punto de adyacencia). La implementación actual tiene como característica el reconocer una sola figura compuesta en una escena. Una FC está definida por una o varias FBs, con la condición de tener un punto de adyacencia entre sí. Por tanto, si una FB no cumple con esta premisa, no será tomada en cuenta como parte de la FC. Consúltense las extensiones propuestas a SIREI en el capítulo 7 para la implementación de esta adición al sistema.

- El rango de la pendiente aceptada para las diagonales

es entre 15 y 75 grados en cualquier sentido. Esto se requiere ya que la facilidad de graficación del sistema de cómputo utilizado (APPLE II e) es inadecuada en cuanto a la resolución de la pantalla para la representación de diagonales con una pendiente fuera del rango indicado.

- Las FBs que pueden ser cargadas deberán ser figuras cerradas formadas por líneas rectas. Al definir como FBs únicamente aquellas que describan trayectorias cerradas, el universo de SIREI se restringe, considerándose inválidas todas aquellas figuras que no cumplan con esta caractéristica.
- El tamaño mínimo de cada uno de los sujetos de las FBs es de cinco "pixels". Con esta restrición aparte de eliminar la posibilidad de generación de una figura geométrica de 1 "pixel" por lado, también se elimina la posibilidad de crear diagonales con una pendiente dentro de un rango entre 15 y 75 grados sin confundirlas con horizontales o verticales según sea el caso.

- Para que una FB sea considerada como válida deberá estar contenida en su totalidad dentro de la pantalla.

- Para que haya dos FBs diferentes, al menos deberán de variar éstas en uno de los dos posibles criterios de identificación (en la RP o en la FN). SIREI utiliza como criterios de identificación de FBs a la RP, la cual asegura que la FB posee un cierta forma describiendo una trayectoria cerrada, y la FN, la cual se encarga de de evaluar y relacionar las longitudes de los lados de una FB. Por tanto, un cuadrado con lados de longitud igual a 100 "pixels" tiene la misma RP y la misma FN que un cuadrado con lados de longitud igual a 20 "pixels". De esto podemos concluir que en cuanto a FBs se logra más que una restricción una característica en cuanto a la independencia del tamaño de una figura con respecto a otra. Así, un cuadrado, aunque sea mayor a otro, siempre va a ser reconocido como cuadrado. Donde esta restricción juega un papael más importante es en la construcción de la RFCG, donde para dos FCG distintas se tenga la misma RFCG.

- Para que la FCG sea válida, cada una de las FBs que la forman deberá existir dentro de la BCFB. Todas aquellas FCs para las cuales alguna de las FBs que la forman no cumpla con las restricciones anteriores, o no haya sido definida como FB dentro del sistema, tampoco formará parte del universo que puede manejar SIREI. Si se cargan otras FBs este universo aumentará considerablemente.

4.4.2 RESTRICCIONES DE LA IMPLEMENTACION.

Las principales restricciones de la implementación del módulo reconocedor de SIREI son :

- Una FB no puede tener un número de lados mayor al definido como "MNL". En esta implementación MNL=10, ya que con esta restricción se asegura que una figura generada con MNL o menos, no posea obligadamente diagonales con una pendiente fuera del rango especificado en el punto 4.4.1.

- No puede haber un número de FBs mayor al definido como "MNFB". MNFB se definió igual a 10 ya que contado con las limitaciones de memoria del equipo, 10 FBs cargadas en memoria ocupan aproximadamente 1Kbyte, no siendo esta cantidad significativa para resultar un problema adicional a la poca memoria disponible.

- Una FB no puede tener un número de igualdades y/o desigualdades mayor al especificado por "MNI". En la implementación se definió MNI=10, ya que éste tiene que estar relacionado con MNL que se explicó anteriormente.

- La longitud de la RFCG no puede ser mayor al tamaño máximo definido en la BCFC (cadena de caracteres), el cual esta definido como BCFC=32. Así se permite poder tener una RFCG que posea hasta 16 FCs.

5. APRENDIZAJE

5.1 INTRODUCCION

Aprendizaje es un término sumamente general que denota la manera en la cual las personas y en este caso, las computadoras, incrementan sus conocimientos y mejoran sus habilidades.

Este módulo se encarga del aprendizaje de las figuras compuestas que no fueron identificadas por el módulo Reconocedor.

El aprendizaje es un aspecto muy importante en cualquier sistema de inteligencia artificial, ya que le permite incrementar el conocimiento que contiene, convirtiéndolo cada vez más, en un sistema "experto". Como se mencionó en el capítulo 1, un sistema experto es aquél que auxilia en la solución de problemas específicos, por lo que, al ir acumulando SIREI más conocimiento, se asemeja cada vez más a este tipo de sistema.

Los cuatro principales tipos de aprendizaje son :

Aprendizaje por Observación. En él, se debe memorizar la información que está siendo recibida, para su uso posterior, sin necesidad de efectuar ningún proceso.

Aprendizaje de lo escuchado "Tomar consejos". La característica principal de este tipo de aprendizaje es la transformación de la información que está siendo recibida,

mediante un proceso denominado "operacionalización", lo cual consiste en entender e interpretar el conocimiento de alto nivel que se le proporciona y relacionarlo con lo que ya se sabe.

Aprendizaje por ejemplos (Inducción). Este tipo de aprendizaje consiste en enseñar cómo hacer una tarea, mediante ejemplos de cómo se comportará. El sistema debe generalizar estos ejemplos para encontrar reglas de más alto nivel, que lo guíen en futuras tareas.

Aprendizaje por Analogía. Si un sistema tiene a su disposición una base de conocimiento que indica cómo ejecutar tareas específicas, debe ser capaz de implementar tareas propias, basándose en el reconocimiento de características comunes y transfiriendo el conocimiento relevante de la otra base de conocimiento. Este tipo de aprendizaje está en sus inicios y existen aún ciertas preguntas en espera de ser resueltas, como pueden ser: Que es exactamente una analogía? Como se reconocen ? [3 vol. III pags. 325-511]

Revisando los distintos tipos de aprendizaje se aprecia que SIREI posee aprendizaje del tipo "por observación", ya que memoriza la nueva información para su uso futuro. Además, este aprendizaje es opcional, ya que una vez que el proceso de reconocimiento ha determinado que la figura compuesta generada no existe dentro de su base de conocimiento, SIREI concede al usuario la opción de incluir dicha figura dentro de la base de conocimiento

respectiva. Esta facilidad permite que aquellas figuras que no interesen al usuario o que hayan sido generadas por error, sean desechadas, con lo cual se logra una base de conocimiento confiable y eficiente.

El proceso de aprendizaje resulta muy simple para el usuario, ya que éste se limita a indicar si quiere o no que se efectúe. SIREI a través del módulo reconocedor se encarga de asignar un nombre a la figura y buscar la posición que le corresponde en la base de conocimiento. Además, en el módulo de aprendizaje se inserta la nueva RFCG en la BCFC.

5.2 REPRESENTACION DE LA INFORMACION

La información que se maneja en este módulo se refiere tanto a la figura compuesta generada como a la base de conocimiento respectiva, las cuales se describen en esta sección.

La representación de una figura compuesta se realiza mediante una regla de producción (RP), que contiene las FBs que la forman y la relación existente entre ellas. Para una mayor descripción de esta RFCG, consultar la sección 4.2.

Por lo que respecta a la base de conocimiento de figuras compuestas, ésta consiste en un archivo secuencial de acceso aleatorio, donde cada registro almacena la representación de una figura compuesta, es decir, su regla de producción. Estas representaciones se almacenan siguiendo un orden alfabético ascenden-

te, con lo cual se facilita el proceso de búsqueda durante la etapa de reconocimiento.

5.3 ALGORITMO

El proceso de aprendizaje consiste en registrar una figura compuesta que el Reconocedor no ha podido identificar en la base de conocimiento de figuras compuestas (BCFC).

En primer lugar, SIREI pregunta al usuario si desea que la nueva figura sea incluida en la BCFC, esto es, que sea "aprendida". En caso afirmativo sucede lo siguiente:

- El módulo de aprendizaje toma los parámetros que le manda el Reconocedor: la representación de la figura compuesta a aprender (RFCG) y la posición (P) que le corresponde dentro de la BCFC. Esta posición es determinada por el Reconocedor durante la búsqueda fallida de dicha figura. Se utiliza una búsqueda binaria sobre el archivo que contiene la BCFC, el cual está ordenado alfabéticamente, lo que garantiza que una vez concluido el proceso de aprendizaje, el archivo quedará igualmente ordenado.

- Las figuras que se encuentran de la posición (P) a la posición NAFC - número actual de figuras compuestas -, (que se encuentra grabado en el primer registro del archivo), son recorridas una posición.

- La representación de la figura a aprender (RFCG) se graba en el registro (P) del archivo.

- Se actualiza el número actual de figuras compuestas (NAFC), incrementándolo en uno y se graba en el primer registro del mismo archivo.

En caso que el usuario no desee que esta nueva figura sea aprendida, el proceso finaliza, dejando sin modificar la BCFC.

La única restricción que posee el aprendizaje se refiere al número máximo de figuras compuestas que el sistema puede aprender. Este número está limitado por la capacidad de los discos de la computadora.

Específicamente, cada figura compuesta ocupa treinta y dos bytes. Como los bloques (unidades lógicas del disco) constan de 512 bytes, cada bloque puede almacenar dieciséis figuras compuestas. Así mismo, el disco completo consta de 280 bloques, de los cuales 274 son accesibles al usuario mientras que los seis restantes son de uso exclusivo del sistema operativo. Por tanto, el número total de figuras compuestas que el sistema puede aprender, considerando que se cuenta únicamente con una unidad de disco destinada a este propósito, es 4,384 figuras. Si fuera necesario, SIREI podría manejar más figuras (el máximo número de unidades de disco que se pueden usar para almacenar figuras es de cinco, lo cual significa 4,384 x 5 = 21,920 figuras). Para ello sería necesario conectar más unidades de disco a la computadora, así

como incluir la lógica necesaria para manejar varios discos en las rutinas que interactúan con la BCFC, que son IDENTIFICA_FC - que identifica la figura compuesta -, y APREND - que "aprende" dicha figura.

Hasta este momento se ha hablado únicamente de lo referente a la implementación del sistema (capítulos 3, 4 y 5), pero, para poder efectuar algunos otros procesos de interés para el usuario (como listados del contenido de las Bases de Conocimiento, inicialización de archivos, etc.), se incluyó un módulo de utilerías que le facilita al usuario esta labor. Dicho módulo se describirá a detalle en el siguiente capítulo.

6. UTILERIAS

6.1 INTRODUCCION.

La finalidad de este módulo es hacer más amigable la interacción del usuario con SIREI; se pretende que ciertos procesos relativamente externos al sistema le sean transparentes.

Se proporcionan herramientas para la inicialización de las bases de conocimiento, para conocer el estado actual del sistema y para modificar la información que maneja. Estas herramientas son las siguientes.

6.2 INICIALIZACION DE LAS BASES DE CONOCIMIENTO.

Primeramente, la base de conocimiento de figuras básicas (BCFB), se almacena físicamente en un archivo en disco. Al inicio del sistema, esta base de conocimiento no existe, por lo cual es necesario ejecutar esta rutina.

Este proceso crea un nuevo archivo para la BCFB, lo que implica que si ya existía información en ella, ésta se eliminará. Por lo anterior, esta rutina no debe utilizarse excepto la primera vez que se va a usar el sistema o cuando se desea eliminar toda la información de la BCFB. Cuando se corre esta rutina, se avisa al usuario que la BCFB será eliminada, dándole la opción de no continuar con el proceso si es que éste fue activado por equivocación.

Por lo que se refiere a la base de conocimiento de figuras compuestas (BCFC), ésta también se almacena físicamente en un archivo en disco. Al inicio del sistema la BCFC no existe, por lo que para generarla se debe correr este proceso que se encarga de crear un nuevo archivo para ella. Es importante indicar que si este proceso se utiliza cuando ya existe la BCFC, la información que contiene se borrará; por esto, al escoger esta opción, SIREI avisa al usuario que de continuar con el proceso, la BCFC se eliminará, dándole la opción de continuar o no con este proceso.

6.3 CARGADOR DE FIGURAS BASICAS.

Esta opción tiene como función lograr una independencia entre la información elemental que maneja el sistema y los distintos procesos de que consta.

Esta información elemental es la que se refiere a las figuras básicas, que es el conocimiento mínimo necesario para que el sistema pueda reconocer y aprender imágenes más complejas (figuras compuestas).

La rutina permite al usuario agregar nuevas figuras básicas a la base se conocimiento respectiva. Para cada una de éstas se debe proporcionar su regla de producción y la función correspondiente, así como su nombre. El cargador se encarga de asignar un identificador por medio del cual se hará referencia a dicha figura dentro del sistema.

La única restricción existente es que no pueden agregarse nuevas figuras básicas si el número de FBs existentes en la BCFB es igual al número máximo de figuras básicas (MNFB), el cual es una constante en los programas cuyo valor actual es de diez.

Si se desea profundizar más, tanto en la representación de estas figuras así como en las estructuras que la almacenan, se puede consultar la sección 4.2.

6.4 BORRADO DE FIGURAS COMPUESTAS.

Esta opción permite la eliminación de RPs de FCs que no se deseen. Es de gran utilidad para poder mantener la base de conocimiento compacta y sin información irrelevante.

El procedimiento que sigue este proceso es el siguiente:

- Se pide al usuario el número de la RP de la FC que se desea eliminar, el cual se puede consultar obteniendo un catálogo de figuras compuestas según se describe más adelante en el punto 6.5.

- Si el número proporcionado por el usuario es menor que uno, o mayor que el número actual de figuras compuestas (NAFC), se le informa al usuario que el movimiento es inválido.

- En caso contrario, se elimina la figura y se recorre un lugar la información restante (de la posición siguiente a la

eliminada en adelante), con lo que la BCFC queda ordenada nuevamente.

- Finalmente se regresa al menú de utilerías.

Se observa que no existe un proceso de borrado de FBs ya que podría suceder que se borrara una FB que estuviera contenida dentro de una FC, por lo que al tratar de trabajar con esta última, se crearía una inconsistencia.

6.5 CATALOGOS DE FIGURAS

Para conocer el contenido actual de la BCFB cada vez que se desee, se puede correr este proceso que despliega la información existente de todas las figuras básicas que se encuentran dentro de esta base de conocimiento, ya sea en pantalla o en un reporte impreso.

Este proceso es de gran utilidad ya que permite saber si una figura básica existe en el sistema o es necesario incorporarla mediante el cargador de figuras básicas.

En caso que se desee conocer todas las figuras compuestas que han sido aprendidas por el sistema, se puede ejecutar este proceso que muestra la información contenida en la BCFC, ya sea en la pantalla o en un reporte impreso.

Es importante aclarar que no existe un cargador de figuras

compuestas, ya que ésta es precisamente la función que realiza el módulo de aprendizaje, la cual es una de las finalidades básicas de SIREI.

Una vez que se han especificado a detalle los distintos módulos que constituyen a SIREI, únicamente falta el hablar de las conclusiones, las cuales son incluidas en el siguiente capítulo.

7 CONCLUSIONES

7.1 CONOCIMIENTO

Uno de los puntos más importantes en un sistema inteligente se refiere al conocimiento que maneja : qué clase de conocimiento, cómo se representa, cómo se maneja. La definición del conocimiento necesario determina el tipo de problemas que el sistema puede atacar. La manera de representarlo y manejarlo influye grandemente en la eficiencia de operación, así como en la capacidad de modificación del sistema.

Con respecto a la eficiencia de operación se puede observar que para poder decidir entre las distintas opciones existentes, un sistema inteligente debe recurrir a la información que posee para ayudarse en su decisión, por lo que mientras más sencillo y rápido sea el acceso a ella, el tiempo requerido para la solución escogida disminuirá considerablemente. Además, la manera como la información es almacenada influye en el aprovechamiento de memoria.

En relación a la capacidad de modificación, se tiene que tomar en cuenta la independencia de la información en las bases de conocimiento. Si se considera que las estructuras de datos en un programa son piezas de conocimiento, luego entonces, el agregar nuevas estructuras de datos equivale a añadir conocimiento al sistema. Una característica esencial es la modularidad, la cual facilita la capacidad de añadir, modificar y borrar estructuras de datos con un alto grado de independencia del resto de la base

de conocimiento, siendo los efectos causados dentro de la base de datos nulos o conocidos. [3 vol. 1 cap. 3]

SIREI emplea dos clases de conocimiento:

- Figuras básicas, formadas a partir de líneas rectas. Por el momento, la implementación permite el reconocimiento de líneas horizontales y verticales, aunque en la teoría descrita fueron consideradas también las diagonales.

- Figuras compuestas, formadas por la combinación de figuras básicas adyacentes (puede ser una sola).

En ambos casos, las figuras se presentan en un espacio bidimensional.

Estas dos clases de conocimiento orientan la aplicación de SIREI, en su estado actual, a la solución de problemas que involucran figuras geométricas simples, pudiendo ser éste el punto de partida para el manejo de figuras más complejas.

Para representar este conocimiento se escogió un método sintáctico empleando la técnica de "reglas de producción". Los patrones están considerados como enunciados en un lenguaje definido por una gramática. Esta aproximación asume una gramática de la imagen para construir las representaciones formales de los objetos y sus interrelaciones en la escena, así como un conjunto de primitivos. En el caso de las figuras básicas, estos primiti-

vos son "items", modificadores y operadores; para las figuras compuestas los primitivos son figuras básicas y operadores. La implementación de las reglas de producción de las FBs se detalla en la sección 4.2.

El empleo de reglas de producción presenta varias ventajas:

- Modularidad.

Las producciones individuales en la base de conocimiento se pueden añadir, borrar o cambiar independientemente. Cada regla posee piezas independientes de conocimiento. Al cambiar una regla, aunque afecta al sistema, no afecta a las otras reglas, ya que éstas no dependen unas de otras.

- Uniformidad.

Otro de los atributos de estos sistemas es la estructura uniforme, ya que la información debe estar codificada dentro de la estructura de las reglas de producción, lo que permite un manejo muy similar para figuras muy diferentes.

- Sencillez.

Una característica adicional es la facilidad con la cual se pueden representar estos tipos de conocimiento. En particular, las figuras básicas y compuestas de SIREI se definen mediante estructuras muy sencillas, que constan de combinaciones de sujetos y operadores. Para las FBs los sujetos son los lados que poseen, de manera conjunta con la dirección en que se recorren; mientras que el único operador

definido por el momento es el de concatenación, el cual une la cabeza del "item" actual con la cola del siguiente de la regla de producción. Dentro de las FCs sus sujetos vienen siendo los identificadores, o lado izquierdo, de las reglas de producción de las FBs; y sus operadores son el de "fin de lado o sujeto" y el de "fin de regla de producción", de las figuras básicas involucradas. Para mayor detalle puede consultarse la sección 4.2.

- Eficiencia.

El manejo de información a través de reglas de producción es muy eficiente cuando éstas se aplican a figuras construidas por una serie de primitivos bien definidos, fácilmente reconocibles y cuando las relaciones entre estos primitivos son simples, como en el caso de SIREI.

Algunas desventajas de las reglas de producción son:

- Falta de capacidad descriptiva.

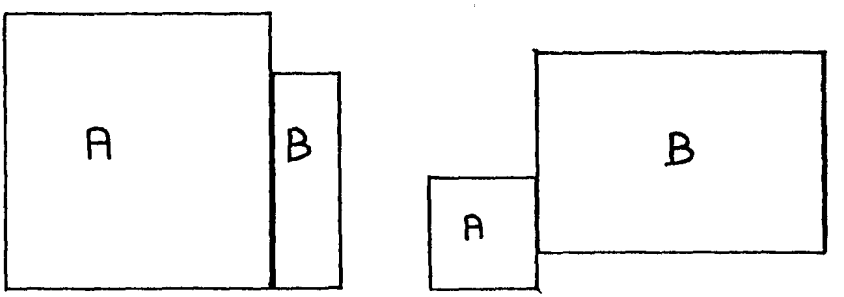
En general, cuando se efectúa una transferencia de un hecho o acontecimiento real a una representación se pierden algunas de sus características descriptivas; las reglas de producción no son la excepción, y en particular, las de SIREI no describen dimensiones tales como longitudes, giros, puntos de adyacencia, etc., ya que el abarcar todo esto llevaría a un aumento considerable de la información a manejar. Debe existir un compromiso entre la cantidad de detalle deseado para cada elemento representado por una

regla de producción y el espacio disponible en memoria con que se cuenta para la implantación del sistema.

- Difícil reconstrucción de la imagen.

Dada una regla de producción, la reconstrucción de la figura que representa puede ser considerablemente distinta de la figura original, debido también a la falta de capacidad descriptiva, ya que mientras más específica es la información de un sistema se necesitarán más características que la definan. A continuación se ilustrará este problema:

+ Supóngase que se tienen las dos FCs siguientes :



Como se pude notar, ambas son visualmente diferentes, pero debido a que están formadas por las mismas FBs y éstas son adyacentes al mismo lado, quedan representadas por la siguiente regla de producción :

RP::=A/B**

- Rigidez en la representación.

Puesto que se trabaja con un conjunto de primitivos previamente definidos, la representación de una figura que no esté formada exclusivamente por ellos resulta imposible o implica modificaciones sustanciales en la representación del conocimiento e incluso, en la programación desarrollada [1 pags. 190-199;2 pags. 143-155; 3 vol. 3 pags. 287-291].

7.2 EVALUACION GENERAL [3 vol. 1 cap III]

Para evaluar un sistema de Inteligencia Artificial, se tienen que considerar tres puntos esenciales que son: la recuperación de hechos de la base de conocimiento relevantes al problema actual, el razonamiento acerca de estos hechos en busca de una solución y la facilidad de adquisición de más conocimiento.

Adquisición. Usualmente se piensa en el aprendizaje como la acumulación de conocimiento, pero involucra mucho más que esto. En realidad, la adquisición de conocimiento implica relacionar algo nuevo con lo que ya se conoce. Los sistemas de IA, usualmente clasifican una nueva estructura de datos antes de ser agregada a la base de conocimiento. Estas nuevas estructuras pueden interactuar con las anteriores. Un sistema de IA es bueno si realiza su proceso de adquisición de conocimiento de manera tal que después pueda extraerlo fácilmente, y además tenga cuidado de que el nuevo conocimiento ayude a solucionar nuevos problemas, sin generar interferencia en el uso del mismo para solucionar aquéllos previamente resueltos. En caso contrario, la adquisición de

conocimiento solamente aumentará la información del sistema sin tener mejoras reales en su funcionamiento.

Recuperación. Se refiere al acceso de la base de conocimiento para obtener la información necesaria durante el proceso. Un buen sistema de IA es aquél que encuentra el conocimiento relevante al problema dentro del existente, de una manera rápida y precisa. Después que se ha encontrado el vértice inicial de una FB, se escoge de entre el conjunto de éstas, únicamente aquellas que posean el vértice encontrado, desechando para ese análisis las demás. Al trabajar con un subconjunto del total de la información que se posee, el proceso de identificación se agiliza.

Razonamiento. Todo sistema de IA debe ser capaz de determinar una solución a un nuevo problema a partir de sus conocimientos actuales, es decir, razonar. Existen varios métodos de razonamiento usados actualmente en IA : formal, procesal, por analogía,generalización-abstracción y "meta-level" [3 vol. 1 cap. III]. En relación a este punto, se considera que un sistema de IA es bueno cuando utiliza, de entre los métodos de razonamiento, aquél que mejor resuelve el tipo de problema para el cual fue creado.

En lo referente al primer criterio de evaluación, se puede decir que SIREI cuenta con dos bases de conocimiento, de Figuras Básicas y de Figuras Compuestas. El usuario es el encargado de modular la adquisición de conocimiento mediante el proceso de cargado de las figuras básicas y con base en esto, el proceso de

aprendizaje dinámico de FCs formadas por FBs. En esta última parte lo único que debe hacer el usuario es el autorizar al sistema para incluir nuevas FCs. Al evaluar a SIREI en este aspecto se puede concluir que su proceso de adquisición de conocimiento es bueno, en especial el que se refiere al aprendizaje de nuevas figuras compuestas, ya que antes de incluirlas en la base de conocimiento las clasifica y ordena, lo que facilita su futura extracción sin interferir con FCs previamente aprendidas.

En relación al segundo criterio de evaluación se tiene lo siguiente. Para la base de conocimiento de figuras básicas (BCFB) su manejo es muy sencillo, mediante acceso secuencial, pues el número de figuras posibles es pequeño (en la implementación actual es dos, el máximo está alrededor de 50 ya que se utilizan las letras como identificadores, -de la "A" a la"Z" tanto para mayúsculas como para minúsculas-). El manejo de la base de conocimiento de figuras conpuestas (BCFC) es más complejo, ya que ésta puede crecer considerablemente y un acceso secuencial resultaría ineficiente. La heurística utilizada en ella consiste en extraer la información mediante una búsqueda binaria basada en el único campo de la base de conocimiento, que es su regla de producción. Este acceso es posible ya que cada vez que se añade una nueva regla a la BCFC, se coloca en el lugar que le corresponde, de acuerdo a un ordenamiento alfabético ascendente sobre la regla de producción, teniendo únicamente que recorrer las reglas que se encuentren de este lugar en adelante. Por ello, en todo momento la BCFC estará ordenada bajo este criterio. Se concluye que el

proceso de extracción de conocimiento de SIREI es bueno: en cuanto a las FBs, la búsqueda secuencial es la mejor solución considerando el número de éstas. Con respecto a las FCs, la búsqueda binaria implementada es eficiente, ya que para el peor de los casos, cuando se tengan "n" elementos en la base de conocimiento, el número de accesos necesarios para encontrar el elemento deseado o determinar que éste no existe, es log2 (n) [21 vol. 3 pags 422-450].

En cuanto al tercer criterio, se aprecia que el razonamiento que usa SIREI es mínimo. Posee algunas características del "razonamiento procesal" y el "razonamiento por analogía". El primero sigue un orden específico de pasos para resolver la pregunta, tanto para el reconocimiento de las FBs como el de las compuestas. El razonamiento por analogía obtiene las similitudes entre lo ya aprendido y lo que está en proceso de reconocimiento, como cuando SIREI emplea los sujetos de un vértice encontrado y los compara con las FBs existentes para poder suponer que la que se está tratando de reconocer es alguna de ellas.

7.3 LIMITACIONES

Aunque en la actual implementación de SIREI existen algunas limitaciones, se puede consultar la parte de extensiones en donde se habla de la manera como se pueden solucionar. Las principales limitaciones que presenta el sistema son:

A.- Las figuras básicas susceptibles de ser cargadas dentro del sistema deben cumplir con las siguientes características:

- Estar formadas por líneas rectas. Por el momento la implementación permite el reconocimiento de horizontales y verticales.
- Describir trayectorias cerradas.
- Ser bidimensionales.

B.- Los principales problemas encontrados durante la identificación de las figuras básicas, que impiden que ésta se realice son :

- Las figuras, por el momento, no pueden estar giradas (consultar la solución propuesta en el punto 7.4).

- No se puede identificar una figura básica que no esté contenida en su totalidad dentro de la pantalla.

- El reconocimiento de figuras básicas intercaladas, sobrepuestas y/o contenida dentro de otra (s), no es posible.

- Aquellas figuras básicas que no tengan al menos un punto de adyacencia con el resto de las figuras contenidas en la pantalla y estén formando la figura compuesta no serán consideradas por el reconocedor. Por el momento, el procedimiento que efectúa el barrido de la matriz gráfica (MG) busca renglón por renglón puntos iluminados. Cuando encuentra uno, el

proceso no hace una búsqueda exhaustiva dentro de la MG, ya que esto sería tardado, sino que se limita a buscar puntos contiguos al encontrado. Por tanto, SIREI detectará únicamente a las FBs que se encuentren adyacentes a la que corresponda el primer punto iluminado encontrado. En caso de existir otras figuras no adyacentes a la analizada, no serían tomadas en cuenta. Esto podría ampliarse para reconocer todas las FCs existentes dentro del espacio de reconocimiento, lo que permitiría reconocer distintas FCs que se encontraran al mismo tiempo en el espacio de reconocimiento. Una aplicación inmediata donde puede apreciarse la finalidad de esta mejora es en el sistema Visión-Robot, sistema para reconocimiento y adquisición de piezas planas para tareas de ensamblaje [22], cuya pecularidad reside en deshacer y desplazar aquellas piezas que por estar superpuestas en la escena resultan irreconocibles.

Al manejar únicamente parte de la posición entre figuras básicas en el proceso de formación de la compuesta, algunas RFCG con características visuales diferentes, (como el tamaño relativo de las figuras básicas que las forman), poseen la misma regla de producción. Por tanto se necesitarían mayores características descriptivas de la regla de producción de la figura compuesta para que durante el armado de ella, se solucione este problema.

proceso no hace una búsqueda exhaustiva dentro de la MG, ya que esto sería tardado, sino que se limita a buscar puntos contiguos al encontrado. Por tanto, SIREI detectará únicamente a las FBs que se encuentren adyacentes a la que corresponda el primer punto iluminado encontrado. En caso de existir otras figuras no adyacentes a la analizada, no serían tomadas en cuenta. Esto podría ampliarse para reconocer todas las FCs existentes dentro del espacio de reconocimiento, lo que permitiría reconocer distintas FCs que se encontraran al mismo tiempo en el espacio de reconocimiento. Una aplicación inmediata donde puede apreciarse la finalidad de esta mejora es en el sistema Visión-Robot, sistema para reconocimiento y adquisición de piezas planas para tareas de ensamblaje [22], cuya pecularidad reside en deshacer y desplazar aquellas piezas que por estar superpuestas en la escena resultan irreconocibles.

Al manejar únicamente parte de la posición entre figuras básicas en el proceso de formación de la compuesta, algunas RFCG con características visuales diferentes, (como el tamaño relativo de las figuras básicas que las forman), poseen la misma regla de producción. Por tanto se necesitarían mayores características descriptivas de la regla de producción de la figura compuesta para que durante el armado de ella, se solucione este problema.

7.4 EXTENSIONES.

Las posibles mejoras que pueden realizarse en SIREI se hayan intimamente asociadas a las limitaciones que el sistema presenta, las cuales en este trabajo no fue posible implementar debido a limitaciones del equipo disponible. A continuación se mencionarán las principales extensiones que pueden llevarse a cabo:

- Manejo de otro tipo de sujetos como podrían ser las diagonales, lo cual permitiría la incorporación de nuevas FBs más complejas, con lo que se podría reconocer una mayor cantidad de FCs. La manera como podrá implementarse ésto, aparece descrita dentro del capítulo 4 (sección 4.3.2).

- Considerar para el proceso de reconocimiento toda la información existente en la pantalla. Para ello bastará con aumentar, en el proceso de armado de la FCG, un proceso recursivo al final del reconocimiento de la FCG actual, que permita continuar con el barrido en busca de puntos iluminados correspondientes a nuevas FCGs.

- Aumentar a los criterios de diferenciación algunas otras características de las figuras básicas como pueden ser la longitud de éstas, lo cual lleva a un mejoramiento en cuanto a la definición de las figuras compuestas.

- Aumentar los operadores existentes para las figuras básicas. Con ello se lograría una mayor versatilidad, tanto en el recorrido de ellas, como en algunas de sus características.

7.5 APLICACIONES

Una aplicación inmediata de la Tesis se sitúa dentro de la educación a nivel universitario como práctica introductoria al campo de la Inteligencia Artificial, ya que muestra conceptos básicos de detección de bordes y segmentación que se presentan en forma gráfica. A su vez contribuye a fortalecer los conceptos de la programación estructurada, como son : el diseño " Top-Down ", la modularidad y la cohesión.

Por otro lado, SIREI puede integrarse como un subsistema de una aplicación más completa, como podría ser un robot dotado de visión con capacidad de aprendizaje. En este sentido, SIREI tiene la ventaja de ser un módulo compacto, que no requiere de grandes recursos de procesamiento.

Dentro de la industria, SIREI puede aplicarse en el ensamble de piezas que consten de varios componentes de conformación geométrica simple. Actualmente existen robots que realizan esta función, pero requieren que los distintos componentes le sean suministrados en un orden estricto y altamente estructurado, ya que la mayoría carecen de sistemas de reconocimiento de imágenes. Al incorporar SIREI en estos robots, los componentes podrían aparecer en una banda de producción distribuidos sin un orden

específico, puesto que SIREI se encargaría de identificar cada uno de ellos. Esto significa que tales robots podrían trabajar en ambientes menos estructurados, lo cual aumentaría su versatilidad.

En resumen, la gama de aplicaciones de SIREI es muy amplia. Para cada caso en particular, se requerirían las interfaces apropiadas, así como algunas de las mejoras de operación mencionadas en la sección 7.4 de extensiones. Por ejemplo, en cuanto al "software", se podría mejorar el algoritmo que construye la matriz gráfica; en lo que se refiere al "hardware", podría mejorarse la memoria RAM que requiere, tanto en tamaño como en tiempo de acceso, así como el procesador, que en nuestra implementación es un 6502, que es lento comparado con otros existentes (8086, 68000 ó Z80000 por ejemplo).

7.6 OBJETIVOS ALCANZADOS Y NO ALCANZADOS.

Analizando los diversos objetivos propuestos para la Tesis (referirse al punto 1.2) podemos destacar lo siguiente :

A.- El objetivo general del sistema se alcanzó plenamente ya que SIREI es un sistema capaz de reconocer ciertas figuras además de aprenderlas para poderlas identificar en experiencias posteriores, logrando además, un sistema compacto.

B.- Con respecto a los objetivos específicos se puede afirmar que el sistema presentado sigue rigurosamente las reglas del diseño estructurado y su programación cumple satisfactoriamente las exigencias de la modularidad y la cohesión, además de que es altamente modificable, flexible y de fácil manejo para cualquier usuario.

C.- Uno de los objetivos específicos más ambiciosos es el de contribuir con algunas bases para la creación de sistemas reconocedores, y a este respecto se puede afirmar que las reglas de producción es un método confiable y de fácil programación que permite generar un sistema de buena calidad, sin requerir muchos recursos.

D.- Dentro de los objetivos no cumplidos se puede mencionar la implementación del reconocimiento de diagonales, que no se logró por limitaciones de equipo. Sin embargo, la teoría para implementar este punto se desarrolló completamente y se encuentra descrito en la sección 4.2.

BIBLIOGRAFIA Y REFERENCIAS CONSULTADAS.

1. WINSTON, P. H. "The Psychology of Computer Vision"; N.Y., Mc-Graw Hill Book Company, 1975.

2. WINSTON, P. H. "Artificial Inteligence"; 2 a edición. Massachusetts, Addison-Wesley Publishing Company, 1979.

3. BARR, A., FEIGENBAUM, E. A. "The Handbook of Artificial Inteligence"; Stanford Cal. Heristech Press, 1981, Volumenes 1, 2 y 3.

4. DUDA, R., HART, P. "Pattern Recognition and Scene Analysis". N.Y., Witey, 1973.

5. ROSENFELD, A. "Picture Languages. Formal Models for Picture Recognition ". New York, Academic Press, 1979.

6. FU, K.S. "Syntactic Methods in Pattern Recognition". New York, Academic Press, 1974.

7. PAULIDIS, T. " Structural Pattern Recognition "; Berlin Sprinnger - Verlag, 1977.

8. GUZMAN ARENAS, A. "Computer Recognition of Three-Dimensional Objects in a Visual Scene "; Tech. Rep. MAC-TR-59, AI Laboratory,Massacusetts Institute of Technology, 1968.

9. YOURDON, E., CONSTANTINE, L. " Structured Design"; New Jersey, Prentice-Hall, 1979.

10. IEEE Transactions on Pattern Analysis and Machine Inteligence; March 1984. The IEEE Computer Society.

11. GRAHAM, N. "Artificial Inteligence". P.A., Tab Books Inc, 1979.

12. GROGONO, P. "Programing in PASCAL"; 2a edición. Massachusetts, Addison-Wesley Publ. Inc., 1978.

13. Apple II Pascal. "Language Reference Manual"; Apple Computer Inc, 1980.

14. Apple II Pascal. "Operating System Reference Manual". Apple Computer Inc, 1980.

15. Apple II. "Reference Manual". Apple Computer Inc, 1980.

16. COMPUTER. "Knowlegde Representation". Octubre 1983.

17. SLAGE, J. "Artificial Intelligence: The Heuristic Programming Approach"; McGraw-Hill Book Company, 1971.

18. NILSSON, N. "Problem-Solving Methods in Artificial Intelligence ;McGraw-Hill Book Company, 1971.

19. RICH, E. "Artificial Intelligence"; McGraw-Hill Book Company, 1984.

20. WIRTH, N. "Algorithms + Data Structures = Programs"; Prentice-Hall Inc, 1976.

21. KNUTH, D. "The Art of Computer Programming, Sorting and Searching; Addison-Wesley, volume 3, 1973.

22. SANFELIU A., TORRAS C. "Cooperación Visión-Robot en Tareas de Ensamblaje"; Institut de Cibernética, II Simposium Nacional de Automática en la Industria, Zaragoza, España. nov. 1984.

APENDICE A

ESTANDARES

A.1. Método de Diseño Usado.

A.1.1 Descripción

A.1.2 Estándares del Sistema

A.2. Diagramas de Estructura.

A.1. Método de Diseño Usado.

A.1.1 Descripción.

Para el diseño del Sistema SIREI, se usó la técnica de "Programación Estructurada". Por medio de ésta, se minimizan los siguientes problemas futuros del Sistema como podrían ser:

- Mantenimiento
- Facilidad de modificación
- Facilidad de uso
- Flexibilidad
- Calidad

El proceso de desarrollo de sistemas consta de cuatro pasos como se puede ver en la figura A.1.

El primero de ellos es el llamado "Análisis Estructural". En esta fase se deben tener claras las especificaciones funcionales de las diferentes etapas del sistema. Aquí se definen las entradas, las salidas y ciertos algoritmos que deben usarse en cada una de estas etapas.

El segundo paso es el "Diseño Estructural". Este diseño es el proceso utilizado para decidir en que forma se elaboran e interconectan los componentes del sistema para resolver problemas específicos en forma óptima.

FLUJO DE DISENO

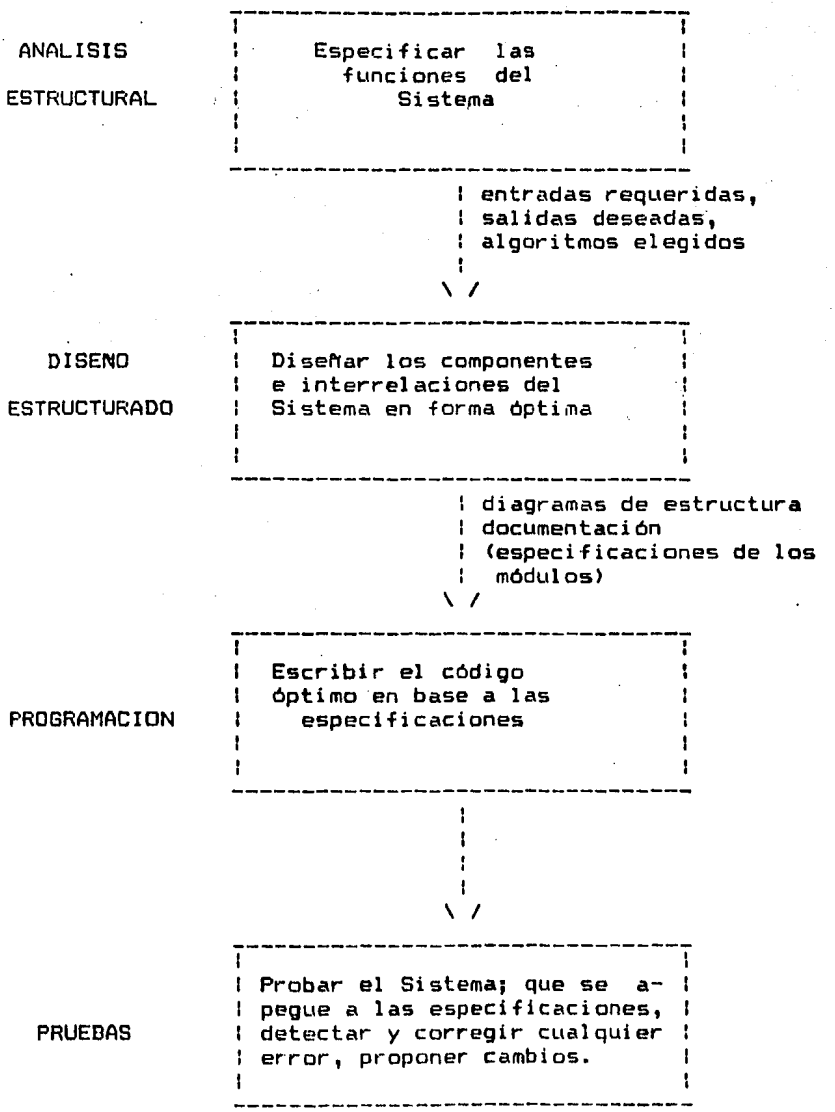
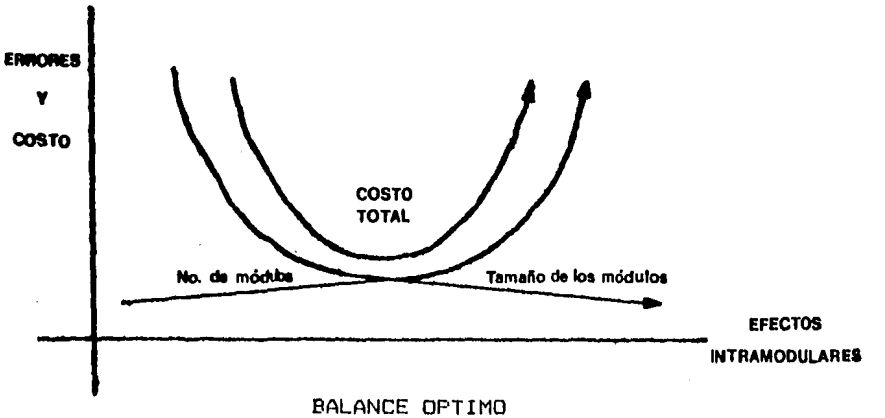


FIGURA A1

Resultando de mayor complejidad el diseño y escritura de programas muy grandes, se incluyó el concepto de modularidad, esto es, la división de grandes problemas en subproblemas. Lo importante de esto, como se observa en la figura A.2, es llegar a un equilibrio entre el tamaño de los subproblemas y la complejidad de la interfaz. Esto tiene que lograr una independencia entre los módulos subdivididos y evitar la introducción de errores por la interrelación de los mismos. Para esto, se introducirán dos conceptos, el de acoplamiento y el de cohesión entre módulos.

Por acoplamiento se entiende la medida de independencia entre módulos, esto es, que ellos puedan funcionar con la menor interconexión o presencia de otro módulo. Lo óptimo es lograr un acoplamiento mínimo.

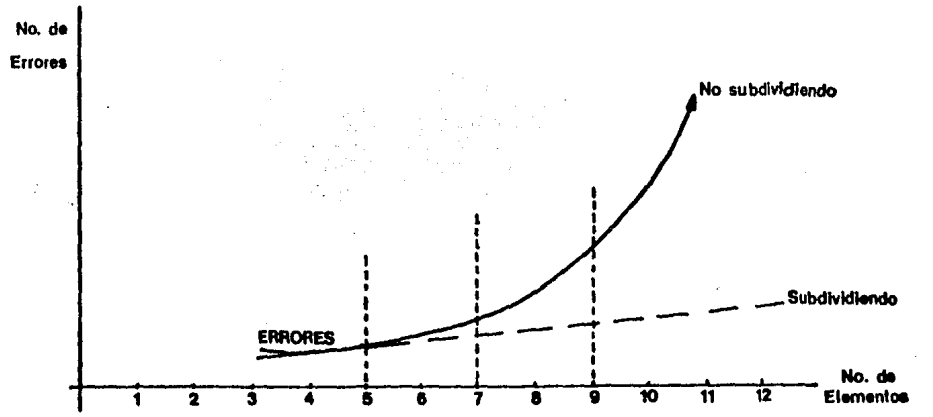
La cohesión es un indicador de que tanto cada módulo es una parte integral y esencial de una única función para la cual fue diseñado. Lo óptimo en este caso es lograr una alta cohesión.



BALANCE OPTIMO
Figura A.2

Otra característica importante que debe considerarse en el diseño de cualquier sistema es el " FAN-OUT ", que indica el número de subordinados inmediatos que posee cierto módulo.

El psicólogo George Miller [9 cap. 5 pags. 68-73] realizó investigaciónes donde pudo determinar las limitaciones humanas en el proceso de información, llegando a la conclusión de que el ser humano sólo puede tener presentes y manipular siete objetos, entidades o conceptos aproximadamente en un tiempo determinado. Como se ilustara en la figura A.3, si se incrementa este número, los errores en el proceso se disparaban desmesuradamente.



NUMERO DE ERRORES POR NUMERO DE ELEMENTOS

Figura A.3

Este estudio se tomó en cuenta para definir y delimitar algunas características del diseño del sistema, las cuales son tratadas en la parte de "Estándares del Sistema".

Al finalizar el diseño estructural, se creará el Diagrama de Estructura, el cual muestra la interrelación de los módulos, su jerarquía, y estará acompañado por una tabla de parámetros de entrada y salida. Haciendo distinción entre los parámetros de control y los de información. Así mismo se contará tanto con el Pseudocódigo, como con la documentación del algoritmo. Dentro de la documentación se dará la función del módulo y demás detalles para tener todos los elementos necesarios para que el paso subsiguiente, el de Programación, pueda efectuar su función.

El tercer paso, el de programación, es el encargado de escribir el código óptimo, apegándose a lo diseñado en el paso anterior. En este paso es de suma importancia que el programador tenga toda la información necesaria para poder transformar, lo diseñado y analizado en la fase de diseño estructurado, en líneas de programación usando el lenguaje adecuado a la aplicación. Para el desarrollo de SIREI, se eligió PASCAL (UCSD), ya que por medio de éste se pueden implementar, de forma inmediata, los conceptos del diseño estructrado, y transportabilidad a otros sistemas con mínimos cambios.

Para finalizar, la fase de pruebas, se encargará de localizar y eliminar los cualquier tipo de error existente en el sistema, de comprobar que el sistema cumpla con lo especificado, o sugerir algún cambio para hacer el sistema más fácil de usar. Para lograr esto, se efectúan pruebas sucesivas, con todo aquello que puede causar problemas, sobre todo en lo que se refiere a los límites que tenga el sistema, para que cuando éste sistema sea liberado, se encuentre libre de errores.

A.1.2 Estándares del Sistema.

Como se describió anteriormente, el método usado para el diseño del sistema es el de Programación Estructurada, con las siguientes características a cumplir:

- Diseño modular de Alta Cohesión y Bajo Acoplamiento. Lo cual reduce significativamente la dificultad de mantenimiento y modificación del sistema, por la independencia de un módulo con respecto a otro.

- Número máximo de 60 instrucciones por módulo. Esto permite que los módulos posean el tamaño ideal, para desarrollar su función, sin que esto se convierta en un problema de mantenimiento del sistema por contar con número elevado de instrucciones.

- Número máximo de 10 parámetros de paso. Cumpliendo este estandar, según el estudio de Miller [9 cap. 5 pags. 68-73], explicado anteriormente, se reduce el número de errores humanos en cualquier modificación a cualquier módulo.

- Número máximo de 4 condiciones anidadas, para evitar confusiones de en el entendimiento de la lógica de programación.

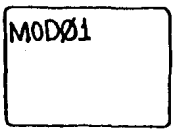
- Máximo nivel de profundidad en la jerarquía de 10. Una mayor profundidad, llevaría al sistema a un falta de claridad en cuanto al flujo control del sistema cual causaría problemas de mantenimiento.

- " Fan Out " máximo de 9. Este es el máximo numero de flujo de control, que puede retener la mente humana sin caer en errores.

A.2. DIAGRAMAS DE ESTRUCTURA

Como apoyo visual al diseño estructrado, se utilizó las representaciones gráficas conocidas como H.I.P.O. (" Hierarchical Input Process Output") [9]. Estas se usaron exclusivamente para representar la interrelación y jerarquía entre módulos.

Las representaciones gráficas y su significado son las siguientes:

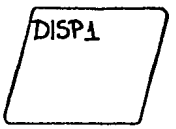


Un módulo simple se representa como un rectángulo con su nombre en la esquina superior izquierda.

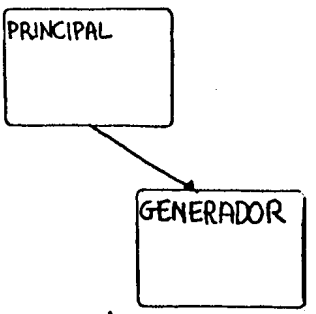
Un módulo predefinido o preexistente.

Un módulo consistente unicamente de elementos de datos.

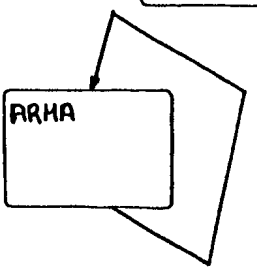
Cualquier dispositivo de Entrada-Salida.



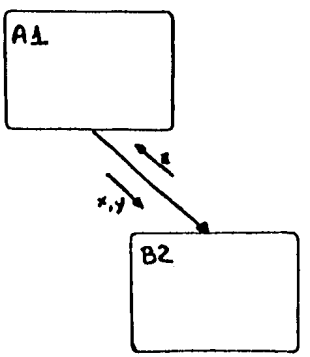
CONECTORES.



Subordinación normal. Existe una referencia de PRINCIPAL a GENERADOR que lo invoca, esto es, existe una subordinación de GENERADOR a PRINCIPAL

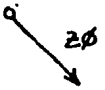


Recursividad. En este caso el módulo ARMA se llama a si mismo creando recursividad. Esto puede darse igualmente en módulos de jerarquías inferiores que llamen recursivamente a su predecesor.

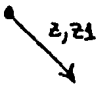


NOTACIONES

Flujo de información. Las anotaciones adyacentes indican la dirección del flujo de la información ya sea de datos o de control.



Flujo de datos.



Flujo de control.

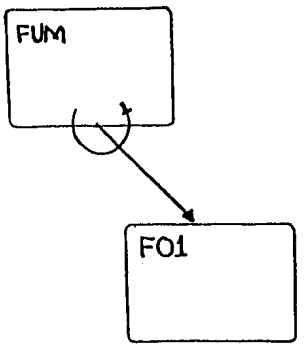
Indice que hace referencia a una tabla de parámetros donde los de control aparecerán subrayados para hacer distinción.

TABLA DE PARAMETROS

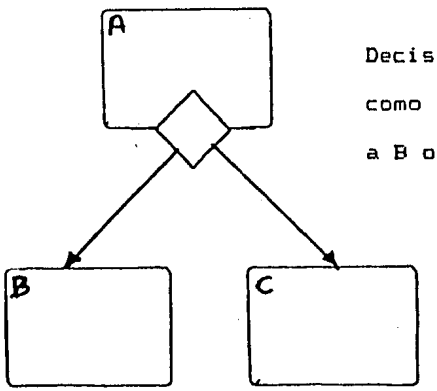
ENTRADA	SALIDA
1:	: LISTA, FIN
2: NOMBRE	: NOMBRE
3: MATRIZ	: FIN

Comentarios. Cualquier nota aclaratoria se ilustrará de la siguiente forma.

PROCEDIMIENTO.



Iteración. Aqui FO1 se repite como subrutina.



Decisión. La referencia marcada se usa como un proceso de decisión en A para ir a B o C.

APENDICE B

DIAGRAMAS DE ESTRUCTURA

Se incluye este apéndice como ayuda a aquellas personas que deseen conocer la manera como se implementó el sistema. Consultando diagramas se puede seguir el flujo del sistema y además, con las tablas de parámetros queda más clara la finalidad de cada una de las variables y las rutinas que las manejan.

A continuación se definen cada una de las variables utilizadas e inmediatamente después, se incluyen los diagramas de estructura del sistema y sus tablas de parámetros.

escenario	:	Imagen generada en la pantalla.
rfcg	:	Representación de la figura compuesta generada.
existe	:	Indica si la RFCG ya existe dentro de la BCFC.
ptralta	:	Indicador del lugar que ocupa la RFCG en la BCFG.
mat-crt	:	Matriz gráfica (arreglo booleano que contiene la información de la pantalla).
vec-entrada	:	Vector de entrada a las RPs de las FBs en memoria principal.
vec-funcion	:	Vector que contiene las funciones de las FBs en memoria principal.
ultimo	:	Indicador del número de FBs existentes en el sistema.

punto	:	Variable que contiene la posición dentro de la matriz gráfica del último punto considerado durante el reconocimiento.
indice-rfcg	:	Indicador de la longitud actual de la RFCG.
correcta	:	Indica si la información de una FB considerada como la que representa a la FB en análisis, se pudo recorrer.
a-regla	:	Variable "buffer" del archivo de la BCFB.
reng	:	Contiene la posición del renglón en análisis dentro de la MG.
sren	:	Posición del subrenglón en análisis de la matriz gráfica.
inre	:	Localidad inicial para el renglón actual de la pantalla gráfica.
inigr	:	Localidad inicial de la pantalla gráfica.
cont3	:	Contador auxiliar.
funcion	:	Función de la RP de la FB actual.
encontro	:	Indica si el punto que se desea considerar como vértice realmente lo es o nó.
sujeto1	:	Es uno de los sujetos que constituye el vértice inicial de la FB en análisis.
sujeto2	:	Es el otro sujeto que constituye el vértice inicial de la FB en análisis.
regla	:	Indicador de la FB que está siendo considerada.

lado	:	Indicador del lado de la RP de la FB actual que se considera como inicial.
inf-temporal	:	Contiene la información de la posición y de los sujetos de la FB actual.
recorre	:	Punto actual (coordenada dentro de la MG) del recorrido de la FB en análisis.
exito	:	Indica si la función de la FB en análisis se cumplió.
col	:	Columna (dentro de la MG) del punto a borrar.
ren	:	Renglón (dentro de la MG) del punto a borrar.
fijo	:	Posición del renglón o de la columna que debe mantenerse fijo en el barrido.
movili	:	Posición inicial (del renglón o de la columna) que va a modificarse.
movilf	:	Posición final (del renglón o de la columna) que va a modificarse.
sujeto	:	Sujeto que está siendo recorrido.
x1	:	Coordenada inicial en x del primer sujeto cuando se prueba su función.
x2	:	Coordenada final en x del primer sujeto cuando se prueba su función.
x3	:	Coordenada inicial en x del segundo sujeto cuando se prueba su función.
x4	:	Coordenada final en x del segundo sujeto cuando se prueba su función.

y1 : Coordenada inicial en y del primer sujeto cuando se prueba su función.

y2 : Coordenada final en y del primer sujeto cuando se prueba su función.

y3 : Coordenada inicial en y del segundo sujeto cuando se prueba su función.

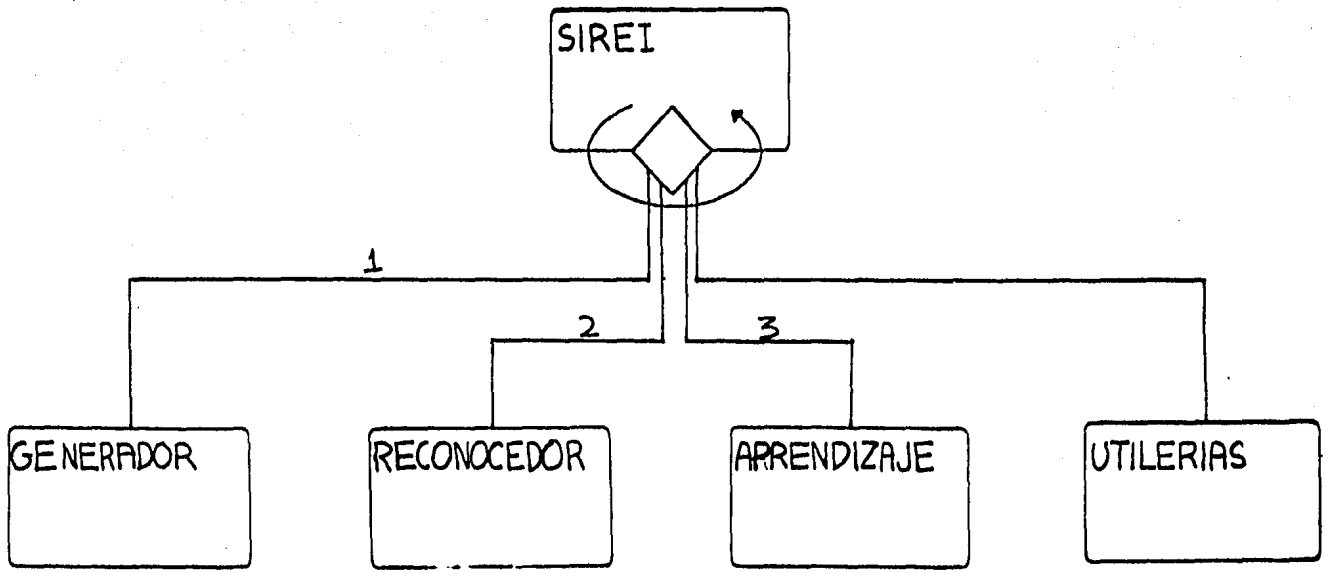
y4 : Coordenada final en y del segundo sujeto cuando se prueba su función.

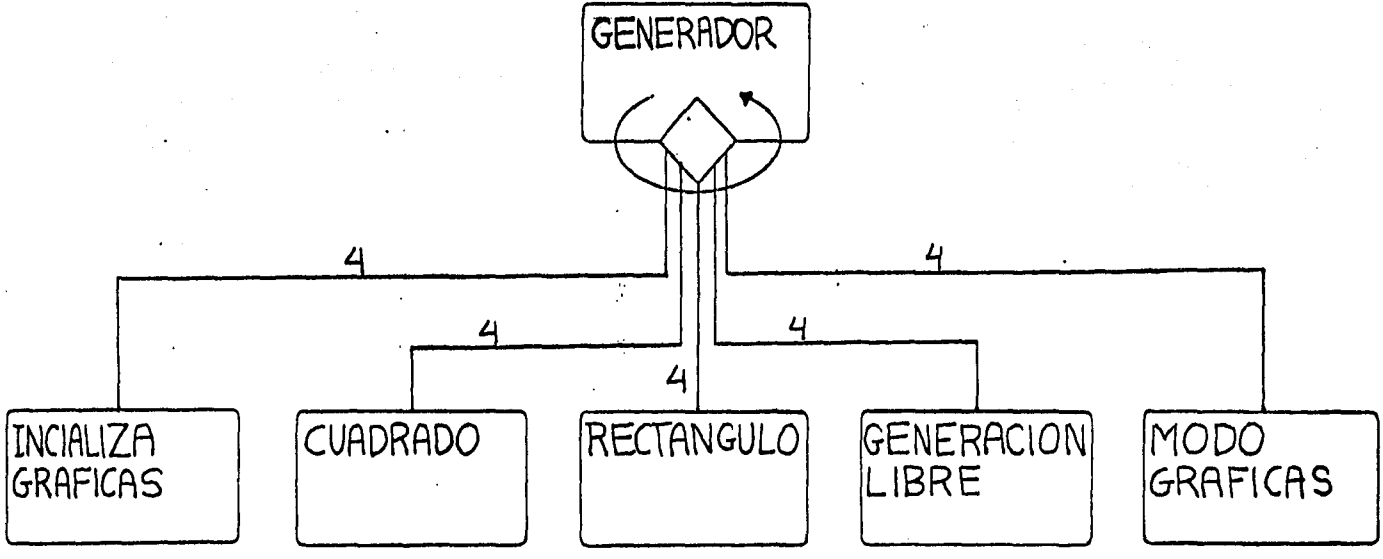
Tabla B1. Parámetros de Paso

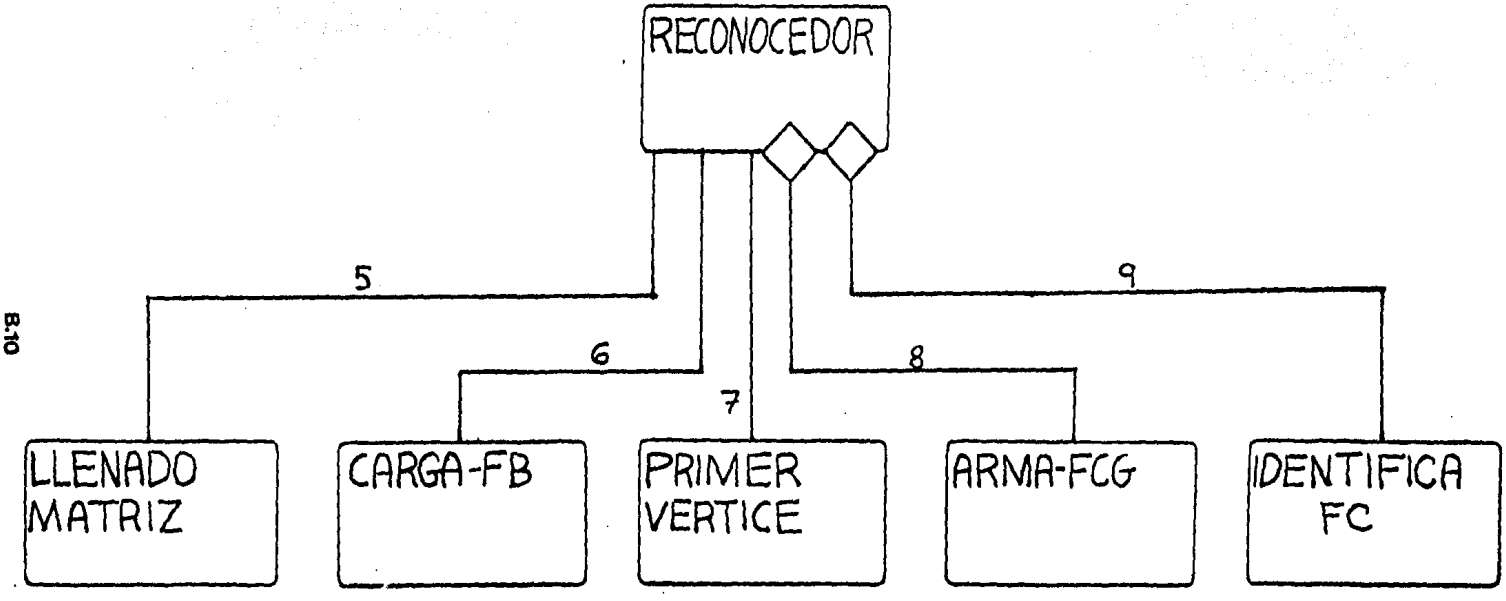
INDICE	ENTRADA	SALIDA
1		escenario
2	escenario	rfcg existe ptralta
3	existe ptralta rfcg	existe
4	escenario	escenario
5	escenario	mat-crt
6		vec-entrada vec-funcion ultimo
7	mat-crt	existe punto
8	punto rfcg indice-rfcg mat-crt vec-funcion a-regla vec-entrada ultimo	punto rfcg indice-rfcg correcta
9	rfcg	existe ptralta
10		reng sren inre inigr
11	reng cont3 sren col	mat-crt
12		funcion

INDICE	ENTRADA	SALIDA
13	mat-crt punto	encontro sujeto1 sujeto2
14	sujeto1 sujeto2 regla ultimo vec-entrada	sujeto1 sujeto2 regla lado ptr-regla
15	punto ptr-regla lado vec-funcion mat-crt	inf-temp exito
16	mat-crt inf-temporal	mat-crt
17	rfcg indice-rfcg regla	rfcg indice-rfcg
18	indice-rfcg rfcg punto vec-entrada vec-funcion inf-temporal mat-crt ultimo	indice-rfcg rfcg correcta
19	sujeto1 sujeto2	sujeto1 sujeto2
20	inf-temporal n-lado mat-crt sujeto1 sujeto2	recorre n-lado error
21	lado inf-temporal funcion	exito
22	escenario col ren	escenario

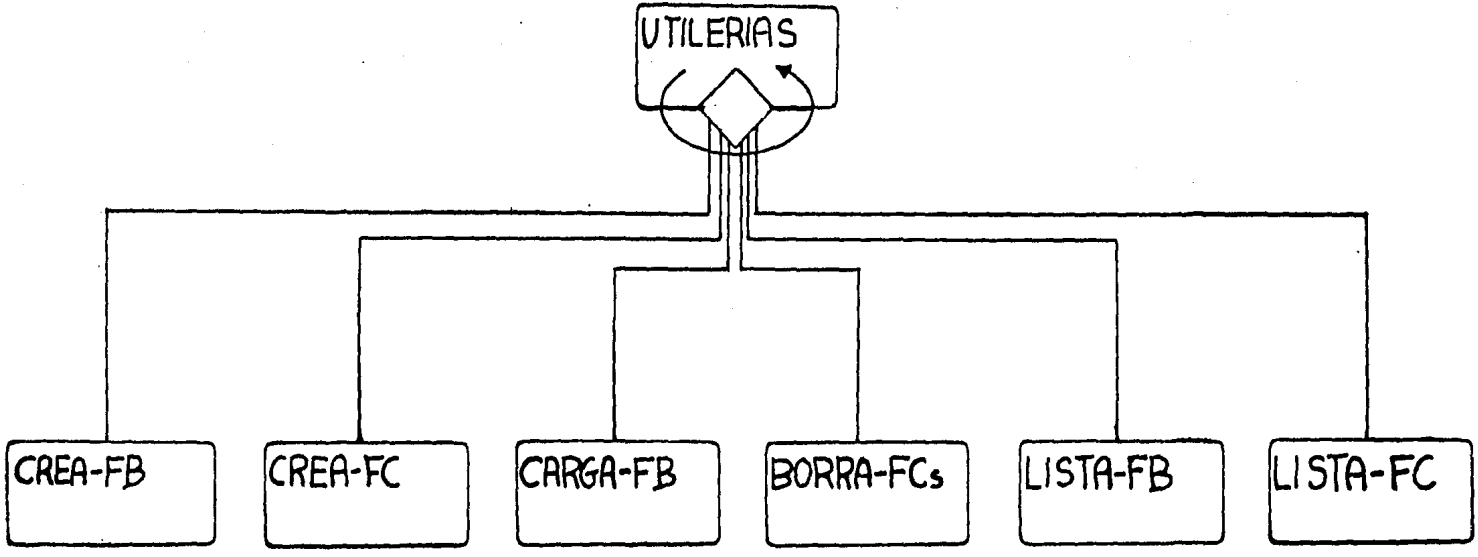
INDICE	ENTRADA	SALIDA
23	punto indice-rfcg rfcg fijo movili movilf sujeto mat-crt	punto indice-rfcg rfcg correcta
24	x1, x2, x3, x4 y1, y2, y3, y4	exito
25	punto sujeto mat-crt	punto

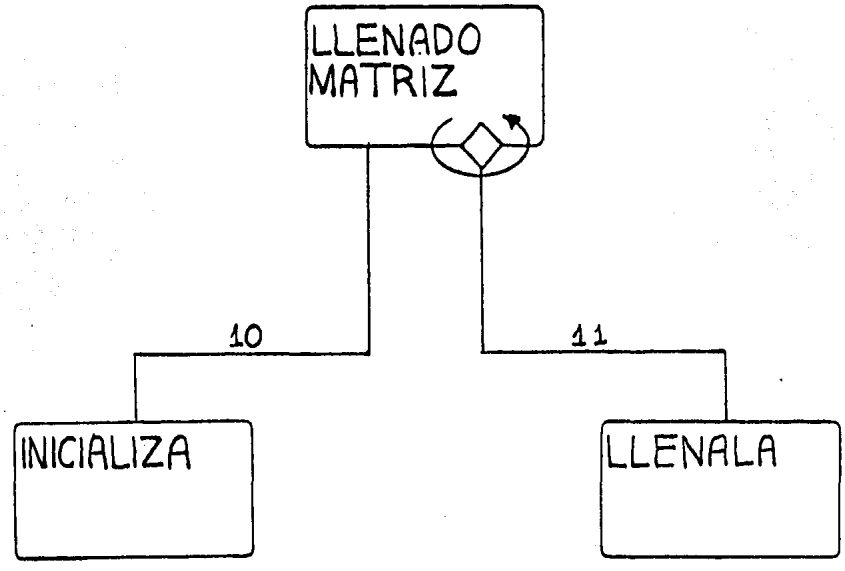

SIREI
1
2
3
GENERADOR
RECONOCEDOR
APRENDIZAJE
UTILERIAS

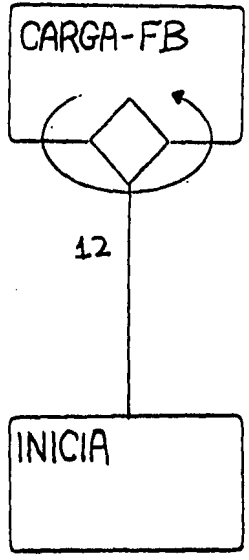

GENERADOR
4
4
4
4
4
INCIALIZA GRAFICAS
CUADRADO
RECTANGULO
GENERACION LIBRE
MODO GRAFICAS

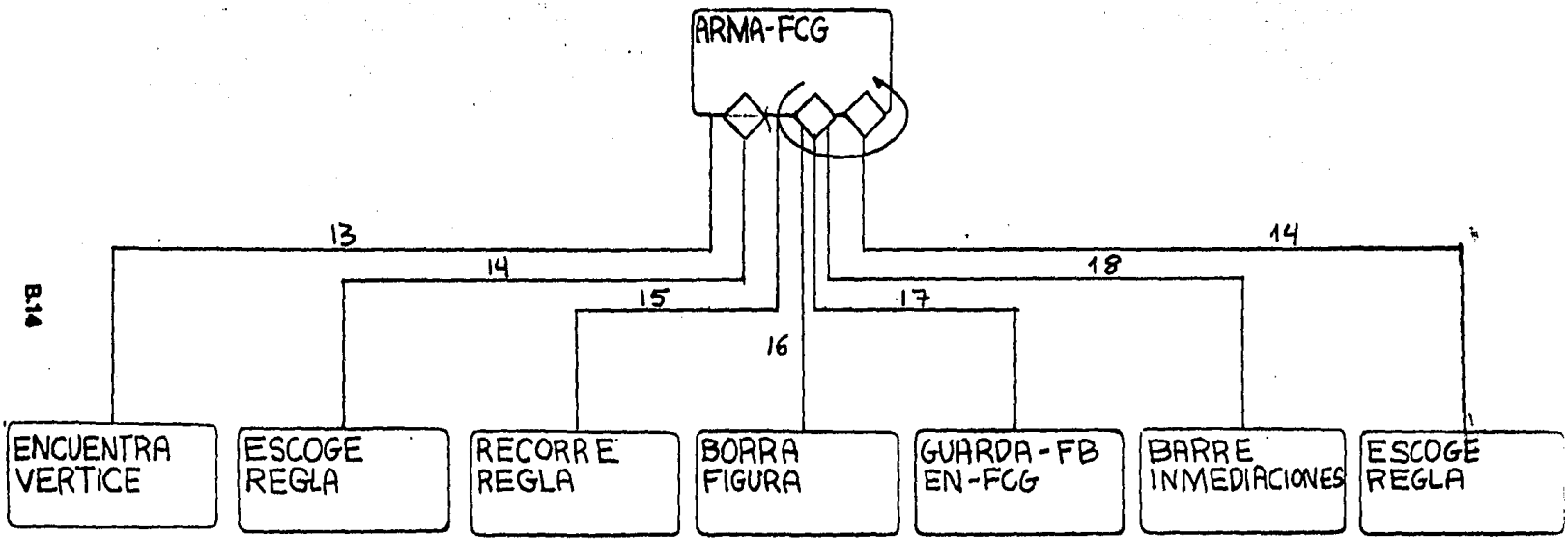

RECONOCEDOR
5
6
7
8
9
LLENADO MATRIZ
CARGA-FB
PRIMER VERTICE
ARMA-FCG
IDENTIFICA FC

B.10

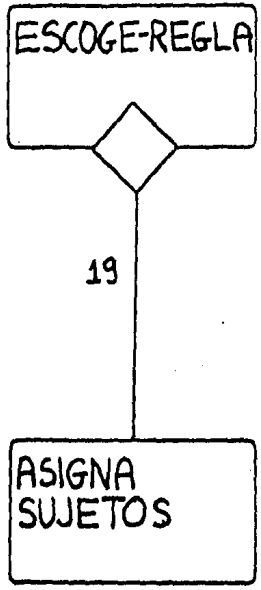

UTILERIAS
CREA-FB
CREA-FC
CARGA-FB
BORRA-FCs
LISTA-FB
LISTA-FC

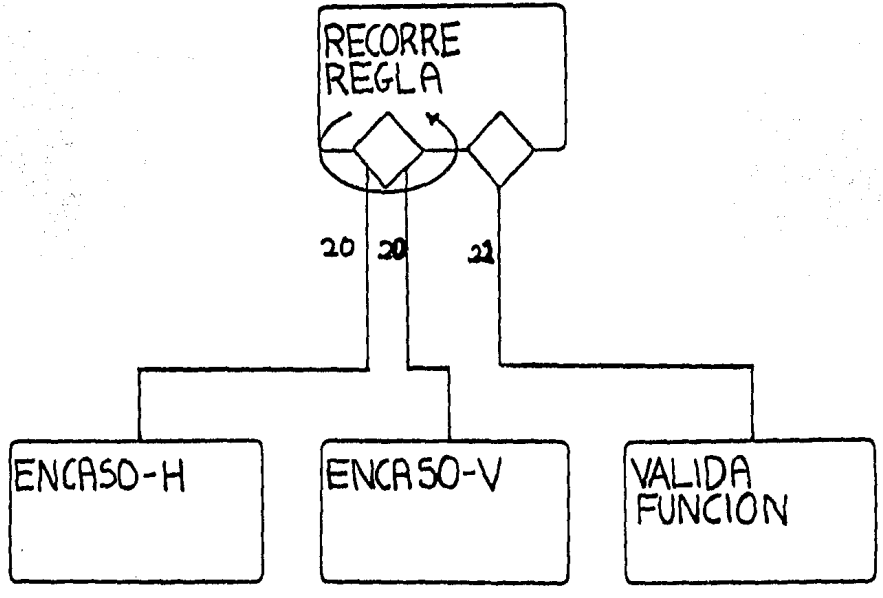

LLENADO
MATRIZ
10
11
INICIALIZA
LLENALA


CARGA-FB
12
INICIA

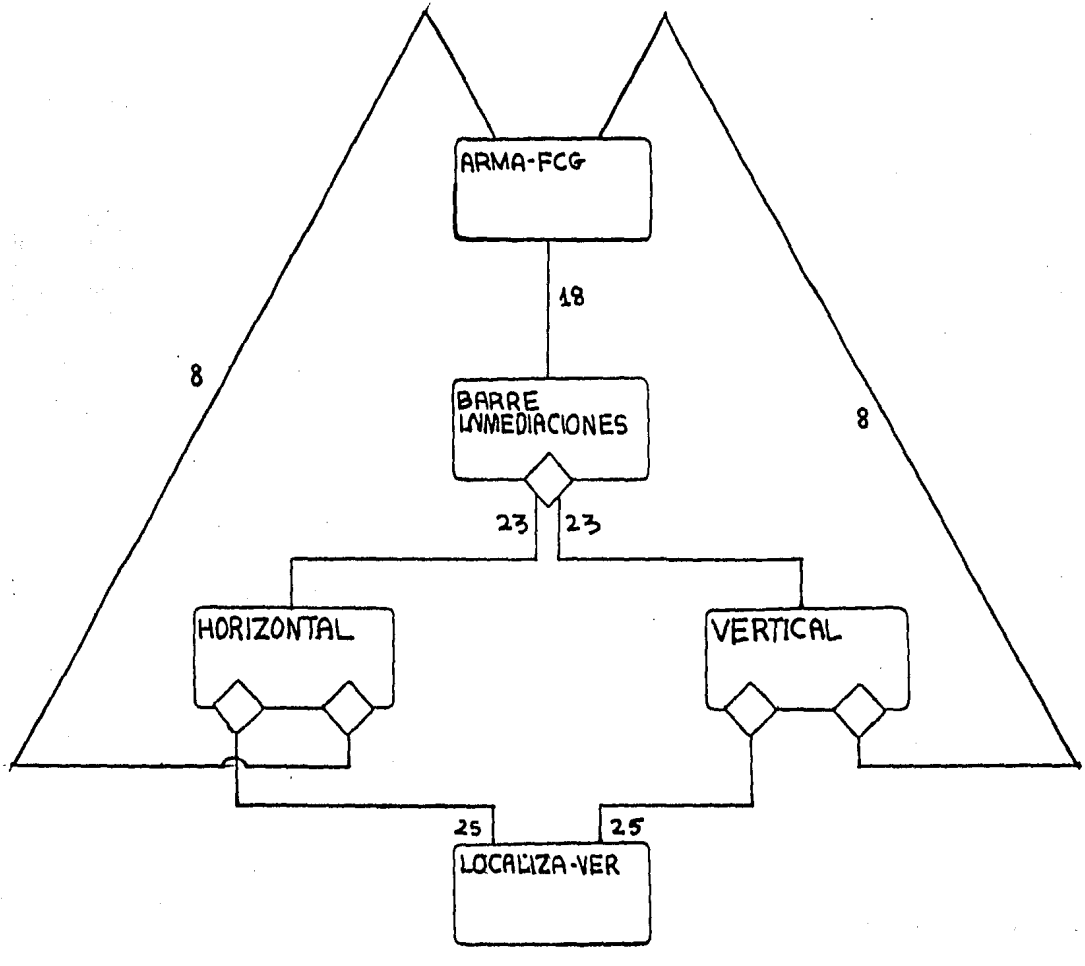

ARMA-FCG
13
14
15
16
17
18
14
ENCUENTRA VERTICE
ESCOGE REGLA
RECORRE REGLA
BORRA FIGURA
GUARDA-FB EN-FCG
BARRE INMEDIACIONES
ESCOGE REGLA

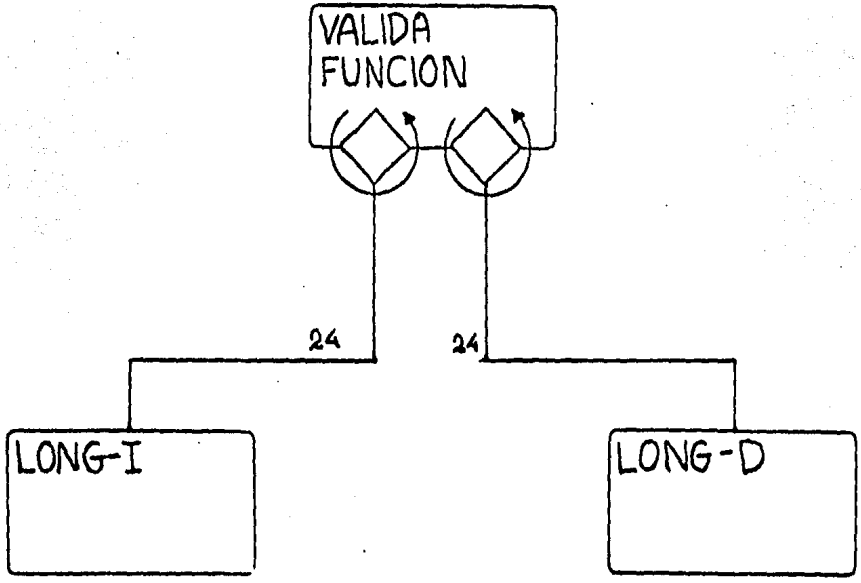
B.14


ESCOGE-REGLA
19
ASIGNA
SUJETOS


RECORRE
REGLA
20
20
21
ENCASO-H
ENCASO-V
VALIDA
FUNCION

BORRA
FIGURA
22
DIBUJA
PUNTO


ARMA-FCG
18
8
8
BARRE
LNMEDIACIONES
23
23
HORIZONTAL
VERTICAL
25
25
LOCALIZA-VER


VALIDA
FUNCION
24
24
LONG-I
LONG-D

APENDICE C

LISTADOS DE PROGRAMAS DEL SISTEMA SIREI

C.1 Programa Principal

C.2 Generador

C.3 Reconocedor

C.4 Aprendizaje

C.5 Utilerías

```
{ ================================================================== }
{                                                                    }
{                  UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO           }
{                                                                    }
{                          FACULTAD DE INGENIERIA                    }
{                                                                    }
{                                                                    }
{                                                                    }
{                                T E S I S                           }
{                                                                    }
{     RECONOCIMIENTO DE FORMAS CON TECNICAS DE INTELIGENCIA ARTIFICIAL }
{                                                                    }
{                                                                    }
{     PARTICIPANTES :        Alvaro Estandia Gonzalez Luna           }
{                            Rafael Guzman Mejia                     }
{                            Jorge Ruiz Garza                        }
{                            Hector Zavala Luna                      }
{                                                                    }
{     DIRECTOR      :        Dr. Francisco Cervantes                 }
{                                                                    }
{ ================================================================== }
{                                                                    }
{     MODULO : PROGRAMA PRINCIPAL                                    }
{              ------------------                                    }
{                                                                    }
{     Este modulo es el controlador de todo el sistema y permite el  }
{     acceso a los cuatro componentes principales del mismo que son :}
{                                                                    }
{              1 .-  GENERADOR DE FIGURAS                            }
{                                                                    }
{              2 .-  RECONOCEDOR DE FIGURAS                          }
{                                                                    }
{              3 .-  APRENDIZAJE DE FIGURAS                          }
{                                                                    }
{              4. -  UTILERIAS                                       }
{                                                                    }
{ ================================================================== }

{$S+}
{$Q+}
{ 13 DE ABRIL   DE 1985} { 18 DE MAYO  DE 1985 }
{ TESIS : PROGRAMA PRINCIPAL                    }

PROGRAM PRINCIPAL;
USES
    CHAINSTUFF,
    {$U #5:CADENAS.CODE} CADENAS;

VAR
    OPCION    : CHAR;
    FECHAHOY : STRING;
```

```
PROCEDURE PANT_SUP1;
{ PANT_SUP1 }
    BEGIN
    PAGE (OUTPUT);
    PROMPT ( 0, 0,FECHAHOY);
    PROMPT (21, 0,'RECONOCIMIENTO DE FORMAS');
    PROMPT (63, 0,'MODULO PRINCIPAL')
    END
{ ENDPRO };

PROCEDURE MENU;
{ MENU }
    BEGIN
    PANT_SUP1;
    PROMPT (24, 7,'0 - Fin');
    PROMPT (24, 9,'1 - Generador');
    PROMPT (24, 11,'2 - Reconocedor');
    PROMPT (24,13,'3 - Aprendizaje');
    PROMPT (24,15,'4 - Utilerias  ');
    PROMPT (24,19,'Teclea la opcion que desees - > ')
    END
{ ENDPRO };

{ PRINCIPAL }
    BEGIN
    GETCVAL (FECHAHOY);
    IF LENGTH (FECHAHOY) = 0 THEN { ES PRIMERA VEZ }
        BEGIN
        PAGE (OUTPUT);
        PROMPT (15,9,'TECLEA LA FECHA DEL DIA (AA/MM/DD -> ');
        OBTCAD ( 8,52,9,['0'..'9','/'],FECHAHOY)
        END
    { ENDIF };
    MENU;
    OPCION := GETCHAR (57,19,['0'..'4']);
    SETCVAL (FECHAHOY);
    CASE OPCION OF
        '1':SETCHAIN ('#4:GENERADOR.CODE' );
        '2':SETCHAIN ('#4:RECONOCE.CODE');
        '3':SETCHAIN ('#4:APREN.CODE');
        '4':SETCHAIN ('#4:UTILERIAS.CODE')
        END
    { ENDCASE }
    END
{ ENDPROGRAM }.
```

```
{ ========================================================================
{
{                    UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO
{
{                            FACULTAD DE INGENIERIA
{
{
{
{                                  T E S I S
{
{       RECONOCIMIENTO DE FORMAS CON TECNICAS DE INTELIGENCIA ARTIFICIAL
{
{
{       PARTICIPANTES :         Alvaro Estandia Gonzalez Luna
{                               Rafael Guzman Mejia
{                               Jorge Ruiz Garza
{                               Hector Zavala Luna
{
{       DIRECTOR      :         Dr. Francisco Cervantes
{
{ ========================================================================
{
{       MODULO : GENERADOR DE FIGURAS
{                ---------------------
{
{       Este modulo permite al usuario la generacion de figuras en la
{       pantalla.  Estas figuras quedan almacenadas en la "pagina cero"
{       de graficas de la memoria de la computadora.
{
{       El usuario puede combinar figuras basicas para realizar figuras
{       compuestas.
{
{       Este modulo realiza validaciones sobre las dimensiones de las
{       figuras de acuerdo con las dimensiones maximas de la pantalla
{       ( 192 renglones por 280 columnas )
{
{ ========================================================================

{$S+}
{ 10 DE ENERO DE 1985     }
{ TESIS : RECONOCIMIENTO DE FORMAS
          PROGRAMA GENERADOR DE FIGURAS }

PROGRAM FIGURAS;

USES
    TURTLEGRAPHICS, CHAINSTUFF,
    {$U #5:CADENAS.CODE} CADENAS;

CONST
    FIN = 999;       { si me arrepiento de dibujar una figura }
```

```
    MIN = 5;          { minimo tamano en pixels de una figura }
    SEN_60 = 0.866;
    CRUZX = 20;       (* TAMANO DE CRUZ EN X *)
    CRUZY = 20;       (* TAMANO DE LA CRUZ EN Y *)

TYPE
    TCRUZ = PACKED ARRAY [0:CRUZX,0:CRUZY] OF BOOLEAN;

VAR
    OPCION   : CHAR;
    FECHAHOY : STRING;
    CRUZ     : TCRUZ;
    X,Y      : INTEGER;

PROCEDURE MENU;

    PROCEDURE PANT_SUP1;
    { PANT_SUP1 }
        BEGIN
        PAGE (OUTPUT);
        PROMPT ( 0, 0,FECHAHOY);
        PROMPT (24, 0,'RECONOCIMIENTO DE FORMAS');
        PROMPT (59, 0,'GENERADOR DE FIGURAS')
        END
    { ENDPRO };

{ MENU }
    BEGIN
    PANT_SUP1;
    PROMPT (24, 5,'0 - Fin');
    PROMPT (24, 7,'1 - Inicializa graficas');
    PROMPT (24, 9,'2 - Cuadrado');
    PROMPT (24,11,'3 - Rectangulo');
    PROMPT (24,13,'4 - Generacion libre');
    PROMPT (24,15,'5 - Modo grafica');
    PROMPT (24,19,'Teclea la opcion que desees - > ')
    END
{ ENDPRO };



PROCEDURE PINTACRUZ ( VAR X,Y :INTEGER;
                          CRUZ:TCRUZ);

VAR
    SIZE,
    X1,
    Y1,
    INC1,
    INC2,
    I,
```

```
    J,
    DECX,
    DECY : INTEGER;

(* PINTACRUZ *)
    BEGIN
    INC1:=0;
    INC2:=0;
    DECX:=10;
    DECY:=10;
    X1:=X-DECX;
    Y1:=Y-DECY;
    IF X1< 0 THEN
        BEGIN
        INC1:=ABS(X1);
        X1:=X1+INC1
        END
    (* ENDIF *);
    IF Y1 < 0 then
        BEGIN
        FOR I:=0 TO CRUZX DO
            FOR J:=0 TO CRUZY DO
                CRUZ[I,J]:=FALSE
            (* END FOR *);
        (* ENDFOR *)
        FOR I:=0 TO CRUZX DO
            CRUZ[I,DECX]:=TRUE
        (* ENDFOR *);
        FOR J:=0 TO CRUZY DO
            CRUZ[Y,J]:=TRUE
        (* ENDFOR *);
        INC2 :=ABS(Y1);
        Y1:=Y1+INC2
        END
    (* ENDIF *);
    SIZE:=2*((CRUZX+1+15) DIV 16);
    DRAWBLOCK(CRUZ,SIZE,INC1,7,CRUZX-INC1,CRUZY-INC2,X1,Y1,6)
    END
(* ENDPRO *);

PROCEDURE HAZCRUZ( VAR  CRUZ : TCRUZ);

VAR
    I,
    J    : INTEGER;

(* HAZCRUZ *)
    BEGIN
    FOR I:=0 TO CRUZX DO
        FOR J:=0 TO CRUZY DO
            CRUZ[I,J]:=FALSE
```

```
        (* ENDFOR *);
    (* ENDFOR *);
    FOR I:=0 TO CRUZX DO
        CRUZ[I,(CRUZY DIV 2)]:=TRUE
    (* ENDFOR *);
    FOR J:=0 TO CRUZY DO
        CRUZ[(CRUZX DIV 2),J]:=TRUE
    (* ENDFOR *)
    END
(* ENDPRO *);


PROCEDURE MUEVECRUZ ( VAR X,Y    : INTEGER;
                          CRUZ   : TCRUZ   );

VAR
    A    : CHAR;
    FLAG : BOOLEAN;

(* MUEVECRUZ *)
    BEGIN
    FLAG := TRUE;
    WHILE FLAG DO
        BEGIN
        REPEAT
            READ(A);
            WRITELN;
            IF A='P' THEN
                BEGIN
                TEXTMODE;
                PAGE (OUTPUT);
                PROMPT (17,16,'Posicion actual de la cruz ');
                WRITELN (' REN = ',Y:3,'COL = ',X:3);
                READLN;
                GRAFMODE
                END
            (* ENDIF *);
        UNTIL
            (A='S') OR (A='D') OR
            (A='E') OR (A='X') OR
            (A=' ');
        PINTACRUZ(X,Y,CRUZ);
        IF A='S' THEN
            X:=PRED(X)
        (* ENDIF *);
        IF A='D' THEN
            X:=SUCC(X)
        (* ENDIF *);
        IF A='E' THEN
            Y:=SUCC(Y)
        (* ENDIF *);
```

```
        IF A='X' THEN
            Y:=PRED(Y)
        (* ENDIF *);
        IF X < 0 THEN
            X:=0
        (* ENDIF *);
        IF X > 278 THEN
            X:=278
        (* ENDIF *);
        IF Y < 0 THEN
            Y:=0
        (* ENDIF *);
        IF Y > 190 THEN
            Y:=190
        (* ENDIF *);
        IF A=' ' THEN
            FLAG:=FALSE
        ELSE
            PINTACRUZ(X,Y,CRUZ)
        (* ENDIF *);
        END
    (* ENDWHILE *);
    READLN;
    END
(* ENDPRO *);

PROCEDURE CUADRADO;
VAR
    X,Y,L:INTEGER;

    FUNCTION CHECA_CUAD (X,Y,L:INTEGER):BOOLEAN;
    VAR
        ERROR : BOOLEAN;
    (CHECA_CUAD)
        BEGIN
        CHECA_CUAD := (X < 0 ) OR (Y<0) OR
                      (X+L > 279) OR (Y+L > 191) OR
                      ( L < MIN )
        END
    ( ENDFUNCTION );

( CUADRADO )
    BEGIN
    REPEAT
      PAGE (OUTPUT);
      TEXTMODE;
      PROMPT (24, 7,'Esquina (COL) -> ');PUNTOS (3,42, 7);X :=OBTENT(3,42, 7)
      PROMPT (24, 8,'Esquina (REN) -> ');PUNTOS (3,42, 8);Y :=OBTENT(3,42, 8)
      PROMPT (24, 9,'Lado          -> ');PUNTOS (3,42, 9);L :=OBTENT(3,42, 9)
      IF ( X <> FIN ) AND NOT CHECA_CUAD(X,Y,L) THEN
        BEGIN
```

```
        GRAFMODE;
        HAZCRUZ(CRUZ);
        PINTACRUZ(X,Y,CRUZ);
        MUEVECRUZ(X,Y,CRUZ);
        IF NOT CHECA_CUAD (X,Y,L) THEN
          BEGIN
          GRAFMODE;
          PENCOLOR (NONE);
          MOVETO (X,Y);
          PENCOLOR (WHITE);
          TURNTO (0); { VIENDO A LA DERECHA }
          MOVE (L);TURN (90);MOVE (L);
          TURN (90);MOVE (L);TURN (90);MOVE (L)
          END
        ELSE
          BEGIN
          TEXTMODE;
          PAGE (OUTPUT);
          PROMPT (15,22,'* *  La figura sale de la pantalla (RETURN)  * *');
          READLN
          END
        { ENDIF }
        END
      ELSE
        IF X <> FIN THEN
          BEGIN
          TEXTMODE;
          PROMPT (23,22,'* *  Error en datos (RETURN)  * *');
          READLN
          END
        { ENDIF }
      { ENDIF };
      UNTIL
        (X = FIN) OR NOT CHECA_CUAD (X,Y,L);
      GRAFMODE;
      READLN
    END
{ ENDPRO };

PROCEDURE RECTANGULO;
VAR
    X,Y,B,H:INTEGER;

    FUNCTION CHECA_RECT (X,Y,B,H:INTEGER):BOOLEAN;
    VAR
        ERROR : BOOLEAN;
    {CHECA_RECT}
        BEGIN
        CHECA_RECT := (X < 0) OR (X+B > 279) OR
                      (Y < 0) OR (Y+H > 191) OR
                      (B < MIN) OR (H < MIN)
```

```
        END
    { ENDFUNCTION };

{ RECTANGULO }
    BEGIN
    REPEAT
      PAGE (OUTPUT);
      PROMPT (24, 7,'Esquina (COL) -> ');PUNTOS (3,42, 7);X :=OBTENT(3,42, 7)
      PROMPT (24, 8,'Esquina (REN) -> ');PUNTOS (3,42, 8);Y :=OBTENT(3,42, 8)
      PROMPT (24, 9,'Base          -> ');PUNTOS (3,42, 9);B :=OBTENT(3,42, 9)
      PROMPT (24,10,'Altura        -> ');PUNTOS (3,42,10);H :=OBTENT(3,42,10)
    IF (X <> FIN) AND NOT CHECA_RECT(X,Y,B,H) THEN
      BEGIN
      GRAFMODE;
      HAZCRUZ(CRUZ);
      PINTACRUZ(X,Y,CRUZ);
      MUEVECRUZ(X,Y,CRUZ);
      IF NOT CHECA_RECT(X,Y,B,H) THEN
        BEGIN
        PENCOLOR (NONE);
        MOVETO (X,Y);
        PENCOLOR (WHITE);
        TURNTO (0); { VIENDO A LA DERECHA }
        MOVE (B);TURN (90);MOVE (H);
        TURN (90);MOVE (B);TURN (90);MOVE (H)
        END
      ELSE
        BEGIN
        TEXTMODE;
        PAGE (OUTPUT);
        PROMPT (15,22,'* *  La figura sale de la pantalla (RETURN)  * *');
        READLN
        END
      { ENDIF }
      END
    ELSE
      IF X <> FIN THEN
        BEGIN
        PROMPT (23,22,'* *  Error en datos (RETURN)  * *');
        READLN
        END
      { ENDIF }
    { ENDIF };
    UNTIL
      (X = FIN) OR NOT CHECA_RECT (X,Y,B,H);
    GRAFMODE;
    READLN
    END
{ ENDPRO };

PROCEDURE GENERA_LIBRE;
```

```
VAR
    X1,Y1,X2,Y2:INTEGER;

    FUNCTION CHECA_LINEA (X1,Y1,X2,Y2:INTEGER):BOOLEAN;
    VAR
        ERROR : BOOLEAN;
    {CHECA_LINEA}
        BEGIN
        ERROR := (X1 < 0) OR (X1 > 279) OR
                 (X2 < 0) OR (X2 > 279) OR
                 (Y1 < 0) OR (Y1 > 191) OR
                 (Y2 < 0) OR (Y2 > 191) OR
                 (SQR(X1-X2)+SQR(Y1-Y2) < SQR(MIN));
        IF ERROR THEN
            BEGIN
            PROMPT (24,11,'Error en datos (teclea <RETURN>');
            READLN
            END
        { ENDIF };
        CHECA_LINEA := ERROR
        END
    { ENDFUNCTION };

{ GENERA_LIBRE }
    BEGIN
    PAGE (OUTPUT);
    REPEAT
      PROMPT (24, 7,'Punto 1    (X) -> ');PUNTOS (3,42, 7);X1:=OBTENT(3,42, 7)
      PROMPT (24, 8,'Punto 1    (Y) -> ');PUNTOS (3,42, 8);Y1:=OBTENT(3,42, 8)
      PROMPT (24, 9,'Punto 2    (X) -> ');PUNTOS (3,42, 9);X2:=OBTENT(3,42, 9)
      PROMPT (24,10,'Punto 2    (Y) -> ');PUNTOS (3,42,10);Y2:=OBTENT(3,42,10)
    UNTIL
        (X1 = FIN) OR NOT CHECA_LINEA (X1,Y1,X2,Y2);
    IF X1 <> FIN THEN
        BEGIN
        GRAFMODE;
        PENCOLOR (NONE);
        MOVETO (X1,Y1);
        PENCOLOR (WHITE);
        MOVETO (X2,Y2)
        END
    { ENDIF }
    END
{ ENDPRO };

{ FIGURAS }
    BEGIN
    GETCVAL (FECHAHOY); { obtiene la fecha que le manda el MODULO PRINCIPAL }
    MENU;
    OPCION := GETCHAR (57,19,['0'..'5']);
    WHILE OPCION <> '0' DO
```

```
        BEGIN
        CASE OPCION OF
             '1':INITTURTLE;
             '2':CUADRADO;
             '3':RECTANGULO;
             '4':GENERA_LIBRE;
             '5':BEGIN GRAFMODE; READLN END;
             END
        { ENDCASE };
        TEXTMODE;
        IF (OPCION <> '1') AND (OPCION <> '5') THEN MENU;
        OPCION := GETCHAR (57,19,['0'..'5'])
        END
    { ENDWHILE };
    SETCHAIN ('#4:PRINCIPAL.CODE') { regresa el MODULO PRINCIPAL }
    END
{ ENDPROGRAM }.
```

```
{ 16 DE MARZO 1986 }
(*$Q+*)
(*$S+*)
PROGRAM RECONOCEDOR  ;
USES TURTLEGRAPHICS,CHAINSTUFF;

CONST
    MNFB = 26; {maximo numero de figuras basicas}
    MNL  = 10; {maximo numero de lados por figura}
    MNI  = 10; {maximo numero de igualdades y/o desigualdades por figura}

TYPE
    T_SUJETO            = RECORD
                          MODIFICADOR       : CHAR;
                          ITEM              : CHAR
                          END;
   T_OPERADOR           = CHAR;
   T_PTR_SUJETO         = ^T_NODO_SUJETO;
   T_PTR_OPERADOR       = ^T_NODO_OP;
   T_NODO_SUJETO        = RECORD
                          SUJETO            : T_SUJETO;
                          SIG_OP            : T_PTR_OPERADOR
                          END;
   T_NODO_OP            = RECORD
                          OPERADOR          : T_OPERADOR;
                          SIG_SUJ           : T_PTR_SUJETO
                          END;
   T_VEC_ENTRADA        = ARRAY [1:MNFB] OF T_PTR_SUJETO;
   T_IGUALDAD           = RECORD
                          PRIMER            : INTEGER;
                          SEGUNDO           : INTEGER
                          END;
   T_REGLA              = RECORD
                          NOMBRE            : STRING [15];
                          LADO_IZQ          : CHAR;
                          LADO_DER          : RECORD
                                  SUJETO    : ARRAY [1:MNL] OF T_SUJETO;
                                  OPERADOR  : ARRAY [1:MNL] OF T_OPERADOR;
                                  IGUALDAD  : ARRAY [1:MNI] OF T_IGUALDAD;
                                  DESIGUAL  : ARRAY [1:MNI] OF T_IGUALDAD
                                  END
                          END;
   T_FCG                = STRING[30];
   T_REGPAR             = RECORD
                          EXISTE            : BOOLEAN;
                          POSICION          : INTEGER;
                          FC                : T_FCG;
                          END;
   T_A_REG              = FILE OF T_REGPAR;
   T_A_REGLA            = FILE OF T_REGLA;
   T_FUNCION            = RECORD
```

```
                              IGUALDAD          : ARRAY [1:MNI] OF TIGUALDAD;
                              DESIGUAL          : ARRAY [1:MNI] OF T_IGUALDAD
                              END;
    T_VEC_FUNCION         = ARRAY [1:MNFB] OF T_FUNCION;
    T_INF_TEMP            = RECORD
                              PX                : ARRAY [1:MNL] OF INTEGER;
                              PY                : ARRAY [1:MNL] OF INTEGER;
                              S                 : ARRAY [1:MNL] OF T_SUJETO;
                              B                 : ARRAY [1:MNL] OF BOOLEAN
                              END;
    VECTOR                = PACKED ARRAY [0:279] OF BOOLEAN;
    PANTALLA              =        ARRAY [0:191] OF VECTOR;
    T_PUNTO               = RECORD
                              R                 : INTEGER;
                              C                 : INTEGER
                              END;
    T_A_FC                = FILE OF T_FCG ;


VAR
    FECHAHOY : STRING;
    MAT_CRT  : PANTALLA;
    OPCION   : CHAR;
    PUNTO    : T_PUNTO; { primer vertice }
    EXISTE   ,          { hay algo en la pantalla }
    CORRECTA : BOOLEAN; { se encontro la regla adecuada }
    VEC_ENTRADA  : T_VEC_ENTRADA;
    VEC_FUNCION  : T_VEC_FUNCION;
    A_REGLA      : T_A_REGLA;
    A_REG        : T_A_REG;
    FCG          : T_FCG;
    ULTIMO,
    INDICE_FCG,
    CONTADOR,
    PTRALTA      : INTEGER;

SEGMENT PROCEDURE LLENADO ( VAR MAT_CRT : PANTALLA);

CONST
    MAXTOT  =     3; { maximo numero de grupos por pantalla }
    MAXGRU  =     8; { renglones por grupo                  }
    MAXREN  =    24; { maximo numero de renglones           }
    MAXCOL  =    40; { columnas de la pantalla              }
    SUBREN  =     8; { subrenglones en un renglon           }
    SUBCOL  =     7; { subcolumnas en una columna           }
    BRINSUB = 1024; { distancia entre subrenglones          }
    BRINREN =  128; { distancia entre renglones             }
    BRINGRU =   40; { distancia entre grupos                }
    INICIAL = 8192; { localidad de inicio                   }

VAR
```

```
    CONT1,
    CONT2,
    CONT3,
    CONT4,
    RENG,
    SREN,
    COL,
    INIRE,
    INIGR : INTEGER;

SEGMENT PROCEDURE  INICIALIZA (VAR RENG, SREN, INIRE, INIGR : INTEGER);

{ BEGIN INICIALIZA }
    BEGIN
    RENG := MAXREN;
    SREN := INICIAL;
    INIRE:= INICIAL;
    INIGR:= INICIAL
    END
{ ENDPRO };



PROCEDURE LLENALA (RENG,CONT3,SREN,COL : INTEGER;
                           VAR MAT_CRT : PANTALLA );

TYPE
     TRICK = RECORD  CASE BOOLEAN OF
             TRUE :  (I: INTEGER);
             FALSE:  (IPTR:^INTEGER)
             END;
     VECT  = RECORD  CASE BOOLEAN OF
             TRUE :  (I:INTEGER);
             FALSE:  (VECT:PACKED ARRAY [0:15] OF BOOLEAN)
             END;
VAR
    VECTOR : VECT;
    LEE    : TRICK;
    X,Y,I  : INTEGER;

{ BEGIN LLENALA }
    BEGIN
    X:=RENG*SUBREN-CONT3;
    Y:=COL * 7 ;
    I:= SREN + COL;
    LEE.I:=I;
    VECTOR.I:=LEE.IPTR^;
    MAT_CRT[X,Y]    := VECTOR.VECT[0];
    MAT_CRT[X,Y+1]  := VECTOR.VECT[1];
    MAT_CRT[X,Y+2]  := VECTOR.VECT[2];
    MAT_CRT[X,Y+3]  := VECTOR.VECT[3];
```

```
    MAT_CRT[X,Y+4]  := VECTOR.VECT[4];
    MAT_CRT[X,Y+5]  := VECTOR.VECT[5];
    MAT_CRT[X,Y+6]  := VECTOR.VECT[6];
    MAT_CRT[X,Y+7]  := VECTOR.VECT[8];
    MAT_CRT[X,Y+8]  := VECTOR.VECT[9];
    MAT_CRT[X,Y+9]  := VECTOR.VECT[10];
    MAT_CRT[X,Y+10] := VECTOR.VECT[11];
    MAT_CRT[X,Y+11] := VECTOR.VECT[12];
    MAT_CRT[X,Y+12] := VECTOR.VECT[13];
    MAT_CRT[X,Y+13] := VECTOR.VECT[14]
    END
{ ENDPRO };

{ BEGIN LLENADO }
    BEGIN
    INICIALIZA (RENG,SREN,INIRE,INIGR);
    FOR CONT1:=1 TO MAXTOT DO
        BEGIN
        FOR CONT2:= 1 TO MAXGRU DO
            BEGIN
            FOR CONT3:=1 TO SUBREN DO
                BEGIN
                COL:=0;
                WHILE COL<39 DO
                    BEGIN
                    LLENALA (RENG, CONT3, SREN, COL, MAT_CRT);
                    COL:=COL+2
                    END
                { ENDWHILE };
                SREN:= SREN + BRINSUB;
                END
            { ENDFOR };
            PAGE(OUTPUT);
            GOTOXY (22,22);
            IF RENG MOD 2 = 0 THEN
                BEGIN
                TEXTMODE;
                WRITELN ('* *  ESTOY PROCESANDO LA IMAGEN  * *')
                END
            ELSE
                GRAFMODE
            { ENDIF };
            RENG := RENG -1;
            SREN := INIRE + BRINREN;
            INIRE:=SREN
            END
        { ENDFOR };
        SREN := INIGR + BRINGRU;
        INIRE:= SREN;
        INIGR:= SREN
        END
```

```
    { ENDFOR }
    END
{ ENDPRO };

SEGMENT PROCEDURE PRIMER_VERTICE (VAR MAT_CRT:PANTALLA;VAR PUNTO:T_PUNTO;
                          VAR EXISTE :BOOLEAN);
VAR
    I,J:INTEGER;
{ PRIMER_VERTICE }
  . BEGIN
    EXISTE := FALSE;
    FOR I:= 0 TO 191 DO
        FOR J:= 0 TO 279 DO
            IF MAT_CRT [I,J] THEN
                BEGIN
                PUNTO.R := I; PUNTO.C := J;
                EXISTE := TRUE;
                EXIT (PRIMER_VERTICE)
                END
            { ENDIF }
      . { ENDFOR }
    { ENDFOR };
    END
{ ENDPRO };

SEGMENT PROCEDURE CARGA_FB(VAR VEC_ENTRADA : T_VEC_ENTRADA;
                  VAR VEC_FUNCION : T_VEC_FUNCION;
                  VAR A_REGLA     : T_A_REGLA;
                  VAR INDICE      : INTEGER);
VAR
    SUJETO_SIGUIENTE   : T_PTR_SUJETO;
    OPERADOR_SIGUIENTE : T_PTR_OPERADOR;
    ELEMENTO           : INTEGER;

SEGMENT PROCEDURE INICIA (VAR FUNCION:T_FUNCION);
VAR
    I:INTEGER;
{ INICIA }
    BEGIN
    FOR I := 1 TO MNI DO
        BEGIN
        FUNCION.IGUALDAD[I].PRIMER := 0;
        FUNCION.IGUALDAD[I].SEGUNDO:= 0;
        FUNCION.DESIGUAL[I].PRIMER := 0;
        FUNCION.DESIGUAL[I].SEGUNDO:= 0
        END
    { ENDFOR }
    END
{ ENDPRO };

{ CARGA_FB }
```

```
BEGIN
RESET(A_REGLA,'#5:BAS_MAST.DATA');
INDICE :=   0;
WHILE (NOT EOF(A_REGLA)) AND (A_REGLA^.NOMBRE <> ' ') DO
    BEGIN
    INDICE := SUCC(INDICE);
    ELEMENTO := 1;
    NEW (SUJETO_SIGUIENTE);
    VEC_ENTRADA [INDICE] := SUJETO_SIGUIENTE;
    SUJETO_SIGUIENTE^.SUJETO := A_REGLA^.LADO_DER.SUJETO[ELEMENTO];
    NEW (OPERADOR_SIGUIENTE);
    SUJETO_SIGUIENTE^.SIG_OP := OPERADOR_SIGUIENTE;
    OPERADOR_SIGUIENTE^.OPERADOR :=
                                A_REGLA^.LADO_DER.OPERADOR[ELEMENTO];
    OPERADOR_SIGUIENTE^.SIG_SUJ := NIL;
    ELEMENTO := SUCC(ELEMENTO);
    WHILE A_REGLA^.LADO_DER.SUJETO[ELEMENTO].ITEM <> ' ' DO
        BEGIN
        NEW (SUJETO_SIGUIENTE);
        OPERADOR_SIGUIENTE^.SIG_SUJ := SUJETO_SIGUIENTE;
        SUJETO_SIGUIENTE^.SUJETO := A_REGLA^.LADO_DER.SUJETO[ELEMENTO];
        NEW (OPERADOR_SIGUIENTE);
        SUJETO_SIGUIENTE^.SIG_OP := OPERADOR_SIGUIENTE;
        OPERADOR_SIGUIENTE^.OPERADOR :=
                                  A_REGLA^.LADO_DER.OPERADOR[ELEMENTO];
        OPERADOR_SIGUIENTE^.SIG_SUJ := NIL;
        ELEMENTO := SUCC(ELEMENTO)
        END
    { ENDWHILE };
    OPERADOR_SIGUIENTE^.SIG_SUJ := VEC_ENTRADA[INDICE];
    ELEMENTO := 1;
    INICIA (VEC_FUNCION[INDICE]);
    WHILE A_REGLA^.LADO_DER.IGUALDAD[ELEMENTO].PRIMER <> 0 DO
        BEGIN
        VEC_FUNCION[INDICE].IGUALDAD[ELEMENTO] :=
               A_REGLA^.LADO_DER.IGUALDAD[ELEMENTO];
        ELEMENTO := SUCC(ELEMENTO)
        END
    { ENDWHILE };
    ELEMENTO := 1;
    WHILE A_REGLA^.LADO_DER.DESIGUAL[ELEMENTO].PRIMER <> 0 DO
        BEGIN
        VEC_FUNCION[INDICE].DESIGUAL[ELEMENTO] :=
               A_REGLA^.LADO_DER.DESIGUAL[ELEMENTO];
        ELEMENTO := SUCC(ELEMENTO)
        END
    { ENDWHILE };
    GET (A_REGLA)
    END
{ ENDWHILE };
CLOSE (A_REGLA)
```

```
    END
( ENDPRO );
SEGMENT PROCEDURE ENC_VERTICE (PUNTO          : T_PUNTO;
                               VAR MAT_CRT    : PANTALLA;
                               VAR SUJETO1,
                                   SUJETO2    : T_SUJETO;
                               VAR ENCONTRO   : BOOLEAN);
CONST
     NMP=5;
VAR
     RECORRE    : T_PUNTO;
     CONT       : INTEGER;

{ ENC_VERTICE }
    BEGIN
    {ASUME H }
    ENCONTRO := TRUE;
    SUJETO1.MODIFICADOR:=' ';
    SUJETO1.ITEM        :='H';
    SUJETO2.MODIFICADOR:=' ';
    SUJETO2.ITEM        :='V';
    RECORRE.C:=PUNTO.C+1;
    RECORRE.R:=PUNTO.R;
    CONT:=1;


    TEXTMODE;
    WRITELN('PUNTO ',PUNTO.R,';',PUNTO.C);
    WHILE (MAT_CRT[RECORRE.R,RECORRE.C]) AND (CONT <= NMP) DO
        BEGIN
        WRITELN('H ',RECORRE.R,';',RECORRE.C);
        RECORRE.C:=SUCC(RECORRE.C);
        CONT:=SUCC(CONT)
        END
    { ENDWHILE };

    IF CONT <= NMP THEN
        BEGIN
        RECORRE.C:=PUNTO.C-1;
        RECORRE.R:=PUNTO.R;
        CONT:=1;

        WHILE (MAT_CRT[RECORRE.R,RECORRE.C]) AND (CONT <= NMP) DO
            BEGIN
            WRITELN ('@H ',RECORRE.R,';',RECORRE.C);
            RECORRE.C:=PRED(RECORRE.C);
            CONT:=SUCC(CONT)
            END
        { ENDWHILE };
        IF CONT > NMP THEN
            SUJETO1.MODIFICADOR:='@'
```

```
        ELSE
            SUJETO1.ITEM := ' '
        { ENDIF }
        END
    { ENDIF };

    {BUSCA V O @V}

    RECORRE.C:=PUNTO.C;
    RECORRE.R:=PUNTO.R+1;
    CONT:=1;

    WHILE (MAT_CRT[RECORRE.R,RECORRE.C]) AND (CONT <= NMP) DO
         BEGIN
         WRITELN('V ',RECORRE.R,';',RECORRE.C);
         RECORRE.R:=SUCC(RECORRE.R);
         CONT:=SUCC(CONT)
         END
    { ENDWHILE };

    IF CONT <= NMP THEN
        BEGIN
        RECORRE.C:=PUNTO.C;
        RECORRE.R:=PUNTO.R-1;
        CONT:=1;

        WHILE (MAT_CRT[RECORRE.R,RECORRE.C]) AND (CONT <= NMP) DO
             BEGIN
             WRITELN('@V ',RECORRE.R,';',RECORRE.C);
             RECORRE.R:=PRED(RECORRE.R);
             CONT:=SUCC(CONT)
             END
        { ENDWHILE };
        IF CONT > NMP THEN
            SUJETO2.MODIFICADOR:='@'
        ELSE
            SUJETO2.ITEM := ' '
        { ENDIF }
        END
    { ENDIF };
    READLN;
    GRAFMODE;
    IF (SUJETO1.ITEM=' ') OR (SUJETO2.ITEM=' ') THEN
        BEGIN
        TEXTMODE;
        PAGE (OUTPUT);
        GOTOXY(18,22);
        WRITELN('* *  EL PUNTO ',PUNTO.R:3,';',PUNTO.C:3,
                ' NO ES UN VERTICE  * *');
        READLN;
        ENCONTRO := FALSE
```

```
        END
    { ENDIF }
    END
{ ENDPRO };

SEGMENT PROCEDURE ESCOGE_REGLA (    ULTIMO          : INTEGER;
                                VAR SUJETO1,
                                    SUJETO2         : T_SUJETO;
                                VAR REGLA           : INTEGER;
                                VAR VEC_ENTRADA     : T_VEC_ENTRADA;
                                VAR PTR_REGLA       : T_PTR_SUJETO;
                                VAR LADO            : INTEGER      );
VAR
    EXITO       : BOOLEAN;

PROCEDURE ASIGNA_SUJETOS (VAR SUJETO1,
                              SUJETO2    : T_SUJETO);
VAR
    SUJ_AUX : T_SUJETO;
    DOMINA  : CHAR;

{ ASIGNA_SUJETO }
    BEGIN
    IF SUJETO1.MODIFICADOR = SUJETO2.MODIFICADOR THEN
        DOMINA := 'H'                              { DOMINA LA H }
    ELSE
        DOMINA := 'V'                              { DOMINA LA V }
    { ENDIF };
    IF SUJETO2.ITEM = DOMINA THEN
        BEGIN
        SUJ_AUX := SUJETO2;
        SUJETO2 := SUJETO1;
        SUJETO1 := SUJ_AUX
        END
    { ENDIF };                  { Se coloco como SUJETO1 el dominador }
    IF SUJETO2.MODIFICADOR = ' ' THEN
        SUJETO2.MODIFICADOR := '@'
    ELSE
        SUJETO2.MODIFICADOR := ' '
    { ENDIF }                   { Se cambio el modificador del sujeto
                                  dominado                             }
    END
{ ENDPRO };

{ ESCOGE_REGLA }
    BEGIN
    EXITO := FALSE;
    IF REGLA = 1 THEN
        ASIGNA_SUJETOS (SUJETO1,SUJETO2);
```

```
    (* ENDIF *)
    WHILE (REGLA <= ULTIMO) AND (NOT EXITO) DO
        BEGIN
        PTR_REGLA := VEC_ENTRADA [REGLA];
        LADO := 2;
        REPEAT
            IF PTR_REGLA^.SUJETO = SUJETO2 THEN
                IF PTR_REGLA^.SIG_OP^.SIG_SUJ^.SUJETO = SUJETO1 THEN
                    EXITO := TRUE
                ELSE
                    PTR_REGLA := VEC_ENTRADA [REGLA]
                { ENDIF }
            ELSE
                PTR_REGLA := PTR_REGLA^.SIG_OP^.SIG_SUJ
            { ENDIF };
            LADO := SUCC (LADO)
        UNTIL
            EXITO OR (PTR_REGLA = VEC_ENTRADA [REGLA]);
        IF NOT EXITO THEN
            REGLA := SUCC (REGLA)
        ELSE
            BEGIN
            PTR_REGLA := PTR_REGLA^.SIG_OP^.SIG_SUJ;
            IF PTR_REGLA = VEC_ENTRADA [REGLA] THEN
                LADO := 1
            ELSE
                LADO:=PRED(LADO)
            { ENDIF }
            END
        { ENDIF }
        END
    { ENDWHILE }
    END
{ ENDPRO };

SEGMENT PROCEDURE REC_REGLA(      PUNTO          : T_PUNTO;
                                  APUNTADOR      : T_PTR_SUJETO;
                                  LADO           : INTEGER;
                            VAR FUNCION          : T_FUNCION;
                            VAR INF_TEMPORAL     : T_INF_TEMP;
                            VAR MAT_CRT          : PANTALLA;
                            VAR EXITO            : BOOLEAN);

VAR
    RECORRE     : T_PUNTO;
    APUNTA      : T_PTR_SUJETO;
    N_LADO      : INTEGER;
    ERROR       : BOOLEAN;
    J           : INTEGER;
    SUJETO1,
    SUJETO2     : T_SUJETO;
```

```
PROCEDURE ENCASOH ( VAR SUJETO1,
                        SUJETO2     : TSUJETO;
                    VAR INFTEMPORAL: TINFTEMPORAL;
                    VAR MATCRT      : PANTALLA;
                    VAR RECORRE     : TPUNTO;
                    VAR ERROR       : BOOLEAN;
                    VAR NLADO       : INTEGER);

VAR
    INCR,
    INCC    : INTEGER;

(* ENCASOH *)
    BEGIN
    IF SUJETO2.ITEM='V' THEN
        BEGIN
        IF SUJETO2.MODIFICADOR = '@' THEN
            BEGIN
            INC_R:=-1;
            INC_C:=0
            END
        { ENDIF };
        IF SUJETO2.MODIFICADOR = ' ' THEN
            BEGIN
            INC_R:=1;
            INC_C:=0
            END
        { ENDIF };
        END
    { ENDIF };

    REPEAT
       IF SUJETO1.MODIFICADOR = ' ' THEN
          RECORRE.C:=SUCC(RECORRE.C)
       { ENDIF };
       IF SUJETO1.MODIFICADOR = '@' THEN
          RECORRE.C:=PRED(RECORRE.C)
       { ENDIF };
    UNTIL
       NOT MAT_CRT[ RECORRE.R,RECORRE.C] OR
       MAT_CRT[RECORRE.R+INC_R,RECORRE.C+INC_C];


    IF NOT MAT_CRT[RECORRE.R,RECORRE.C] THEN
        ERROR := TRUE
    ELSE
        BEGIN
        WITH INF_TEMPORAL DO
            BEGIN
            S [N_LADO].ITEM := SUJETO1.ITEM;
```

```
            S [N_LADO].MODIFICADOR := SUJETO1.MODIFICADOR;
            PX[N_LADO+1] := RECORRE.C ;
            PY[N_LADO+1] := RECORRE.R;
            END
        { ENDWITH };
        N_LADO := SUCC(NLADO)
        END
    { ENDIF }
    END
(* ENDPRO  *);


PROCEDURE ENCASOV  (VAR SUJETO1,
                        SUJETO2     : TSUJETO;
                    VAR INFTEMPORAL: TINFTEMPORAL;
                    VAR MATCRT      : PANTALLA;
                    VAR RECORRE     : TPUNTO;
                    VAR ERROR       : BOOLEAN;
                    VAR NLADO       : INTEGER);

VAR
    INCR,
    INCC   : INTEGER;

(* ENCASOV *)
    BEGIN
    IF SUJETO2.ITEM='H' THEN
        BEGIN
        IF SUJETO2.MODIFICADOR='@' THEN
            BEGIN
            INC_C:=-1;
            INC_R:=0
            END
        { ENDIF };
        IF SUJETO2.MODIFICADOR = ' ' THEN
            BEGIN
            INC_C:=1;
            INC_R:=0
            END
        { ENDIF };
        REPEAT
            IF SUJETO1.MODIFICADOR = ' ' THEN
                RECORRE.R:=SUCC(RECORRE.R)
            { ENDIF };
            IF SUJETO1.MODIFICADOR = '@' THEN
                RECORRE.R:=PRED(RECORRE.R)
            { ENDIF };
        UNTIL
             NOT MAT_CRT[ RECORRE.R,RECORRE.C] OR
             MAT_CRT[RECORRE.R+INC_R,RECORRE.C+INC_C];
```

```
        IF NOT MAT_CRT[RECORRE.R,RECORRE.C] THEN
            ERROR:= TRUE
        ELSE
            BEGIN
            WITH INF_TEMPORAL DO
                BEGIN
                S [N_LADO].ITEM:=SUJETO1.ITEM;
                S [N_LADO].MODIFICADOR:=SUJETO1.MODIFICADOR;
                PX[N_LADO+1]:=RECORRE.C;
                PY[N_LADO+1]:=RECORRE.R;
                END
            { ENDWITH };
            N_LADO:=SUCC(N_LADO)
            END
        { ENDIF }
        END
    { ENDIF }
    END
(* ENDPRO  *);


PROCEDURE VALIDA_FUNCION (    LADO          : INTEGER;
                              INF_TEMPORAL  : T_INF_TEMP;
                              FUNCION       : T_FUNCION;
                          VAR EXITO         : BOOLEAN);

VAR
    ACOMODADO  : T_INF_TEMP;
    CONTADOR,
    DIF,
    AUX        : INTEGER;

FUNCTION LONG ( X1,Y1,X2,Y2 : INTEGER) : INTEGER;

{ LONG }
    BEGIN
    LONG := ABS(X2-X1)+ABS(Y2-Y1)
    END
{ ENDFUNCTION };

PROCEDURE LONG_I (X1,Y1,X2,Y2,X3,Y3,X4,Y4  : INTEGER);

VAR
    LONG1,LONG2  : INTEGER;

{ LONG_I }
    BEGIN
    LONG1 := LONG (X1,Y1,X2,Y2);
    LONG2 := LONG (X3,Y3,X4,Y4);
    IF LONG1 <> LONG2 THEN
```

```
        EXITO := FALSE
    { ENDIF }
    END
{ ENDPRO };

PROCEDURE LONG_D (X1,Y1,X2,Y2,X3,Y3,X4,Y4  : INTEGER);

VAR
    LONG1,LONG2  : INTEGER;

{ LONG_D }
    BEGIN
    LONG1 := LONG (X1,Y1,X2,Y2);
    LONG2 := LONG (X3,Y3,X4,Y4);
    IF LONG1 = LONG2 THEN
        EXITO := FALSE
    { ENDIF }
    END
{ ENDPRO };


{ VALIDA_FUNCION }
    BEGIN
    CONTADOR := 0;
    WHILE INF_TEMPORAL.S[CONTADOR+1].ITEM <>' ' DO
        CONTADOR := SUCC(CONTADOR)  { Ve cuantos sujetos posee la regla }
    { ENDWHILE };
    DIF := CONTADOR - LADO;
    FOR AUX := 0 TO DIF DO
        BEGIN
        ACOMODADO.PX [AUX+LADO] := INF_TEMPORAL.PX [AUX +1];
        ACOMODADO.PY [AUX+LADO] := INF_TEMPORAL.PY [AUX +1];
        ACOMODADO.S  [AUX+LADO] := INF_TEMPORAL.S  [AUX +1]
        END
    { ENDFOR };
    IF LADO > 1 THEN
        BEGIN
        FOR AUX := 1 TO LADO - 1 DO
            BEGIN
            ACOMODADO.PX [AUX] := INF_TEMPORAL.PX [DIF + 1 + AUX];
            ACOMODADO.PY [AUX] := INF_TEMPORAL.PY [DIF + 1 + AUX];
            ACOMODADO.S  [AUX] := INF_TEMPORAL.S  [DIF + 1 + AUX]
            END
        { ENDFOR }
        END
    (* ENDIF *);
    WITH ACOMODADO DO
        BEGIN
        PX[CONTADOR+1]:=PX[1];
        PY[CONTADOR+1]:=PY[1];
        S[CONTADOR+1].ITEM:=' ';
```

```
        S[CONTADOR+1].MODIFICADOR:=' '
        END
    (* ENDWITH *);

    EXITO := TRUE;
    CONTADOR := 1;
    WHILE (EXITO) AND (FUNCION.IGUALDAD[CONTADOR].PRIMER <> 0) DO
        BEGIN
        LONGI (ACOMODADO.PX[FUNCION.IGUALDAD[CONTADOR].PRIMER],
               ACOMODADO.PY[FUNCION.IGUALDAD[CONTADOR].PRIMER],
               ACOMODADO.PX[FUNCION.IGUALDAD[CONTADOR].PRIMER+1],
               ACOMODADO.PY[FUNCION.IGUALDAD[CONTADOR].PRIMER+1],
               ACOMODADO.PX[FUNCION.IGUALDAD[CONTADOR].SEGUNDO],
               ACOMODADO.PY[FUNCION.IGUALDAD[CONTADOR].SEGUNDO],
               ACOMODADO.PX[FUNCION.IGUALDAD[CONTADOR].SEGUNDO+1],
               ACOMODADO.PY[FUNCION.IGUALDAD[CONTADOR].SEGUNDO+1]);
        CONTADOR := SUCC(CONTADOR)
        END
    { ENDWHILE };
    IF EXITO THEN
        BEGIN
        CONTADOR := 1;
        WHILE (EXITO) AND (FUNCION.DESIGUAL[CONTADOR].PRIMER <> 0) DO
            BEGIN
            LONGD (ACOMODADO.PX[FUNCION.DESIGUAL[CONTADOR].PRIMER],
                   ACOMODADO.PY[FUNCION.DESIGUAL[CONTADOR].PRIMER],
                   ACOMODADO.PX[FUNCION.DESIGUAL[CONTADOR].PRIMER+1],
                   ACOMODADO.PY[FUNCION.DESIGUAL[CONTADOR].PRIMER+1],
                   ACOMODADO.PX[FUNCION.DESIGUAL[CONTADOR].SEGUNDO],
                   ACOMODADO.PY[FUNCION.DESIGUAL[CONTADOR].SEGUNDO],
                   ACOMODADO.PX[FUNCION.DESIGUAL[CONTADOR].SEGUNDO+1],
                   ACOMODADO.PY[FUNCION.DESIGUAL[CONTADOR].SEGUNDO+1]);
            CONTADOR := SUCC(CONTADOR)
            END
        { ENDWHILE }
        END
    { ENDIF }
    END
{ ENDPRO };


{ REC_REGLA }
    BEGIN
    RECORRE.C:=PUNTO.C;
    RECORRE.R:=PUNTO.R;
    APUNTA   :=APUNTADOR;
    EXITO    :=FALSE;
    ERROR    :=FALSE;
    N_LADO   :=1;

    FOR J:=1 TO MNL DO
```

```
    BEGIN
        WITH INFTEMPORAL DO
        BEGIN
            S[J].ITEM:=' ';
            S[J].MODIFICADOR:=' ';
            PX[J]:=0;
            PY[J]:=0
            END
        (* ENDWITH *);
        END
    (* ENDFOR *);

    INFTEMPORAL.PX[NLADO]:=PUNTO.C;
    INFTEMPORAL.PY[NLADO]:=PUNTO.R;


    WHILE ( NOT ERROR ) AND ( NOT EXITO ) DO
        BEGIN
        SUJETO1.ITEM:=APUNTA^.SUJETO.ITEM;
        SUJETO1.MODIFICADOR:=APUNTA^.SUJETO.MODIFICADOR;
        APUNTA:=APUNTA^.SIG_OP^.SIG_SUJ;
        SUJETO2.ITEM:=APUNTA^.SUJETO.ITEM;
        SUJETO2.MODIFICADOR:=APUNTA^.SUJETO.MODIFICADOR;


        CASE SUJETO1.ITEM OF

        'H' :
              ENCASOH(SUJETO1,SUJETO2,INFTEMPORAL,MATCRT,
                      RECORRE,ERROR,NLADO);

        'V' :
              ENCASOV(SUJETO1,SUJETO2,INFTEMPORAL,MATCRT,
                      RECORRE,ERROR,NLADO);
        END
        ( ENDCASE );
        IF NOT ERROR AND ( PUNTO.C = RECORRE.C ) AND
                         ( PUNTO.R = RECORRE.R ) THEN
            EXITO := TRUE
        ELSE
            EXITO := FALSE
        ( ENDIF );
        END
    ( ENDWHILE );

    IF EXITO THEN
        VALIDAFUNCION(LADO,INFTEMPORAL,FUNCION,EXITO)
    (* ENDIF *)
    END
( ENDPRO );
```

```
SEGMENT PROCEDURE BORRA_FIGURA ( VAR MAT_CRT      : PANTALLA;
                                 VAR INF_TEMPORAL : T_INF_TEMPORAL);

VAR
    CONTADOR,
    RECORRE_C,
    RECORRE_R   : INTEGER;

PROCEDURE DIBUJA_PUNTO(R, C : INTEGER);

VAR
    PUNTO : BOOLEAN;

(* DIBUJA_PUNTO *)
    BEGIN
    DRAWBLOCK(PUNTO,1,0,0,1,1,C,R,3)
    END
(* ENDPRO *);


(* BORRA_FIGURA *)
    BEGIN
    CONTADOR:=1;
    WHILE INF_TEMPORAL.S[CONTADOR].ITEM <> ' ' DO
        BEGIN
        RECORRE_C:=INF_TEMPORAL.PX[CONTADOR];
        RECORRE_R:=INF_TEMPORAL.PY[CONTADOR];
        CASE INF_TEMPORAL.S[CONTADOR].ITEM OF

        'H':
            BEGIN
            REPEAT
                IF INF_TEMPORAL.S[CONTADOR].MODIFICADOR=' ' THEN
                    RECORRE_C:=SUCC(RECORRE_C)
                (* ENDIF *);
                IF INF_TEMPORAL.S[CONTADOR].MODIFICADOR='@' THEN
                    RECORRE_C:=PRED(RECORRE_C)
                (* ENDIF *);
                MAT_CRT[RECORRE_R,RECORRE_C]:=FALSE;
                DIBUJA_PUNTO(RECORRE_R,RECORRE_C);
            UNTIL RECORRE_C=INF_TEMPORAL.PX[CONTADOR+1]
            END
         (* ENDH *);

        'V' :
            BEGIN
            REPEAT
                IF INF_TEMPORAL.S[CONTADOR].MODIFICADOR=' ' THEN
                    RECORRE_R:=SUCC(RECORRE_R)
                (* ENDIF *);
                IF INF_TEMPORAL.S[CONTADOR].MODIFICADOR='@' THEN
```

```
                    RECORRE_R:=PRED(RECORRE_R)
                (* ENDIF *);
                MAT_CRT[RECORRE_R,RECORRE_C]:=FALSE;
                DIBUJA_PUNTO(RECORRE_R,RECORRE_C);
            UNTIL RECORRE_R=INF_TEMPORAL.PY[CONTADOR+1]
            END
        (* ENDV *)
        END
        (* ENDCASE *);
        CONTADOR:=SUCC(CONTADOR)
        END
    (* ENDWHILE *);
    READLN;
    END
(* ENDPRO *);


SEGMENT PROCEDURE GUARDA_FB_EN_FCG (    REGLA        : INTEGER;
                                    VAR FCG          : T_FCG;
                                    VAR INDICE_FCG   : INTEGER;
                                    VAR A_REGLA      : T_A_REGLA);

{ GUARDA_FB_EN_FCG }
    BEGIN
    RESET (A_REGLA, '#5:BAS_MAST.DATA');
    SEEK (A_REGLA,PRED(REGLA));
    GET(A_REGLA);
    INDICE_FCG := SUCC(INDICE_FCG);
    FCG [INDICE_FCG] := A_REGLA^.LADO_IZQ;
    CLOSE (A_REGLA)
    END
{ ENDPRO };


PROCEDURE ARMA_FCG(VAR PUNTO        : TPUNTO;
                   VAR ULTIMO       : INTEGER;
                   VAR VEC_ENTRADA  : T_VEC_ENTRADA;
                   VAR MAT_CRT      : PANTALLA;
                   VAR FCG          : T_FCG;
                   VAR CORRECTA     : BOOLEAN;
                   VAR VEC_FUNCION  : T_VEC_FUNCION;
                   VAR A_REGLA      : T_A_REGLA;
                   VAR INDICE_FCG   : INTEGER);
FORWARD;

PROCEDURE BARR_INMED (    INF_TEMPORAL  : T_INF_TEMP;
                      VAR MAT_CRT       : PANTALLA;
                      VAR INDICE_FCG    : INTEGER;
                      VAR FCG           : T_FCG;
                      VAR CORRECTA      : BOOLEAN
```

```
                        VAR PUNTO          : T_PUNTO;
                        VAR ULTIMO         : INTEGER;
                        VAR VEC_ENTRADA    : T_VEC_ENTRADA;
                        VAR VEC_FUNCION    : T_VEC_FUNCION;
                        VAR A_REGLA        : T_A_REGLA);

VAR
    INDICE,
    ULT_SUJ     : INTEGER;

PROCEDURE LOCALIZA_VERTICE (     S              : T_SUJETO;
                            VAR PUNTO          : T_PUNTO;
                            VAR MAT_CRT        : PANTALLA);

VAR
    P_AUX       : T_PUNTO;

{ BEGIN LOCALIZA_VERTICE }
    BEGIN
    P_AUX := PUNTO;
    IF S.ITEM = 'V' THEN
        BEGIN
        IF S.MODIFICADOR = ' ' THEN
            WHILE (MAT_CRT[PRED(P_AUX.R),P_AUX.C]) AND NOT
                  (MAT_CRT[P_AUX.R,SUCC(P_AUX.C)])  DO
                P_AUX.R := PRED(P_AUX.R)
            { ENDWHILE }
        ELSE
            WHILE (MAT_CRT [SUCC(P_AUX.R),P_AUX.C]) AND NOT
                  (MAT_CRT [P_AUX.R,PRED(P_AUX.C)]) DO
                P_AUX.R := SUCC(P_AUX.R)
            { ENDWHILE }
        { ENDIF };
        PUNTO.R := P_AUX.R
        END
    ELSE
        BEGIN
        IF S.MODIFICADOR = '@' THEN
            WHILE (MAT_CRT[P_AUX.R,SUCC(P_AUX.C)]) AND NOT
                  (MAT_CRT[SUCC(P_AUX.R),P_AUX.C]) DO
                    P_AUX.C := SUCC(P_AUX.C)
            { ENDWHILE }
        ELSE
            WHILE (MAT_CRT [P_AUX.R,PRED(P_AUX.C)]) AND NOT
                  (MAT_CRT [PRED(P_AUX.R),P_AUX.C]) DO
                P_AUX.C := PRED(P_AUX.C)
            { ENDWHILE }
        { ENDIF };
        PUNTO.C := P_AUX.C
        END
    { ENDIF };
```

```
    END
{ ENDPRO };

PROCEDURE HORIZONTAL (    FIJO,
                          MOVILI,
                          MOVILF          : INTEGER;
                          S               : T_SUJETO;
                      VAR PUNTO           : TPUNTO;
                      VAR MAT_CRT         : PANTALLA;
                      VAR INDICE_FCG      : INTEGER;
                      VAR FCG             : T_FCG;
                      VAR CORRECTA        : BOOLEAN);

VAR
    AUXR,
    AUXC      : INTEGER;
    P_AUX     : T_PUNTO;
    PIX       : BOOLEAN;

{ BEGIN HORIZONTAL }
    BEGIN
    CORRECTA:=TRUE;
    IF S.MODIFICADOR = ' ' THEN
        BEGIN
        AUXR := PRED(FIJO);
        AUXC:=MOVILI;
        WHILE CORRECTA AND (AUXC <= MOVILF ) DO
            BEGIN
            DRAWBLOCK(PIX,1,0,0,1,1,AUXC,AUXR,15);
            IF MAT_CRT [AUXR, AUXC] THEN
                BEGIN
                P_AUX.R := AUXR;
                P_AUX.C := AUXC;
                IF MATCRT[AUXR,PRED(AUXC)] THEN
                    LOCALIZA_VERTICE (S,P_AUX,MAT_CRT)
                { ENDIF };
                ARMA_FCG(PAUX,ULTIMO,VEC_ENTRADA,MAT_CRT,FCG,CORRECTA,
                        VEC_FUNCION,A_REGLA,INDICE_FCG)
                END
            { ENDIF };
            AUXC:=SUCC(AUXC);
            END
        { ENDWHILE }
        END
    ELSE
        BEGIN
        AUXR := SUCC(FIJO);
        AUXC:=MOVILI;
        WHILE CORRECTA AND (AUXC >= MOVILF) DO
            BEGIN
            DRAWBLOCK(PIX,1,0,0,1,1,AUXC,AUXR,15);
```

```
            IF MAT_CRT [AUXR, AUXC] THEN
                BEGIN
                P_AUX.R := AUXR;
                P_AUX.C := AUXC;
                IF MATCRT[AUXR,SUCC(AUXC)] THEN
                    LOCALIZA_VERTICE (S,P_AUX,MAT_CRT)
                { ENDIF };
                ARMA_FCG(PAUX,ULTIMO,VEC_ENTRADA,MAT_CRT,FCG,CORRECTA,
                        VEC_FUNCION,A_REGLA,INDICE_FCG)
                END
            { ENDIF };
            AUXC:=PRED(AUXC);
            END
        { ENDWHILE }
        END
    { ENDIF };
    INDICE_FCG := SUCC (INDICE_FCG);
    FCG [INDICE_FCG] := '/';
    END
{ ENDPRO };

PROCEDURE VERTICAL    (     FIJO,
                            MOVILI,
                            MOVILF              : INTEGER;
                            S                   : T_SUJETO;
                       VAR PUNTO              : TPUNTO;
                       VAR MAT_CRT            : PANTALLA;
                       VAR INDICE_FCG         : INTEGER;
                       VAR FCG                : T_FCG;
                       VAR CORRECTA           : BOOLEAN);

VAR
    AUXR,
    AUXC        : INTEGER;
    P_AUX       : T_PUNTO;
    PIX         : BOOLEAN;

{ BEGIN VERTICAL }
    BEGIN
    CORRECTA:=TRUE;
    IF S.MODIFICADOR = ' ' THEN
        BEGIN
        AUXC := SUCC(FIJO);
        AUXR:=MOVILI;
        WHILE CORRECTA AND (AUXR <= MOVILF) DO
            BEGIN
            DRAWBLOCK(PIX,1,0,0,1,1,AUXC,AUXR,15);
            IF MAT_CRT [AUXR, AUXC] THEN
                BEGIN
                P_AUX.R := AUXR;
                P_AUX.C := AUXC;
```

```
                IF MATCRT[PRED(AUXR),AUXC] THEN
                    LOCALIZA_VERTICE (S,P_AUX,MAT_CRT)
                { ENDIF };
                ARMA_FCG(PAUX,ULTIMO,VEC_ENTRADA,MAT_CRT,FCG,CORRECTA,
                        VEC_FUNCION,A_REGLA,INDICE_FCG)
                END
            { ENDIF };
            AUXR:=SUCC(AUXR);
            END
        { ENDWHILE }
        END
    ELSE
        BEGIN
        AUXC := PRED(FIJO);
        AUXR:=MOVILI;
        WHILE CORRECTA AND (AUXR >= MOVILF) DO
            BEGIN
            DRAWBLOCK(PIX,1,0,0,1,1,AUXC,AUXR,15);
            IF MAT_CRT [AUXR, AUXC] THEN
                BEGIN
                P_AUX.R := AUXR;
                P_AUX.C := AUXC;
                IF MATCRT[PRED(AUXR),AUXC] THEN
                    LOCALIZA_VERTICE (S,P_AUX,MAT_CRT)
                { ENDIF };
                ARMA_FCG(PAUX,ULTIMO,VEC_ENTRADA,MAT_CRT,FCG,CORRECTA,
                        VEC_FUNCION,A_REGLA,INDICE_FCG)
                END
            { ENDIF };
            AUXR:=PRED(AUXR);
            END
        { ENDWHILE }
        END
    { ENDIF };
    INDICE_FCG := SUCC (INDICE_FCG);
    FCG [INDICE_FCG] := '/';
    END
ENDPRO };

BEGIN BARRIDO_INMEDIACIONES }
    BEGIN
    ULT_SUJ:= 1;
    WHILE INF_TEMPORAL.S[SUCC(ULT_SUJ)].ITEM <> ' ' DO
        ULT_SUJ:= SUCC(ULT_SUJ)
    { ENDWHILE };
    FOR INDICE := 1 TO ULTSUJ DO
        BEGIN
        WITH INF_TEMPORAL DO
            CASE S[INDICE].ITEM OF
                'H' : HORIZONTAL (PY[INDICE],PX[INDICE],PX[SUCC(INDICE)],
                                  S[INDICE],PUNTO,MAT_CRT,INDICE_FCG,FCG,
```

```
                                        CORRECTA);
                'V' : VERTICAL     (PX[INDICE],PY[INDICE],PY[SUCC(INDICE)],
                                        S[INDICE],PUNTO,MAT_CRT,INDICE_FCG,FCG,
                                        CORRECTA);
                END
            { ENDCASE }
        { ENDWITH };
        END
    { ENDFOR };
    WHILE FCG[INDICE_FCG] = '/' DO
        BEGIN
        FCG[INDICEFCG]:=' ';
        INDICE_FCG := PRED(INDICE_FCG)
        END
    { ENDWHILE };
    INDICE_FCG := SUCC (INDICE_FCG);
    FCG[INDICE_FCG] := '*'
    END
{ ENDFRO };

PROCEDURE IDENTIFICA_FC (     FC       :T_FCG;
                          VAR EXISTE   :BOOLEAN;
                          VAR PTR_ALTA:INTEGER);
VAR
    N_A_FC,              { numero actual de figuras compuestas }
    I         :INTEGER;
    A_FC      :T_A_FC;

PROCEDURE BUSCA_FC (     FC       :T_FCG;
                         INICIO,
                         FIN      :INTEGER;
                     VAR EXISTE   :BOOLEAN;
                     VAR PTR_ALTA:INTEGER);
{ BUSCA_FC }
    BEGIN
    PTR_ALTA := (FIN-INICIO) DIV 2 + INICIO;
    SEEK (A_FC,PTR_ALTA);
    GET (A_FC);
    IF FC <> A_FC^ THEN
      BEGIN
      IF INICIO <> FIN THEN
        IF FC < A_FC^ THEN
            BUSCA_FC (FC,INICIO,PTR_ALTA,EXISTE,PTR_ALTA)
        ELSE
            BUSCA_FC (FC,PTR_ALTA+1,FIN,EXISTE,PTR_ALTA)
        { ENDIF }
      { ENDIF }
      END
    ELSE
        EXISTE := TRUE
    { ENDIF }
```

```
    END
{ ENDPRO };

{ IDENTIFICA_FC }
    BEGIN
    RESET (A_FC,'#5:A_FC.DATA');
    N_A_FC:=0;
    FOR I:= 1 TO LENGTH(A_FC^) DO
        N_A_FC:=N_A_FC*10+ORD(A_FC^[I])-48
    { ENDFOR };
    EXISTE := FALSE;
    IF N_A_FC > 0 THEN
        BEGIN
        SEEK (A_FC,N_A_FC); { recupera el ultimo registro }
        GET (A_FC);
        PTR_ALTA:=1;
        IF FC>A_FC^ THEN
            BEGIN
            PTR_ALTA := N_A_FC + 1;
            PAGE (OUTPUT);
            GOTOXY(22,22);
            WRITELN ('* *  LA FIGURA NO HA SIDO APRENDIDA  * *');
            READLN
            END
        ELSE
            BEGIN
            BUSCA_FC (FC,1,N_A_FC,EXISTE,PTR_ALTA);
            IF EXISTE THEN
                BEGIN
                PAGE (OUTPUT);
                GOTOXY(14,22);
                WRITELN('* *  SE IDENTIFICO UN  -> ',FC,'  * *');
                READLN
                END
            ELSE
                BEGIN
                PAGE (OUTPUT);
                GOTOXY(20,22);
                WRITELN ('* *  LA FIGURA NO HA SIDO APRENDIDA  * *');
                READLN
                END
            { ENDIF }
            END
        { ENDIF }
        END
    ELSE
        BEGIN
        PTR_ALTA := N_A_FC + 1;
        PAGE (OUTPUT);
        GOTOXY(23,22);
        WRITELN ('* *  NO SE HAN APRENDIDO FCS  * *');
```

```
        READLN
        END
    { ENDIF };
    CLOSE (A_FC)
    END
{ ENDPRO };


PROCEDURE ARMA_FCG;

VAR
    LADO,
    REGLA          : INTEGER;
    SUJETO1,
    SUJETO2        : T_SUJETO;
    INF_TEMPORAL   : T_INF_TEMP;
    FIN,
    EXITO,
    ENCONTRO       : BOOLEAN;
    PTR_REGLA      : T_PTR_SUJETO;


{ ARMA_FCG }
    BEGIN
    FIN := FALSE;
    CORRECTA := FALSE;
    ENC_VERTICE(PUNTO,MAT_CRT,SUJETO1,SUJETO2,ENCONTRO);
    IF ENCONTRO THEN
        BEGIN
        REGLA :=1;
        ESCOGE_REGLA(ULTIMO,SUJETO1,SUJETO2,REGLA,VEC_ENTRADA,PTR_REGLA,LADO
        EXITO := FALSE;
        WHILE (REGLA<=ULTIMO) AND NOT EXITO DO
            BEGIN
            REC_REGLA(PUNTO,PTR_REGLA,LADO,VEC_FUNCION[REGLA],
                      INF_TEMPORAL,MAT_CRT,EXITO);
            IF EXITO THEN
                BEGIN
                BORRA_FIGURA(MAT_CRT,INF_TEMPORAL);
                GUARDA_FB_EN_FCG(REGLA,FCG,INDICE_FCG,A_REGLA);
                BARR_INMED(INF_TEMPORAL,MAT_CRT,INDICE_FCG,FCG,CORRECTA,
                           PUNTO,ULTIMO,VEC_ENTRADA,VEC_FUNCION,A_REGLA)
                END
            ELSE
                BEGIN
                REGLA:=SUCC(REGLA);
                ESCOGE_REGLA (ULTIMO,SUJETO1,SUJETO2,REGLA,VEC_ENTRADA,
                              PTR_REGLA,LADO);
                END
            { ENDIF};
            IF FIN THEN
```

```
                CORRECTA := TRUE
            { ENDIF }
            END
        { ENDWHILE }
        END
    { ENDIF }
    END
{ ENDPRO };

{ RECONOCEDOR }
    BEGIN
    {GETCVAL (FECHAHOY);} { obtiene la fecha que le manda el MODULO PRINCIPAL
    PAGE (OUTPUT);

    LLENADO (MAT_CRT);  { llena la matriz con la grafica }

    CARGA_FB (VEC_ENTRADA,VEC_FUNCION,A_REGLA,ULTIMO);
    PRIMER_VERTICE (MAT_CRT,PUNTO,EXISTE);
    IF EXISTE THEN
        BEGIN
        INDICE_FCG := 0;
        FCG:='                                        ';
        GRAFMODE;
        ARMA_FCG (PUNTO,ULTIMO,VEC_ENTRADA,MAT_CRT,FCG,CORRECTA,
                  VEC_FUNCION,A_REGLA,INDICE_FCG);
        TEXTMODE;
        IF CORRECTA THEN
            BEGIN
            PAGE(OUTPUT);
            GOTOXY(7,22);
            WRITELN('* *  LA FIGURA COMPUESTA GENERADA ES ',FCG,'  * *');
            READLN;
            IDENTIFICA_FCG(FCG,EXISTE,PTRALTA);
            IF NOT EXISTE THEN
                BEGIN
                REWRITE(AREG,'#5:A_REGPAR.DATA');
                AREG^.EXISTE:=EXISTE;
                AREG^.FC:=FCG;
                AREG^.POSICION:=PTRALTA;
                PUT(AREG);
                CLOSE(AREG,LOCK)
                END
            (* ENDIF *)
            END
        ELSE
            BEGIN
            PAGE (OUTPUT);
            GOTOXY(20,22);
            WRITELN('* *  ERROR EN LA GENERACION DE LA IMAGEN  * *');
            READLN
            END
```

```
        { END IF }
        END
    { END IF };
    SETCHAIN ('#4:PRINCIPAL.CODE')
    END
{ END PROGRAM }.
```

```
(* 16 DE MARZO 1986 *)
(*$S+*)
PROGRAM APRENDIZAJE ;
USES CHAINSTUFF;

TYPE
    TFCG        = STRING[30];
    TREGPAR     = RECORD
                  EXISTE    : BOOLEAN;
                  POSICION  : INTEGER;
                  FCG       : TFCG
                  END;
    TAREGPAR    = FILE OF TREGPAR;
    TAFCG       = FILE OF TFCG;
VAR
    FECHAHOY    : STRING;
    AREGPAR     : TAREGPAR;
    AFCG        : TAFCG;
    REGISTRO    : TREGPAR;
    AUX         : TFCG;
    CONTADOR,
    ULTIMO      : INTEGER;
    OPCION      : CHAR;

PROCEDURE TRAEULTIMO (    VALOR    : TFCG;
                      VAR ULTIMO   : INTEGER);
VAR
    I : INTEGER;

(* TRAEULTIMO *)
    BEGIN
    ULTIMO := 0;
    FOR I := 1 TO LENGTH (VALOR) DO
        ULTIMO := ULTIMO*10 + ORD(VALOR[I])-48
    (* ENDFOR *)
    END
(* ENDPRO *);

(* APRENDIZAJE *)
    BEGIN
    PAGE(OUTPUT);
    GOTOXY(10,20);
    (*I-*)
    RESET (AREGPAR,'#5:A_REGPAR.DATA');
    IF IORESULT = 0 THEN
        BEGIN
    (*I+*)
        REGISTRO := AREGPAR^;
        IF NOT(REGISTRO.EXISTE) THEN
            BEGIN
```

```
            PAGE(OUTPUT);
            GOTOXY(20,20);
            WRITE ('DESEA APRENDER LA FC ->  ',REGISTRO.FCG,'  <S>I O <N>O  ')
            READLN(OPCION);
            IF OPCION = 'S' THEN
                BEGIN
    (*I-*)
                RESET(AFCG,'#5:A_FC.DATA');
                IF IORESULT = 0 THEN
                    BEGIN
        (*I+*)
                    TRAEULTIMO (AFCG^,ULTIMO);
                    FOR CONTADOR := REGISTRO.POSICION TO ULTIMO DO
                        BEGIN
                        SEEK(AFCG,CONTADOR);
                        GET(AFCG);
                        AUX := AFCG^;
                        AFCG^ := REGISTRO.FCG;
                        REGISTRO.FCG := AUX;
                        SEEK(AFCG,CONTADOR);
                        PUT (AFCG)
                        END
                    (* ENDFOR *);
                    SEEK (AFCG,SUCC(ULTIMO));
                    AFCG^_:= REGISTRO.FCG;
                    PUT (AFCG);
                    SEEK (AFCG,0);
                    STR(SUCC(ULTIMO),AFCG^);
                    PUT (AFCG);
                    AREGPAR^.EXISTE := TRUE;
                    SEEK(AREGPAR,0);
                    PUT(AREGPAR);
                    CLOSE(AREGPAR,LOCK);
                    CLOSE(AFCG,LOCK)
                    END
                ELSE
                    BEGIN
                    PAGE (OUTPUT);
                    GOTOXY(18,22);
                    WRITELN ('* *  NO EXISTE EL ARCHIVO DE LAS FC"S  * *')
                    END
                (* ENDIF *)
                END
            (* ENDIF *);
            PAGE (OUTPUT);
            GOTOXY(12,22);
            WRITELN ('* *  SE APRENDIO LA FIGURA --> ',REGISTRO.FCG,'  * *');
            END
        ELSE
            BEGIN
            PAGE (OUTPUT);
```

```
            GOTOXY(27,22);
            WRITELN ('* *  YA EXISTE LA FC  * *')
            END
        (* ENDIF *)
        END
    ELSE
        BEGIN
        PAGE (OUTPUT);
        GOTOXY(21,22);
        WRITELN ('* *  NO EXISTE FIGURA A RECONOCER  * *')
        END
    (* ENDIF *);
    GOTOXY(24,23);
    WRITELN('* *  RETURN PARA CONTINUAR  * *');
    READLN(OPCION);
    SETCHAIN ('#4:PRINCIPAL.CODE')
    END
```

```
{$S+}
{ 17 DE MARZO DE 1986}
{ TESIS : PROGRAMA DE UTILERIAS                    }

PROGRAM UTILERIAS;
USES
    CHAINSTUFF,
    {$U #5:CADENAS.CODE} CADENAS;

CONST
    MNL =  10;
    MNI =  10;

TYPE
    T_SUJETO        = RECORD
                      MODIFICADOR      : CHAR;
                      ITEM             : CHAR
                      END;
    T_OPERADOR      = CHAR;
    T_IGUALDAD      = RECORD
                      PRIMER   : INTEGER;
                      SEGUNDO  : INTEGER
                      END;
    T_REGLA         = RECORD
                      NOMBRE           : STRING [15];
                      LADO_IZQ         : CHAR;
                      LADO_DER         : RECORD
                                         SUJETOS : ARRAY [1:MNL] OF
                                                   T_SUJETO;
                                         OPERADORES : ARRAY [1:MNL]
                                                   OF T_OPERADOR;
                                         IGUALDADES : ARRAY [1:MNI]
                                                   OF T_IGUALDAD;
                                         DESIGUALES : ARRAY [1:MNI]
                                                   OF T_IGUALDAD
                                         END
                      END;
    T_A_REGLA       = FILE OF T_REGLA;
    T_FC            = STRING [30];
    T_A_FC          = FILE OF T_FC;

VAR
    OPCION   : CHAR;
    FECHAHOY : STRING;

PROCEDURE PANTSUP;

{ PANTSUP }
    BEGIN
    PAGE (OUTPUT);
    PROMPT (0,0,FECHAHOY);
```

```
    PROMPT (21,0,'RECONOCIMIENTO DE FORMAS');
    PROMPT (63,0,'MODULO UTILERIAS')
    END
(* ENDPRO *);

PROCEDURE MENU;

{ MENU }
    BEGIN
    PANTSUP;
    PROMPT (24, 7,'0 - Fin');
    PROMPT (24, 8,'1 - Creacion del Archivo de las FBs');
    PROMPT (24, 9,'2 - Creacion del Archivo de las FCs');
    PROMPT (24,10,'3 - Cargador de las FBs');
    PROMPT (24,11,'4 - BORRADO DE FCS');
    PROMPT (24,12,'5 - Listado de FBs');
    PROMPT (24,13,'6 - Listado de FCs');
    PROMPT (24,18,'Teclea la opcion que desees - > ')
    END
{ ENDPRO };

PROCEDURE CREA_FB;

VAR
    I,
    J                : INTEGER;
    IDENT,
    OPCION           : CHAR;
    MASTER           : T_A_REGLA;

{ CREA_FB }
    BEGIN
    PAGE (OUTPUT);
    WRITELN;
    WRITELN ('Esta opcion destruye el Archivo de las FBs si existe');
    WRITE ('          Desea Continuar   <S>i o <N>o  ');
    READ (OPCION);
    IF OPCION = 'S' THEN
        BEGIN
        REWRITE (MASTER, '#5:BAS_MAST.DATA');
        MASTER^.NOMBRE := ' ';
        WITH MASTER^.LADO_DER DO
            BEGIN
            FOR J:=1 TO MNL DO
                BEGIN
                SUJETOS[J].MODIFICADOR := ' ';
                SUJETOS[J].ITEM        := ' ';
                OPERADOR[J]            := ' ';
                IGUALDAD[J].PRIMER     := 0;
                IGUALDAD[J].SEGUNDO    := 0;
                DESIGUAL[J].PRIMER     := 0;
```

```
                DESIGUAL[JJ].SEGUNDO    := 0
                END
            { ENDFOR }
            END
        { ENDWITH };

        IDENT := PRED('A');
        FOR I:= 1 TO 26 DO
            BEGIN
            IDENT := SUCC(IDENT);
            MASTER^.LADO_IZQ := IDENT;
            PUT (MASTER)
            END
        { ENDFOR };
        CLOSE (MASTER,LOCK);
        RESET (MASTER,'#5:BAS_MAST.DATA')
        END
    { ENDIF }
    END
{ ENDPRO };

PROCEDURE CREA_FC;

VAR
    I,
    J                : INTEGER;
    OPCION           : CHAR;
    REGISTRO         : T_FC;
    A_FC             : T_A_FC;

{ CREA_FC }
    BEGIN
    PAGE (OUTPUT);
    WRITELN;
    WRITELN ('Esta opcion destruye el Archivo de las FCs si existe');
    WRITE ('         Desea Continuar   <S>i o <N>o  ');
    READ (OPCION);
    IF OPCION = 'S' THEN
        BEGIN
        REWRITE (A_FC, '#5:A_FC.DATA');
        STR(0,A_FC^);
        PUT(A_FC);
        REGISTRO :=
        FOR I:= 1 TO 50 DO
            PUT (A_FC)
        { ENDFOR };
        CLOSE (A_FC,LOCK)
        END
    { ENDIF }
    END
{ ENDPRO };
```

```
PROCEDURE CARGA_FB;

VAR
    MASTER          : T_A_REGLA;
    APUN,
    N,
    NR              : INTEGER;
    FLAG,
    NEW             : BOOLEAN;
    IDENT           : CHAR;


{ CARGA_FB }
    BEGIN
    {$I-}
    PAGE (OUTPUT);
    NR :=0;
    RESET (MASTER,'#5:BAS_MAST.DATA');
    IF IORESULT <> 0 THEN
    {$I+}
        BEGIN
        WRITELN;
        WRITELN ('NO EXISTE EL ARCHIVO DE FBS, TECLEAR LA OPCION 1 DEL MENU')
        WRITELN ('              RETURN  para continuar');
        WRITELN
        END
    ELSE
    {$I+}
        BEGIN
        WHILE (NOT EOF (MASTER)) AND (MASTER^.NOMBRE <> ' ') DO
            BEGIN
            GET (MASTER);
            NR := SUCC(NR);
            END
        { ENDWHILE };
        FLAG := TRUE;
        WHILE FLAG DO
            BEGIN
            N := 0;
            SEEK (MASTER,NR);
            GET (MASTER);
            WRITE (
            'NOMBRE DE LA FIGURA BASICA (15 CAR. MAX. / 0 = FIN) -> ');
            READLN(MASTER^.NOMBRE);
            IF MASTER^.NOMBRE <> '0' THEN
                BEGIN
                REPEAT
                    N := SUCC(N);
                    WRITE ('DAME EL SUJETO  (00 = FIN) -> ');
                    READLN (MASTER^.LADO_DER.SUJETOS[N].MODIFICADOR,
```

```
                        MASTER^.LADO_DER.SUJETOS[N].ITEM);
                    IF MASTER^.LADO_DER.SUJETOS[N].ITEM <> '0' THEN
                        BEGIN
                        WRITE ('DAME EL OPERADOR ');
                        READLN (MASTER^.LADO_DER.OPERADOR[N])
                        END
                    { ENDIF };
                UNTIL MASTER^.LADO_DER.SUJETOS[N].ITEM = '0';
                WRITELN;
                N := 0;
                REPEAT
                    N := SUCC(N);
                    WRITELN ('DAME LA IGUALDAD ',N);
                    WRITE ('DAME EL PRIMER DATO (0 = FIN) -> ');
                    READLN (MASTER^.LADO_DER.IGUALDAD[N].PRIMER);
                    IF MASTER^.LADO_DER.IGUALDAD[N].PRIMER <> 0 THEN
                        BEGIN
                        WRITE ('DAME EL SEGUNDO DATO              ');
                        READLN(MASTER^.LADO_DER.IGUALDAD[N].SEGUNDO);
                        END
                    { ENDIF };
                UNTIL (MASTER^.LADO_DER.IGUALDAD[N].PRIMER = 0);
                N := 0;
                REPEAT
                    N := SUCC(N);
                    WRITELN ('DAME LA DESIGUALDAD ',N);
                    WRITE ('DAME EL PRIMER DATO (0 = FIN) -> ');
                    READLN (MASTER^.LADO_DER.DESIGUAL[N].PRIMER);
                    IF MASTER^.LADO_DER.DESIGUAL[N].PRIMER <> 0 THEN
                        BEGIN
                        WRITE ('DAME EL SEGUNDO DATO              ');
                        READLN(MASTER^.LADO_DER.DESIGUAL[N].SEGUNDO);
                        END
                    { ENDIF };
                UNTIL (MASTER^.LADO_DER.DESIGUAL[N].PRIMER = 0);
                SEEK (MASTER,NR);
                PUT (MASTER);
                NR := SUCC(NR)
                END
            ELSE
                FLAG := FALSE
            { ENDIF }
            END
        { ENDWHILE };
        CLOSE (MASTER,LOCK)
        END
    { ENDIF }
    END
ENDPRO };

PROCEDURE LISTA_FB;
```

```
VAR
    MASTER : T_A_REGLA;
    N      : INTEGER;

{ L_FB }
    BEGIN
    PANTSUP;
    PROMPT(21,2,'CATALOGO DE FIGURAS BASICAS');
    RESET (MASTER,'#5:BAS_MAST.DATA');
    IF IORESULT <> 0 THEN
    {$I+}
        BEGIN
        PROMPT (0,4,'NO EXISTE EL ARCHIVO DE FIGURAS BASICAS');
        PROMPT (0,6,'RETURN para continuar ');
        READLN;
        EXIT (LISTAFB)
        END
    (* ENDIF *);
    {$I-}

    PROMPT (0,4,
    'LISTADO EN PANTALLA O EN IMPRESORA (P/I/RETURN=FIN) -> ');
    OPCION := GETCHAR(55,4,['P','I',CHR(13)]);
    IF OPCION = CHR(13) THEN
        EXIT (LISTAFB)
    ELSE
        IF OPCION = 'P' THEN
            RESET(SAL,'CONSOLE:')
        ELSE
            BEGIN
            PROMPT (0,6,'* * ENCENDER LA IMPRESORA, POR FAVOR * *');
            READLN;
            RESET(SAL,'PRINTER:');
            WRITELN(SAL);
            WRITELN(SAL,'CATALOGO DE FIGURAS BASICAS');
            WRITELN(SAL,'============================');
            WRITELN(SAL);
            END
        (* ENDIF *)
    (* ENDIF *);

    WRITELN(SAL);WRITELN(SAL);
    WHILE (NOT EOF (MASTER)) AND (MASTER^.NOMBRE <> ' ') DO
        BEGIN
        WRITE(SAL,MASTER^.LADO_IZQ,' ; ',MASTER^.NOMBRE:16, '::=');
        N := 1;
        REPEAT
            WRITE(SAL,MASTER^.LADO_DER.SUJETOS[N].MODIFICADOR,
                  MASTER^.LADO_DER.SUJETOS[N].ITEM,
                  MASTER^.LADO_DER.OPERADORES[N]);
```

```
            N := SUCC(N);
        UNTIL
             MASTER^.LADO_DER.SUJETOS[N].ITEM = ' ';
        WRITELN(SAL);
        WRITE(SAL,'     IGUALDADES : ');
        N := 1;
        REPEAT
            WRITE(SAL,MASTER^.LADO_DER.IGUALDAD[N].PRIMER,',',
                  MASTER^.LADO_DER.IGUALDAD[N].SEGUNDO,':');
            N := SUCC(N);
        UNTIL
             N > MNI;
        WRITELN(SAL);
        WRITE(SAL,'     DESIGUALES : ');
        N := 1;
        REPEAT
            WRITE(SAL,MASTER^.LADO_DER.DESIGUAL[N].PRIMER,',',
                  MASTER^.LADO_DER.DESIGUAL[N].SEGUNDO,':');
            N := SUCC(N);
        UNTIL
             N > MNI;
        WRITELN(SAL);
        GET (MASTER)
        END
    { ENDWHILE };
    CLOSE (MASTER,LOCK);
    WRITELN(SAL);
    WRITELN(SAL);
    WRITELN('      RETURN  para continuar');
    READLN;
    CLOSE (SAL)
    END
{ ENDPRO };

PROCEDURE LISTA_FC;

VAR
    A_FC   : T_A_FC;
    N,
    ULTIMO : INTEGER;
    OPCION : CHAR;

{ LISTAFC }
    BEGIN
    PANTSUP;
    PROMPT(21,2,'CATALOGO DE FIGURAS COMPUESTAS');
    RESET (A_FC,'#5:A_FC.DATA');
    IF IORESULT <> 0 THEN
    {$I+}
       BEGIN
       PROMPT (0,4,'NO EXISTE EL ARCHIVO DE FIGURAS COMPUESTAS');
```

```
        PROMPT (0,6,'RETURN para continuar ');
        READLN;
        EXIT (LISTAFC)
        END
    (* ENDIF *);
    {$I-}

    PROMPT (0,4,
    'LISTADO EN PANTALLA O EN IMPRESORA (P/I/RETURN=FIN) -> ');
    OPCION := GETCHAR(55,4,['P','I',CHR(13)]);
    IF OPCION = CHR(13) THEN
        EXIT (LISTAFC)
    ELSE
        IF OPCION = 'P' THEN
            RESET(SAL,'CONSOLE:')
        ELSE
            BEGIN
            PROMPT (0,6,'* * ENCENDER LA IMPRESORA, POR FAVOR * *');
            READLN;
            RESET(SAL,'PRINTER:');
            WRITELN(SAL);
            WRITELN(SAL,'CATALOGO DE FIGURAS COMPUESTAS');
            WRITELN(SAL,'==============================');
            WRITELN(SAL);
            END
        (* ENDIF *)
    (* ENDIF *);

    (* OBTIENE EL NUMERO ACTUAL DE FIGURAS COMPUESTAS 'ULTIMO' *)
    ULTIMO := 0;
    FOR N := 1 TO LENGTH(A_FC^) DO
        ULTIMO := ULTIMO*10 + ORD (A_FC^[N]) - 48
    { ENDFOR };

    WRITELN(SAL);
    N := 1;
    WHILE N <= ULTIMO DO
        BEGIN
        GET (A_FC);
        WRITELN (SAL);
        WRITELN (SAL,'FIGURA ',N,'  ::= ',A_FC^);
        N := N+1
        END
    { ENDWHILE };
    CLOSE (A_FC,LOCK);
    WRITELN;
    WRITELN;
    WRITELN('        RETURN  para continuar ');
    READLN;
    CLOSE (SAL)
    END
```

```
{ ENDPRO };

PROCEDURE BORRAFCS;
VAR
    A_FC   : T_A_FC;
    ULTIMO,
    CUAL,N : INTEGER;

PROCEDURE BORRALA (CUAL,ULTIMO:INTEGER);
VAR
    I:INTEGER;
    S:STRING;
(* BORRALA *)
    BEGIN
    FOR I := CUAL TO ULTIMO-1 DO
        BEGIN
        SEEK (AFC,I+1);
        GET (AFC);
        SEEK (AFC,I);
        PUT (AFC)
        END
    (* ENDFOR *);

    (* ACTUALIZA EL NUMERO ACTUAL DE FIGURAS COMPUESTAS *)
    ULTIMO := ULTIMO -1;
    STR (ULTIMO,S);
    AFC^ := S;
    SEEK (AFC,0);
    PUT (AFC)
    END
(* ENDPRO *);

{ BORRAFCS }
    BEGIN
    {$I-}
    PANTSUP;
    PROMPT(21,2,'BORRADO DE UNA FIGURA COMPUESTA');
    RESET (A_FC,'#5:A_FC.DATA');
    IF IORESULT <> 0 THEN
    {$I+}
        BEGIN
        PROMPT (0,4,'NO EXISTE EL ARCHIVO DE FIGURAS COMPUESTAS');
        PROMPT (0,6,'RETURN para continuar ');
        READLN;
        EXIT (BORRAFCS)
        END
    (* ENDIF *);

    (* OBTIENE EL NUMERO ACTUAL DE FIGURAS COMPUESTAS 'ULTIMO' *)
    ULTIMO := 0;
    FOR N := 1 TO LENGTH(A_FC^) DO
```

```
        ULTIMO := ULTIMO*10 + ORD (A_FC^[N]) - 48
    { ENDFOR };

    PROMPT (0,4,'NUMERO DE LA RP DE LA FC QUE DESEA ELIMINAR  -> ');
    CUAL := OBTENT(5,48,4);
    IF (CUAL<1) OR (CUAL>ULTIMO) THEN
        BEGIN
        PROMPT (0,6,'NO EXISTE TAL FIGURA. <RETURN> PARA CONTINUAR.');
        READLN
        END
    ELSE
        BEGIN
        SEEK (AFC,CUAL);
        GET (A_FC);
        WRITELN;
        WRITELN ('FIGURA ',CUAL:5,'  ::= ',A_FC^);
        PROMPT (0,22,'CORRECTO (S/N) -> ');
        OPCION := GETCHAR(18,22,['S','N']);
        IF OPCION = 'S' THEN
            BORRALA (CUAL,ULTIMO)
        (* ENDIF *)
        END
    (* ENDIF *);
    CLOSE (A_FC,LOCK)
    END
{ ENDPRO };

{ PRINCIPAL }
    BEGIN
    GETCVAL(FECHAHOY);
    MENU;
    OPCION := GETCHAR (57,18,['0'..'6']);
    WHILE OPCION <> '0' DO
        BEGIN
        CASE OPCION OF
            '1':CREA_FB;
            '2':CREA_FC;
            '3':CARGA_FB;
            '4':BORRAFCS;
            '5':LISTA_FB;
            '6':LISTA_FC
            END
        { ENDCASE };
        MENU;
        OPCION := GETCHAR (57,18,['0'..'6'])
        END
    { ENDWHILE };
    SETCHAIN ('#4:PRINCIPAL.CODE')
    END
{ ENDPROGRAM }.
```

APENDICE D.

MANUAL DE USUARIO DEL SISTEMA " SIREI "

TABLA DE CONTENIDO

1. INTRODUCCION

SIREI - SIstema REconocedor de Imágenes - es un sistema capaz de reconocer imágenes bidimensionales basándose en información digitalizada de las mismas y empleando "reglas de producción" (ver capítulo 4 de la Tesis); así mismo tiene la capacidad de "aprender" nuevas figuras para su posterior identificación (ver capítulo 5). Para la programación de SIREI se utilizó el lenguaje Pascal que permite crear programas de alta calidad y estructuración. Se utilizó una microcomputadora Apple IIe con 64 KBytes de memoria principal y dos unidades de disco flexibles con capacidad de 140 KBytes cada uno. El presente manual tiene el objetivo de guiar al usuario en el uso de SIREI permitiéndole utilizar el sistema sin ser necesario el pleno conocimiento del mismo.

Antes de intentar correr el Sistema, es necesario verificar los siguientes puntos :

+ Se debe disponer de dos diskettes marcados con los siguientes nombres:

  " SIREI 1 "

  " SIREI 2 "

+ Se debe contar con una microcomputadora Apple IIe o IIplus
+ Se debe tener a la mano este manual.
+ Se sugiere leer la Tesis para entender fácilmente los resultados obtenidos en una sesión con el sistema.

2. COMO CORRER EL SISTEMA

Para asegurar una corrida del sistema libre de errores es necesario seguir todos los pasos que a continuación se listan :

a) Insertar el disco " SIREI 1 " en la unidad marcada con el número 1 (uno).

b) Insertar el disco " SIREI 2 " en la unidad marcada con el número 2 (dos).

c) Encender la computadora.

d) Esperar a que el sistema sea cargado en la memoria principal y empiece a ejecutarse.

Nota : Si el usuario del sistema conoce el funcionamiento de la computadora, es conveniente que verifique el contenido de cada disco, para asegurar la presencia de los siguientes archivos en cada uno de ellos :

1) Para el disco marcado " SIREI 1 ":

- SYSTEM.APPLE
- SYSTEM.PASCAL
- SYSTEM.LIBRARY
- SYSTEM.MISCINFO
- SYSTEM.CHARSET
- PRINCIPAL.CODE
- GENERADOR.CODE
- RECONOCE.CODE
- APREN.CODE
- UTILERIAS.CODE

2) Para el disco marcado " SIREI 2 ":

- BAS_MAST.DATA
- A_FC.DATA
- A_REGPAR.DATA

Cabe mencionar que en el disco " SIREI 2 " los archivos se crean en la primera sesión con el sistema, asi que la primera vez que se consulte el directorio de este disco es posible que no se encuentren en él (referirse al capítulo de utilerías).

e) Una vez que el sistema despliegue la pregunta de la fecha, indica al usuario que el mismo se encuentra corriendo en memoria principal y es él quien ahora tiene el control de la computadora.

f) El formato que se debe seguir para darle la fecha correcta al sistema se indica entre paréntesis de la siguiente manera : (AA/MM/DD). Ejemplo : si la fecha es 25 de Diciembre de 1985, se debe responder a la pregunta 85/12/25.

g) Una vez aceptada la fecha, el sistema responde desplegando el Menú Principal de la siguiente forma :

0 - Fin

1 - Generador

2 - Reconocedor

3 - Aprendizaje

4 - Utilerías

Cada parte se selecciona tecleando el número correspondiente.

Nota : si es la primera vez que se corre el sistema, es necesario entrar a la sección de utilerías para crear los archivos de datos que se utilizarán posteriormente, asi como también dar de alta las reglas de producción de las figuras básicas susceptibles a reconocimiento.

A continuación se explica la operación de cada una de las secciones del sistema.

3. GENERADOR

Seleccionando el número 1 del menú principal, se da entrada al módulo generador de figuras, que en primera instancia, despliega el siguiente menú :

0 - Fin

1 - Inicializa Gráficas

2 - Cuadrado

3 - Rectángulo

4 - Generación Libre

5 - Modo de Graficación

Tecleando el número 0, indica al sistema la finalización del uso del generador, por lo que despliega el Menú Principal dando la opción de ejecutar otro parte del mismo, o bien, terminar la sesión con SIREI.

3.1 INICIALIZA GRAFICA.

Esta opción debe ser la primera seleccionada al entrar al módulo Generador ya que inicializa la pantalla de gráficas para asegurar la creación de figuras sobre un escenario limpio, ya que de otra manera, podría existir basura en el área de memoria reservada para la gráfica. El Generador despliega su menú después de que este proceso se complete (1 segundo aproximadamente).

3.2 CUADRADO.

Tecleando el número 2 del menú, el generador está listo para dibujar en la pantalla un cuadrado en la posición y con las dimensiones que el usuario le indique contestando a las preguntas desplegadas de la siguiente manera :

A la pregunta " Esquina (COL) -> " se responde un número que corresponde a la columna del vértice inferior izquierdo del cuadrado.

A la pregunta " Esquina (REN) -> " se responde un número que corresponde al renglón del mismo vértice.

Por último, la pregunta " Lado -> " se contesta con la longitud en "pixels" de un lado del cuadrado. Un pixel es un punto dentro de la pantalla, la cual contiene 280 x 192 pixels y el origen de la misma es la esquina inferior izquierda (0,0).

Si el vértice definido es válido, el generador responde poniendo una cruz (+) en el punto definido como vértice inferior izquierdo del cuadrado, dando la posibilidad de poder moverlo dentro de toda el área de la pantalla. Esta función tiene la finalidad de situar la figura visualmente en el punto deseado.

El vértice definido se puede mover utilizando las siguientes teclas:

- Tecla " E " movimiento hacia arriba.
- Tecla " X " movimiento hacia abajo.
- Tecla " S " movimiento hacia la izquierda.
- Tecla " D " movimiento hacia la derecha.

Cuando el vértice se encuentre en la posición deseada, sólo basta presionar RETURN y el cuadrado es dibujado automáticamente y el control del programa regresa al menú del Generador. Si se entra a esta sección por error, se puede evitar la creación del cuadrado respondiendo RETURN a la primera pregunta, situación válida para las demás rutinas.

3.3 RECTANGULO.

Si el usuario opta por dibujar un rectángulo, sólo será necesario que conteste las siguientes preguntas :

```
Esquina   (COL) ->
Esquina   (REN) ->
Base            ->
Altura          ->
```

en donde las dos primeras definen la posición del vértice inferior izquierdo de la figura, la tercera el tamaño de su base y la cuarta la dimensión de su altura.

Al igual que en los puntos anteriores, se puede variar la posición del vértice con las teclas definidas.

3.4 GENERACION LIBRE.

Dentro de esta opción, se permite al usuario generar cualquier figura con líneas rectas definiendo el punto inicial de cada una, ya que a continuación la pantalla cambia a modo de graficación y aparece la cruz en el punto definido dando oportunidad de variar la posición de éste con las teclas que permiten el movimiento de la misma dentro de la pantalla. En el momento que se presione RETURN queda definida la posición inicial de la línea y el usuario queda en libertad de definir el punto final moviendo la cruz a la posición deseada. Una vez definido este punto, basta teclear RETURN para que la línea sea dibujada. Este proceso continúa hasta que el usuario teclee "999" en la definición del primer punto de la línea a dibujar, con lo que se despliega el menú del Generador.

3.5 MODO DE GRAFICACION.

Al seleccionar este inciso, el programa despliega la gráfica que se esta generando para poder observar detalles o ajustes pertinentes. Es conveniente observar la(s) figura(s) generadas antes de pasar a la etapa reconocedora para comprobar que se generó el escenario deseado. Tecleando RETURN se regresa al menú del Generador.

4. RECONOCEDOR

Esta etapa del sistema es la más importante ya que es la que se encarga de reconocer la(s) figura(s) generada(s) por el usuario, por lo que se le sugiere leer las restricciones del reconocedor tratadas en el capítulo 4 de la Tesis.

Su primera función es crear la matriz gráfica para facilitar el procesamiento de las figuras (referirse a la Tesis para detalles a este respecto) y durante esta función despliega el siguiente mensaje en la pantalla :

* * ESTOY PROCESANDO LA IMAGEN * *

El usuario puede observar como se van reconociendo las figuras ya que el programa borra los lados de cada una de ellas conforme los va reconociendo, y cuando reconoce una figura completa, procede a analizar sus inmediaciones o fronteras para buscar alguna figura en cualquiera de sus lados, pintando en la pantalla el recorrido de este análisis. Este proceso sigue hasta que el Reconocedor identifica todas las figuras que componen la gráfica, o bien, hasta encontrar una figura que no pueda reconocer.

Si el sistema logra armar toda la gráfica (se reconocieron todas las figuras en la misma), utilizando la llave única generada, busca la figura compuesta dentro de la base de conocimiento

correspondiente. Si nó se encuentra ésta en el conjunto de figuras previamente aprendidas, el usuario es notificado de este hecho mediante el siguiente mensaje :

* * LA FIGURA NO EXISTE * *

En caso de que la figura sea encontrada en la base de conocimiento, el Reconocedor responde :

LA FIGURA COMPUESTA GENERADA ES A/BA//B****

RETURN, PARA CONTINUAR

que, como ejemplo indica que se encontró una figura compuesta por cuatro figuras básicas, dos del tipo A y dos del B.

Nota : Si el reconocedor encuentra alguna figura básica que no pueda reconocer dentro de la figura compuesta, desplegará en la pantalla el siguiente mensaje :

!! ERROR !! FIGURA BASICA INVALIDA

En este caso el control regresa al Menú Principal y el usuario tiene las siguientes posibilidades para corregir el error :

- Dar de alta la regla de producción de la figura básica no reconocida.
- Generar nuevamente la grafica cuidando los errores (ver Restricciones del Reconocedor en la Tesis).

Al final de la etapa reconocedora, el control del sistema se regresa al Menú Principal.

5. APRENDIZAJE

Seleccionado la opción 3 del Menú Principal de SIREI se entra a la etapa de aprendizaje del mismo.

La función principal de este módulo es agregar, si el usuario así lo desea, aquella figura compuesta reconocida que no exista en la Base de Conocimiento correspondiente, para que el sistema sea capaz de identificarla en futuras experiencias si se vuelve a presentar.

En caso de que se desee aprender la figura reconocida en la etapa anterior, el usuario de sistema deberá contestar " S " a la siguiente pregunta que aparecerá en la pantalla :

DESEA APRENDER LA FIGURA COMPUESTA RECONOCIDA (S/N) ->

Despues de agregada la figura a la base de conocimiento, el control del programa regresa al Menú principal.

6. UTILERIAS

Este módulo, aunque secundario, es muy importante para el sistema y de gran ayuda para el usuario, ya que en él se inicializan los archivos de datos utilizados por SIREI además de generar los listados del contenido de los mismos para consulta del usuario.

En primera instancia, el programa despliega el menú correspondiente de la siquiente forma :

```
0 - Fin
1 - Inicialización de la BCFB
2 - Inicialización de la BCFC
3 - Cargador de las FBs
4 - Borrado de FCs
5 - Catálogo de FBs
6 - Catálogo de FCs
```

Tecleando la opción 1 de este menú, el usuario tiene la posibilidad de crear el archivo que contendrá las reglas de producción de las figuras básicas, el mismo se llama " BAS_MAST.DATA " y estará contenido en el disco marcado " SIREI 2 ". Cabe mencionar que si el archivo ya existe en el disco, la información contenida en él se perderá, por esto, el programa despliega la siguiente pregunta :

```
Esta opción destruye el Archivo de las FBs si existe
Desea continuar <S>i o <N>o
```

En caso de contestar " S ", el archivo será creado en el disco correspondiente sin tomar en cuenta su previa existencia. Si la contestación es " N " el control de este módulo regresa al menú listado anteriormente.

De igual forma, si se selecciona la opción 2 del menú, el usuario tiene la oportunidad de crear o borrar - si existe el archivo - la base de conocimiento para las figuras compuestas. El usuario tiene la opción de continuar o no, mediante una pregunta parecida a la del inciso anterior. El archivo creado se encontrará en el disco " SIREI 2 " bajo el nombre " A_FC.DATA ".

Si el usuario opta por la opción 3 del menú, tendrá la posibilidad de agregar reglas de producción de figuras básicas a la base de conocimiento correpondiente, en caso de no existir ésta, se desplegará el siguiente mensaje :

No existe el Archivo de FBs, teclear la opción 1 del Menú
RETURN para continuar

En caso de que el archivo exista, el usuario debe responder las siguientes preguntas que aparecerán en la pantalla (referirse al capítulo 4 de la Tesis donde se explica qué es una regla de producción de figuras básicas y cómo se construye) :

Nombre de la Figura Básica (15 car. máx. / 0 = Fin) ->

Esta pregunta se responde con un nombre, escogido por el usuario, que se asociará a la regla de producción de la figura básica que se prentende dar de alta, con la restricción de que debe empezar con una letra y no puede ser mayor a 15 caracteres alfanuméricos. En caso de entrar por error a esta sección, o bien, de haber terminado de dar de alta FBs, bastará con teclear RETURN para regresar al menú de utilerías.

Si el usuario definió un nombre válido para la nueva FB, se procede a capturar la regla de producción correspondiente de la siguiente forma :

Dame el sujeto (00 = Fin) ->

pregunta que sólo se puede responder tecleando " H", "@H", " V" o "@V" (referirse al capítulo 4 de la Tesis) o "00" para terminar de dar de alta la regla. Si se teclea "00" la primera vez que se pide un sujeto, el programa asume que se desea abortar este proceso, con lo que se regresa automáticamente al menú de esta sección; en otro caso, se asume que el usuario ha terminado de construir la regla de producción correspondiente y se pasa a la definición de las funciones de igualdad y desigualdad para la nueva FB.

Una vez aceptado el sujeto, se despliega la siguiente pregunta :

Dame el operador ->

que se contesta con un " + ". Este proeso se repite hasta que se defina por completo la regla de producción.

A continuación el sistema requiere del usuario las funciones de igualdad y desigualdad para la FB en cuestión. Estas se dan de alta respondiendo las siguientes preguntas :

Dame la igualdad #
Dame el primer dato (0 = Fin) ->
Dame el segundo dato (0 = Fin) ->

donde el signo # indica el número de la función igualdad a definir y los datos son los números correspondientes a los lados iguales que definen la igualdad. Si se teclea cero en estas preguntas, esto indica la finalización de definición de igualdades, por lo que se pasa a una sección similar pa las funciones desigualdad. Cuando las funciones desigualdad se den de alta y se teclee cero a las preguntas correspondientes, la FB en cuestión es dada alta en la BCFB y se le da oportunidad al usuario de dar de alta otra FB.

El cuarto punto del menú permite al usuario dar de baja figuras compuestas de la BCFC, para lo cual el sistema despliega lo siguiente :

BORRADO DE UNA FIGURA COMPUESTA

Número de la RP de la FC que desea eliminar ->

Figura # ::= A/B**

Correcto (S/N)

donde el signo # corresponde al número de la RP de la FC qye se desea eliminar y después del signo = aparece la RP mencionada. Esto último es necesario para que el usuario esté seguro de que la regla dé producción desplegada es la que se desea dar de baja. Si se responde la última pregunta con una "S", la RP será borrada del archivo correspondiente a la BCFC, en caso de no querer proceder, bastará con teclear "N" a esta pregunta. A continuación, se despliega el menú de Utilerías.

Los dos últimos puntos son similares en su operación por lo que se tratarán como uno solo. Si se desea obtener un listado de las FBs o FCs existentes, sólo es necesario contestar la siguiente pregunta :

CATALOGO DE FIGURAS BASICAS

Listado en Pantalla o en Impresora (P / I / RETURN=Fin) ->

donde tecleando " P " el listado es direccionado a la pantalla, " I " a la impresora y RETURN finaliza la ejecución de esta sección. Cuando se selecciona la impresora, se despliega, además, el mensaje :

* * Encender la Impresora, por favor * *

Cuando esta operación se completa, se regresa automaticamente al menú de la sección utilerías.

Una vez que el usuario termina de usar esta sección, basta que teclee " 0 ", estando en el menú, para que el sistema responda desplegando el Menú Principal de SIREI.