SOFTWARE SIMULADOR DE CIRCUITOS NEUMATICOS

YASMINA GOMEZ PEREZ

ENDER GAMEZ MONTERO

CORPORACION UNIVERSITARIA TECNOLOGICA DE BOLIVAR

FACULTAD DE INGENIERIA DE SISTEMAS

CARTAGENA D.T Y C.

2000

SOFTWARE SIMULADOR DE CIRCUÍTOS NEUMÁTICOS

YASMINA GOMEZ PEREZ

ENDER GAMEZ MONTERO

Trabajo de Grado presentado
para optar el título de
Ingeniero de Sistemas

Director
CARLOS J. FERRIOL
Licenciado en Matemática Cibernética

CORPORACION UNIVERSITARIA TECNOLÓGICA DE BOLIVAR

FACULTAD DE INGENIERÍA DE SISTEMAS

CARTAGENA D.T Y C.

2000

Cartagena de indias, octubre 19 de 1999

Señores:

CORPORACION UNIVERSITARIA TECNOLOGICA DE BOLIVAR

Comité de Evaluación de Proyectos

Facultad de Ingeniería de Sistemas.

La Ciudad.

Respetados señores:

Por medio de la presente nos permitimos presentar para su estudio y calificación el proyecto de grado " SOFTWARE SIMULADOR DE CIRCUITOS NEUMATICOS ", con el propósito de obtener el título de Ingeniero de Sistemas.

Cordialmente,

ENDER H. GAMEZ MONTERO **YASMINA JUDITH GOMEZ PEREZ**

Cartagena de indias, octubre 19 de 1999

Señores:

CORPORACION UNIVERSITARIA TECNOLOGICA DE BOLIVAR

Comité de Evaluación de Proyectos

Facultad de Ingeniería de Sistemas.

La Ciudad.

Respetados señores:

Por medio de la presente me permito presentar para su estudio y calificación el proyecto de grado " SOFTWARE SIMULADOR DE CIRCUITOS NEUMATICOS ".

Cordialmente,

_____

CARLOS FERRIOL DORTICOS

Lic. Matemática Cibernética.

Nota de aceptación

_____

_____

_____

_____

Presidente del jurado

_____

jurado

_____

jurado

Cartagena de Indias D.T. Y C., octubre 4 de 1999

“La corporación se reserva el derecho de propiedad intelectual de todos los trabajos de grado aprobados, los cuales no pueden ser explotados comercialmente sin su autorización. Esta observación debe quedar impresa en parte visible del proyecto”.

DEDICATORIA

A la memoria de mi padre Moisés Gómez Soto y a mi madre Vilma Pérez González

Yasmina.

DEDICATORIA

A mi padre Heriberto Gámez Calderón y a mi madre Ana Montero de Gámez.

A mis hermanos y a todos mis familiares:

Ender.

AGRADECIMIENTOS

Los autores expresan sus agradecimientos:

Gracias a Dios por darnos la fuerza y la sabiduría para seguir adelante.

Gracias a nuestros hermanos por su apoyo y colaboración.

Gracias a todas esas personas a las cuales nos une un lazo de la amistad y que nos acompañaron en esta etapa de la carrera.

Gracias a Eduardo Concepción a Carlos Ferriol y a Miguel Angel Romero por su asesoría y apoyo incondicional.

CONTENIDO

LISTA DE CUADROS

LISTA DE FIGURAS

LISTA DE ANEXOS

GLOSARIO

ABSTRACCIÓN: Facilidad mental que permite a los humanos ver los problemas del mundo real con grados variables de detalle, dependiendo del contexto vigente del problema.

ACCIÓN: (En el modelo dinámico) operación instantánea. Las acciones se asocian a los sucesos y se suelen encontrar de manera explicita en la naturaleza.

ALMACÉN DE DATOS: Objeto pasivo que almacena datos para un acceso posterior.

ANÁLISIS: Fase del ciclo de desarrollo en el que se examina un problema del mundo real para comprender sus requisitos, sin planear su implementación.

APUNTADOR: Atributo de un objeto que contiene una referencia explícita a otro objeto.

ATRIBUTO: Propiedad con nombre de una clase que describe el valor de un dato guardado por cada uno de los objetos de una clase.

CARRERA DEL CILINDRO: Longitud del cilindro

CILINDRO: Elemento actuador de un circuito neumático. Es un elemento capaz de convertir la energía contenida en el aire comprimido en energía mecánica.

CLASE: Descripción de un grupo de objetos con propiedades similares, comportamientos comunes , interrelaciones comunes y semántica común.

CLASE ABSTRACTA: Clase que no puede tener instancias directas pero cuyos descendientes si pueden tenerlas.

COMPRESORES: Dispositivo neumático cuya función es aspirar el aire a presión de la atmósfera y comprimirlo a una presión elevada.

CONSUMO DE AIRE: Volumen de aire consumido en cada ciclo de trabajo

DIAGRAMA DE ESTADOS: Grafo dirigido en el que los nodos representan los estados del sistema y los arcos las transacciones entre estados

DIAGRAMA DE OBJETOS: Representación gráfica del modelo de objeto que muestra interrelaciones, atributos y operaciones.

DIAGRAMA DE ELIMINACIÓN DE SEÑALES: Es un diagrama que muestra la activación de las válvulas rodillo en una fase determinada.

DIAGRAMA ESPACIO FASE: Diagrama para la representación de las formas de estado de los elementos más importantes o bien de todos los elementos de un mando, dependientes de la correspondiente fase de conmutación.

ENCAPSULACIÓN: Técnica del modelado e implementación que separa los aspectos externos de un objeto, de los internos, detalles e implementación de un objeto.

ESCENARIO: (En el modelo dinámico) secuencia de sucesos que ocurren durante la ejecución de un determinado sistema.

ESTADO: Valores de los atributos y ligaduras de un objeto en un instante determinado.

FUERZA DEL CILINDRO: Es el producto de la presión por la superficie del cilindro.

HERENCIA: Mecanismo orientado a objetos que permite a las clases compartir atributos y operaciones, apoyando en una interrelación, normalmente de generalización.

HILO DE CONTROL: Ruta elemental de ejecución que atraviesa por un programa, un modelo dinámico o cualquier otra representación de control de flujo.

INSTANCIA: Objeto descrito por una clase

MÉTODO: Implementación de una operación para una determinada clase.

MODELO: Abstracción de algo con el propósito de comprenderlo antes de construirlo.

NEUMÁTICA: Parte de la mecánica que se encarga de desarrollar métodos que excluyan el trabajo manual y no dependan de la habilidad humana. Entre estos trabajos están: los accionamientos, elevación, alimentación y expulsión de materiales y transporte.

OBJETO: Concepto, abstracción o cosa con frontera y significado débil, perteneciente al problema que se trata; instancia de una clase.

POLIMORFISMO: Propiedad que permite una operación tener distintos comportamientos en diferentes clases.

PROCESO: Algo que transforma valores de datos.

SIMULACIÓN: Sistema que modela o sigue la pista de los objetos del mundo real.

SISTEMA: Colección organizada de componentes que interactúan.

UNIDAD DE MANTENIMIENTO: Dispositivo neumático cuya función es filtrar el aire comprimido, regular su presión, y lubricarlo para engrasar todos los elementos del circuito.

VARIABLE DE CLASE: Atributo de un objeto descriptor de clases.

VÁLVULAS: Dispositivo neumático que tiene la función de mantener o cambiar según ordenes o señales recibidas, las conexiones entre los conductos a ellos conectados para obtener una señal de salida de acuerdo con el programa establecido.

VELOCIDAD DEL ÉMBOLO: Relación de la carrera del cilindro y el tiempo.

1. ANTECEDENTES, EL PROBLEMA Y LOS OBJETIVOS DE LA INVESTIGACION

1.1 ANTECEDENTES DEL PROBLEMA

A medida que el tiempo transcurre, el Hardware y el Software siguen constituyendo un papel fundamental en la computación de tal modo que las organizaciones que están relacionadas con esta tecnología, van modificando y creando sus aplicaciones de acuerdo a sus necesidades.

Así mismo las computadoras cada día adquieren más poder y para hacer buen uso de ellas surge la necesidad de crear programas de mejor calidad y de menor costo.

La necesidad de crear aplicaciones ha sido aplicada en las industrias del software, así como también en las empresas que crean sus propias aplicaciones. Por ésta razón en el software se ha notado un salto en la complejidad, capacidad de diseño, flexibilidad, rapidez de desarrollo, y confiabilidad, que contribuyeron con la aparición de las primeras herramientas poderosas(I-CASE) y los métodos de ingeniería de gran utilidad para sintetizar los diseños con la ayuda de las computadoras.

Así es como se origina una combinación de herramientas y técnicas llamada “Programación Orientada a Objetos”, la cual data desde los principios de los años 60 cuando nace la idea de crear un lenguaje que sirva no sólo para describir los sistemas, sino también para programar la simulación.

Al madurar la técnica Orientada a Objetos, se facilita la actividad de programar, debido a que se crean bibliotecas con clases reutilizables con una complejidad cada vez mayor, que ayudan tanto al diseñador como al programador a especificar sus necesidades, ya que la definición de los datos implicaba definir la forma de “comportarse el mismo y así se lograban crear bibliotecas donde procedimientos y variables respondían a un único elemento llamado clase.

Esta es la forma como las herramientas poderosas modifican los métodos de construcción de aplicaciones. Por esto se trata de mejorar con dichas herramientas todo el proceso de desarrollo de las mismas, y estas dan lugar a la necesidad de representar modelos de la realidad, como lo son los sistemas de comunicación, los sistemas de producción, los sistemas mecánicos e incluso los sistemas sociales.

Igualmente se puede observar que las aplicaciones que actualmente se utilizan son realizadas con herramientas que soportan este estilo de Programación que permiten un desarrollo detallado y su ejecución activa.

Teniendo en cuenta el poder y facilidad de manejo que nos brinda la Programación Orientada a Objetos en la actualidad, se considera necesario su buen aprovechamiento en la CUTB y se propone implementar un Software de apoyo en la asignatura Sistemas Automáticos de Producción que sea capaz de representar el comportamiento de los circuitos neumáticos.

1.2 PLANTEAMIENTO DEL PROBLEMA

1.2.1 Descripción y análisis del proyecto. Actualmente los estudiantes de la CUTB de la asignatura Sistemas Automáticos de Producción, al realizar las prácticas de la materia, solo cuentan con un banco de circuitos neumáticos para simular las características y el funcionamiento de los mismos, así como los diferentes elementos que lo conforman como lo son: los cilindros, las válvulas, etc.

Por tratarse de un banco con elementos reales requiere de la supervisión del profesor a fin de tener en cuenta todas las normas de seguridad para su manejo.

Igualmente, es incomodo tanto para los estudiantes como para el profesor realizar sus prácticas de neumática. Además la CUTB no cuenta con un software que permita simular el funcionamiento de los circuitos neumáticos y los costos para adquirir nuevos bancos neumáticos son muy elevados.

Teniendo en cuenta que la Programación Orientada a Objetos es una herramienta de gran poder y muy útil para el desarrollo de aplicaciones, con la que se pueden

representar modelos de la realidad, nace así nuestra inquietud de aprovechar esta herramienta en la CUTB para implementar un Software capaz de representar el comportamiento de los circuitos neumáticos y de esta manera favorecer el proceso de enseñanza aprendizaje de esta asignatura.

1.2.2 Formulación del problema. Se desea realizar un software que permita diseñar circuitos neumáticos simples, con los elementos básicos que hacen parte de ellos, tales como; cilindros de simple y doble efecto, válvulas de vías, válvulas de caudal, etc.; y que permita realizar una simulación de su funcionamiento. El trabajo se pretende desarrollar como una herramienta para el S.O Windows para así aprovechar el entorno gráfico de este.

Dicho software se complementará con una ayuda teórica referente a los principales conceptos de la neumática.

1.3 JUSTIFICACION

En la actualidad un círculo cada vez más amplio de técnicos, ingenieros y especialistas deben enfrentarse con problemas de sistemas neumáticos de mando en las ramas más variadas de la producción. Por esta razón es esencial una buena preparación, y complementaria a esta se propone la implementación de un Software Simulador de Circuitos Neumáticos para su aplicación en la asignatura de Sistemas Automáticos de Producción, de la facultad de Ingeniería Mecánica de

la CUTB. El Software posibilita a los estudiantes realizar un circuito neumático y visualizar su funcionamiento en un computador, además ayuda a los estudiantes en la adquisición de conocimientos teóricos y prácticos.

El profesor de la asignatura contará con un medio de enseñanza que puede ser utilizado en los laboratorios de Informática de la CUTB. con un ambiente agradable, mejorando el proceso de enseñanza aprendizaje. Con este software se pretende ayudar a facilitar los primeros pasos en el campo de la Neumática.

1.4 OBJETIVOS

1.4.1 Objetivo general

- Diseñar e implementar un software que permita la creación y simulación de un circuito neumático, bajo un ambiente gráfico y de fácil manejo.

1.4.2 OBJETIVOS ESPECIFICOS

- Analizar todos los datos referentes al a la construcción y simulación de circuitos neumáticos simples.
- Crear la jerarquía de clases, identificando los métodos y atributos de cada clase.

- Realizar el diseño lógico de la creación y del funcionamiento de los circuitos neumáticos.
- Codificar utilizando como herramienta un lenguaje de programación orientado a objetos.

2. ESTRATEGIA METODOLOGICA DE LA INVESTIGACION

2.1 MARCO TEORICO

2.1.1 Programación orientada a objetos. La programación orientada a objetos (POO) es un método de implementación en el cual los programas se organizan como colecciones cooperativas de objetos, cada una de esas colecciones representa una instancia de una clase. Las clases son miembros de una jerarquía de clases unidas mediante relaciones de herencia**.**

La Programación Orientada a Objetos (POO) utiliza objetos, no algoritmos como bloques de construcción lógicos(jerarquía de objetos). Un **Objeto** una abstracción o cosa con frontera y significado débil.

Cada objeto es una instancia de una clase, las clases se relacionan unas con otras por medio de relaciones de herencia.

En POO se define una **clase** como una colección de elementos de datos, que se denominan *atributos* y junto con las funciones asociadas, que describen su comportamiento que se denominan *métodos*, utilizadas para operar sobre estos datos. Dicho de otro modo una clase es una declaración de un tipo de objeto.

Debido a esta característica de las clases se incluye el concepto de encapsulaciòn de los datos que no es más que la combinación de los datos y del comportamiento en un solo paquete.

Los **atributos** describen el estado de un objeto y se comportan como variables. El valor de un atributo puede ser de tipo de dato primario como: enteros, caracteres, booleanos, reales etc. o tipos complejos como: listas, arreglos, pilas e incluso clases.

Los **métodos** describen el comportamiento asociado a un objeto. Representan las acciones que puede realizarse por un objeto o sobre un objeto. La ejecución de un método puede conducir a cambiar el estado actual o el dato local del objeto. Cada método tiene un nombre y un cuerpo que realiza la acción o comportamiento asociado con el nombre del método. Todos los métodos que acceden o alteran a los datos de un objeto se definen dentro del objeto. Un objeto puede acceder o modificar los datos de otro objeto si este se lo permite.

Las clases son similares a los tipos de datos. Cada vez que se construye un objeto a partir de una clase, se crea lo que se llama ***instancia de una clase***.

La **herencia** es una propiedad de la orientación a objetos, con la capacidad de construir clases a partir de otras clases y el propósito de esta es construir código

para funciones especializadas, dicho de otro modo, la capacidad de un objeto para utilizar las estructuras de datos y métodos previstos en antepasados o ascendientes. El objetivo final es la reutilizabilidad, es decir reutilización de código.

La herencia supone una jerarquía de clases compuesta por una ***clase base*** y las clases derivadas de la clase base. Las clases derivadas heredan código y los datos de su clase base, añadiendo su propio código especial y datos a ella, incluso cambiar aquellos elementos de la clase base que necesitan sean diferentes. Cualquier clase derivada puede convertirse en la clase base para otra derivada de la jerarquía.

Existe un tipo de herencia denominada herencia múltiple donde una clase derivada puede tener varias clases bases.

El **polimorfismo** permite desarrollar sistemas en que los objetos diferentes puedan responder de modo diferente al mismo mensaje.

Ejecución de un programa orientado a objeto

Cuando se ejecuta un programa orientado a objeto ocurren tres sucesos:

1. Los objetos se crean a medida que se necesitan.

2. Los mensajes se mueven de un objeto a otro (o del usuario al objeto) a medida que el programa procesa información internamente o responde a la entrada del usuario.
3. Cuando los objetos ya no son necesarios, se borra y se libera la memoria.

2.1.2 Neumática. Actualmente para desarrollar cualquier proceso de producción, se desarrollan métodos que excluyan el trabajo manual y no dependan de la habilidad humana. La energía neumática puede realizar muchas funciones mejor y más rápidamente y sobretodo por mas tiempo sin sufrir efectos de fatiga.

Entre los trabajos más importantes que puede realizar la energía neumática se citan los siguientes:

1. accionamiento
2. elevación
3. Alimentación y expulsión de materiales.
4. transporte

Los principales elementos para el diseño de circuitos neumáticos son los siguientes:

1. El cilindro neumático que es un elemento capas de convertir la energía contenida en el aire comprimido en energía mecánica. Los cilindros neumáticos pueden ser:

- *cilindros de simple efecto*
- *cilindros de doble efecto*

Los cilindros de doble efecto tienen dos entradas de aire, están formados por un émbolo y un vástago unidos que se desplazan dentro de un tubo circular cerrado en cada extremo por cada cabezal deslizándose sobre juntas convenientemente situadas para evitar perdidas de aire.

El cilindro de simple efecto tiene una sola entrada de aire y el retorno lo hace por el muelle interior.

2. Las válvulas neumáticas son las que gobiernan el movimiento de los cilindros entre los más importantes tenemos:

Las válvulas distribuidoras que controlan la dirección del aire comprimido permitiendo su arranque, paro o sentido de circulación. El objetivo de las válvulas distribuidoras es el de mantener o cambiar según ordenes o señales recibidas, las conexiones entre los conductos a ellos conectados para obtener una señal de salida de acuerdo con el programa establecido.

Las válvulas de bloqueo cortan el paso del aire comprimido. Estas válvulas están construidas de manera que el aire comprimido actúa sobre la pieza de bloqueo reforzando el efecto de cierre. Las válvulas de bloqueo más utilizadas son:

- válvulas antiretorno
- válvulas selectoras
- válvulas de simultaneidad

Las válvulas antiretorno permiten el paso del aire en un sentido pero no en el contrario.

Las válvulas selectoras tienen dos entradas y una salida, el efecto de bloqueo actúa siempre en el sentido de la entrada purgada, por lo que queda libre el paso de la otra entrada. Si en ambas entradas hay simultáneamente presión siempre habrá presión en la salida.

Las válvulas de simultaneidad se utilizan para el control. Esta válvula tiene dos entradas y una salida. La señal de salida solo esta presente cuando las dos señales de entrada lo están.

Las válvulas de caudal permiten regular el caudal de aire comprimido para lograr la disminución o aumento de la velocidad de desplazamiento de un cilindro entre estas están:

1. válvulas reguladoras unidimensionales
2. válvulas reguladoras bidimensionales
3. válvulas de escape rápido

Las reguladoras unidimensionales permiten la libre circulación de aire en un sentido y en el contrario intercalan una estrangulación que fija el caudal del aire una vez determinada la presión, mientras que las reguladoras bidimensionales regulan el paso del aire en ambos sentidos.

La válvula de escape rápido permite una salida más rápida del aire comprimido al ambiente, aumentando de esta manera la velocidad de entrada o salida del cilindro. En la neumática se utilizan **compresores** cuya función del compresor neumático es aspirar el aire a presión de la atmosférica y comprimirlo a una presión elevada.

La unidad de mantenimiento consta de un filtro, regulador de presión, y un lubricante esta seguida de la fuente de presión. Su función es filtrar el aire comprimido, regular su presión, y lubricarlo para engrasar todos los elementos del circuito.

2.1.2.1 Características Técnicas para los Cilindros Neumáticos

Fuerza del cilindro

La transmisión de potencia mediante aire comprimido se basa en el principio de Pascal: “Toda presión ejercida sobre un fluido se trasmite íntegramente en todas las direcciones”. Por tanto la fuerza ejercida por un émbolo es igual al producto de la presión por la superficie.

En los cilindros de simple efecto debe reducirse la fuerza del muelle recuperador, y en los cilindros de doble efecto debe reducirse en la carrera de retroceso el área del vástago del área total del émbolo.

Consumo de aire

Otra característica importante es la cantidad de aire a presión necesario para el funcionamiento de un cilindro. Se entiende por consumo de aire de un cilindro, al volumen de aire consumido en cada ciclo de trabajo.

Velocidad del émbolo

La velocidad media del émbolo en los cilindros estándar está comprendida entre 0,1 y 1,5 m/s. En los cilindros especiales la velocidad puede ser mayor.

La velocidad del émbolo es función de la presión de trabajo, de la fuerza antagonista, de las secciones de las tuberías y también del diámetro nominal de la válvula de mando. Además la velocidad del émbolo puede ser afectada por válvulas estranguladoras o por válvulas de escape rápido.

Carrera del cilindro

En comparación con los cilindros de simple efecto con muelle de retorno, la carrera de los de doble efecto está considerablemente menos limitada. Las principales razones para la limitación de las carreras son:

- la disponibilidad comercial de los materiales para la fabricación de piezas largas.

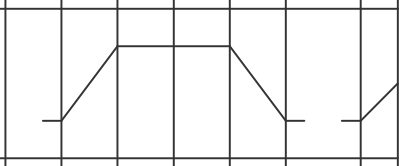
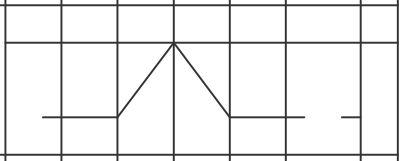
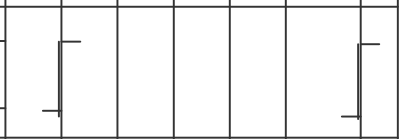
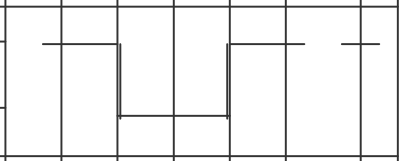
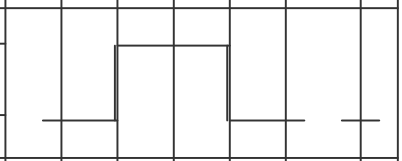
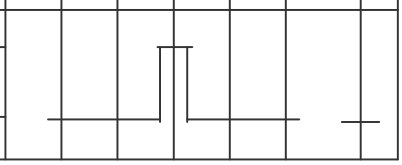
- La proporción entre la longitud del vástago y su diámetro

2.1.2.2 Esquema de un circuito neumático.

Diagrama Espacio Fase

Diagrama para la representación de las formas de estado de los elementos más importantes o bien de todos los elementos de un mando, dependientes de la correspondiente fase de conmutación.

Cuadro 1. Ejemplo de un diagrama espacio fase para una maquina remachadora.

Elemento		Función	Tiempo(s) Paso	1 2 3 4 5 / 1 2 3 4 5 = 1
1.0	Cilindro	Sujetar	Sale Entra	
2.0	Cilindro	Remachar	Sale Entra	
1.2	Válvula 3/2	Manda 1.1	1 0	
1.3	Válvula 3/2	Manda 0.2	1 0	
2.2	Válvula 3/2	Manda 2.1	1 0	
2.3	Válvula 3/2	Manda 0.2	1 0	

El esquema neumático debe diseñarse con normas VDI (VDI 3226 mandos neumáticos, esquemas) y DIN (DIN 24300 oleohidráhulica y neumática, denominaciones y símbolos). El desarrollo del mando debe dibujarse de abajo hacia arriba en el sentido del flujo de la energía, continuando la representación de los diferentes elementos de mando de izquierda a derecha.

Los elementos de trabajo se indican con los números 1, 2, 3, 4,... los elementos de mando con 1.1, 2.1, 3.1,... y los captadores de información y los elementos de tratamiento de señales con 1.2, 1.3, 2.2, 2.3, etc. La enumeración sigue el sentido contrario al flujo de la energía manteniendo el elemento índice de eslabón de mando al que pertenece, 1, 2, 3, etc.

3. ANALISIS Y DISEÑO DEL PROYECTO

3.1 METODOLOGIA OMT

La Técnica del Modelado de Objetos (OMT) es el nombre que se da a una Metodología que combina tres puntos de vistas distintos, pero siendo todos ellos necesarios para una descripción completa. El *modelo de objetos* representa los aspectos estáticos, estructurales "de datos" del sistema. El *modelo funcional* representa los aspectos transformacionales "de función" del sistema. El *modelo dinámico* muestra la secuencia de operaciones en el tiempo.

Las tres clases de modelo desglosan el sistema en puntos de vista ortogonales que se pueden representar y manipular empleando una notación uniforme.

3.1.1 El modelo de objetos. Describe la estructura de los objetos de un sistema identidad, relaciones con otros objetos, atributos y operaciones. El modelo de objetos proporciona el entorno esencial en la cual se pueden situar el modelo dinámico y el modelo funcional.

El modelo de objetos se representa gráficamente mediante diagramas que contienen clases de objetos, organizándose estas en jerarquías que comparten

una estructura y comportamiento comunes y que están asociadas con otras clases. Estas clases definen los valores de los atributos que lleva cada instancia de objeto y las operaciones que efectúa o sufre cada uno.

3.1.1.1 Clases

- **Circuito**

Esta clase está conformada por la unión de elementos y conexiones

- **Elementos**

Los elementos que intervienen en este análisis son:

1. Válvula Memoria
2. Válvula 3/2NCPP
3. Válvula 3/2NCPR
4. Válvula 3/2NCRR
5. Válvula 3/2NCRAR
6. Válvula de Simultaneidad
7. Válvula Selectora
8. Cilindro de simple efecto
9. Cilindros de doble efecto
10. Fuente

- **Secuencias**

Un circuito se puede generar a partir de una secuencia de 4 maneras diferentes.

1. Rodillo abatible
2. Cascada
3. Paso a paso mínimo
4. Paso a paso máximo

3.1.1.1.1 Descripción de las super clase ‘elemento’

ELEMENTO
NOMBRE ENTRADA SALIDA
GENERACION DE SALIDA

3.1.1.1.2 Descripción de las clases

VALVULA 3/2NCPR
ENTRADA X ENTRADA P SALIDA A
GENERACION DE SALIDA

VALVULA 3/2NCRAR
ENTRADA X ENTRADA P SALIDA A
GENERACION DE SALIDA

VALVULA 3/2NCRR
ENTRADA X ENTRADA P SALIDA A
GENERACION DE SALIDA

VALVULA 3/2NCPP
ENTRADA X ENTRADA Y ENTRADA P SALIDA A
GENERACION DE SALIDA

CILINDRO DE SIMPLE EFECTO
ENTRADA X SALIDA A SALIDA B DIAMETRO DEL ÉMBOLO LONGITUD DE CARRERA
GENERACION DE SALIDA EFECTO DE SALIDA Y ENTRADA

CIRCUITO
NOMBRE CONEXIONES ESTADOS
SIMULACION DEL CIRCUITO VERIFICACION DE CONEXIONES CONEXIÓN DE ELEMENTOS

CILINDRO DE DOBLE EFECTO
ENTRADA X ENTRADA Y SALIDA A SALIDA B DIAMETRO DEL ÉMBOLO LONGITUD DE CARRERA ESTRANGULACION ESCAPE RAPIDO
GENERACION DE SALIDA EFECTO DE SALIDA Y ENTRADA

VALVULA MEMORIA
ENTRADA X ENTRADA Y ENTRADA P SALIDA A SALIDA B
GENERACION DE SALIDAS

VALVULA DE SIMULTANEIDAD
ENTRADA X SALIDA A
GENERACION DE SALIDA

VALVULA SELECTORA
ENTRADA X SALIDA Y
GENERACION DE SALIDA

3.1.1.2 Diagrama de referencia

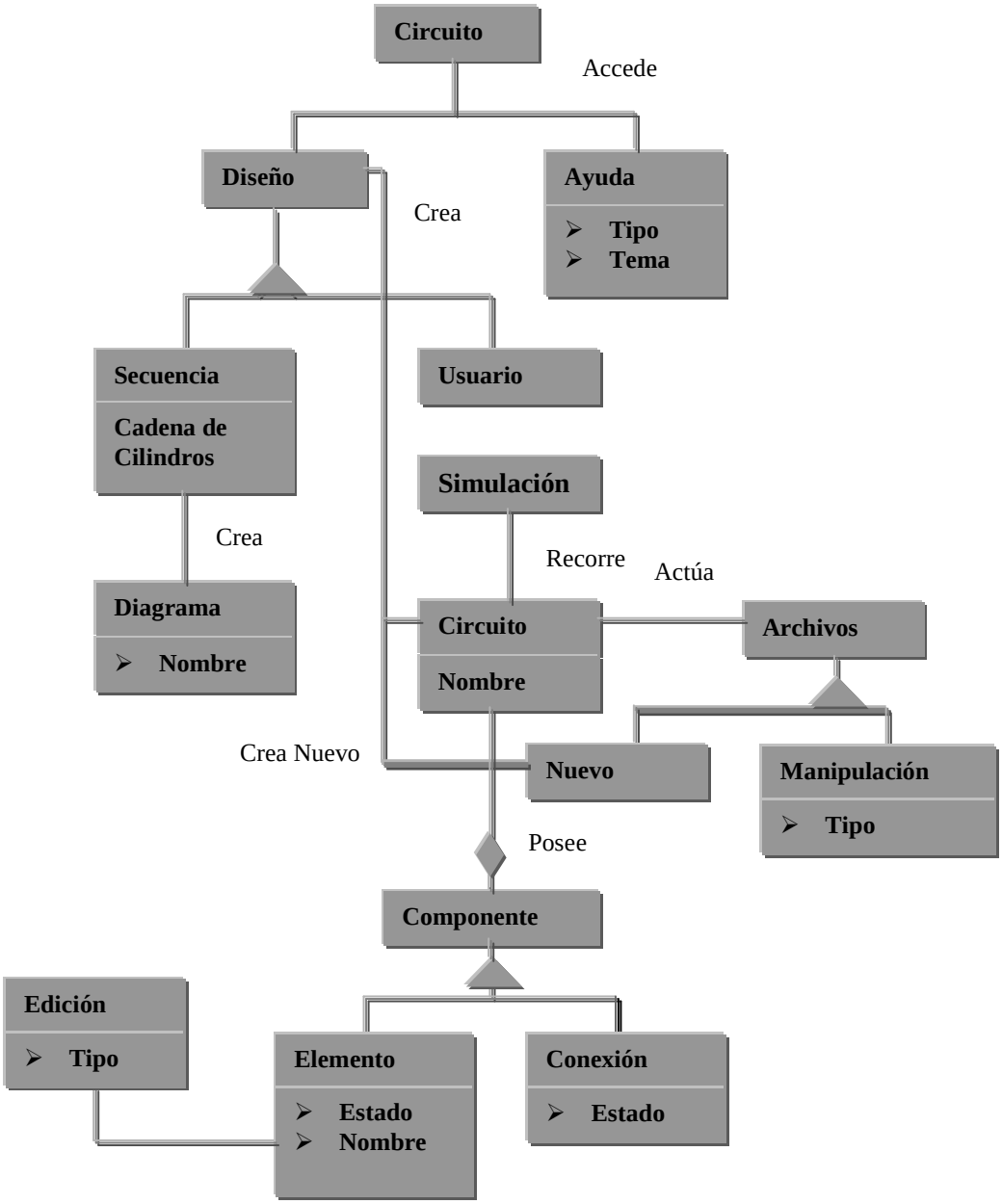


Figura 1. Diagrama de referencia

3.1.1.2.1 Módulo de diseño por usuario

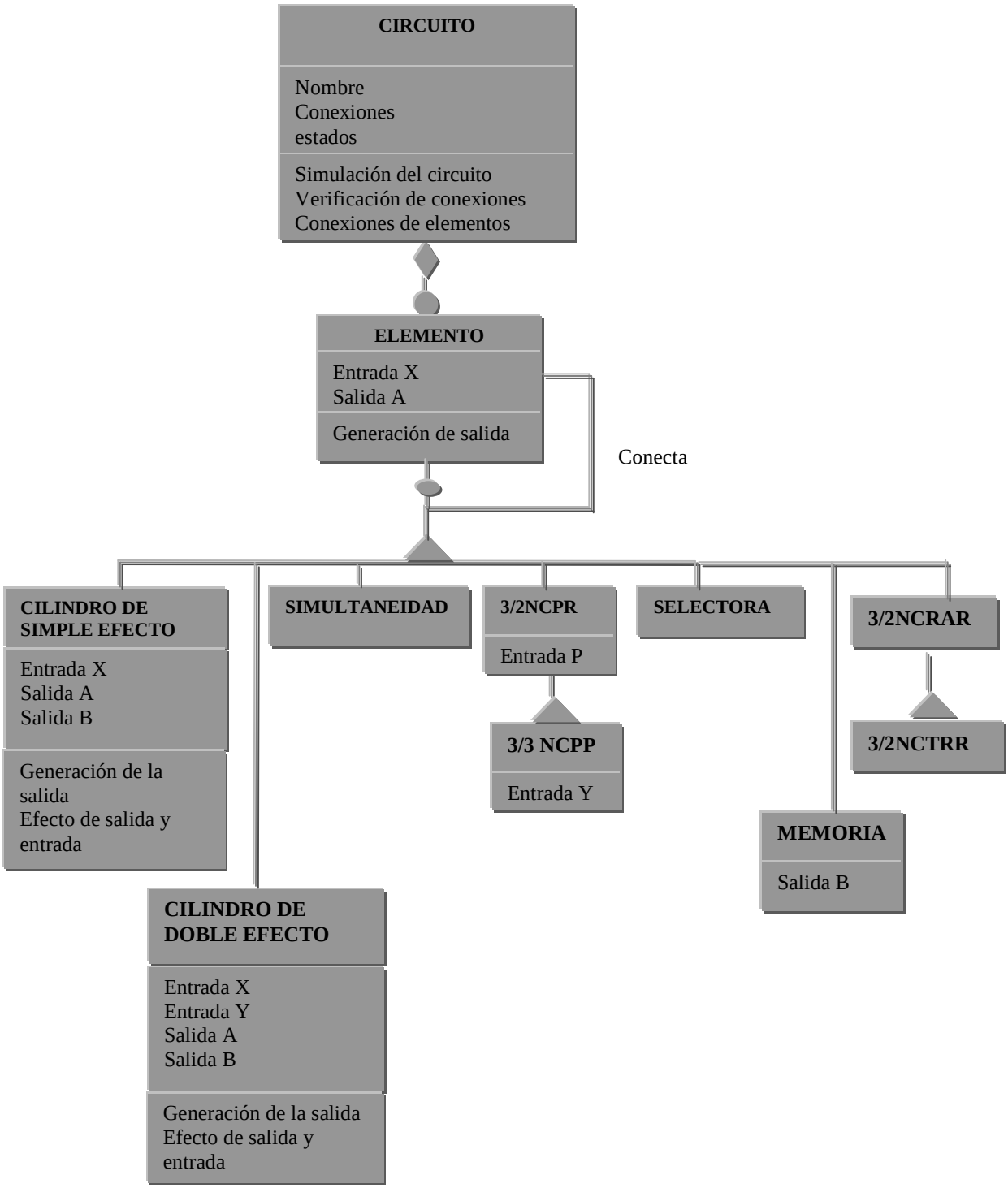


Figura 2. Diagrama del módulo de diseño por usuario

3.1.1.2.2 Modulo de diseño por secuencia

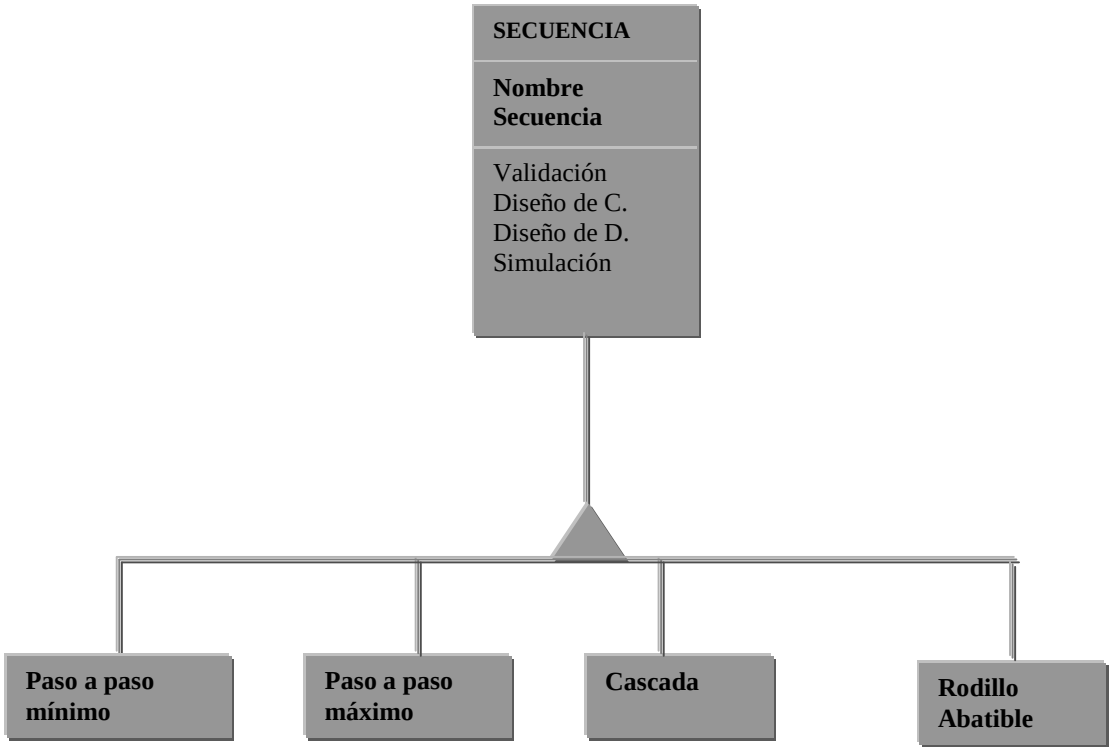


Figura 3. Diagrama del módulo de diseño por secuencia

3.1.2 El modelo dinámico. El *modelo dinámico* describe aquellos aspectos del sistema que tratan de la temporización y secuencia de operaciones, sucesos que marcan los cambios, secuencias de sucesos, estados que definen el contexto para los sucesos y la organización de sucesos y estados. El modelo dinámico captura el *control*, aquel aspecto de un sistema que describe las secuencias de operaciones que se producen sin tener en cuenta lo que hagan las operaciones, aquello a lo que afecten o la forma en la que estén implementadas.

El modelo dinámico se representa gráficamente mediante diagramas de estado. Cada uno de estos diagramas muestra el estado y las secuencias de sucesos que son admisibles en un sistema para una clase de objetos. Los diagramas de estado también hacen alusión a los demás modelos. Las acciones de los diagramas de estado se corresponden con funciones precedentes del modelo funcional; los sucesos de un diagrama de estado pasan a ser operaciones que se aplican a objetos dentro del modelo de objetos.

3.1.2.1 Escenario

3.1.2.1.1 Diseño del circuito por usuario(Escenario normal)

- Se activa el menú principal
- El usuario selecciona un nuevo circuito
- Se crea y se activa el menú gráfico
- El usuario coloca elementos sobre la pantalla de diseño
- El sistema guarda cada uno de los elementos
- El usuario activa la opción de conexión
- La pantalla de diseño se pone en estado de conexión
- El usuario realiza la conexión entre elementos
- El sistema verifica que la conexión sea correcta y relaciona los elementos conectados

- El usuario puede modificar las características de algunos elementos
- El sistema registra las nuevas características de los elementos
- El usuario toma la opción de simulación
- El sistema recorre la conexión de los elementos y muestra gráficamente el movimiento de los cilindros
- El sistema muestra una pantalla que contiene los valores de velocidad, tiempo, fuerza de salida y consumo de aire
- El usuario toma la opción de terminar simulación
- El usuario toma la opción de guardar el circuito
- El sistema guarda la parte gráfica del circuito y las variables asociadas a él que sirven para la simulación
- El usuario tiene la opción de imprimir el archivo
- El sistema prepara la impresión e imprime el archivo
- El usuario toma la opción de abrir un archivo de circuito
- El sistema abre ahora el archivo que el usuario desea abrir
- El usuario toma la opción ayuda, puede elegir entre ayuda del software y ayuda de neumática

3.1.2.1.2 Diseño del circuito por usuario(Escenario con excepciones)

- Se activa el menú principal

- El usuario selecciona un nuevo circuito
- Se crea y se activa el menú gráfico
- El usuario selecciona un elemento del menú gráfico y lo pega fuera de la pantalla de diseño
- El sistema no pega el elemento y tampoco lo registra
- El usuario activa la opción de conexión
- La pantalla de diseño se pone en estado de conexión
- El usuario hace una conexión del elemento con el mismo
- El sistema no acepta esta conexión
- La pantalla de diseño se pone nuevamente en estado de conexión
- El usuario realiza la conexión entre elementos
- El sistema verifica que la conexión sea correcta y relaciona los componentes
- El usuario toma la opción de simulación y los elementos no están conectados en su totalidad
- El sistema recorre el circuito solo con los elementos conectados
- Si hay un elemento conectado pero no completamente, el sistema no simula el circuito y manda un mensaje de error
- Los elementos son conectados correctamente
- El sistema recorre la conexión de los elementos y muestra gráficamente el movimiento de los cilindros
- El sistema detiene el movimiento
- El usuario toma la opción de guardar el circuito

- El sistema guarda la parte gráfica del circuito y las variables asociadas a él que sirven para la simulación
- El usuario tiene la opción de imprimir el archivo
- La impresora no esta preparada
- El usuario prepara la impresora
- El sistema prepara la impresión e imprime el archivo
- El usuario toma la opción de abrir un archivo de circuito
- El usuario da el nombre y el sistema acepta la operación,
- El archivo no existe
- El sistema pide el nombre correcto del archivo.
- El sistema abre ahora el archivo que el usuario desea abrir
- El usuario toma la opción ayuda, puede elegir entre ayuda del software y ayuda de neumática

3.1.2.1.3 Diseño del circuito por secuencia(Escenario sin excepciones)

- El usuario escoge el tipo de diseño a aplicar(cascada, rodillo abatible, paso a paso mínimo ó paso a paso máximo).
- El sistema pasa a una nueva ventana y pide una secuencia que el usuario debe introducir
- El usuario selecciona la secuencia y el sistema acepta la operación

- El sistema verifica que la secuencia sea correcta y realiza la gráfica del circuito generado por dicha secuencia
- El usuario selecciona ver diagramas de movimientos, de espacio fase o de eliminación de señales
- El usuario toma la opción de simulación
- El sistema detiene el movimiento
- El usuario toma la opción de guardar el circuito
- El usuario da un nombre y se acepta la operación
- El usuario tiene la opción de imprimir el archivo
- El sistema prepara la impresión
- El sistema imprime el archivo
- Si el usuario toma la opción ayuda, puede elegir entre ayuda del software y ayuda de neumática

3.1.2.1.4 Diseño del circuito por secuencia(Escenario con excepciones)

- El usuario escoge el tipo de diseño a aplicar(cascada, rodillo abatible, paso a paso mínimo ó paso a paso máximo).
- El sistema pasa a una nueva ventana y pide una secuencia que el usuario debe introducir
- El usuario introduce la secuencia

- El sistema verifica que la secuencia sea correcta y encuentra una secuencia incorrecta
- El sistema manda un mensaje de error y pide que se introduzca otra secuencia hasta que se escriba correctamente.
- El usuario selecciona la opción que muestra el circuito
- El sistema muestra la secuencia de salida y entrada de los cilindros
- El usuario selecciona ver diagramas ya sea de mando o de movimientos
- El usuario toma la opción de terminar simulación
- El sistema detiene el movimiento.
- El usuario toma la opción de guardar circuito
- El usuario da un nombre que ya existe
- El sistema pregunta que si desea sobre escribir el que ya existe
- El archivo se guarda con el nombre del archivo que ya existe y se borra el contenido del archivo anterior
- El sistema falla y el programa se suspende

3.1.2.2 Diagrama de estados

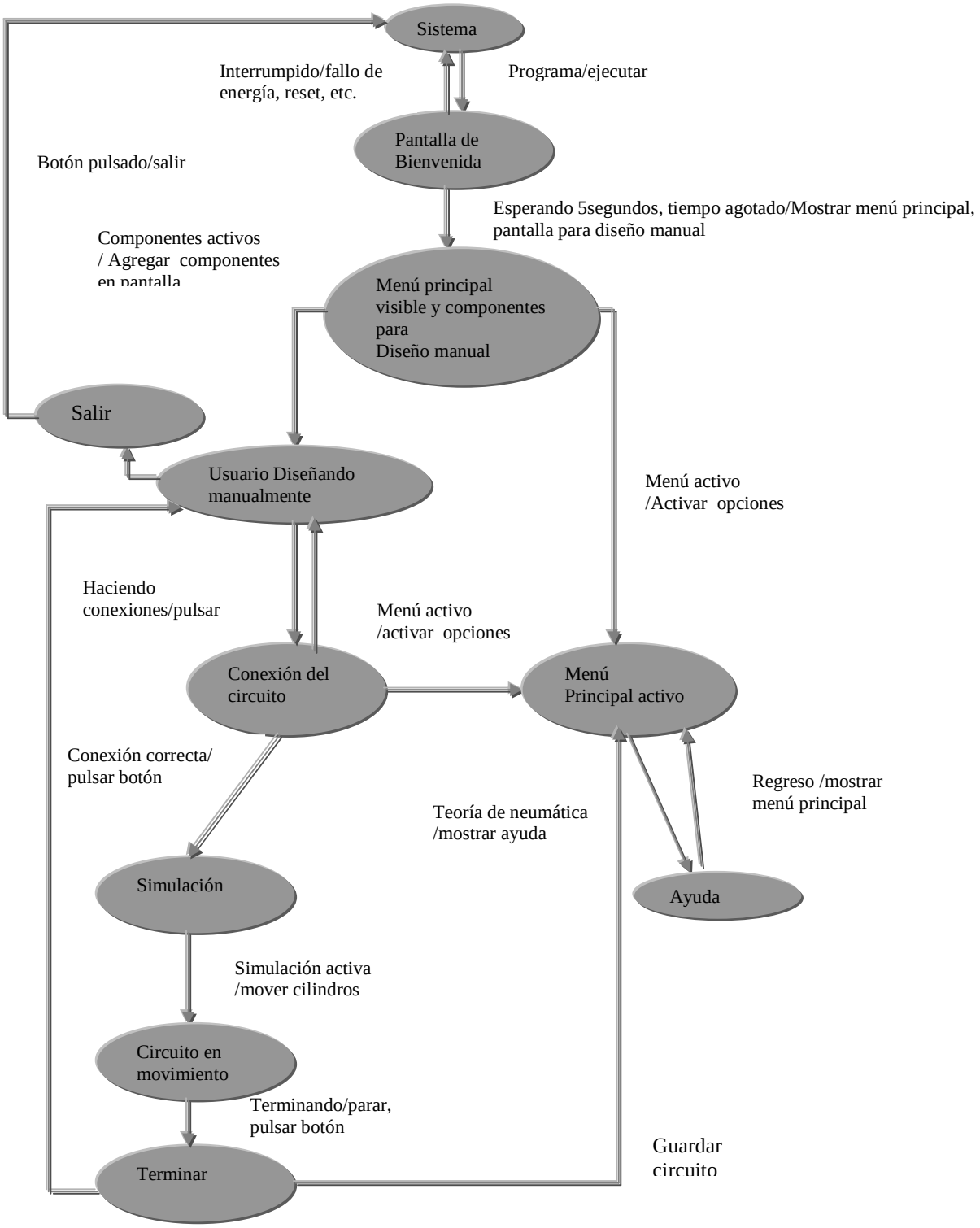


Figura 4. Diagrama de estados

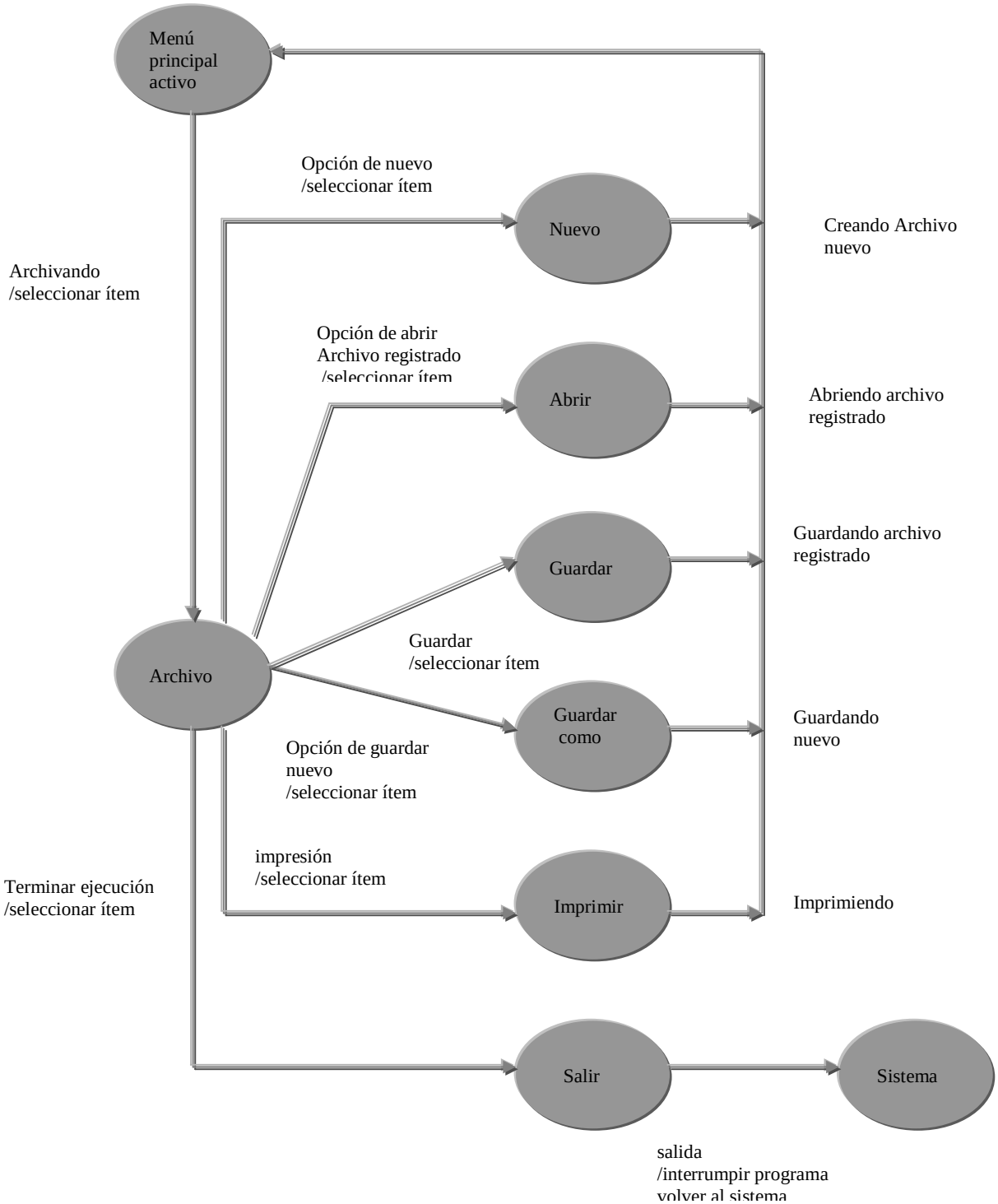


Figura 5. Segunda parte de diagrama de estados

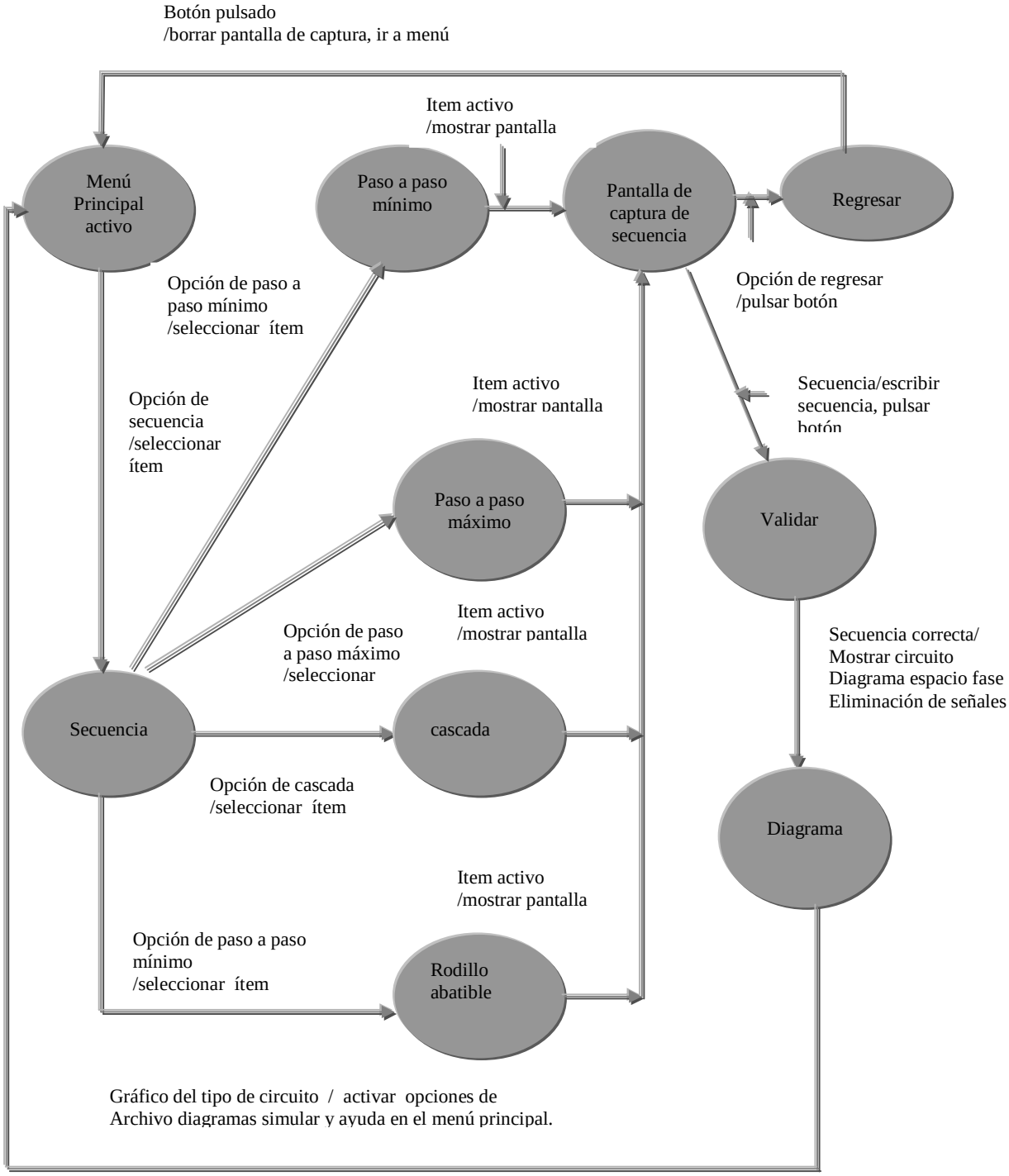


Figura 6. Tercera parte de diagramas de estado

3.1.2 El modelo funcional. El *modelo funcional* describe aquellos aspectos del sistema que tratan de las transformaciones de valores, funciones, correspondencias, restricciones y dependencias funcionales. El modelo funcional captura lo que hace el sistema, independientemente del cuando se haga o de la forma en que se haga.

El modelo funcional se representa mediante diagramas de flujos de datos. Estos muestran las dependencias entre los valores y el cálculo de valores, se da a partir de los datos de entrada y de las funciones, sin considerar cuando se ejecuten las funciones, ni siquiera si llegan a ejecutarse.

3.1.3.1 Proceso de seleccionar y pegar un elemento en la pantalla de diseño.

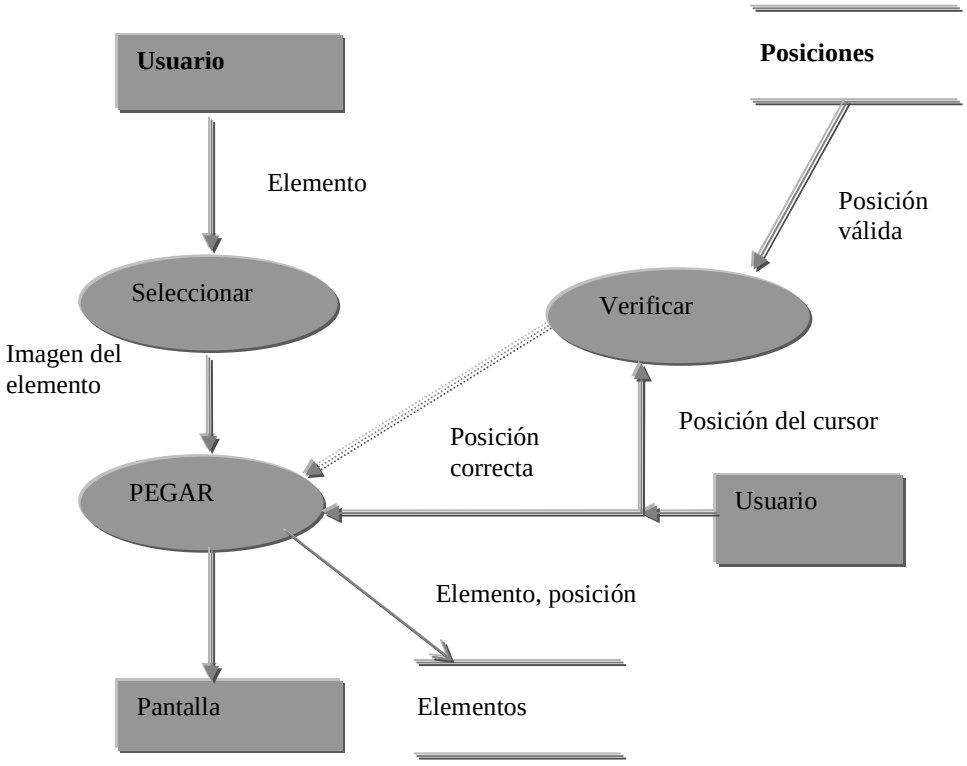


Figura 7. Seleccionar y pegar un elemento en la pantalla de diseño

Procesos:

- Verificar: Determina si la posición donde se colocara el elemento es correcta.
- Pegar: Coloca la imagen del elemento seleccionado en una posición de pantalla determinada.
- Copiar: Copia el elemento que luego será mostrado la pantalla de diseño.

3.1.3.2 Proceso de conectar dos elementos en la pantalla de diseño.

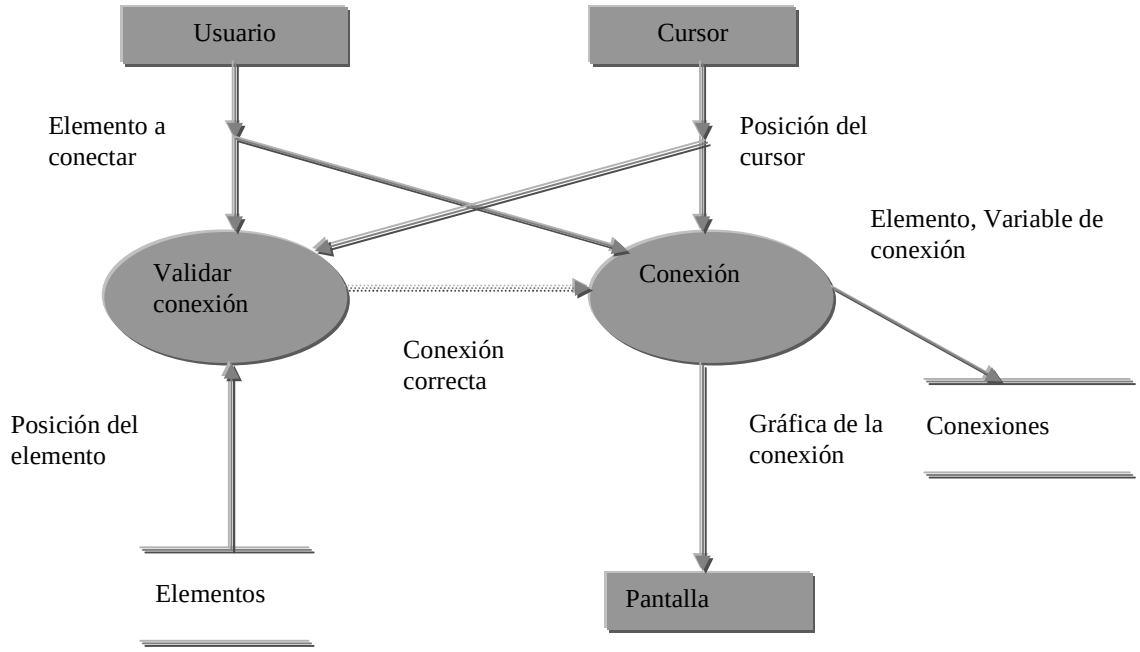


Figura 8. Proceso de conectar dos elementos en la pantalla de diseño

Procesos

- Validar conexión: Este proceso se encarga de verificar que un elemento no se conecte con el mismo y que la conexión termine en un punto valido(entrada de señal).
- Conexión: Realiza la gráfica de la conexión a partir de las coordenadas de los elementos a conectar.

3.1.3.3 Proceso de simular el recorrido del circuito.

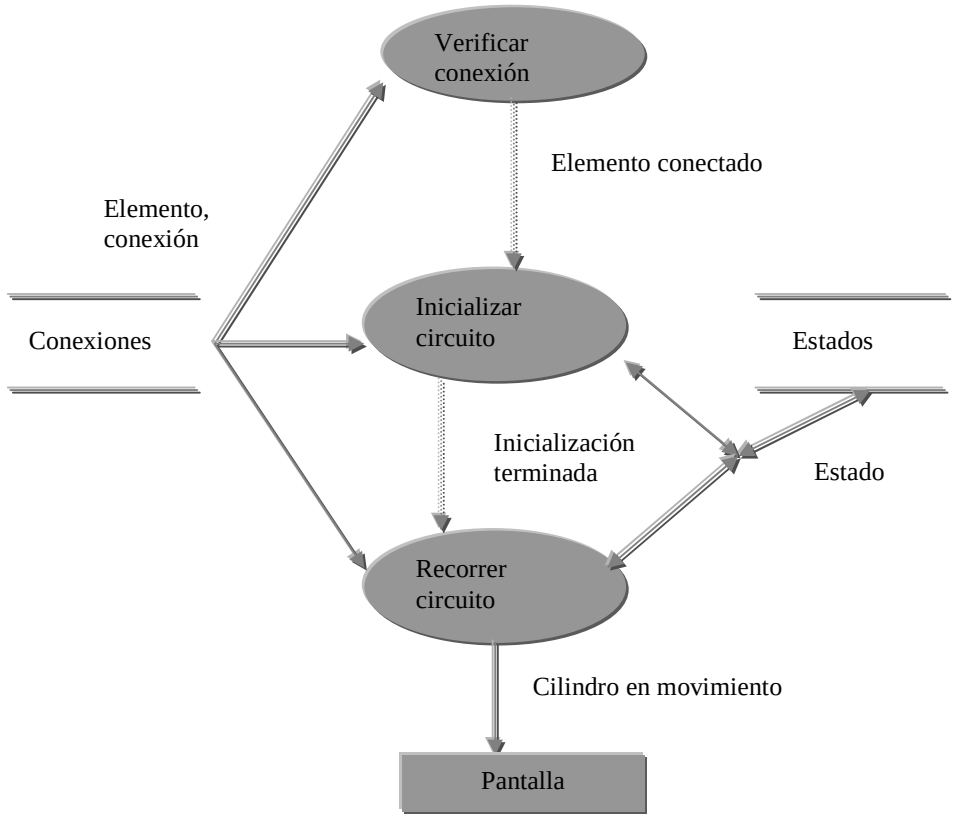


Figura 9. Proceso de simular el recorrido de un circuito

Procesos:

- Verificar conexión: Determina si todos los elementos tienen todas sus conexiones, es decir, que no falta ninguna conexión en un elemento.
- Inicializa circuíto: Se determina el estado inicial de las salidas de los elementos según como esten conectados.

- Recorrer circuito: Hace el recorrido del aire en el circuito, mostrando el movimiento de los cilindros en la pantalla.

3.1.3.4 Proceso de cambiar las características de un cilindro

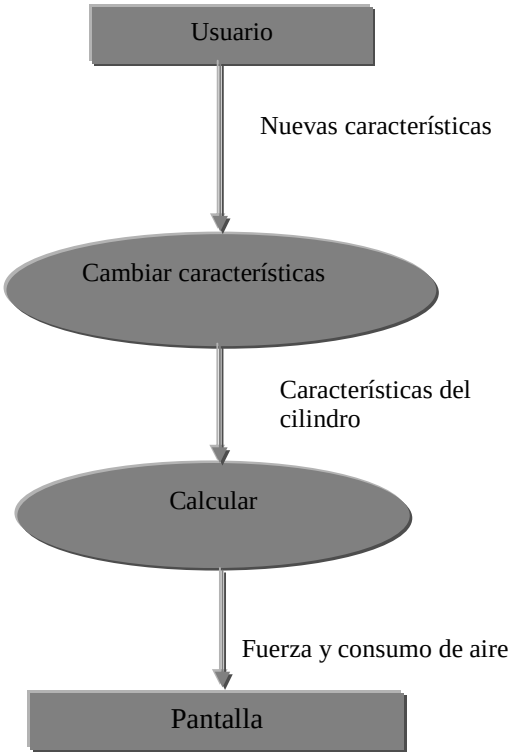


Figura 10. Proceso de cambiar las características de un cilindro

Procesos:

- Cambiar características: Recibe del usuario unas nuevas características las cuales causan efectos en la forma, la entrada y salida de este.
- Calcular: Se encarga de calcular, a partir de las características de un cilindro la fuerza, y consumo de aire del vástago.

3.1.3.5 Proceso de crear circuitos neumáticos a partir de una secuencia de entrada.

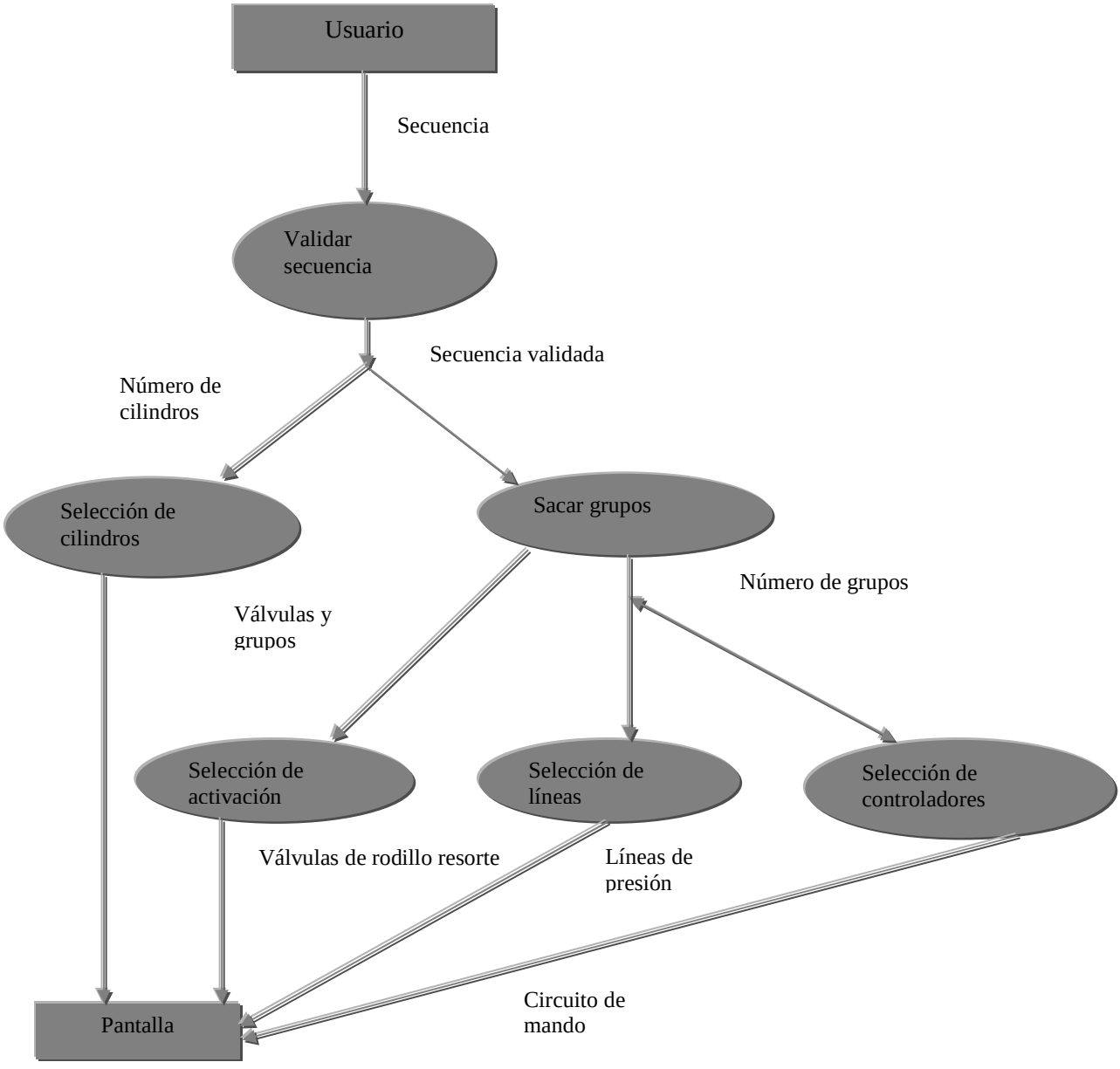


Figura 11. Procesos de crear circuitos neumáticos a partir de una secuencia de entrada

Procesos:

- Validar secuencia: Dada una secuencia de caracteres de entrada, determina si todos los caracteres son validos, igual que su posición en la secuencia.
- Seleccionar los grupos: Dada la secuencia valida determina él número de grupos de la secuencia, igual que las válvulas que componen dichos grupos
- Selección de cilindros: Dibuja los cilindros y memorias que se encuentran dentro de la secuencia.
- Selección de activación: Determina que válvulas activan las memorias y realiza las gráficas.
- Selección de líneas de presión: Determina a partir del número de grupos las líneas de presión que forman el circuito.
- Selección de controladores: Gráfica los circuitos de mando y determina las válvulas de cambio de estado.

3.1.4. **Relaciones entre modelos.** Cada modelo describe un aspecto del sistema, pero contiene referencias a los demás modelos. El modelo de objetos describe la estructura de los datos sobre la cual operan los modelos dinámico y funcional. El modelo dinámico describe la estructura de control de los objetos, muestra decisiones que dependen del valor de los mismos y que dan lugar a acciones que modifican valores y que invocan a funciones. El modelo funcional describe las funciones pedidas por operaciones en el modelo de objetos y

acciones del modelo dinámico. Las funciones operan sobre los valores de datos especificados por el modelo de objetos. El modelo funcional también muestra limitaciones de los valores de los objetos.

4. IMPLEMENTACION DEL SOFTWARE

4.1 LENGUAJE UTILIZADO

El lenguaje utilizado para la elaboración de este proyecto es Delphi 3, es un lenguaje visual orientado a objetos que hereda todas las ventajas de Object Pascal y ofrece una gran flexibilidad para el desarrollo rápido de aplicaciones.

También permite la creación e instalación de componentes visuales y sirve para el diseño de interfaces gráficas de usuario.

Hemos seleccionado este lenguaje porque es el que más se adapta a nuestras necesidades, es potente en la creación y manejo de objetos y ofrece ventajas ante otros lenguajes visuales por poseer una gran cantidad de componentes para el desarrollo de todo tipo de aplicaciones.

Un factor importante para utilizar este lenguaje y no otro en este proyecto es el conocimiento previo que se tiene sobre él y la facilidad de recibir asesorías sobre su manejo.

4.2 JERARQUIA DE CLASES

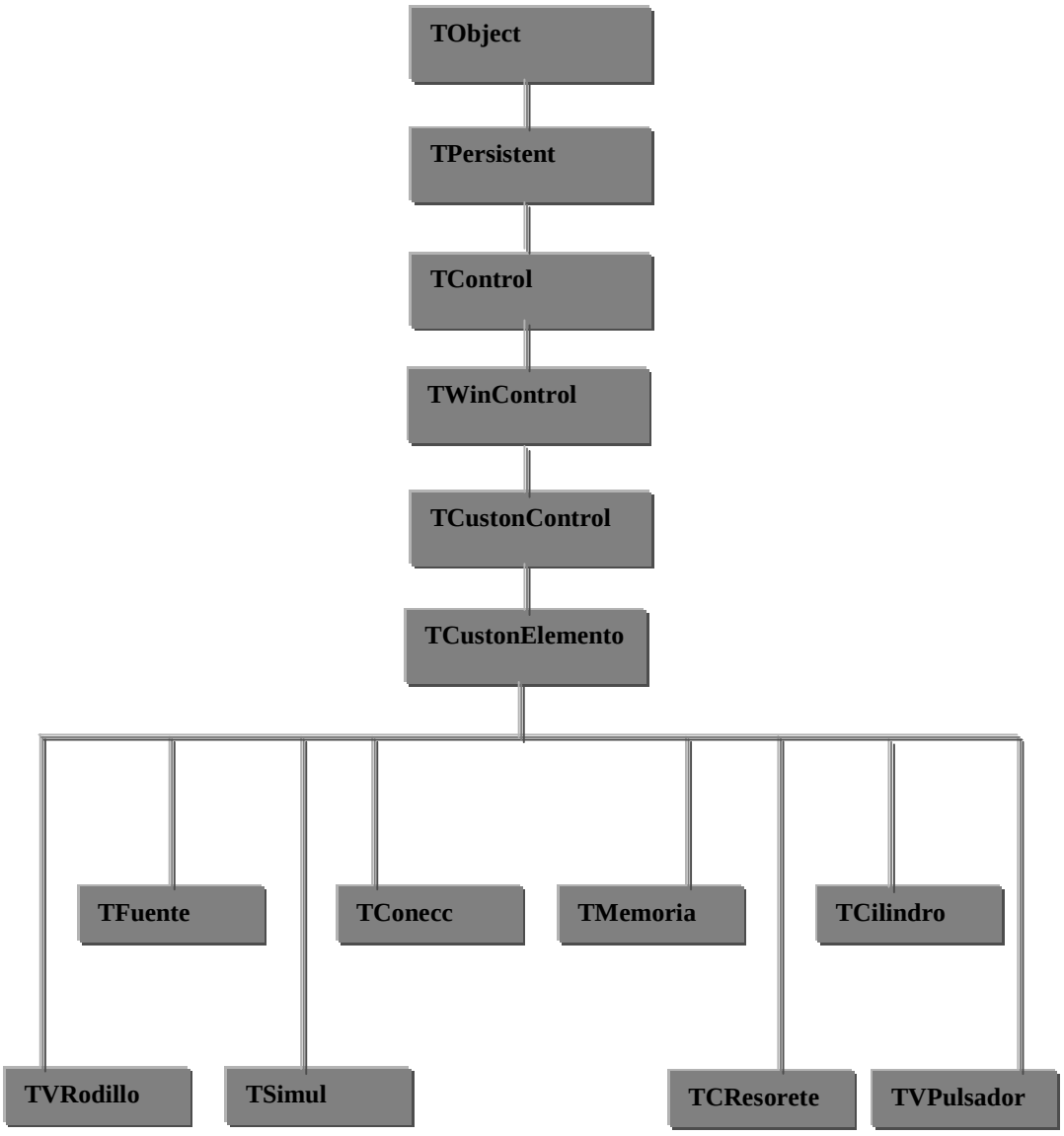


Figura 12. Jerarquía de clases

4.3 CLASES

4.3.1 La clase TcustomElemento. La clase TcustomElemento es una clase abstracta y es el nodo principal de la jerarquía de clases, de esta se derivan otras clases como TMemoria, TCilindro, TConecc, TSimul, TCresorte, TVPulsador, TVRodillo .

4.3.1.1 Tipos de datos

punto=Matriz de 2x20 de enteros

TipoSalida= toma los valores de SalidaA o toma el valor de SalidaB

Puntero= puntero a un elemento de la clase TCustomElemento

4.3.1.2 Propiedades

Xe: Entero, contiene la coordenada X de un punto de salida

Valor inicial: 0

Ye: Entero, contiene la coordenada Y de un punto de salida

Valor inicial: 0

Xs: Entero, Contiene la coordenada X de un punto de Entrada

Valor inicial: 0

Ys: Entero, contiene la coordenada Y de un punto de Entrada

Valor inicial: 0

LineaActiva: Boolean, es verdadero cuando el componente esta listo para ser conectado o conectar.

Valor inicial: verdadero

N: Entero, contiene el número de conexiones que hace un componente.

Valor inicial: 0

ConexiónX: Puntero, contiene la dirección al elemento que ha hecho conexión a la entrada X.

Valor inicial: nulo

ConexiónY: Puntero, contiene la dirección al elemento que ha hecho conexión a la entrada Y.

Valor inicial: nulo

ConexiónP: puntero, contiene la dirección al elemento que ha hecho conexión a la entrada P.

Valor inicial: nulo

SalidaX: TipoSalida, contiene la salida por la cual ha hecho conexión un elemento en la entrada X.

Valor inicial: SalidaA

SalidaY: TipoSalida, contiene la salida por la cual ha hecho conexión un elemento en la entrada Y.

Valor inicial: SalidaA

SalidaP: TipoSalida, contiene la salida por la cual ha hecho conexión un elemento en la entrada P.

Valor inicial: SalidaA

Salida: TipoSalida, contiene la salida por la cual un elemento va hacer una conexión.

Valor inicial: SalidaA

Entrada: Entero, contiene un valor entero de 1 si va a conectar la entrada X, 2 si se va a conectar la entrada Y, y 3 si se va a conectar la entrada P.

Valor inicial: 0

Conectado: Boolean, es verdadero cuando el elemento va hacer una conexión.

Valor inicial: Falso

Conector: Boolean, es verdadero cuando el elemento va ser conectado por otro elemento.

Valor inicial: falso

A: Entero, contiene el estado de la salida del elemento, 1 si hay flujo de aire, 0 si no hay y –1 si no ha sido inicializado

Valor inicial: -1

X: Entero, contiene el valor de la entrada X, es uno cuando hay entrada de presión, 0 si no hay y –1 si no ha sido inicializado.

Valor inicial: -1

Inicializado: Boolean, es verdadero cuando el elemento tiene entradas y salidas validas.

Valor inicial: falso

Puntos: punto, es una matriz que contiene las coordenadas finales de las conexiones realizados por el elemento

Valor inicial: 0

4.3.1.3 Métodos

Inicialización

Es un método abstracto y dinámico, su principal función es leer y evaluar las entradas del elemento para dar una salida válida que determina el estado inicial del elemento. Este estado inicial puede ser de 0 o 1.

Recorrido

Es también un método dinámico y abstracto, lee y evalúa las entradas del elemento para dar una salida de 0 o 1.

4.3.2 La clase Tfuente. Representa una fuente de presión y es parte indispensable de un circuito neumático. Tiene una salida representada en la propiedad A la cual siempre será 1. Tiene una Zona sensible de salida, por lo tanto cualquier elemento del circuito conectado a ella, tendrá sobre su entrada una señal de 1.

4.3.2.1 Principales propiedades de la clase

Nombre: Tlabel, contiene el nombre asignado al elemento.

4.3.2.2 Métodos de la clase

Inicialización: Asigna un valor a la salida A y a la propiedad Inicializado.

Create: Inicializa las propiedades de la clase al momento de ser creado el elemento, crea una instancia de la clase Tlabel que contiene el nombre del elemento.

4.3.2.3 Eventos Utilizados en la clase

MouseDown: Responde a un click del ratón. Evalúa que el click se dé sobre una zona sensible del elemento, si esto se cumple se actualizan las propiedades, relacionadas a cada zona sensible.

MouseMove: Responde al movimiento del señalador del ratón, al mover el señalador sobre una zona sensible de salida A, cambia la forma de este.

4.3.3 La clase Tpulsador. La válvula pulsador tiene la propiedad de controlar el paso de la señal de aire a través del circuito y al momento de la simulación, es el único elemento que permite una interacción entre el circuito y el usuario. La válvula pulsador tiene una entrada de presión representada en la propiedad P y una salida representada en la propiedad A. La válvula pulsador tiene dos estados uno en el cual se bloquea el paso de aire sobre la salida A y otro en que se deja libre el paso del aire. Esta válvula cambia de estado mediante un pulsador.

Existen dos tipos de válvulas pulsador: la pulsador con resorte, tiene tres zonas sensibles, la zona sensible de pulsación al ser activada por el ratón da una señal de salida, la cual es anulada por el resorte. La válvula pulsador _pulsador, tiene

cuatro zonas sensibles: dos zonas de pulsación, las cuales al ser activada uno se desactiva la otra. Las zonas sensibles de salida A y P, son comunes para los dos tipos.

4.3.3.1 Propiedades de la clase:

Nombre: Tlabel, contiene el nombre asignado al elemento.

ActivacionPresion: Boolean, es verdadero si se ha hecho click sobre la zona sensible de presión P.

LineaSalidaA: Boolean, es verdadero si se ha hecho click sobre la zona sensible de salida A.

ActivaciónIzquierda: Boolean, es verdadero si se hace click sobre la zona sensible de pulsación Izquierda y se hace falso al hacer click sobre la zona de pulsación derecha del elemento.

Tipo: Ttipo, determina el tipo de válvula.

4.3.3.2 Métodos de la clase

Inicialización: Lee la entrada de presión, y la propiedad *ActivaciónIzquierda*, a partir de estas se determina el valor de la salida A así: si el tipo de válvula es pulsador con resorte, se hace la salida A siempre 0. Si el tipo es válvula pulsador _pulsador, la salida es uno solo si la propiedad *activaciónizquierda* es verdadera y la entrada P es uno. Si las entradas tienen un [1]valor valido se hace la propiedad *inicializado* verdadero.

Recorrido: Lee las entradas del elemento y da una salida de uno, solo si la entrada *P* es uno y la propiedad *ActivaciónIzquierda* es verdadera.

Create: Inicializa las propiedades de la clase al momento de ser creado el elemento, crea una instancia de la clase Tlabel que contiene el nombre del elemento.

4.3.3.3 Eventos Utilizados en la clase

MouseDown: Responde a un click del ratón. Evalúa que el click se dé sobre una zona sensible del elemento, si esto se cumple se actualizan las propiedades, relacionadas a cada zona sensible.

MouseMove: Responde al movimiento del señalador del ratón, al mover el señalador sobre una zona sensible de salida A, cambia la forma de este.

[1] Valor valido es 0 o 1, valor no valido -1.

4.3.4 La clase Tsimul. Esta clase representa la válvula simultaneidad, tiene dos entradas representadas en las propiedades X y Y. Tiene una salida representada en la propiedad A. Tiene 3 zonas sensibles, dos de conexión y una de salida.

4.3.4.1 Propiedades de la clase

Nombre: Tlabel, contiene el nombre asignado al elemento.

Y: Entero, toma el valor de la entrada Y.

4.3.4.2 Métodos de la clase

inicialización: Lee las entradas X y Y, solo si las dos entradas del elemento son uno, este tendrá una salida de uno. Si una de las entradas es cero la salida inicial del elemento es cero. Si las entradas X y Y tienen valores validos se hace la propiedad inicialización verdadero.

Recorrido: Lee y actualiza las propiedades X y Y, la salida A será uno solo si las dos entradas son uno y cero cualquiera de las dos entradas es cero.

Create: Inicializa todas las propiedades de la clase y crea una instancia de la clase Tlabel, la cual tiene el nombre del elemento.

4.3.4.2 Eventos utilizados por la clase

MouseDown: Responde a un click del ratón. Evalúa que el click se de sobre una zona sensible del elemento, si esto se cumple se actualizan las propiedades, relacionadas a cada zona sensible.

MouseMove: Responde al movimiento del señalador del ratón, el cual al moverse sobre una zona sensible, cambia la forma.

4.3.5 La clase Tconecc. Tiene dos entradas representadas en las propiedades X y Y, una salida representada en la propiedad A. El efecto de bloqueo actúa siempre en sentido de la entrada purgada, por lo que queda libre el paso por la otra entrada hacia la salida. Tiene 3 zonas sensibles dos de entradas y una de salida.

4.3.5.1 Propiedades de la clase

Y: Entero, contiene el valor de la entrada Y

Nombre: Tlabel, contiene el nombre de la válvula

4.3.5.2 Métodos de la clase

Inicialización: Lee las entradas X y Y, si las dos entradas son cero entonces la salida A será cero. Si una de las dos entradas es uno, la salida A es uno. Si las dos entradas tiene valores validos entonces la propiedad inicializado es verdadera.

Recorrido: Actualiza las entradas X y Y, si las dos entradas son cero entonces la salida A es cero, si no, la salida A es uno.

Create: Inicializa las propiedades de la clase y crea una instancia de la clase Tlabel.

4.3.5.3 Eventos utilizados por la clase

MouseDown: Responde a un click del ratón. Evalúa que el click se de sobre una zona sensible del elemento, si esto se cumple se actualizan las propiedades, relacionadas a cada zona sensible.

MouseMove: Responde al movimiento del señalador del ratón. Al mover el señalador sobre una zona sensible de salida A, cambia la forma de este.

4.3.6 La clase Tmemoria. esta clase representa una válvula Memoria, la cual tiene 3 entradas representadas en las propiedades X, Y y P, y dos salidas representadas en las propiedades A y B. Esta válvula tiene dos estados uno en el cual la salida A tiene el paso del aire y bloquea la salida B, el otro estado bloquea

la salida A y deja libre la salida B. Para cada entrada y salida hay una zona sensible. Si la válvula memoria tiene la entrada de presión activa siempre tendrá una señal de uno en una de sus salidas A o B dependiendo de la entrada X o Y que se encuentre activa en el momento.

4.3.6.1 Propiedades de la clase

NB: Entero, contiene el número de activaciones realizadas por la salida B.

PuntosB: Punto, es una matriz que contiene las coordenadas finales de las conexiones realizados por él elemento en su salida B.

B: Entero, contiene el valor de la salida B

Y: Entero, contiene el valor de la entrada Y

4.3.6.2 Métodos de la clase

Inicialización: Lee las entradas X, Y y P, si la entrada P es uno y si la entrada X es uno y la entrada Y es cero entonces la salida A es cero y la salida B es uno. Si la entrada X es uno y la salida Y es cero entonces la salida A es uno y la salida B es cero. Si la entrada P es cero, entonces la salida A y la salida B es cero. Si las entradas tienen un valor valido la propiedad *inicializado* toma el valor de verdadera.

Recorrido: Lee las entradas X, Y y P,. Si la entrada X es cero y la entrada Y es cero, entonces la salida A es cero y la salida B es uno. Si la entrada X es uno y la salida Y es cero, entonces la salida A es uno y la salida B es cero.

4.3.6.3 Eventos utilizados en la clase

MouseDown: Responde a un click del ratón. Evalúa que el click se de sobre una zona sensible del elemento, si esto se cumple se actualizan las propiedades, relacionadas a cada zona sensible.

MouseMove: Responde al movimiento del señalador del ratón. Al mover el señalador sobre una zona sensible de salida A, cambia la forma de este.

4.3.7 La clase Tcilindro. Esta clase representa una cilindro de doble efecto, el cual tiene 2 entradas representadas en las propiedades: Xinicial, YInicial. Este tipo de cilindro tiene 4 zonas sensibles , las que permiten conectarse con los demás elementos del circuito. Para cada entrada hay una zona sensible y para las activaciones de válvulas con la salida y entrada también hay dos zonas sensibles.

4.3.7.1 Principales propiedades

Xinicial: Contiene el valor inicial de la entrada del cilindro, entrada que permite que el cilindro saque el émbolo.

YInicial: Contiene el valor inicial de la entrada Y del cilindro, entrada que permite que el émbolo del cilindro entre.

4.3.7.2 Métodos de la clase

Inicialización: Permite hacer la inicialización de las variables del componente, si la entrada Xinicial es 1, Yinicial es 0 y si Xinicial es 0, entonces, Inicial es 1

Recorrido: Lee y actualiza las propiedades X y Y, del cilindro.

Create: Inicializa todas las propiedades de la clase y crea una instancia de la clase Tlabel, la cual tiene el nombre del elemento.

4.3.7.3 Eventos

MouseDown: Responde a un click del ratón. Evalúa que el click se de sobre una zona sensible del elemento, si esto se cumple se actualizan las propiedades, relacionadas a cada zona sensible.

MouseMove: Responde al movimiento del señalador del ratón. Al mover el señalador sobre una zona sensible del cilindro si se hace clic responde al efecto de conectar el cilindro con otro elemento.

4.3.8 La clase Tcresorte. Esta clase representa una cilindro de simple efecto con retorno por muelle, el cual tiene 1 entrada representada en la propiedad: Xinicial, Este tipo de cilindro tiene 3 zonas sensibles , la que permiten conectarse con los demás elementos del circuito. Para la entrada hay una zona sensible y para las activaciones de válvulas con la salida y entrada del émbolo también hay dos zonas sensibles.

4.3.8.1 Principales propiedades

Xinicial: Contiene el valor inicial de la entrada del cilindro, entrada que permite que el cilindro saque el émbolo.

4.3.8.2 Métodos de la clase

Inicialización: Permite hacer la inicialización de las variables del componente EN en este caso la variable Xinicial.

Recorrido: Lee y actualiza las propiedades X inicial del cilindro.

Create: Inicializa todas las propiedades de la clase y crea una instancia de la clase Tlabel, la cual tiene el nombre del elemento.

4.3.7.3 Eventos

MouseDown: Responde a un click del ratón. Evalúa que el click se de sobre una zona sensible del elemento, si esto se cumple se actualizan las propiedades, relacionadas a cada zona sensible.

MouseMove: Responde al movimiento del señalador del ratón al igual que un cilindro de doble efecto, con la diferencia que hay menos zonas sensibles.

4.4 PRINCIPALES FUNCIONES

Conectar: Este procedimiento se encarga de realizar la conexión entre diferentes elementos. Para que este procedimiento se ejecute se debe haber hecho click sobre la salida de un elemento y sobre la entrada de otro. Este procedimiento actualiza las propiedades de conexión del elemento conectado. La conexión se realiza dibujando una línea desde el punto de salida del elemento conector hasta la entrada seleccionada del elemento conectado.

VerificarConexión: Verifica que todas las entradas de cada elemento estén conectadas, si esto se cumple se puede ejecutar el procedimiento de inicialización del circuito; si no se cumple se notifica al usuario.

Inicialización: Una vez realizada la verificación de la conexión se procede a inicializar el circuito llamando al método de inicialización para cada uno de los elementos que componen el circuito. Según sea la salida de cada elemento se

dibuja la línea de rojo si la salida es cero y de azul si la salida es uno. Si el circuito no se puede inicializar se notifica al usuario. Si el circuito es inicializado correctamente se puede realizar el procedimiento de simulación.

Simulación: Este procedimiento se encarga de llamar al método **recorrer** para cada uno de los elementos del circuito según sea la salida de cada elemento, de esta manera se dibujan las líneas con el color correspondiente.

Eliminar: Toma el elemento activo del circuito y llama al método **Destroy** del elemento y borra todas sus conexiones y actualiza las líneas del circuito.

Desconectar: El usuario determina que salida y que entrada se van a desconectar, el procedimiento borra la línea de conexión y actualiza las líneas del circuito. La entrada del elemento desconectado se habilita para que se pueda volver a conectar.

Validar_Secuencia: Es el encargado de leer la cadena capturada y validar restricciones como las siguientes:

1. Cada letra debe tener su estado, es decir no puede estar sola, ejemplo: A+ ó A-
2. siempre deben existir sólo dos estados para cada letra: A+, A-, B+ B- etc.
3. Cada letra debe comenzar con su estado positivo y luego en su estado negativo
4. las letras se pueden combinar de diferentes formas para generar diferentes circuitos. Ej. A+B+A-C+B-D+D-C-

5. Una secuencia no puede comenzar con un signo, sólo deberá comenzar con letras y finalizar con signos.

En caso de que estas restricciones no se cumplan se debe corregir la secuencia hasta obtener una correcta, este procedimiento se encarga de mostrar cada error.

Este procedimiento se ejecuta cuando es pulsado el botón BitBtn1 que lleva como etiqueta: Aceptar.

Graficacilindros: Es el encargado de dibujar los cilindros, las memorias y sus conexiones, recibe el valor global cc de la unidad 3 que indica la cantidad de cilindros para el circuito, se ejecuta cuando es llamado para el diseño de cada una de las 4 clases de circuitos.

Integrados: Realiza las gráficas de los circuitos integrados pequeños que contienen las válvulas selectora, simultaneidad y la 3/2 normalmente cerrada. Toma de la unidad 3 la variable ma-1 que corresponde al número de grupos para luego diseñar los integrados con la fórmula:

Nintegrados=#Grupos-1

Se ejecuta cuando es llamado desde el procedimiento estado1.

Simular: Este procedimiento realiza la simulación de la entrada y la salida de los cilindros, toma como referencia la secuencia de entrada capturada en edit1 de la unidad 3. Se ejecuta cuando se pulsa la opción simular del menú principal de la pantalla de diseño por secuencia.

Eliminaciondeseales1: Este Procedimiento realiza el diagrama de eliminación de señales y muestra gráficamente las señales que se deben eliminar colocando válvulas de rodillo abatible. Se ejecuta cuando se pulsa la opción eliminación de señales de la pantalla principal de diseño de circuitos por secuencia.

Espacio_fase: Este procedimiento realiza el diagrama del comportamiento de los cilindros en cada una de las fases de la simulación del circuito. Se ejecuta cuando se pulsa la opción espacio fase del menú principal o pulsando el botón etiquetado con esta opción.

5. REQUERIMIENTOS

Para la instalación y ejecución optima de este programa se requiere un equipo compatible con Windows con una memoria RAM de 16 MB. Un procesador de 133 Mhz o más, Mouse, impresora, unidad de CD-ROM y tarjeta de videos.

En caso de que la persona encargada de modificar el código de este programa desee editar el programa fuente deberá hacerlo ejecutando el paquete de Delphi 3. e instalar los componentes creados para este proyecto en la librería de Delphi.

Para una mejor ejecución del programa se requiere una configuración de pantalla de 800x600 píxeles.

6. RESULTADOS DE LA INVESTIGACION

Como resultados de la investigación se obtuvo un completo informe sobre el diseño y elaboración del software que contiene lo siguiente:

6.1 Documentos

- Manual del usuario
- Manual del sistema
- Documento que contiene los fundamentos teóricos y los pasos realizados que llevo a la creación del software.

6.2 Medio magnético

- Código fuente
- Instaladores

En las pruebas realizadas al programa se ha comprobado un buen funcionamiento en el diseño y simulación y hasta el momento se ha comprobado que cumple con los objetivos propuestos en este proyecto.

7. RECOMENDACIONES

- Para que el proyecto sea ampliado y mejorado se recomienda:

- Incluir en la ayuda mas teoría a cerca de la neumática tales como: ejemplos de circuitos neumáticos de casos reales.

- Ampliar el software para que sea un tutorial que abarque todos los temas de la materia sistemas automáticos de producción.

- Agregar nuevos componentes al menú.

- Utilizar la interfaz de usuario.

8. CONCLUSIONES

La culminación del proyecto “Software Simulador de Circuitos Neumáticos” refleja varios años de aprendizaje.

Este Software muestra los resultados de una larga investigación y proporciona una manera fácil de diseñar y simular circuitos neumáticos corroborando la inmensa posibilidad que pueden brindar las herramientas computacionales a otras ciencias e ingenierías y en este caso al diseño mecánico.

Además es importantísimo el aporte que hace el desarrollo educativo al desarrollo educativo al proporcionar con este software un “Taller virtual de neumática”

Su ambiente agradable y sencillo muestra las grandes ventajas que esta herramienta RAD (Desarrollo Rápido de Aplicaciones) proporciona. Otro logro importante relacionado con la implementación es el uso de la POO (Programación Orientada a Objetos) y la aplicación de la metodología OMT.

BIBLIOGRAFIA

CIRCUITOS BASICOS DE NEUMATICA. Carulla Miguel.

México. Alfaomega, Marcombo, 1995

INTRODUCCIÓN A LA NEUMATICA. Guillen Salvador Antonio.

Barcelona. Marcombo, 1993.

DISPOSITIVOS NEUMATICOS INTRODUCCION Y FUNDAMENTOS. Deppert w. Barcelona. Marcombo,1982.

PROGRAMACION ORIENTADA A OBJETOS. Luís Joyanes Aguilar. Mc Graw Hill, España 1996.

APRENDIENDO DELPHI 3. Dan Osier, Steve Grobman y Steve Batson.

Prentice Hall Hispanoamericana, S.A. México 1998.

MODELADO Y DISEÑO ORIENTADOS A OBJETOS. Rumbaugh, Blaha,

Premerlani, Lorensen y Eddy.

Prentice Hall Hispanoamericana, S.A. ESPAÑA 1991.

MANUAL DEL SISTEMA

INTRODUCCION

1. **REQUERIMIENTOS**
2. **PRINCIPALES VARIABLES**
3. **PRINCIPALES FUNCIONES**
4. **ORGANIZACIÓN DE LOS DIRECTORIOS**
4. **RECOMENDACIONES**

INTRODUCCION

1. REQUERIMIENTOS

2. PRINCIPALES VARIABLES

A continuación se presentan las principales variables que intervienen en la implementación de este software, se hace una pequeña descripción de ellas, su utilización y se especifica en que funciones intervienen.

VARIABLES

- *NuevoCilindro*
- *NuevaMemoria*
- *NuevaRodillo*
- *NuevaPulsador*
- *NuevaSimultaneidad*
- *NuevaSelectora*

- *NuevoCResorte*

Descripción

Estas variables son arreglos los cuales guardan en cada posición un elemento o instancia de la clase Tcilindro, Tmemoria, TVRodillo, TVPulsador, Tsimul, Tconecc y TCresorte respectivamente.

Utilización

Se utilizan paran para manejar los elementos del circuito de una manera más fácil ya que están dividido por clases.

Funciones

- *Conectar*
- *Simular*
- *Inicializar*
- *Eliminar*
- *Línea*
- *Compilar*
- *Desconectar*
- *Nuevo*
- *Abrir*
- *Guardarcomo*
- *Parar.*

VARIABLE NuevaFuente

Es una variable la clase Tfuente la cual contiene el elemento fuente del circuito.

Se utiliza para manejar el elemento **Fuente** del circuito.

VARIABLES

- *A*
- *B*
- *C*
- *D*
- *E*
- *F*
- *G*
- *H*

Descripción

Son variables de tipo entero, cada variable esta ligada a una clase.

Utilización

Controlan el número de elementos por clase que se han creado para el circuito. También se utilizan para verificar cuando sé ha creado un elemento de una clase especifica.

Funciones

- *Compilar*
- *Simular*
- *Inicializar*
- *Abrir*
- *Mousedown*

VARIABLES

- *ConCa*
- *ConCb*
- *ConCc*
- *ConCd*
- *ConCe*
- *ConCf*
- *ConCg*
- *ConCh*:

Descripción

Variables de tipo enteros, cada una esta relacionada con un arreglo de clase especifica.

Utilización

Se utilizan para controlar los elementos eliminados del circuito, también se utilizan al momento de crear un nuevo elemento.

Funciones

- *Eliminar*
- *mousedown(crear)*

VARIABLES

- *Ca*
- *Cb*
- *Cc*
- *Cd*
- *Ce*
- *Cf*
- *Cg*

- *Ch*

Descripción

Arreglos de enteros, cada arreglo representa a una clase determinada.

Utilización

Estas variables permiten controlar el borrado de los elementos

Funciones

- *Eliminar*
- *MouseDown(crear)*

VARIABLES

- *Xi*
- *Yi*
- *Xf*
- *Yf*
- *X1*
- *Y1*

Descripción

Variables de tipo entero, las cuales representan coordenadas.

Utilización

Se trabajan como auxiliares pare el manejo del dibujo de las líneas.

Funciones

- *mousedown (botón línea activa)*
- *mousemove(mover el cursor)*

VARIABLE *circuito*

Descripción

Archivo de registros.

Utilización

Guarda dentro de su registro las principales propiedades de los elementos que forman el circuito neumático.

Funciones

- *Abrir*
- *Guardarcomo*

VARIABLE *Ecircuito:*

Descripción

Es una variable del tipo de registro declarado para el archivo, contiene los siguientes campos:

Clase: Guarda un numero entero que va desde 1 a 8 para cada numero hay una clase asignada.

SalidaX, salidaY, salidaP: Estos campos guardan la propiedad del elemento llamada con el mismo nombre.

Nombre: es una cadena que guarda el nombre del elemento

conexiónX, ConexiónY, ConexiónP: Son matrices de 1X2 de tipo entero cuya columna indica la clase del elemento conector y la segunda columna indica la posición del elemento conector dentro del arreglo de su tipo de clase.

Tipo : Variable de tipo entero utilizado para aquellos elementos que tienen un tipo especifico.

Top: Este campo contiene la coordenada Y sobre la ventana que contiene el elemento.

Left: campo que contiene la coordenada X sobre la ventana que contiene el elemento.

Posi : Campo del tipo entero que contiene la posición del elemento dentro del arreglo de su clase.

N : Este campo contiene la propiedad con el mismo nombre del elemento.

LíneaSalidaA, LíneaSalidaB: Contienen la propiedad del elemento con el mismo nombre.

ActivaciónIzquierda, ActivaciónDerecha, ActivaciónPresion: Campos que contienen las propiedades del mismo nombre de los elementos.

NB: Contiene la propiedad del mismo nombre específicamente para el elemento de la clase Tmemoria.

Puntos: Campo que contiene la propiedad del mismo nombre para los elementos que tienen la característica de dibujar líneas

puntosB: Guarda la propiedad puntosB de la clase Tmemoria que es la única que tiene dos salidas neumáticas.

Utilización

Se utiliza para llenar el archivo y leer los registros del archivo.

Funciones

- *Abrir*
- *GuardarComo.*

VARIABLES

- *cadena2*
- *arre*
- *arreglo*
- *arreglo1*
- *arreglo2*
- *arreglo3*

Descripción

Son variables de tipo arreglo con posiciones que almacenan cadenas de 2 caracteres, en ellas se guarda partes de la secuencia de caracteres que se captura para el diseño de circuitos por secuencia.

Utilización

Se usa para hacer las validaciones de entrada de una secuencia, para hacer comparaciones y para selección de grupos. Se manejan a nivel global y son llamadas desde otras unidades de código.

Funciones

- *SpeedButton1Click*
- *SpeedButton2Click*

VARIABLE cadena

Es una variable de tipo string y es de gran uso porque sirve como variable auxiliar para la variable edit1. Esta variable es manipulada desde las funciones *SpeedButton1Click, SpeedButton2Click.*

VARIABLES

- *Ma*
- *Cc*

Descripción

Variables de tipo entero declaradas a nivel global en la unidad de captura y validación de secuencia.

Utilización

usadas para almacenar valores importantes relacionados con el número de cilindros a usar y las de líneas de grupos.

Funciones

- *SpeedButton1Click*
- *SpeedButton2Click*

Estas variables también son llamadas desde otros procedimientos pero no son modificados sus valores.

VARIABLES

- *K*
- *I*
- *J*
- *X*
- *Y*
- *X1*
- *Z*

Descripción

Variables de tipo entero declaradas a nivel global.

Utilización

De gran manejo especialmente en los bucles y como coordenadas para el diseño de circuitos paso a paso mínimo, paso a paso máximo, rodillo abatible y cascada.

Funciones

- *Estado1Click*
- *Mando1Click*
- *Empezar1Click*
- *Eliminaciondeseales1Click*
- *Pmínimo*
- *Rodilloabatible*
- *Graficacilindros*
- *Líneasdegrupo*
- *Memorias*
- *Integrados*
- *Unimantenimiento*
- *Startreset*
- *Válvularodillo*
- *Colocarmemorias*
- *Conectar*

- *Comparar*
- *Espaciofase*

3. PRINCIPALES FUNCIONES O PROCEDIMIENTOS

CONECTAR

Este procedimiento se encarga de realizar la conexión entre diferentes elementos. Para que este procedimiento se ejecute se debe haber hecho click sobre la salida de un elemento y sobre la entrada de otro. Este procedimiento actualiza las propiedades de conexión del elemento conectado.

CODIGO

```
procedure TForm1.botonconectarClick(Sender: TObject);
var
i,j,Xp,Yp,s:integer;
begin
with form2 do
begin
 for j:=1 to  5  do
 begin
  if (NuevoCilindro[j]<>nil)  then
   if NuevoCilindro[j].conector then
     begin
     conectora:=@NuevoCilindro[j];
     Nconectora:=j;
     end
   else
    if NuevoCilindro[j].conectado then
    begin
    conectado:=@NuevoCilindro[j];
    Nconectado:=j;
   end;

 if (NuevoCResorte[j]<>nil)  then
  if form2.NuevoCResorte[j].conector then
  begin
```

```
    conectora:=@NuevoCResorte[j];
    Nconectora:=j;
   end
   else
    if NuevoCResorte[j].conectado then
    begin
     conectado:=@NuevoCResorte[j];
     Nconectado:=j;
    end;
 end;// fin del for

 for j:=1 to  10  do
 begin
 if (NuevaMemoria[j]<>nil)  then
  if NuevaMemoria[j].conector then
   begin
   conectora:=@NuevaMemoria[j];
   Nconectora:=j;
   end
  else
  if NuevaMemoria[j].conectado then
  begin
   conectado:=@NuevaMemoria[j];
   Nconectado:=j;
  end;

 if (NuevaSimultaneidad[j]<>nil)  then
  if NuevaSimultaneidad[j].conector then
  begin
  conectora:=@NuevaSimultaneidad[j];
  Nconectora:=j;
  end
  else
  if NuevaSimultaneidad[j].conectado then
  begin
  conectado:=@NuevaSimultaneidad[j];
  Nconectado:=j;
  end;

 if (NuevaSelectora[j]<>nil)  then
  if NuevaSelectora[j].conector then
  begin
  conectora:=@NuevaSelectora[j];
  Nconectora:=j;
  end
 else
  if NuevaSelectora[j].conectado then
  begin
  conectado:=@NuevaSelectora[j];
  Nconectado:=j;
  end;
```

```
 if (NuevaFuente<>nil)  then
  if NuevaFuente.conector then
  conectora:=@NuevaFuente;

if (NuevaRodillo[j]<>nil)  then
 if NuevaRodillo[j].conector then
 begin
  conectora:=@NuevaRodillo[j] ;
  Nconectora:=j;
  end
  else
   if NuevaRodillo[j].conectado then
   begin
   conectado:=@NuevaRodillo[j];
   Nconectado:=j;
   end;

if (NuevaPulsador[j]<>nil)  then
 if NuevaPulsador[j].conector then
 begin
    conectora:=@NuevaPulsador[j];
    Nconectora:=j;
    end
 else
  if NuevaPulsador[j].conectado then
  begin
  conectado:=@NuevaPulsador[j];
  Nconectado:=j;
  end;
end;// del for

if (conectora <> nil) and (conectado<>nil) then
begin
//******************conexion
if conectado^.Entrada = 1 then
begin
// si se activa a un cilindro
if (conectado^ is Tcilindro) then
begin
if NuevoCilindro[NConectado].conexionX<>nil then
 if NuevoCilindro[NConectado].conexionX^=nil then
 NuevoCilindro[NConectado].conexionX:=nil;

 if NuevoCilindro[NConectado].conexionX=nil then
 begin
  if (conectora^ is Tfuente) then
  begin
  NuevoCilindro[NConectado].conexionX:=@nuevafuente;//conectora;
  NuevoCilindro[NConectado].salidaX:=Nuevafuente.salidas;
  Nuevafuente.N:=Nuevafuente.N+1;
```

```
  Nuevafuente.puntos[Nuevafuente.N,1]:=NuevoCilindro[NConectado].Xs;
  Nuevafuente.puntos[Nuevafuente.N,2]:=NuevoCilindro[NConectado].Ys;
  NuevoCilindro[NConectado].conectado:=false;//conectora;
  {if Nuevafuente.TipoConexion=Cindirecta then
  begin
  nohacerlinea:=true;
  form2.image1.canvas.Brush.style:=bsSolid;
  form2.image1.canvas.Brush.color:=clblue;
  form2.image1.Canvas.ellipse(conectado.Xs-
5,conectado.Ys,conectado.Xs+5,conectado.Ys+10);
  end;}
  end;

  if (conectora^ is Tmemoria) then
  begin
  NuevoCilindro[NConectado].conexionX:=@NuevaMemoria[NConectora];
  NuevoCilindro[NConectado].salidaX:=NuevaMemoria[NConectora].salidas;
  if NuevaMemoria[NConectora].salidas = salidaA then
   begin
   NuevaMemoria[NConectora].N:=NuevaMemoria[NConectora].N+1;

NuevaMemoria[NConectora].puntos[NuevaMemoria[NConectora].N,1]:=NuevoCilindro[NCon
ectado].Xs;

NuevaMemoria[NConectora].puntos[NuevaMemoria[NConectora].N,2]:=NuevoCilindro[NCon
ectado].Ys;
   end
  else
   begin
   NuevaMemoria[NConectora].NB:=NuevaMemoria[NConectora].NB+1;

NuevaMemoria[NConectora].puntosB[NuevaMemoria[NConectora].NB,1]:=NuevoCilindro[NC
onectado].Xs;

NuevaMemoria[NConectora].puntosB[NuevaMemoria[NConectora].NB,2]:=NuevoCilindro[NC
onectado].Ys;
   end;
  end;//si es una memoria

  if (conectora^ is Tsimul) then
  begin
  NuevoCilindro[NConectado].conexionX:=@Nuevasimultaneidad[NConectora];
  NuevoCilindro[NConectado].salidaX:=Nuevasimultaneidad[NConectora].salidas;
  Nuevasimultaneidad[NConectora].N:=Nuevasimultaneidad[NConectora].N+1;

Nuevasimultaneidad[NConectora].puntos[Nuevasimultaneidad[NConectora].N,1]:=NuevoCili
ndro[NConectado].Xs;

Nuevasimultaneidad[NConectora].puntos[Nuevasimultaneidad[NConectora].N,2]:=NuevoCili
ndro[NConectado].Ys;
  end;
```

```
  if  (conectora^ is Tconecc) then
   begin
   NuevoCilindro[NConectado].conexionX:=@NuevaSelectora[NConectora];
   NuevoCilindro[NConectado].salidaX:=NuevaSelectora[NConectora].salidas;
   NuevaSelectora[NConectora].N:=NuevaSelectora[NConectora].N+1;

NuevaSelectora[NConectora].puntos[NuevaSelectora[NConectora].N,1]:=NuevoCilindro[NCo
nectado].Xs;

NuevaSelectora[NConectora].puntos[NuevaSelectora[NConectora].N,2]:=NuevoCilindro[NCo
nectado].Ys;
    end;

  if (conectora^ is TVRodillos) then
  begin
  NuevoCilindro[NConectado].conexionX:=@Nuevarodillo[NConectora];
  NuevoCilindro[NConectado].salidaX:=Nuevarodillo[NConectora].salidas;
  Nuevarodillo[NConectora].N:=Nuevarodillo[NConectora].N+1;

Nuevarodillo[NConectora].puntos[Nuevarodillo[NConectora].N,1]:=NuevoCilindro[NConecta
do].Xs;

Nuevarodillo[NConectora].puntos[Nuevarodillo[NConectora].N,2]:=NuevoCilindro[NConecta
do].Ys;
  end;

  if  (conectora^ is Tvpulsador) then
  begin
  NuevoCilindro[NConectado].conexionX:=@Nuevapulsador[NConectora];
  NuevoCilindro[NConectado].salidaX:=Nuevapulsador[NConectora].salidas;
  //Nuevapulsador[NConectora].conector:=false;
  Nuevapulsador[NConectora].N:=Nuevapulsador[NConectora].N+1;

Nuevapulsador[NConectora].puntos[Nuevapulsador[NConectora].N,1]:=NuevoCilindro[NCon
ectado].Xs;

Nuevapulsador[NConectora].puntos[Nuevapulsador[NConectora].N,2]:=NuevoCilindro[NCon
ectado].Ys;
  end;
end // si la entrada esta conectada
else
 begin
 error:=1;// entrada ya conectada
 validaError(Error);
 nohacerlinea:=true;
 end;
end;//si es un cilindro


// si se activa a un cilindro con muelle
```

```
if (conectado^ is TCResorte) then
begin
if NuevoCresorte[NConectado].conexionX<>nil then
if NuevoCresorte[NConectado].conexionX^=nil then
NuevoCresorte[NConectado].conexionX:=nil;
if NuevoCresorte[NConectado].conexionX=nil then
begin
if (conectora^ is Tfuente) then
begin
NuevoCresorte[NConectado].conexionX:=@nuevafuente;//conectora;
NuevoCresorte[NConectado].salidaX:=Nuevafuente.salidas;
Nuevafuente.N:=Nuevafuente.N+1;
Nuevafuente.puntos[Nuevafuente.N,1]:=NuevoCresorte[NConectado].Xs;
Nuevafuente.puntos[Nuevafuente.N,2]:=NuevoCresorte[NConectado].Ys;

if Nuevafuente.TipoConexion=Cindirecta then
begin
nohacerlinea:=true;
form2.image1.canvas.Brush.style:=bsSolid;
form2.image1.canvas.Brush.color:=clblue;
form2.image1.Canvas.ellipse(conectado.Xs-
5,conectado.Ys,conectado.Xs+5,conectado.Ys+10);
end;
end;

if (conectora^ is Tmemoria) then
begin
NuevoCresorte[NConectado].conexionX:=@NuevaMemoria[NConectora];
NuevoCresorte[NConectado].salidaX:=NuevaMemoria[NConectora].salidas;
if NuevaMemoria[NConectora].salidas = salidaA then
begin
NuevaMemoria[NConectora].N:=NuevaMemoria[NConectora].N+1;
NuevaMemoria[NConectora].puntos[NuevaMemoria[NConectora].N,1]:=NuevoCresorte[NCon
ectado].Xs;
NuevaMemoria[NConectora].puntos[NuevaMemoria[NConectora].N,2]:=NuevoCresorte[NCon
ectado].Ys;
end
else
begin
NuevaMemoria[NConectora].NB:=NuevaMemoria[NConectora].NB+1;
NuevaMemoria[NConectora].puntosB[NuevaMemoria[NConectora].NB,1]:=NuevoCresorte[N
Conectado].Xs;
NuevaMemoria[NConectora].puntosB[NuevaMemoria[NConectora].NB,2]:=NuevoCresorte[N
Conectado].Ys;
end;
end;

if(conectora^ is Tsimul) then
begin
NuevoCresorte[NConectado].conexionX:=@Nuevasimultaneidad[NConectora];
NuevoCresorte[NConectado].salidaX:=Nuevasimultaneidad[NConectora].salidas;
```

```
Nuevasimultaneidad[NConectora].N:=Nuevasimultaneidad[NConectora].N+1;
Nuevasimultaneidad[NConectora].puntos[Nuevasimultaneidad[NConectora].N,1]:=NuevoCre
sorte[NConectado].Xs;
Nuevasimultaneidad[NConectora].puntos[Nuevasimultaneidad[NConectora].N,2]:=NuevoCre
sorte[NConectado].Ys;
end;

if (conectora^ is Tconecc) then
begin
NuevoCresorte[NConectado].conexionX:=@NuevaSelectora[NConectora];
NuevoCresorte[NConectado].salidaX:=NuevaSelectora[NConectora].salidas;
NuevaSelectora[NConectora].N:=NuevaSelectora[NConectora].N+1;
NuevaSelectora[NConectora].puntos[NuevaSelectora[NConectora].N,1]:=NuevoCresorte[NC
onectado].Xs;
NuevaSelectora[NConectora].puntos[NuevaSelectora[NConectora].N,2]:=NuevoCresorte[NC
onectado].Ys;
end;

if (conectora^ is TVRodillos) then
begin
NuevoCresorte[NConectado].conexionX:=@Nuevarodillo[NConectora];
NuevoCresorte[NConectado].salidaX:=Nuevarodillo[NConectora].salidas;
Nuevarodillo[NConectora].N:=Nuevarodillo[NConectora].N+1;
Nuevarodillo[NConectora].puntos[Nuevarodillo[NConectora].N,1]:=NuevoCresorte[NConecta
do].Xs;
Nuevarodillo[NConectora].puntos[Nuevarodillo[NConectora].N,2]:=NuevoCresorte[NConecta
do].Ys;
end;

if (conectora^ is TVpulsador) then
begin
NuevoCresorte[NConectado].conexionX:=@Nuevapulsador[NConectora];
NuevoCresorte[NConectado].salidaX:=Nuevapulsador[NConectora].salidas;
Nuevapulsador[NConectora].N:=Nuevapulsador[NConectora].N+1;
Nuevapulsador[NConectora].puntos[Nuevapulsador[NConectora].N,1]:=NuevoCresorte[NCo
nectado].Xs;
Nuevapulsador[NConectora].puntos[Nuevapulsador[NConectora].N,2]:=NuevoCresorte[NCo
nectado].Ys;
end;
end
else
begin
error:=1;// entrada ya conectada
validaError(Error);
nohacerlinea:=true;
end;
end;

// si se activa a un memoria
if (conectado^ is Tmemoria)then
begin
```

```
if Nuevamemoria[NConectado].conexionX<>nil then
if Nuevamemoria[NConectado].conexionX^=nil then
Nuevamemoria[NConectado].conexionX:=nil;
if Nuevamemoria[NConectado].conexionX=nil then
begin
if (conectora^ is Tfuente) then
begin
Nuevamemoria[NConectado].conexionX:=@nuevafuente;//conectora;
Nuevamemoria[NConectado].salidaX:=Nuevafuente.salidas;
Nuevafuente.N:=Nuevafuente.N+1;
Nuevafuente.puntos[Nuevafuente.N,1]:=Nuevamemoria[NConectado].Xs;
Nuevafuente.puntos[Nuevafuente.N,2]:=Nuevamemoria[NConectado].Ys;
if Nuevafuente.TipoConexion=Cindirecta then
begin
nohacerlinea:=true;
form2.image1.canvas.Brush.style:=bsSolid;
form2.image1.canvas.Brush.color:=clblue;
form2.image1.Canvas.ellipse(conectado.Xs,conectado.Ys-5,conectado.Xs-
10,conectado.Ys+5);
end;
end;

if (conectora^ is Tmemoria) then
begin
Nuevamemoria[NConectado].conexionX:=@NuevaMemoria[NConectora];
Nuevamemoria[NConectado].salidaX:=NuevaMemoria[NConectora].salidas;
if NuevaMemoria[NConectora].salidas = salidaA then
begin
NuevaMemoria[NConectora].N:=NuevaMemoria[NConectora].N+1;
NuevaMemoria[NConectora].puntos[NuevaMemoria[NConectora].N,1]:=Nuevamemoria[NCon
ectado].Xs;
NuevaMemoria[NConectora].puntos[NuevaMemoria[NConectora].N,2]:=Nuevamemoria[NCon
ectado].Ys;
end
else
begin
NuevaMemoria[NConectora].NB:=NuevaMemoria[NConectora].NB+1;
NuevaMemoria[NConectora].puntosB[NuevaMemoria[NConectora].NB,1]:=Nuevamemoria[N
Conectado].Xs;
NuevaMemoria[NConectora].puntosB[NuevaMemoria[NConectora].NB,2]:=Nuevamemoria[N
Conectado].Ys;
end;
end;

if (conectora^ is Tsimul) then
begin
Nuevamemoria[NConectado].conexionX:=@Nuevasimultaneidad[NConectora];
Nuevamemoria[NConectado].salidaX:=Nuevasimultaneidad[NConectora].salidas;
Nuevasimultaneidad[NConectora].N:=Nuevasimultaneidad[NConectora].N+1;
Nuevasimultaneidad[NConectora].puntos[Nuevasimultaneidad[NConectora].N,1]:=Nuevame
moria[NConectado].Xs;
```

```
Nuevasimultaneidad[NConectora].puntos[Nuevasimultaneidad[NConectora].N,2]:=Nuevame
moria[NConectado].Ys;
end;

if (conectora^ is Tconecc) then
begin
Nuevamemoria[NConectado].conexionX:=@NuevaSelectora[NConectora];
Nuevamemoria[NConectado].salidaX:=NuevaSelectora[NConectora].salidas;
NuevaSelectora[NConectora].N:=NuevaSelectora[NConectora].N+1;
NuevaSelectora[NConectora].puntos[NuevaSelectora[NConectora].N,1]:=Nuevamemoria[NC
onectado].Xs;
NuevaSelectora[NConectora].puntos[NuevaSelectora[NConectora].N,2]:=Nuevamemoria[NC
onectado].Ys;
end;

if  (conectora^ is TVRodillos) then
begin
Nuevamemoria[NConectado].conexionX:=@Nuevarodillo[NConectora];
Nuevamemoria[NConectado].salidaX:=Nuevarodillo[NConectora].salidas;
Nuevarodillo[NConectora].N:=Nuevarodillo[NConectora].N+1;
Nuevarodillo[NConectora].puntos[Nuevarodillo[NConectora].N,1]:=Nuevamemoria[NConecta
do].Xs;
Nuevarodillo[NConectora].puntos[Nuevarodillo[NConectora].N,2]:=Nuevamemoria[NConecta
do].Ys;
end;

if  (conectora^ is TVpulsador) then
begin
Nuevamemoria[NConectado].conexionX:=@Nuevapulsador[NConectora];
Nuevamemoria[NConectado].salidaX:=Nuevapulsador[NConectora].salidas;
Nuevapulsador[NConectora].N:=Nuevapulsador[NConectora].N+1;
Nuevapulsador[NConectora].puntos[Nuevapulsador[NConectora].N,1]:=Nuevamemoria[NCo
nectado].Xs;
Nuevapulsador[NConectora].puntos[Nuevapulsador[NConectora].N,2]:=Nuevamemoria[NCo
nectado].Ys;
end;
end
else
begin
error:=1;// entrada ya conectada
validaError(Error);
nohacerlinea:=true;
end;
end;

// si se activa a una selectora
if (conectado^ is Tconecc)then
begin
if Nuevaselectora[NConectado].conexionX<>nil then
if Nuevaselectora[NConectado].conexionX^=nil then
Nuevaselectora[NConectado].conexionX:=nil;
```

```
if NuevaSelectora[NConectado].conexionX=nil then
begin
if(conectora^ is Tfuente) then
begin
NuevaSelectora[NConectado].conexionX:=@nuevafuente;//conectora;
NuevaSelectora[NConectado].salidaX:=Nuevafuente.salidas;
Nuevafuente.N:=Nuevafuente.N+1;
Nuevafuente.puntos[Nuevafuente.N,1]:=NuevaSelectora[NConectado].Xs;
Nuevafuente.puntos[Nuevafuente.N,2]:=NuevaSelectora[NConectado].Ys;
if Nuevafuente.TipoConexion=Cindirecta then
begin
nohacerlinea:=true;
form2.image1.canvas.Brush.style:=bsSolid;
form2.image1.canvas.Brush.color:=clblue;
form2.image1.Canvas.ellipse(conectado.Xs,conectado.Ys-5,conectado.Xs-
10,conectado.Ys+5);
end;
end;

if (conectora^ is Tmemoria) then
begin
NuevaSelectora[NConectado].conexionX:=@NuevaMemoria[NConectora];
NuevaSelectora[NConectado].salidaX:=NuevaMemoria[NConectora].salidas;
if NuevaMemoria[NConectora].salidas = salidaA then
begin
NuevaMemoria[NConectora].N:=NuevaMemoria[NConectora].N+1;
NuevaMemoria[NConectora].puntos[NuevaMemoria[NConectora].N,1]:=NuevaSelectora[NCo
nectado].Xs;
NuevaMemoria[NConectora].puntos[NuevaMemoria[NConectora].N,2]:=NuevaSelectora[NCo
nectado].Ys;
end
else
begin
NuevaMemoria[NConectora].NB:=NuevaMemoria[NConectora].NB+1;
NuevaMemoria[NConectora].puntosB[NuevaMemoria[NConectora].NB,1]:=NuevaSelectora[N
Conectado].Xs;
NuevaMemoria[NConectora].puntosB[NuevaMemoria[NConectora].NB,2]:=NuevaSelectora[N
Conectado].Ys;
end;

end;

if (conectora^ is Tsimul) then
begin
NuevaSelectora[NConectado].conexionX:=@Nuevasimultaneidad[NConectora];
NuevaSelectora[NConectado].salidaX:=Nuevasimultaneidad[NConectora].salidas;
Nuevasimultaneidad[NConectora].N:=Nuevasimultaneidad[NConectora].N+1;
Nuevasimultaneidad[NConectora].puntos[Nuevasimultaneidad[NConectora].N,1]:=NuevaSel
ectora[NConectado].Xs;
Nuevasimultaneidad[NConectora].puntos[Nuevasimultaneidad[NConectora].N,2]:=NuevaSel
ectora[NConectado].Ys;
```

```
end;

if  (conectora^ is Tconecc) then
begin
NuevaSelectora[NConectado].conexionX:=@NuevaSelectora[NConectora];
NuevaSelectora[NConectado].salidaX:=NuevaSelectora[NConectora].salidas;
NuevaSelectora[NConectora].N:=NuevaSelectora[NConectora].N+1;
NuevaSelectora[NConectora].puntos[NuevaSelectora[NConectora].N,1]:=NuevaSelectora[NC
onectado].Xs;
NuevaSelectora[NConectora].puntos[NuevaSelectora[NConectora].N,2]:=NuevaSelectora[NC
onectado].Ys;
end;

if  (conectora^ is TVRodillos) then
begin
NuevaSelectora[NConectado].conexionX:=@Nuevarodillo[NConectora];
NuevaSelectora[NConectado].salidaX:=Nuevarodillo[NConectora].salidas;
Nuevarodillo[NConectora].N:=Nuevarodillo[NConectora].N+1;
Nuevarodillo[NConectora].puntos[Nuevarodillo[NConectora].N,1]:=NuevaSelectora[NConect
ado].Xs;
Nuevarodillo[NConectora].puntos[Nuevarodillo[NConectora].N,2]:=NuevaSelectora[NConect
ado].Ys;
end;

if  (conectora^ is TVpulsador) then
begin
NuevaSelectora[NConectado].conexionX:=@Nuevapulsador[NConectora];
NuevaSelectora[NConectado].salidaX:=Nuevapulsador[NConectora].salidas;
Nuevapulsador[NConectora].N:=Nuevapulsador[NConectora].N+1;
Nuevapulsador[NConectora].puntos[Nuevapulsador[NConectora].N,1]:=NuevaSelectora[NCo
nectado].Xs;
Nuevapulsador[NConectora].puntos[Nuevapulsador[NConectora].N,2]:=NuevaSelectora[NCo
nectado].Ys;
end;
end
else
begin
error:=1;// entrada ya conectada
validaError(Error);
nohacerlinea:=true;
end;
end;

// si se activa a una simultaneidad
if (conectado^ is Tsimul)then
begin
if Nuevasimultaneidad[NConectado].conexionX<>nil then
if Nuevasimultaneidad[NConectado].conexionX^=nil then
Nuevasimultaneidad[NConectado].conexionX:=nil;
if Nuevasimultaneidad[NConectado].conexionX=nil then
begin
```

```
if (conectora^ is Tfuente) then
begin
Nuevasimultaneidad[NConectado].conexionX:=@nuevafuente;//conectora;
Nuevasimultaneidad[NConectado].salidaX:=Nuevafuente.salidas;
Nuevafuente.N:=Nuevafuente.N+1;
Nuevafuente.puntos[Nuevafuente.N,1]:=Nuevasimultaneidad[NConectado].Xs;
Nuevafuente.puntos[Nuevafuente.N,2]:=Nuevasimultaneidad[NConectado].Ys;
if Nuevafuente.TipoConexion=Cindirecta then
begin
nohacerlinea:=true;
form2.image1.canvas.Brush.style:=bsSolid;
form2.image1.canvas.Brush.color:=clblue;
form2.image1.Canvas.ellipse(conectado.Xs,conectado.Ys-5,conectado.Xs-
10,conectado.Ys+5);
end;
end;

if  (conectora^ is Tmemoria) then
begin
Nuevasimultaneidad[NConectado].conexionX:=@NuevaMemoria[NConectora];
Nuevasimultaneidad[NConectado].salidaX:=NuevaMemoria[NConectora].salidas;
if NuevaMemoria[NConectora].salidas = salidaA then
begin
NuevaMemoria[NConectora].N:=NuevaMemoria[NConectora].N+1;
NuevaMemoria[NConectora].puntos[NuevaMemoria[NConectora].N,1]:=Nuevasimultaneidad[
NConectado].Xs;
NuevaMemoria[NConectora].puntos[NuevaMemoria[NConectora].N,2]:=Nuevasimultaneidad[
NConectado].Ys;
end
else
begin
NuevaMemoria[NConectora].NB:=NuevaMemoria[NConectora].NB+1;
NuevaMemoria[NConectora].puntosB[NuevaMemoria[NConectora].NB,1]:=Nuevasimultaneid
ad[NConectado].Xs;
NuevaMemoria[NConectora].puntosB[NuevaMemoria[NConectora].NB,2]:=Nuevasimultaneid
ad[NConectado].Ys;
end;

end;

if  (conectora^ is Tsimul) then
begin
Nuevasimultaneidad[NConectado].conexionX:=@Nuevasimultaneidad[NConectora];
Nuevasimultaneidad[NConectado].salidaX:=Nuevasimultaneidad[NConectora].salidas;
Nuevasimultaneidad[NConectora].N:=Nuevasimultaneidad[NConectora].N+1;
Nuevasimultaneidad[NConectora].puntos[Nuevasimultaneidad[NConectora].N,1]:=Nuevasim
ultaneidad[NConectado].Xs;
Nuevasimultaneidad[NConectora].puntos[Nuevasimultaneidad[NConectora].N,2]:=Nuevasim
ultaneidad[NConectado].Ys;
end;
```

```
if (conectora^ is Tconecc) then
begin
Nuevasimultaneidad[NConectado].conexionX:=@NuevaSelectora[NConectora];
Nuevasimultaneidad[NConectado].salidaX:=NuevaSelectora[NConectora].salidas;
NuevaSelectora[NConectora].N:=NuevaSelectora[NConectora].N+1;
NuevaSelectora[NConectora].puntos[NuevaSelectora[NConectora].N,1]:=Nuevasimultaneida
d[NConectado].Xs;
NuevaSelectora[NConectora].puntos[NuevaSelectora[NConectora].N,2]:=Nuevasimultaneida
d[NConectado].Ys;
end;

if  (conectora^ is TVRodillos) then
begin
Nuevasimultaneidad[NConectado].conexionX:=@Nuevarodillo[NConectora];
Nuevasimultaneidad[NConectado].salidaX:=Nuevarodillo[NConectora].salidas;
Nuevarodillo[NConectora].N:=Nuevarodillo[NConectora].N+1;
Nuevarodillo[NConectora].puntos[Nuevarodillo[NConectora].N,1]:=Nuevasimultaneidad[NCo
nectado].Xs;
Nuevarodillo[NConectora].puntos[Nuevarodillo[NConectora].N,2]:=Nuevasimultaneidad[NCo
nectado].Ys;
end;

if  (conectora^ is TVpulsador) then
begin
Nuevasimultaneidad[NConectado].conexionX:=@Nuevapulsador[NConectora];
Nuevasimultaneidad[NConectado].salidaX:=Nuevapulsador[NConectora].salidas;
Nuevapulsador[NConectora].N:=Nuevapulsador[NConectora].N+1;
Nuevapulsador[NConectora].puntos[Nuevapulsador[NConectora].N,1]:=Nuevasimultaneidad
[NConectado].Xs;
Nuevapulsador[NConectora].puntos[Nuevapulsador[NConectora].N,2]:=Nuevasimultaneidad
[NConectado].Ys;
end;
end
else
begin
error:=1;// entrada ya conectada
validaError(Error);
nohacerlinea:=true;
end;
end;

// si se activa a un rodillo
if (conectado^ is TVrodillos) then
begin
if NuevaRodillo[NConectado].conexionX<>nil then
if NuevaRodillo[NConectado].conexionX^=nil then
 NuevaRodillo[NConectado].conexionX:=nil;
 if NuevaRodillo[NConectado].conexionX= nil then
  begin
  if (conectora^ is Tcilindro) then
  begin
```

```
 NuevaRodillo[NConectado].conexionX:=@Nuevocilindro[NConectora];
 NuevaRodillo[NConectado].salidaX:=NuevoCilindro[NConectora].salidas;
 if NuevoCilindro[NConectora].salidas = salidaA then
 NuevaRodillo[NConectado].Nombre.caption:=NuevoCilindro[NConectora].val0.caption
 else
 NuevaRodillo[NConectado].Nombre.caption:=NuevoCilindro[NConectora].val1.caption;
 nohacerlinea:=true;
 end;//si conectora es Tcilindro

 if (conectora^ is TCresorte) then
 begin
 NuevaRodillo[NConectado].conexionX:=@NuevoCresorte[NConectora];
 NuevaRodillo[NConectado].salidaX:=NuevoCresorte[NConectora].salidas;
 nohacerlinea:=true;
 if NuevoCilindro[NConectora].salidas = salidaA then
 NuevaRodillo[NConectado].Nombre.caption:=NuevoCilindro[NConectora].val0.caption
 else
 NuevaRodillo[NConectado].Nombre.caption:=NuevoCilindro[NConectora].val1.caption;
 nohacerlinea:=true;
 end;//si es Tcresorte
end
else
begin
error:=1;// entrada ya conectada
validaError(Error);
nohacerlinea:=true;
end;
end; //si es Trodillo

end;// si la entrada es 1


if conectado^.entrada = 2 then
begin
// si se activa a un cilindro
if (conectado^ is Tcilindro) then
begin
if NuevoCilindro[NConectado].conexionY<>nil then
if NuevoCilindro[NConectado].conexionY^=nil then
NuevoCilindro[NConectado].conexionY:=nil;
if NuevoCilindro[NConectado].conexionY =nil then
begin
if (conectora^ is Tfuente) then
begin
NuevoCilindro[NConectado].conexionY:=@nuevafuente;//conectora;
NuevoCilindro[NConectado].salidaY:=Nuevafuente.salidas;
Nuevafuente.N:=Nuevafuente.N+1;
Nuevafuente.puntos[Nuevafuente.N,1]:=NuevoCilindro[NConectado].Xs;
Nuevafuente.puntos[Nuevafuente.N,2]:=NuevoCilindro[NConectado].Ys;

if Nuevafuente.TipoConexion=Cindirecta then
```

```
begin
nohacerlinea:=true;
form2.image1.canvas.Brush.style:=bsSolid;
form2.image1.canvas.Brush.color:=clblue;
form2.image1.Canvas.ellipse(conectado.Xs-
5,conectado.Ys,conectado.Xs+5,conectado.Ys+10);
end;
end;

if  (conectora^ is Tmemoria) then
begin
NuevoCilindro[NConectado].conexionY:=@NuevaMemoria[NConectora];
NuevoCilindro[NConectado].salidaY:=NuevaMemoria[NConectora].salidas;
if NuevaMemoria[NConectora].salidas = salidaA then
begin
NuevaMemoria[NConectora].N:=NuevaMemoria[NConectora].N+1;
NuevaMemoria[NConectora].puntos[NuevaMemoria[NConectora].N,1]:=NuevoCilindro[NCon
ectado].Xs;
NuevaMemoria[NConectora].puntos[NuevaMemoria[NConectora].N,2]:=NuevoCilindro[NCon
ectado].Ys;
end
else
begin
NuevaMemoria[NConectora].NB:=NuevaMemoria[NConectora].NB+1;
NuevaMemoria[NConectora].puntosB[NuevaMemoria[NConectora].NB,1]:=NuevoCilindro[NC
onectado].Xs;
NuevaMemoria[NConectora].puntosB[NuevaMemoria[NConectora].NB,2]:=NuevoCilindro[NC
onectado].Ys;
end;

end;

if  (conectora^ is Tsimul) then
begin
NuevoCilindro[NConectado].conexionY:=@Nuevasimultaneidad[NConectora];
NuevoCilindro[NConectado].salidaY:=Nuevasimultaneidad[NConectora].salidas;
Nuevasimultaneidad[NConectora].N:=Nuevasimultaneidad[NConectora].N+1;
Nuevasimultaneidad[NConectora].puntos[Nuevasimultaneidad[NConectora].N,1]:=NuevoCili
ndro[NConectado].Xs;
Nuevasimultaneidad[NConectora].puntos[Nuevasimultaneidad[NConectora].N,2]:=NuevoCili
ndro[NConectado].Ys;
end;

if (conectora^ is Tconecc) then
begin
NuevoCilindro[NConectado].conexionY:=@NuevaSelectora[NConectora];
NuevoCilindro[NConectado].salidaY:=NuevaSelectora[NConectora].salidas;
NuevaSelectora[NConectora].N:=NuevaSelectora[NConectora].N+1;
NuevaSelectora[NConectora].puntos[NuevaSelectora[NConectora].N,1]:=NuevoCilindro[NCo
nectado].Xs;
```

```
NuevaSelectora[NConectora].puntos[NuevaSelectora[NConectora].N,2]:=NuevoCilindro[NCo
nectado].Ys;
end;

if (conectora^ is TVRodillos) then
begin
NuevoCilindro[NConectado].conexionY:=@Nuevarodillo[NConectora];
NuevoCilindro[NConectado].salidaY:=Nuevarodillo[NConectora].salidas;
Nuevarodillo[NConectora].N:=Nuevarodillo[NConectora].N+1;
Nuevarodillo[NConectora].puntos[Nuevarodillo[NConectora].N,1]:=NuevoCilindro[NConecta
do].Xs;
Nuevarodillo[NConectora].puntos[Nuevarodillo[NConectora].N,2]:=NuevoCilindro[NConecta
do].Ys;
end;

if  (conectora^ is Tvpulsador) then
begin
NuevoCilindro[NConectado].conexionY:=@Nuevapulsador[NConectora];
NuevoCilindro[NConectado].salidaY:=Nuevapulsador[NConectora].salidas;
Nuevapulsador[NConectora].N:=Nuevapulsador[NConectora].N+1;
Nuevapulsador[NConectora].puntos[Nuevapulsador[NConectora].N,1]:=NuevoCilindro[NCon
ectado].Xs;
Nuevapulsador[NConectora].puntos[Nuevapulsador[NConectora].N,2]:=NuevoCilindro[NCon
ectado].Ys;
end;
end
else
begin
error:=1;// entrada ya conectada
validaError(Error);
nohacerlinea:=true;
end;
end;

// si se activa a un memoria
if (conectado^ is Tmemoria) then
begin
if Nuevamemoria[NConectado].conexionY<>nil then
if Nuevamemoria[NConectado].conexionY^=nil then
Nuevamemoria[NConectado].conexionY:=nil;
if Nuevamemoria[NConectado].conexionY=nil then
begin

if (conectora^ is Tfuente) then
begin
Nuevamemoria[NConectado].conexionY:=@nuevafuente;//conectora;
Nuevamemoria[NConectado].salidaY:=Nuevafuente.salidas;
Nuevafuente.N:=Nuevafuente.N+1;
Nuevafuente.puntos[Nuevafuente.N,1]:=Nuevamemoria[NConectado].Xs;
Nuevafuente.puntos[Nuevafuente.N,2]:=Nuevamemoria[NConectado].Ys;
if Nuevafuente.TipoConexion=Cindirecta then
```

```
begin
nohacerlinea:=true;
form2.image1.canvas.Brush.style:=bsSolid;
form2.image1.canvas.Brush.color:=clblue;
form2.image1.Canvas.ellipse(conectado.Xs,conectado.Ys-
5,conectado.Xs+10,conectado.Ys+5);
end;
end;

if  (conectora^ is Tmemoria) then
begin
Nuevamemoria[NConectado].conexionY:=@NuevaMemoria[NConectora];
Nuevamemoria[NConectado].salidaY:=NuevaMemoria[NConectora].salidas;
if NuevaMemoria[NConectora].salidas = salidaA then
begin
NuevaMemoria[NConectora].N:=NuevaMemoria[NConectora].N+1;
NuevaMemoria[NConectora].puntos[NuevaMemoria[NConectora].N,1]:=Nuevamemoria[NCon
ectado].Xs;
NuevaMemoria[NConectora].puntos[NuevaMemoria[NConectora].N,2]:=Nuevamemoria[NCon
ectado].Ys;
end
else
begin
NuevaMemoria[NConectora].NB:=NuevaMemoria[NConectora].NB+1;
NuevaMemoria[NConectora].puntosB[NuevaMemoria[NConectora].NB,1]:=Nuevamemoria[N
Conectado].Xs;
NuevaMemoria[NConectora].puntosB[NuevaMemoria[NConectora].NB,2]:=Nuevamemoria[N
Conectado].Ys;
end;
end;

if  (conectora^ is Tsimul) then
begin
Nuevamemoria[NConectado].conexionY:=@Nuevasimultaneidad[NConectora];
Nuevamemoria[NConectado].salidaY:=Nuevasimultaneidad[NConectora].salidas;
Nuevasimultaneidad[NConectora].N:=Nuevasimultaneidad[NConectora].N+1;
Nuevasimultaneidad[NConectora].puntos[Nuevasimultaneidad[NConectora].N,1]:=Nuevame
moria[NConectado].Xs;
Nuevasimultaneidad[NConectora].puntos[Nuevasimultaneidad[NConectora].N,2]:=Nuevame
moria[NConectado].Ys;
end;

if  (conectora^ is Tconecc) then
begin
Nuevamemoria[NConectado].conexionY:=@NuevaSelectora[NConectora];
Nuevamemoria[NConectado].salidaY:=NuevaSelectora[NConectora].salidas;
NuevaSelectora[NConectora].N:=NuevaSelectora[NConectora].N+1;
NuevaSelectora[NConectora].puntos[NuevaSelectora[NConectora].N,1]:=Nuevamemoria[NC
onectado].Xs;
NuevaSelectora[NConectora].puntos[NuevaSelectora[NConectora].N,2]:=Nuevamemoria[NC
onectado].Ys;
```

```
end;

if  (conectora^ is TVRodillos) then
begin
Nuevamemoria[NConectado].conexionY:=@Nuevarodillo[NConectora];
Nuevamemoria[NConectado].salidaY:=Nuevarodillo[NConectora].salidas;
Nuevarodillo[NConectora].N:=Nuevarodillo[NConectora].N+1;
Nuevarodillo[NConectora].puntos[Nuevarodillo[NConectora].N,1]:=Nuevamemoria[NConecta
do].Xs;
Nuevarodillo[NConectora].puntos[Nuevarodillo[NConectora].N,2]:=Nuevamemoria[NConecta
do].Ys;
end;

if  (conectora^ is TVpulsador) then
begin
Nuevamemoria[NConectado].conexionY:=@Nuevapulsador[NConectora];
Nuevamemoria[NConectado].salidaY:=Nuevapulsador[NConectora].salidas;
Nuevapulsador[NConectora].N:=Nuevapulsador[NConectora].N+1;
Nuevapulsador[NConectora].puntos[Nuevapulsador[NConectora].N,1]:=Nuevamemoria[NCo
nectado].Xs;
Nuevapulsador[NConectora].puntos[Nuevapulsador[NConectora].N,2]:=Nuevamemoria[NCo
nectado].Ys;
end;
end
else
begin
error:=1;// entrada ya conectada
validaError(Error);
nohacerlinea:=true;
end;
end;

// si se activa a una selectora
if (conectado^ is Tconecc) then
begin
if Nuevaselectora[NConectado].conexionY<>nil then
if Nuevaselectora[NConectado].conexionY^=nil then
Nuevaselectora[NConectado].conexionY:=nil;
if NuevaSelectora[NConectado].conexionY=nil then
begin
if (conectora^ is Tfuente) then
begin
NuevaSelectora[NConectado].conexionY:=@nuevafuente;//conectora;
NuevaSelectora[NConectado].salidaY:=Nuevafuente.salidas;
Nuevafuente.N:=Nuevafuente.N+1;
Nuevafuente.puntos[Nuevafuente.N,1]:=NuevaSelectora[NConectado].Xs;
Nuevafuente.puntos[Nuevafuente.N,2]:=NuevaSelectora[NConectado].Ys;

if Nuevafuente.TipoConexion=Cindirecta then
begin
nohacerlinea:=true;
```

```
form2.image1.canvas.Brush.style:=bsSolid;
form2.image1.canvas.Brush.color:=clblue;
form2.image1.Canvas.ellipse(conectado.Xs,conectado.Ys-
5,conectado.Xs+10,conectado.Ys+5);
end;
end;

if  (conectora^ is Tmemoria) then
begin
NuevaSelectora[NConectado].conexionY:=@NuevaMemoria[NConectora];
NuevaSelectora[NConectado].salidaY:=NuevaMemoria[NConectora].salidas;
if NuevaMemoria[NConectora].salidas = salidaA then
begin
NuevaMemoria[NConectora].N:=NuevaMemoria[NConectora].N+1;
NuevaMemoria[NConectora].puntos[NuevaMemoria[NConectora].N,1]:=NuevaSelectora[NCo
nectado].Xs;
NuevaMemoria[NConectora].puntos[NuevaMemoria[NConectora].N,2]:=NuevaSelectora[NCo
nectado].Ys;
end
else
begin
NuevaMemoria[NConectora].NB:=NuevaMemoria[NConectora].NB+1;
NuevaMemoria[NConectora].puntosB[NuevaMemoria[NConectora].NB,1]:=NuevaSelectora[N
Conectado].Xs;
NuevaMemoria[NConectora].puntosB[NuevaMemoria[NConectora].NB,2]:=NuevaSelectora[N
Conectado].Ys;
end;

end;

if (conectora^ is Tsimul) then
begin
NuevaSelectora[NConectado].conexionY:=@Nuevasimultaneidad[NConectora];
NuevaSelectora[NConectado].salidaY:=Nuevasimultaneidad[NConectora].salidas;
Nuevasimultaneidad[NConectora].N:=Nuevasimultaneidad[NConectora].N+1;
Nuevasimultaneidad[NConectora].puntos[Nuevasimultaneidad[NConectora].N,1]:=NuevaSel
ectora[NConectado].Xs;
Nuevasimultaneidad[NConectora].puntos[Nuevasimultaneidad[NConectora].N,2]:=NuevaSel
ectora[NConectado].Ys;
end;

if  (conectora^ is Tconecc) then
begin
NuevaSelectora[NConectado].conexionY:=@NuevaSelectora[NConectora];
NuevaSelectora[NConectado].salidaY:=NuevaSelectora[NConectora].salidas;
NuevaSelectora[NConectora].N:=NuevaSelectora[NConectora].N+1;
NuevaSelectora[NConectora].puntos[NuevaSelectora[NConectora].N,1]:=NuevaSelectora[NC
onectado].Xs;
NuevaSelectora[NConectora].puntos[NuevaSelectora[NConectora].N,2]:=NuevaSelectora[NC
onectado].Ys;
end;
```

```
if  (conectora^ is TVRodillos) then
begin
NuevaSelectora[NConectado].conexionY:=@Nuevarodillo[NConectora];
NuevaSelectora[NConectado].salidaY:=Nuevarodillo[NConectora].salidas;
Nuevarodillo[NConectora].N:=Nuevarodillo[NConectora].N+1;
Nuevarodillo[NConectora].puntos[Nuevarodillo[NConectora].N,1]:=NuevaSelectora[NConect
ado].Xs;
Nuevarodillo[NConectora].puntos[Nuevarodillo[NConectora].N,2]:=NuevaSelectora[NConect
ado].Ys;
end;

if  (conectora^ is TVpulsador) then
begin
NuevaSelectora[NConectado].conexionY:=@Nuevapulsador[NConectora];
NuevaSelectora[NConectado].salidaY:=Nuevapulsador[NConectora].salidas;
Nuevapulsador[NConectora].N:=Nuevapulsador[NConectora].N+1;
Nuevapulsador[NConectora].puntos[Nuevapulsador[NConectora].N,1]:=NuevaSelectora[NCo
nectado].Xs;
Nuevapulsador[NConectora].puntos[Nuevapulsador[NConectora].N,2]:=NuevaSelectora[NCo
nectado].Ys;
end;
end
else
begin
error:=1;// entrada ya conectada
validaError(Error);
nohacerlinea:=true;
end;
end;

// si se activa a una simultaneidad
if (conectado^ is Tsimul) then
begin
if Nuevasimultaneidad[NConectado].conexionY<>nil then
if Nuevasimultaneidad[NConectado].conexionY^=nil then
Nuevasimultaneidad[NConectado].conexionY:=nil;
if Nuevasimultaneidad[NConectado].conexionY=nil then
begin
if (conectora^ is Tfuente) then
begin
Nuevasimultaneidad[NConectado].conexionY:=@nuevafuente;//conectora;
Nuevasimultaneidad[NConectado].salidaY:=Nuevafuente.salidas;
Nuevafuente.N:=Nuevafuente.N+1;
Nuevafuente.puntos[Nuevafuente.N,1]:=Nuevasimultaneidad[NConectado].Xs;
Nuevafuente.puntos[Nuevafuente.N,2]:=Nuevasimultaneidad[NConectado].Ys;

if Nuevafuente.TipoConexion=Cindirecta then
begin
nohacerlinea:=true;
form2.image1.canvas.Brush.style:=bsSolid;
```

```
form2.image1.canvas.Brush.color:=clblue;
form2.image1.Canvas.ellipse(conectado.Xs,conectado.Ys-
5,conectado.Xs+10,conectado.Ys+5);
end;
end;

if (conectora^ is Tmemoria) then
begin
Nuevasimultaneidad[NConectado].conexionY:=@NuevaMemoria[NConectora];
Nuevasimultaneidad[NConectado].salidaY:=NuevaMemoria[NConectora].salidas;
if NuevaMemoria[NConectora].salidas = salidaA then
begin
NuevaMemoria[NConectora].N:=NuevaMemoria[NConectora].N+1;
NuevaMemoria[NConectora].puntos[NuevaMemoria[NConectora].N,1]:=Nuevasimultaneidad[
NConectado].Xs;
NuevaMemoria[NConectora].puntos[NuevaMemoria[NConectora].N,2]:=Nuevasimultaneidad[
NConectado].Ys;
end
else
begin
NuevaMemoria[NConectora].NB:=NuevaMemoria[NConectora].NB+1;
NuevaMemoria[NConectora].puntosB[NuevaMemoria[NConectora].NB,1]:=Nuevasimultaneid
ad[NConectado].Xs;
NuevaMemoria[NConectora].puntosB[NuevaMemoria[NConectora].NB,2]:=Nuevasimultaneid
ad[NConectado].Ys;
end;

end;

if(conectora^ is Tsimul) then
begin
Nuevasimultaneidad[NConectado].conexionY:=@Nuevasimultaneidad[NConectora];
Nuevasimultaneidad[NConectado].salidaY:=Nuevasimultaneidad[NConectora].salidas;
Nuevasimultaneidad[NConectora].N:=Nuevasimultaneidad[NConectora].N+1;
Nuevasimultaneidad[NConectora].puntos[Nuevasimultaneidad[NConectora].N,1]:=Nuevasim
ultaneidad[NConectado].Xs;
Nuevasimultaneidad[NConectora].puntos[Nuevasimultaneidad[NConectora].N,2]:=Nuevasim
ultaneidad[NConectado].Ys;
end;

if (conectora^ is Tconecc) then
begin
Nuevasimultaneidad[NConectado].conexionY:=@NuevaSelectora[NConectora];
Nuevasimultaneidad[NConectado].salidaY:=NuevaSelectora[NConectora].salidas;
NuevaSelectora[NConectora].N:=NuevaSelectora[NConectora].N+1;
NuevaSelectora[NConectora].puntos[NuevaSelectora[NConectora].N,1]:=Nuevasimultaneida
d[NConectado].Xs;
NuevaSelectora[NConectora].puntos[NuevaSelectora[NConectora].N,2]:=Nuevasimultaneida
d[NConectado].Ys;
end;
```

```
if  (conectora^ is TVRodillos) then
begin
Nuevasimultaneidad[NConectado].conexionY:=@Nuevarodillo[NConectora];
Nuevasimultaneidad[NConectado].salidaY:=Nuevarodillo[NConectora].salidas;
Nuevarodillo[NConectora].N:=Nuevarodillo[NConectora].N+1;
Nuevarodillo[NConectora].puntos[Nuevarodillo[NConectora].N,1]:=Nuevasimultaneidad[NCo
nectado].Xs;
Nuevarodillo[NConectora].puntos[Nuevarodillo[NConectora].N,2]:=Nuevasimultaneidad[NCo
nectado].Ys;
end;

if  (conectora^ is TVpulsador) then
begin
Nuevasimultaneidad[NConectado].conexionY:=@Nuevapulsador[NConectora];
Nuevasimultaneidad[NConectado].salidaY:=Nuevapulsador[NConectora].salidas;
Nuevapulsador[NConectora].N:=Nuevapulsador[NConectora].N+1;
Nuevapulsador[NConectora].puntos[Nuevapulsador[NConectora].N,1]:=Nuevasimultaneidad
[NConectado].Xs;
Nuevapulsador[NConectora].puntos[Nuevapulsador[NConectora].N,2]:=Nuevasimultaneidad
[NConectado].Ys;
end;
end
else
begin
error:=1;// entrada ya conectada
validaError(Error);
nohacerlinea:=true;
end;
end;

end;//  fin si la entrada es 2


// si  la entrada es 3
if conectado^.entrada = 3 then
begin
// si se activa a un memoria
if (conectado^ is Tmemoria) then
begin
if Nuevamemoria[NConectado].conexionP<>nil then
if Nuevamemoria[NConectado].conexionP^=nil then
Nuevamemoria[NConectado].conexionP:=nil;
if Nuevamemoria[NConectado].conexionP=nil then
begin
if (conectora^ is Tfuente) then
begin
Nuevamemoria[NConectado].conexionP:=@nuevafuente;//conectora;
Nuevamemoria[NConectado].salidaP:=Nuevafuente.salidas;
Nuevafuente.N:=Nuevafuente.N+1;
Nuevafuente.puntos[Nuevafuente.N,1]:=Nuevamemoria[NConectado].Xs;
Nuevafuente.puntos[Nuevafuente.N,2]:=Nuevamemoria[NConectado].Ys;
```

```
if Nuevafuente.TipoConexion=Cindirecta then
begin
nohacerlinea:=true;
form2.image1.canvas.Brush.style:=bsSolid;
form2.image1.canvas.Brush.color:=clblue;
form2.image1.Canvas.ellipse(conectado.Xs-
5,conectado.Ys,conectado.Xs+5,conectado.Ys+10);
end;
end;

if(conectora^ is Tmemoria) then
begin
Nuevamemoria[NConectado].conexionP:=@NuevaMemoria[NConectora];
Nuevamemoria[NConectado].salidaP:=NuevaMemoria[NConectora].salidas;
if NuevaMemoria[NConectora].salidas = salidaA then
begin
NuevaMemoria[NConectora].N:=NuevaMemoria[NConectora].N+1;
NuevaMemoria[NConectora].puntos[NuevaMemoria[NConectora].N,1]:=Nuevamemoria[NCon
ectado].Xs;
NuevaMemoria[NConectora].puntos[NuevaMemoria[NConectora].N,2]:=Nuevamemoria[NCon
ectado].Ys;
end
else
begin
NuevaMemoria[NConectora].NB:=NuevaMemoria[NConectora].NB+1;
NuevaMemoria[NConectora].puntosB[NuevaMemoria[NConectora].NB,1]:=Nuevamemoria[N
Conectado].Xs;
NuevaMemoria[NConectora].puntosB[NuevaMemoria[NConectora].NB,2]:=Nuevamemoria[N
Conectado].Ys;
end;
end;

if (conectora^ is Tsimul) then
begin
Nuevamemoria[NConectado].conexionP:=@Nuevasimultaneidad[NConectora];
Nuevamemoria[NConectado].salidaP:=Nuevasimultaneidad[NConectora].salidas;
Nuevasimultaneidad[NConectora].N:=Nuevasimultaneidad[NConectora].N+1;
Nuevasimultaneidad[NConectora].puntos[Nuevasimultaneidad[NConectora].N,1]:=Nuevame
moria[NConectado].Xs;
Nuevasimultaneidad[NConectora].puntos[Nuevasimultaneidad[NConectora].N,2]:=Nuevame
moria[NConectado].Ys;
end;

if(conectora^ is Tconecc) then
begin
Nuevamemoria[NConectado].conexionP:=@NuevaSelectora[NConectora];
Nuevamemoria[NConectado].salidaP:=NuevaSelectora[NConectora].salidas;
NuevaSelectora[NConectora].N:=NuevaSelectora[NConectora].N+1;
NuevaSelectora[NConectora].puntos[NuevaSelectora[NConectora].N,1]:=Nuevamemoria[NC
onectado].Xs;
```

```
NuevaSelectora[NConectora].puntos[NuevaSelectora[NConectora].N,2]:=Nuevamemoria[NC
onectado].Ys;
end;

if  (conectora^ is TVRodillos) then
begin
Nuevamemoria[NConectado].conexionP:=@Nuevarodillo[NConectora];
Nuevamemoria[NConectado].salidaP:=Nuevarodillo[NConectora].salidas;
Nuevarodillo[NConectora].N:=Nuevarodillo[NConectora].N+1;
Nuevarodillo[NConectora].puntos[Nuevarodillo[NConectora].N,1]:=Nuevamemoria[NConecta
do].Xs;
Nuevarodillo[NConectora].puntos[Nuevarodillo[NConectora].N,2]:=Nuevamemoria[NConecta
do].Ys;
end;

if  (conectora^ is TVpulsador) then
begin
Nuevamemoria[NConectado].conexionP:=@Nuevapulsador[NConectora];
Nuevamemoria[NConectado].salidaP:=Nuevapulsador[NConectora].salidas;
Nuevapulsador[NConectora].N:=Nuevapulsador[NConectora].N+1;
Nuevapulsador[NConectora].puntos[Nuevapulsador[NConectora].N,1]:=Nuevamemoria[NCo
nectado].Xs;
Nuevapulsador[NConectora].puntos[Nuevapulsador[NConectora].N,2]:=Nuevamemoria[NCo
nectado].Ys;
end;
end
else
begin
error:=1;// entrada ya conectada
validaError(Error);
nohacerlinea:=true;
end;
end;

// si se activa a un rodillo
if (conectado^ is TVrodillos) then
begin
if Nuevarodillo[NConectado].conexionP<>nil then
if Nuevarodillo[NConectado].conexionP^=nil then
Nuevarodillo[NConectado].conexionP:=nil;
if NuevaRodillo[NConectado].conexionP =nil then
begin
if (conectora^ is Tfuente) then
begin
NuevaRodillo[NConectado].conexionP:=@nuevafuente;//conectora;
NuevaRodillo[NConectado].salidaP:=Nuevafuente.salidas;
Nuevafuente.N:=Nuevafuente.N+1;
Nuevafuente.puntos[Nuevafuente.N,1]:=NuevaRodillo[NConectado].Xs;
Nuevafuente.puntos[Nuevafuente.N,2]:=NuevaRodillo[NConectado].Ys;
if Nuevafuente.TipoConexion=Cindirecta then
begin
```

```
nohacerlinea:=true;
form2.image1.canvas.Brush.style:=bsSolid;
form2.image1.canvas.Brush.color:=clblue;
form2.image1.Canvas.ellipse(conectado.Xs-
5,conectado.Ys,conectado.Xs+5,conectado.Ys+10);
end;
end;

if  (conectora^ is Tmemoria) then
begin
NuevaRodillo[NConectado].conexionP:=@NuevaMemoria[NConectora];
NuevaRodillo[NConectado].salidaP:=NuevaMemoria[NConectora].salidas;
if NuevaMemoria[NConectora].salidas = salidaA then
begin
NuevaMemoria[NConectora].N:=NuevaMemoria[NConectora].N+1;
NuevaMemoria[NConectora].puntos[NuevaMemoria[NConectora].N,1]:=NuevaRodillo[NCone
ctado].Xs;
NuevaMemoria[NConectora].puntos[NuevaMemoria[NConectora].N,2]:=NuevaRodillo[NCone
ctado].Ys;
end
else
begin
NuevaMemoria[NConectora].NB:=NuevaMemoria[NConectora].NB+1;
NuevaMemoria[NConectora].puntosB[NuevaMemoria[NConectora].NB,1]:=NuevaRodillo[NCo
nectado].Xs;
NuevaMemoria[NConectora].puntosB[NuevaMemoria[NConectora].NB,2]:=NuevaRodillo[NCo
nectado].Ys;
end;
end;

if (conectora^ is Tsimul) then
begin
NuevaRodillo[NConectado].conexionP:=@Nuevasimultaneidad[NConectora];
NuevaRodillo[NConectado].salidaP:=Nuevasimultaneidad[NConectora].salidas;
Nuevasimultaneidad[NConectora].N:=Nuevasimultaneidad[NConectora].N+1;
Nuevasimultaneidad[NConectora].puntos[Nuevasimultaneidad[NConectora].N,1]:=NuevaRod
illo[NConectado].Xs;
Nuevasimultaneidad[NConectora].puntos[Nuevasimultaneidad[NConectora].N,2]:=NuevaRod
illo[NConectado].Ys;
end;

if  (conectora^ is Tconecc) then
begin
NuevaRodillo[NConectado].conexionP:=@NuevaSelectora[NConectora];
NuevaRodillo[NConectado].salidaP:=NuevaSelectora[NConectora].salidas;
NuevaSelectora[NConectora].N:=NuevaSelectora[NConectora].N+1;
NuevaSelectora[NConectora].puntos[NuevaSelectora[NConectora].N,1]:=NuevaRodillo[NCon
ectado].Xs;
NuevaSelectora[NConectora].puntos[NuevaSelectora[NConectora].N,2]:=NuevaRodillo[NCon
ectado].Ys;
end;
```

```
if  (conectora^ is TVRodillos) then
begin
NuevaRodillo[NConectado].conexionP:=@Nuevarodillo[NConectora];
NuevaRodillo[NConectado].salidaP:=Nuevarodillo[NConectora].salidas;
Nuevarodillo[NConectora].N:=Nuevarodillo[NConectora].N+1;
Nuevarodillo[NConectora].puntos[Nuevarodillo[NConectora].N,1]:=Nuevarodillo[NConectado
].Xs;
Nuevarodillo[NConectora].puntos[Nuevarodillo[NConectora].N,2]:=Nuevarodillo[NConectado
].Ys;
end;

if  (conectora^ is TVpulsador) then
begin
NuevaRodillo[NConectado].conexionP:=@Nuevapulsador[NConectora];
NuevaRodillo[NConectado].salidaP:=Nuevapulsador[NConectora].salidas;
Nuevapulsador[NConectora].N:=Nuevapulsador[NConectora].N+1;
Nuevapulsador[NConectora].puntos[Nuevapulsador[NConectora].N,1]:=NuevaRodillo[NCone
ctado].Xs;
Nuevapulsador[NConectora].puntos[Nuevapulsador[NConectora].N,2]:=NuevaRodillo[NCone
ctado].Ys;
end;
end
else
begin
error:=1;// entrada ya conectada
validaError(Error);
nohacerlinea:=true;
end;
end;

// si se activa a un pulsador
if   (conectado^ is TVpulsador) then
begin
if Nuevapulsador[NConectado].conexionP<>nil then
if Nuevapulsador[NConectado].conexionP^=nil then
Nuevapulsador[NConectado].conexionP:=nil;
if Nuevapulsador[NConectado].conexionP=nil then
begin
if  (conectora^ is Tfuente) then
begin
Nuevapulsador[NConectado].conexionP:=@nuevafuente;//conectora;
Nuevapulsador[NConectado].salidaP:=Nuevafuente.salidas;
Nuevafuente.N:=Nuevafuente.N+1;
Nuevafuente.puntos[Nuevafuente.N,1]:=Nuevapulsador[NConectado].Xs;
Nuevafuente.puntos[Nuevafuente.N,2]:=Nuevapulsador[NConectado].Ys;
if Nuevafuente.TipoConexion=Cindirecta then
begin
nohacerlinea:=true;
form2.image1.canvas.Brush.style:=bsSolid;
form2.image1.canvas.Brush.color:=clblue;
```

```
form2.image1.Canvas.ellipse(conectado.Xs-
5,conectado.Ys,conectado.Xs+5,conectado.Ys+10);
end;
end;

if  (conectora^ is Tmemoria) then
begin
Nuevapulsador[NConectado].conexionP:=@NuevaMemoria[NConectora];
Nuevapulsador[NConectado].salidaP:=NuevaMemoria[NConectora].salidas;
if NuevaMemoria[NConectora].salidas = salidaA then
begin
NuevaMemoria[NConectora].N:=NuevaMemoria[NConectora].N+1;
NuevaMemoria[NConectora].puntos[NuevaMemoria[NConectora].N,1]:=Nuevapulsador[NCon
ectado].Xs;
NuevaMemoria[NConectora].puntos[NuevaMemoria[NConectora].N,2]:=Nuevapulsador[NCon
ectado].Ys;
end
else
begin
NuevaMemoria[NConectora].NB:=NuevaMemoria[NConectora].NB+1;
NuevaMemoria[NConectora].puntosB[NuevaMemoria[NConectora].NB,1]:=Nuevapulsador[N
Conectado].Xs;
NuevaMemoria[NConectora].puntosB[NuevaMemoria[NConectora].NB,2]:=Nuevapulsador[N
Conectado].Ys;
end;
end;

if  (conectora^ is Tsimul) then
begin
Nuevapulsador[NConectado].conexionP:=@Nuevasimultaneidad[NConectora];
Nuevapulsador[NConectado].salidaP:=Nuevasimultaneidad[NConectora].salidas;
Nuevasimultaneidad[NConectora].N:=Nuevasimultaneidad[NConectora].N+1;
Nuevasimultaneidad[NConectora].puntos[Nuevasimultaneidad[NConectora].N,1]:=Nuevapul
sador[NConectado].Xs;
Nuevasimultaneidad[NConectora].puntos[Nuevasimultaneidad[NConectora].N,2]:=Nuevapul
sador[NConectado].Ys;
end;

if  (conectora^ is Tconecc) then
begin
Nuevapulsador[NConectado].conexionP:=@NuevaSelectora[NConectora];
Nuevapulsador[NConectado].salidaP:=NuevaSelectora[NConectora].salidas;
NuevaSelectora[NConectora].N:=NuevaSelectora[NConectora].N+1;
NuevaSelectora[NConectora].puntos[NuevaSelectora[NConectora].N,1]:=Nuevapulsador[NC
onectado].Xs;
NuevaSelectora[NConectora].puntos[NuevaSelectora[NConectora].N,2]:=Nuevapulsador[NC
onectado].Ys;
end;

if  (conectora^ is TVRodillos) then
begin
```

```
Nuevapulsador[NConectado].conexionP:=@Nuevarodillo[NConectora];
Nuevapulsador[NConectado].salidaP:=Nuevarodillo[NConectora].salidas;
Nuevarodillo[NConectora].N:=Nuevarodillo[NConectora].N+1;
Nuevarodillo[NConectora].puntos[Nuevarodillo[NConectora].N,1]:=Nuevapulsador[NConectado].Xs;
Nuevarodillo[NConectora].puntos[Nuevarodillo[NConectora].N,2]:=Nuevapulsador[NConectado].Ys;
end;

if  (conectora^ is TVpulsador) then
begin
Nuevapulsador[NConectado].conexionP:=@Nuevapulsador[NConectora];
Nuevapulsador[NConectado].salidaP:=Nuevapulsador[NConectora].salidas;
Nuevapulsador[NConectora].N:=Nuevapulsador[NConectora].N+1;
Nuevapulsador[NConectora].puntos[Nuevapulsador[NConectora].N,1]:=Nuevapulsador[NConectado].Xs;
Nuevapulsador[NConectora].puntos[Nuevapulsador[NConectora].N,2]:=Nuevapulsador[NConectado].Ys;
end;
end
else
begin
error:=1;// entrada ya conectada
validaError(Error);
nohacerlinea:=true;
end;
end;
end;// fin si la entrada es 3

if (conectora^ is Tfuente) then
begin
  if Nuevafuente <> nil then
  for s:=1 to Nuevafuente.N do  begin
   if  Nuevafuente.Ye<Nuevafuente.puntos[s,2] then
    if Nuevafuente.Xe<nuevafuente.puntos[s,1] then
    begin
    form2.image1.canvas.moveto(nuevafuente.Xe,nuevafuente.Ye);
    form2.image1.canvas.lineto(nuevafuente.Xe,nuevafuente.puntos[s,2]);
    form2.image1.canvas.moveto(nuevafuente.xe,nuevafuente.puntos[s,2]);
    form2.image1.canvas.lineto(nuevafuente.puntos[s,1],nuevafuente.puntos[s,2]);
   end//del if de Xe mayor
 else
   begin
   Form2.image1.canvas.moveto(Nuevafuente.Xe,Nuevafuente.Ye);
   Form2.image1.canvas.lineto(Nuevafuente.Xe,Nuevafuente.puntos[s,2]);
   form2.image1.canvas.moveto(Nuevafuente.Xe,Nuevafuente.puntos[s,2]);
   form2.image1.canvas.lineto(Nuevafuente.puntos[s,1],Nuevafuente.puntos[s,2]);
   end//
else
begin
  form2.image1.canvas.moveto(Nuevafuente.Xe,Nuevafuente.Ye);
```

```
  form2.image1.canvas.lineto(Nuevafuente.puntos[s,1],Nuevafuente.Ye);
  form2.image1.canvas.moveto(Nuevafuente.puntos[s,1],Nuevafuente.Ye);
  form2.image1.canvas.lineto(Nuevafuente.puntos[s,1],Nuevafuente.puntos[s,2]);
end;
  end;
nohacerlinea:=true;
end;

if not(nohacerlinea) then
begin
hacerlinea(conectora.Xe,conectora.Ye,conectado.Xs,conectado.Ys);
end;
conectado:=nil;
conectora:=nil;
nohacerlinea:=false;
end;
limpiaconexion;
end;// del with
end;//fin del click
```

VERIFICAR CONEXIÓN

Verifica que todas las entradas de cada elemento estén conectadas, si esto se cumple se puede ejecutar el procedimiento de inicialización del circuito; si no se cumple se notifica al usuario.

CODIGO DE VERIFICAR_CONEXIÓN

```
procedure TForm1.botoncompilarClick(Sender: TObject);
var
j:integer;
begin
with form2 do
begin
conectadotodo:=true;
if NuevaFuente = nil then
begin
validaerror(2);
exit;
end;
if g<>0 then
for j:=1 to  10  do
```

```
if  ( NuevaPulsador[j]<> nil) then
 begin
  NuevaPulsador[j].Vconexion;
  conectadoTodo:=conectadotodo and NuevaPulsador[j].todoconectado;
 end;// si es TVPulsador

if d<>0 then
 for j:=1 to  10  do
 if   NuevaSimultaneidad[j]<> nil then
 begin
 NuevaSimultaneidad[j].vconexion;
 conectadoTodo:=conectadotodo and NuevaSimultaneidad[j].todoconectado;
 end;// si es Tsimul

if e<>0 then
for j:=1 to  10  do
if   Nuevaselectora[j]<> nil then
 begin
 NuevaSelectora[j].Vconexion;
 conectadoTodo:=conectadotodo and NuevaSelectora[j].todoconectado;
 end;// si es Tconecc

 if c<>0 then
 for j:=1 to  10  do
 if   NuevaMemoria[j]<> nil then
 begin
 NuevaMemoria[j].vconexion;
 conectadoTodo:=conectadotodo and NuevaMemoria[j].todoconectado;
 end;// si es Tmemoria

if a<>0 then
 for j:=1 to 5 do
 if   Nuevocilindro[j]<> nil then
 begin
NuevoCilindro[j].Vconexion;
conectadoTodo:=conectadotodo and NuevoCilindro[j].todoconectado;
 end;

 if b<>o then
for j:=1 to  5  do
if   NuevoCresorte[j]<> nil then
 begin
 NuevoCResorte[j].Vconexion;
 conectadoTodo:=conectadotodo and NuevoCResorte[j].todoconectado;
 end;// si es TCresorte

 if f<>0 then
for j:=1 to  10  do
if   NuevaRodillo[j]<> nil then
 begin
 NuevaRodillo[j].vconexion;
```

```
 conectadoTodo:=conectadotodo and NuevaRodillo[j].todoconectado;
 end;// si es TVRodillos
 if conectadotodo= false then
 showmessage('hay un elemento del circuito no conectado')
 else
 begin
for j:=1 to  10  do
begin
if  ( NuevaPulsador[j]<> nil) then
Paux[j]:=NuevaPulsador[j].left;
if  ( NuevaRodillo[j]<> nil) then
Raux[j]:=NuevaRodillo[j].left;
if  ( Nuevamemoria[j]<> nil) then
Maux[j]:=Nuevamemoria[j].left;
end;// si es TVPulsador
 showmessage('Todos los elementos estan conectados');
 botonInicializar.enabled:=true;
 inicializar1.enabled:=true;
 end;
//until(inicializado);
end;
end;
```

INICIALIZACIÓN

Una vez realizada la verificación de la conexión se procede a inicializar el circuito llamando al método de inicialización para cada uno de los elementos que componen el circuito.

CODIGO DE INICIALIZACIÓN

```
procedure TForm1.botonInicializarClick(Sender: TObject);
var
inicializado:boolean;
j,c,i:integer;
Arect:Trect;
begin
detienemo;
with form2 do
begin

with Image1.Canvas do
begin
CopyMode := cmWhiteness;
```

```
ARect := Rect(0, 0, Image1.Width, Image1.Height);
CopyRect(ARect, Image1.Canvas, ARect);
CopyMode := cmSrcCopy;
end;
(activecontrol as Tcustomelemento).color:=clBtnFace;
c:=0;
if conectadotodo then
repeat
c:=c+1;
inicializado:=true;
if NuevaFuente <> nil then
begin
 NuevaFuente.inicializacion;
 inicializado:=inicializado and NuevaFuente.inicializado;
 image1.canvas.Pen.color:=clblue;
 canvas.Pen.color:=clblue;
 for i:=1 to NuevaFuente.N   do

hacerlinea(NuevaFuente.xe,NuevaFuente.Ye,NuevaFuente.puntos[i,1],NuevaFuente.puntos[i,
2]);
 image1.canvas.Pen.color:=clblack;
end;

if g<>0 then
for j:=1 to  10  do
if  ( NuevaPulsador[j]<> nil) then
 begin
 NuevaPulsador[j].inicializacion;
 inicializado:=inicializado and NuevaPulsador[j].inicializado;
 if (NuevaPulsador[j].X=1) and (paux[j]=NuevaPulsador[j].left) then
 begin
 NuevaPulsador[j].left:=paux[j]+25;
 NuevaPulsador[j].refresh;
end;
 if NuevaPulsador[j].A= 1 then
 begin
 image1.canvas.Pen.color:=clblue;
 canvas.Pen.color:=clblue;
 end
 else
 if NuevaPulsador[j].A=0 then
 begin
 image1.canvas.Pen.color:=clred;
 canvas.Pen.color:=clred;
 end;
 for i:=1 to NuevaPulsador[j].N   do

hacerlinea(NuevaPulsador[j].xe,NuevaPulsador[j].Ye,NuevaPulsador[j].puntos[i,1],NuevaPuls
ador[j].puntos[i,2]);
 image1.canvas.Pen.color:=clblack;
 end;// si es TVPulsador
```

```
if d<>0 then
 for j:=1 to  10  do
 if   NuevaSimultaneidad[j]<> nil then
 begin
 NuevaSimultaneidad[j].inicializacion;
 inicializado:=inicializado and NuevaSimultaneidad[j].inicializado;
if NuevaPulsador[j].A= 1 then
 image1.canvas.Pen.color:=clblue
 else
 if NuevaSimultaneidad[j].A=0 then
 image1.canvas.Pen.color:=clred;

 for i:=1 to NuevaSimultaneidad[j].N   do

hacerlinea(NuevaSimultaneidad[j].xe,NuevaSimultaneidad[j].Ye,NuevaSimultaneidad[j].punto
s[i,1],NuevaSimultaneidad[j].puntos[i,2]);
 image1.canvas.Pen.color:=clblack;
 end;// si es Tsimul

if e<>0 then
for j:=1 to  10  do
if   Nuevaselectora[j]<> nil then
 begin
 NuevaSelectora[j].inicializacion;
 inicializado:=inicializado and NuevaSelectora[j].inicializado;
if Nuevaselectora[j].A= 1 then
 image1.canvas.Pen.color:=clblue
 else
 if Nuevaselectora[j].A=0 then
 image1.canvas.Pen.color:=clred;

 for i:=1 to Nuevaselectora[j].N   do

hacerlinea(Nuevaselectora[j].xe,Nuevaselectora[j].Ye,Nuevaselectora[j].puntos[i,1],Nuevasel
ectora[j].puntos[i,2]);
 image1.canvas.Pen.color:=clblack;

 end;// si es Tconecc

 if c<>0 then
 for j:=1 to  10  do
 if   NuevaMemoria[j]<> nil then
 begin
 NuevaMemoria[j].inicializacion;
 inicializado:=inicializado and NuevaMemoria[j].inicializado;
 if (NuevaMemoria[j].X=1)and(Maux[j]=NuevaMemoria[j].left)and(NuevaMemoria[j].Y=0) then
 begin
 left:= Maux[j]+25;
 NuevaMemoria[j].refresh;
 end;
```

```
 if NuevaMemoria[j].A= 1 then
 begin
 image1.canvas.Pen.color:=clblue;
 canvas.Pen.color:=clblue;
 end
 else
 if NuevaMemoria[j].A=0 then
 begin
 image1.canvas.Pen.color:=clred;
 canvas.Pen.color:=clred;
 end;

 for i:=1 to NuevaMemoria[j].N   do
 hacerlinea(NuevaMemoria[j].left+40,NuevaMemoria[j].top-
2,NuevaMemoria[j].puntos[i,1],NuevaMemoria[j].puntos[i,2]);
 if  NuevaMemoria[j].A=0 then
 image1.canvas.Pen.color:=clblue
 else
  if  NuevaMemoria[j].A=1 then
 image1.canvas.Pen.color:=clred;
 for i:=1 to NuevaMemoria[j].NB   do
 hacerlinea(NuevaMemoria[j].left+60,NuevaMemoria[j].top-
4,NuevaMemoria[j].puntosB[i,1],NuevaMemoria[j].puntosB[i,2]);

 end;// si es Tmemoria

if a<>0 then
 for j:=1 to 5 do
 if   Nuevocilindro[j]<> nil then
 begin
 NuevoCilindro[j].inicializacion;
 inicializado:=inicializado and NuevoCilindro[j].inicializado;
 end;

 if b<>o then
for j:=1 to  5  do
if   NuevoCresorte[j]<> nil then
 begin
 NuevoCResorte[j].inicializacion;
 inicializado:=inicializado and NuevoCResorte[j].inicializado;
 end;// si es TCresorte

if f<>0 then
for j:=1 to  10  do
if   NuevaRodillo[j]<> nil then
 begin
  NuevaRodillo[j].inicializacion;
 inicializado:=inicializado and NuevaRodillo[j].inicializado;
 if (NuevaRodillo[j].X=1) and (raux[j]=NuevaRodillo[j].left) then
 begin
```

```
 NuevaRodillo[j].left:=raux[j]+25;
 NuevaRodillo[j].refresh;
 end;

 if NuevaRodillo[j].A= 1 then
 image1.canvas.Pen.color:=clblue
 else
 if NuevaRodillo[j].A=0 then
 image1.canvas.Pen.color:=clred;
 for i:=1 to NuevaRodillo[j].N   do

hacerlinea(NuevaRodillo[j].xe,NuevaRodillo[j].Ye,NuevaRodillo[j].puntos[i,1],NuevaRodillo[j].
puntos[i,2]);
 end;// si es TVRodillos

until(inicializado) or (c>150);
if c>=50 then
showmessage('Este circuito no es logico')
else
begin
showmessage('circuito inicializado');
botonsimular.enabled:=true;
simular1.enabled:=true;
end;
end;
end;
```

SIMULACIÓN

Este procedimiento se encarga de llamar al método **recorrer** para cada uno de los elementos del circuito según sea la salida de cada elemento; de esta manera se dibujan las líneas con el color correspondiente.

CODIGO DE SIMULACION

```
procedure TForm1.BotonsimularClick(Sender: TObject);
var t:integer;
procedure simular;
var
j,p,i:integer;
termino:boolean;
begin
with form2 do
begin
(activecontrol as Tcustomelemento).color:=clBtnFace;
p:=1;
```

```
repeat
p:=p+1;
termino:=true;
if g<>0 then
for j:=1 to  10  do
if NuevaPulsador[j]<> nil then
 begin
 NuevaPulsador[j].recorrido;
 termino:= termino and NuevaPulsador[j].TerminarSimulacion;
 if (NuevaPulsador[j].X=1) and (NuevaPulsador[j].left=paux[j]) then
 begin
 NuevaPulsador[j].left:=Paux[j]+25;
 refresh; sleep(2);
 NuevaPulsador[j].refresh;
 end
 else
 if (NuevaPulsador[j].X=0) and (NuevaPulsador[j].left<>paux[j])then
 begin
 NuevaPulsador[j].left:=Paux[j];
 refresh; sleep(2);
 NuevaPulsador[j].refresh;
 end;

 if NuevaPulsador[j].A= 1 then
 begin
 image1.canvas.Pen.color:=clblue;
 canvas.Pen.color:=clblue;
 end
 else
 if NuevaPulsador[j].A=0 then
 begin
 image1.canvas.Pen.color:=clred;
 canvas.Pen.color:=clred;
 end;
 for i:=1 to NuevaPulsador[j].N   do

hacerlinea(NuevaPulsador[j].xe,NuevaPulsador[j].Ye,NuevaPulsador[j].puntos[i,1],NuevaPuls
ador[j].puntos[i,2]);
end;// si es TVPulsador

if f<>0 then
for j:=1 to  10  do
if   NuevaRodillo[j]<> nil then
 begin
 NuevaRodillo[j].recorrido;
 termino:= termino  and NuevaRodillo[j].TerminarSimulacion;
 end;// si es TVRodillos

 if d<>0 then
 for j:=1 to  10  do
 begin
```

```
 if   Nuevasimultaneidad[j]<> nil then
 begin
 NuevaSimultaneidad[j].recorrido;
 termino:= termino  and NuevaSimultaneidad[j].TerminarSimulacion;
 if NuevaPulsador[j].A= 1 then
 image1.canvas.Pen.color:=clblue
 else
 if NuevaSimultaneidad[j].A=0 then
 image1.canvas.Pen.color:=clred;

 for i:=1 to NuevaSimultaneidad[j].N   do

hacerlinea(NuevaSimultaneidad[j].xe,NuevaSimultaneidad[j].Ye,NuevaSimultaneidad[j].punto
s[i,1],NuevaSimultaneidad[j].puntos[i,2]);
 end;
 end;// si es Tsimul

if e<>0 then
for j:=1 to  10  do
if   Nuevaselectora[j]<> nil then
 begin
 NuevaSelectora[j].recorrido;
 termino:= termino and NuevaSelectora[j].TerminarSimulacion;
 if Nuevaselectora[j].A= 1 then
 image1.canvas.Pen.color:=clblue
 else
 if Nuevaselectora[j].A=0 then
 image1.canvas.Pen.color:=clred;

 for i:=1 to Nuevaselectora[j].N   do

hacerlinea(Nuevaselectora[j].xe,Nuevaselectora[j].Ye,Nuevaselectora[j].puntos[i,1],Nuevasel
ectora[j].puntos[i,2]);
 end;// si es Tconecc}


 if c<>0 then
 for j:=1 to  10  do
 if   Nuevamemoria[j]<> nil then
 begin
 NuevaMemoria[j].recorrido;
 termino:= termino and NuevaMemoria[j].TerminarSimulacion;
 if (NuevaMemoria[j].X=1)and(NuevaMemoria[j].left=Maux[j])
 and(NuevaMemoria[j].Y=0)then
 begin
 image1.canvas.pen.color:=image1.canvas.pixels[maux[j]-2,NuevaMemoria[j].top+26];
 image1.canvas.moveto(Maux[j],NuevaMemoria[j].top+25);
 image1.canvas.lineto(Maux[j]+25,NuevaMemoria[j].top+25);
 NuevaMemoria[j].left:=Maux[j]+30;
 refresh; sleep(2);
 NuevaMemoria[j].refresh;
```

```
 end
 else
 if (NuevaMemoria[j].X=0) and (NuevaMemoria[j].left<>maux[j])then
 begin
 NuevaMemoria[j].left:=Maux[j];
 refresh; sleep(2);
 NuevaMemoria[j].refresh;
 end;


if NuevaMemoria[j].A= 1 then
 image1.canvas.Pen.color:=clblue
 else
 if NuevaMemoria[j].A=0 then
 image1.canvas.Pen.color:=clred;

 for i:=1 to NuevaMemoria[j].N   do
 hacerlinea(Maux[j]+40,NuevaMemoria[j].top-
2,NuevaMemoria[j].puntos[i,1],NuevaMemoria[j].puntos[i,2]);
 if  NuevaMemoria[j].A=0 then
 image1.canvas.Pen.color:=clblue
 else
 image1.canvas.Pen.color:=clred;
 for i:=1 to NuevaMemoria[j].NB   do
 hacerlinea(Maux[j]+60,NuevaMemoria[j].top-
4,NuevaMemoria[j].puntosB[i,1],NuevaMemoria[j].puntosB[i,2]);
 end;// si es Tmemoria

 if a<>0 then
 for j:=1 to 5 do
 if   Nuevocilindro[j]<> nil then
 begin
 NuevoCilindro[j].recorrido;
 termino:= termino and NuevoCilindro[j].TerminarSimulacion;
 end;

if b<>0 then
for j:=1 to  5  do
if   NuevoCresorte[j]<> nil then
 begin
 NuevoCResorte[j].recorrido;
 termino:= termino and NuevoCResorte[j].TerminarSimulacion;
 end;// si es TCresorte

 if f<>0 then
 for j:=1 to  10  do
 if   Nuevarodillo[j]<> nil then
 begin
 if  NuevaRodillo[j].tipo <> RodilloAbatibleConResorte then
 NuevaRodillo[j].recorrido;
 termino:= termino and NuevaRodillo[j].TerminarSimulacion;
```

```
 if (NuevaRodillo[j].X=1)and(NuevaRodillo[j].left=raux[j])then
 begin
 NuevaRodillo[j].left:=raux[j]+25;
 refresh;
 sleep(2);
 NuevaRodillo[j].refresh;
 end
 else
 if (NuevaRodillo[j].X=0) and (NuevaRodillo[j].left<>raux[j])then
 begin
 NuevaRodillo[j].left:=raux[j];
 refresh;
 sleep(2);
 NuevaRodillo[j].refresh;
 end;

 if NuevaRodillo[j].A=1 then
 image1.canvas.Pen.color:=clblue
 else
 if NuevaRodillo[j].A=0 then
 image1.canvas.Pen.color:=clred;
 for i:=1 to NuevaRodillo[j].N   do

hacerlinea(Raux[j]+40,NuevaRodillo[j].Ye,NuevaRodillo[j].puntos[i,1],NuevaRodillo[j].puntos[
i,2]);

end;// si es TVRodillos
until(termino) or (p>5);
end;
end;
begin
botonselectora.enabled:=false;
botonsimultaneidad.enabled:=false;
botoncilindro.enabled:=false;
botonfuente.enabled:=false;
botonmemoria.enabled:=false;
botonpulsador.enabled:=false;
botonrodillo.enabled:=false;
botoncilindror.enabled:=false;

botondesconectar.enabled:=false;
desconectar1.enabled:=false;

botonelimina.enabled:=false;
borra1.enabled:=false;

botoninicializar.enabled:=false;
inicializar1.enabled:=false;

botonCompilar.enabled:=false;
```

```
verifica1.enabled:=false;
simular;
end;
```

ELIMINAR

Toma el elemento activo del circuito y llama al método **Destroy** del elemento y borra todas sus conexiones y actualiza las líneas del circuito.

CODIGO DE ELIMINAR

```
procedure TForm1.BotonEliminaClick(Sender: TObject);
var
i,s,k,l:integer;
begin
with Form2 do
begin
if (form2.activecontrol is TcustomElemento) then
begin
// es un cilindro
if (form2.activecontrol is TFuente) then
begin
for s:=1 to  nuevafuente.N do
borralinea(nuevafuente.Xe,nuevafuente.Ye,nuevafuente.puntos[s,1],nuevafuente.puntos[s,2]
);
nuevafuente.destroy;
nuevafuente:=nil;
ConCb:=ConCb+1; Cb[ConCb]:=i; b:=b-1;
end;

// es un cilindro
if (form2.activecontrol is TCilindro) then
for i:=1 to 5 do
if  form2.activecontrol = NuevoCilindro[i] then
begin
with  NuevoCilindro[i] do
begin
 //borra las lineas que lo conectan
 if (ConexionX <> nil) then
 if (conexionX^<>nil)  then
 begin
 if ConexionX^ is Tmemoria then
 begin
  if salidaX = salidaA then
   begin
```

```
   borralinea(ConexionX^.left+40,ConexionX^.top-1,left+2,top+40);
   for s:=1 to ConexionX.N do
   if (ConexionX.puntos[s,1]=left+2)and (ConexionX.puntos[s,2]=top+40) then
    begin
    ConexionX.puntos[s,1]:=-1;
    ConexionX.puntos[s,2]:=-1;
    end;//del if
  end//fin si es salidaA
  else
   begin
    borralinea(ConexionX^.left+60,ConexionX^.top-4,left+2,top+40);
    for s:=1 to (ConexionX^ as Tmemoria).NB do
    if ((ConexionX^ as Tmemoria).puntosB[s,1]=left+2)and((ConexionX^ as
Tmemoria).puntosB[s,2]=top+40) then
     begin
     (ConexionX^as tmemoria).puntosB[s,1]:=-1;
     (ConexionX^ as tmemoria).puntosB[s,2]:=-1;
      end;//fin del if
    end; // fin del else
  end //si es memoria
else
begin
borralinea(ConexionX^.Xe,ConexionX^.Ye,left+2,top+40);
for s:=1 to ConexionX.N do
if (ConexionX.puntos[s,1]=left+2)and(ConexionX.puntos[s,2]=top+40) then
begin
ConexionX.puntos[s,1]:=-1;
ConexionX.puntos[s,2]:=-1;
end;//fin del if
end;//fin del else
end;//si la conexion es x

if ConexionY <> nil then
if conexionY^<>nil then
begin
borralinea(ConexionY^.Xe,ConexionY^.Ye,left+58,top+40);
if ConexionY^ is Tmemoria then
begin
  if salidaY = salidaA then
  begin
  borralinea(ConexionY^.left+40,ConexionY^.top-1,left+58,top+40);
  for s:=1 to ConexionY.N do
  if (ConexionY.puntos[s,1]=left+58)and (ConexionY.puntos[s,2]=top+40) then
  begin
  ConexionY.puntos[s,1]:=-1;
  ConexionY.puntos[s,2]:=-1;
  end;
  end
  else
  begin
  borralinea(ConexionY^.left+60,ConexionY^.top-4,left+58,top+40);
```

```
  for s:=1 to (ConexionY^ as Tmemoria).NB do
  if       ((ConexionY^       as       Tmemoria).puntosB[s,1]=left+58)and((ConexionY^       as
Tmemoria).puntosB[s,2]=top+40) then
  begin
 (ConexionY^as tmemoria).puntosB[s,1]:=-1;
 (ConexionY^ as tmemoria).puntosB[s,2]:=-1;
  end;
  end;//fin del else
end //si es memoria
else
begin
borralinea(ConexionY^.Xe,ConexionY^.Ye,left+58,top+40);
for s:=1 to ConexionY.N do
if (ConexionY.puntos[s,1]=left+58)and(ConexionY.puntos[s,2]=top+40) then
begin
ConexionY.puntos[s,1]:=-1;
ConexionY.puntos[s,2]:=-1;
end;//si
end;//else
end;//conexionY
destroy;
end;// del with
NuevoCilindro[i]:=nil;
ConCa:=ConCa+1; Ca[ConCa]:=i; a:=a-1;
break;
end;

//si es un TCresorte
if (form2.activecontrol is TCresorte) then
for i:=1 to 5 do
if  form2.activecontrol = NuevoCresorte[i] then
BEGIN
with NuevoCresorte[i] do
begin//borra las lineas que lo conectan
 if ConexionX <> nil then
  if ConexionX^ is Tmemoria then
   begin
   if salidaX = salidaA then
    begin
    borralinea(ConexionX^.left+40,ConexionX^.top-1,left+2,top+40);
    for s:=1 to ConexionX.N do
     if (ConexionX.puntos[s,1]=left+2)and (ConexionX.puntos[s,2]=top+40) then
     begin
     ConexionX.puntos[s,1]:=-1;
     ConexionX.puntos[s,2]:=-1;
     end;//del if
    end//fin si es salidaA
   else
    begin
    borralinea(ConexionX^.left+60,ConexionX^.top-4,left+2,top+40);
     for s:=1 to (ConexionX^ as Tmemoria).NB do
```

```
    if ((ConexionX^ as Tmemoria).puntosB[s,1]=left+2)and((ConexionX^ as
Tmemoria).puntosB[s,2]=top+40) then
     begin
     (ConexionX^as tmemoria).puntosB[s,1]:=-1;
     (ConexionX^ as tmemoria).puntosB[s,2]:=-1;
     end;//fin del if
   end; // fin del else
  end //si es memoria
 else
  begin
  borralinea(ConexionX^.Xe,ConexionX^.Ye,left+2,top+40);
   for s:=1 to ConexionX.N do
    if (ConexionX.puntos[s,1]=left+2)and(ConexionX.puntos[s,2]=top+40) then
    begin
    ConexionX.puntos[s,1]:=-1;
    ConexionX.puntos[s,2]:=-1;
    end;
 end;//fin del else no es memoria
destroy;
end;// del with
NuevoCresorte[i]:=nil;
ConCb:=ConCb+1; Cb[ConCb]:=i; b:=b-1;
break;
end;//fin del with

//********************** si una memoria
if (form2.activecontrol is TMemoria) then
for i:=1 to 10 do
if  form2.activecontrol = NuevaMemoria[i] then
with NuevaMemoria[i] do
begin
for s:=1 to  N do
borralinea(left+40,top-1,puntos[s,1],puntos[s,2]);
for s:=1 to  NB do
borralinea(left+60,top-4,puntosB[s,1],puntosB[s,2]);
// ******conexion X
if ConexionX <> nil then
if conexionX^<>nil then
 if ConexionX^ is Tmemoria then
 begin
  if salidaX = salidaA then
  begin
   borralinea(ConexionX.left+40,ConexionX^.top-1,left-2,top+25);
   for s:=1 to ConexionX.N do
   begin
   if (ConexionX.puntos[s,1]=left-2)and (ConexionX.puntos[s,2]=top+25) then
    begin
    ConexionX.puntos[s,1]:=-1;
    ConexionX.puntos[s,2]:=-1;
   end;//del if
   end;
```

```
  end//fin si es salidaA
  else
   begin
    borralinea(ConexionX^.left+60,ConexionX^.top-4,left-2,top+25);
    for s:=1 to (ConexionX^ as Tmemoria).NB do
    if       ((ConexionX^       as       Tmemoria).puntosB[s,1]=left-2)and((ConexionX^       as
Tmemoria).puntosB[s,2]=top+25) then
     begin
     (ConexionX^as tmemoria).puntosB[s,1]:=-1;
     (ConexionX^ as tmemoria).puntosB[s,2]:=-1;
  end;//fin del if
  end; // fin del else
end //si es memoria
else
begin
borralinea(ConexionX^.Xe,ConexionX^.Ye,left-2,top+25);
for s:=1 to ConexionX.N do
if (ConexionX.puntos[s,1]=left-2)and(ConexionX.puntos[s,2]=top+25) then
begin
ConexionX.puntos[s,1]:=-1;
ConexionX.puntos[s,2]:=-1;
end;
end;//fin del else

//conexion es Y
if ConexionY <> nil then
if conexionY^<>nil then
begin
borralinea(ConexionY^.Xe,ConexionY^.Ye,left+72,top+25);
if ConexionY^ is Tmemoria then
begin
  if salidaY = salidaA then
  begin
  borralinea(ConexionY^.left+40,ConexionY^.top-1,left+72,top+25);
  for s:=1 to ConexionY.N do
  if (ConexionY.puntos[s,1]=left+72)and(ConexionY.puntos[s,2]=top+25) then
  begin
  ConexionY.puntos[s,1]:=-1;
  ConexionY.puntos[s,2]:=-1;
  end;
  end
  else
  begin
  borralinea(ConexionY^.left+60,ConexionY^.top-4,left+72,top+25);
  for s:=1 to (ConexionY^ as Tmemoria).NB do
  if       ((ConexionY^       as       Tmemoria).puntosB[s,1]=left+72)and((ConexionY^       as
Tmemoria).puntosB[s,2]=top+25) then
  begin
 (ConexionY^as tmemoria).puntosB[s,1]:=-1;
 (ConexionY^ as tmemoria).puntosB[s,2]:=-1;
  end;
```

```
  end;
end //si es memoria
else
begin
borralinea(ConexionY^.Xe,ConexionY^.Ye,left+75,top+25);
for s:=1 to ConexionY.N do
if (ConexionY.puntos[s,1]=left+72)and(ConexionY.puntos[s,2]=top+25) then
begin
ConexionY.puntos[s,1]:=-1;
ConexionY.puntos[s,2]:=-1;
end;
end;
end;// si la conexion es Y

//conexion es P
if ConexionP <> nil then
if conexionP^<>nil then
begin
if ConexionP^ is Tmemoria then
begin
  if salidaY = salidaA then
  begin
  borralinea(ConexionP^.left+40,ConexionP^.top-1,left+40,top+42);
  for s:=1 to ConexionP.N do
  if (ConexionP.puntos[s,1]=left+40)and (ConexionP.puntos[s,2]=top+42) then
  begin
  ConexionP.puntos[s,1]:=-1;
  ConexionP.puntos[s,2]:=-1;
  end;
  end
  else
  begin
  borralinea(ConexionP^.left+60,ConexionP^.top-4,left+40,top+42);
  for s:=1 to (ConexionP^ as Tmemoria).NB do
  if ((ConexionP^ as Tmemoria).puntosB[s,1]=left+40)and((ConexionP^ as
Tmemoria).puntosB[s,2]=top+42) then
  begin
 (ConexionP^as tmemoria).puntosB[s,1]:=-1;
 (ConexionP^ as tmemoria).puntosB[s,2]:=-1;
  end;
  end;
end //si es memoria
else
begin
borralinea(ConexionP^.Xe,ConexionP^.Ye,left+40,top+42);
for s:=1 to ConexionP.N do
if (ConexionP.puntos[s,1]=left+40)and(ConexionP.puntos[s,2]=top+42) then
begin
ConexionP.puntos[s,1]:=-1;
ConexionP.puntos[s,2]:=-1;
end;
```

```
end;
end;
//
form2.NuevaMemoria[i].destroy;NuevaMemoria[i]:=nil;ConCc:=ConCc+1;        Cc[ConCc]:=i;
c:=c-1;
break;
end;
// si es Simultaneidad
if (form2.activecontrol is Tsimul) then
for i:=1 to 10 do
if  form2.activecontrol = NuevaSimultaneidad[i] then
with  NuevaSimultaneidad[i] do
begin
for s:=1 to N do
borralinea(Xe,Ye,puntos[s,1],puntos[s,2]);
//
if ConexionX <> nil then
if conexionX^<>nil then
 if ConexionX^ is Tmemoria then
 begin
  if salidaX = salidaA then
  begin
   borralinea(ConexionX^.left+40,ConexionX^.top-1,left-2,top+25);
   for s:=1 to ConexionX.N do
   if (ConexionX.puntos[s,1]=left-2)and (ConexionX.puntos[s,2]=top+25) then
    begin
    ConexionX.puntos[s,1]:=-1;
    ConexionX.puntos[s,2]:=-1;
   end;//del if
  end//fin si es salidaA
  else
   begin
    borralinea(ConexionX^.left+60,ConexionX^.top-4,left-2,top+25);
    for s:=1 to (ConexionX^ as Tmemoria).NB do
    if       ((ConexionX^       as       Tmemoria).puntosB[s,1]=left-2)and((ConexionX^       as
Tmemoria).puntosB[s,2]=top+25) then
     begin
     (ConexionX^as tmemoria).puntosB[s,1]:=-1;
     (ConexionX^ as tmemoria).puntosB[s,2]:=-1;
  end;//fin del if
  end; // fin del else
end //si es memoria
else
begin
borralinea(ConexionX^.Xe,ConexionX^.Ye,left-2,top+25);
for s:=1 to ConexionX.N do
if (ConexionX.puntos[s,1]=left-2)and(ConexionX.puntos[s,2]=top+25) then
begin
ConexionX.puntos[s,1]:=-1;
ConexionX.puntos[s,2]:=-1;
end;
```

```
end;//fin del else

//borra las lineas de entrada
if ConexionY <> nil then
if conexionY^<>nil then
begin
borralinea(ConexionY^.Xe,ConexionY^.Ye,left+72,top+25);
if ConexionY^ is Tmemoria then
begin
  if salidaY = salidaA then
  begin
  borralinea(ConexionY^.left+40,ConexionY^.top-1,left+72,top+25);
  for s:=1 to ConexionY.N do
  if (ConexionY.puntos[s,1]=left+72)and (ConexionY.puntos[s,2]=top+25) then
  begin
  ConexionY.puntos[s,1]:=-1;
  ConexionY.puntos[s,2]:=-1;
  end;
  end
  else
  begin
  borralinea(ConexionY^.left+60,ConexionY^.top-4,left+72,top+25);
  for s:=1 to (ConexionY^ as Tmemoria).NB do
  if       ((ConexionY^       as       Tmemoria).puntosB[s,1]=left+72)and((ConexionY^       as
Tmemoria).puntosB[s,2]=top+25) then
  begin
 (ConexionY^as tmemoria).puntosB[s,1]:=-1;
 (ConexionY^ as tmemoria).puntosB[s,2]:=-1;
  end;
  end;
end //si es memoria
else
begin
borralinea(ConexionY^.Xe,ConexionY^.Ye,left+75,top+25);
for s:=1 to ConexionY.N do
if (ConexionY.puntos[s,1]=left+72)and(ConexionY.puntos[s,2]=top+25) then
begin
ConexionY.puntos[s,1]:=-1;
ConexionY.puntos[s,2]:=-1;
end;
end;
end;// si la conexion es Y
//
form2.NuevaSimultaneidad[i].destroy;NuevaSimultaneidad[i]:=nil;
ConCd:=ConCd+1; Cd[ConCd]:=i; d:=d-1;
 break;  end;

// si es una selectora
if (form2.activecontrol is Tconecc) then
for i:=1 to 10 do
if  form2.activecontrol = NuevaSelectora[i] then
```

```
with NuevaSelectora[i] do
begin
for s:=1 to N do
borralinea(Xe,Ye,puntos[s,1],puntos[s,2]);
//
if ConexionX <> nil then
if conexionX^<>nil then
 if ConexionX^ is Tmemoria then
 begin
  if salidaX = salidaA then
  begin
   borralinea(ConexionX^.left+40,ConexionX^.top-1,left-2,top+25);
   for s:=1 to ConexionX.N do
   if (ConexionX.puntos[s,1]=left-2)and (ConexionX.puntos[s,2]=top+25) then
    begin
    ConexionX.puntos[s,1]:=-1;
    ConexionX.puntos[s,2]:=-1;
   end;//del if
  end//fin si es salidaA
  else
   begin
    borralinea(ConexionX^.left+60,ConexionX^.top-4,left-2,top+25);
    for s:=1 to (ConexionX^ as Tmemoria).NB do
    if ((ConexionX^ as Tmemoria).puntosB[s,1]=left-2)and((ConexionX^ as
Tmemoria).puntosB[s,2]=top+25) then
     begin
     (ConexionX^as tmemoria).puntosB[s,1]:=-1;
     (ConexionX^ as tmemoria).puntosB[s,2]:=-1;
  end;//fin del if
  end; // fin del else
end //si es memoria
else
begin
borralinea(ConexionX^.Xe,ConexionX^.Ye,left-2,top+25);
for s:=1 to ConexionX.N do
if (ConexionX.puntos[s,1]=left-2)and(ConexionX.puntos[s,2]=top+25) then
begin
ConexionX.puntos[s,1]:=-1;
ConexionX.puntos[s,2]:=-1;
end;
end;//fin del else

//borra las lineas de entrada
if ConexionY <> nil then
if conexionY^<>nil then
begin
borralinea(ConexionY^.Xe,ConexionY^.Ye,left+72,top+25);
if ConexionY^ is Tmemoria then
begin
  if salidaY = salidaA then
  begin
```

```
 borralinea(ConexionY^.left+40,ConexionY^.top-1,left+72,top+25);
 for s:=1 to ConexionY.N do
 if (ConexionY.puntos[s,1]=left+72)and (ConexionY.puntos[s,2]=top+25) then
 begin
 ConexionY.puntos[s,1]:=-1;
 ConexionY.puntos[s,2]:=-1;
 end;
 end
 else
 begin
 borralinea(ConexionY^.left+60,ConexionY^.top-4,left+72,top+25);
 for s:=1 to (ConexionY^ as Tmemoria).NB do
 if ((ConexionY^ as Tmemoria).puntosB[s,1]=left+72)and((ConexionY^ as
Tmemoria).puntosB[s,2]=top+25) then
 begin
(ConexionY^as tmemoria).puntosB[s,1]:=-1;
(ConexionY^ as tmemoria).puntosB[s,2]:=-1;
 end;
 end;
end //si es memoria
else
begin
borralinea(ConexionY^.Xe,ConexionY^.Ye,left+75,top+25);
for s:=1 to ConexionY.N do
if (ConexionY.puntos[s,1]=left+72)and(ConexionY.puntos[s,2]=top+25) then
begin
ConexionY.puntos[s,1]:=-1;
ConexionY.puntos[s,2]:=-1;
end;
end;
end;// si la conexion es Y
//
form2.NuevaSelectora[i].destroy;NuevaSelectora[i]:=nil;ConCe:=ConCe+1;      Ce[ConCe]:=i;
e:=e-1;
 break;
end;
//************************************************
// si es un rodillo
if (form2.activecontrol is TVRodillos) then
for i:=1 to 10 do
if  form2.activecontrol = NuevaRodillo[i] then
with  NuevaRodillo[i] do
begin
for s:=1 to N do
borralinea(Xe,Ye,puntos[s,1],puntos[s,2]);
//conexion es P
if ConexionP <> nil then
if conexionP^<>nil then
begin
borralinea(ConexionP^.Xe,ConexionP^.Ye,left+40,top+42);
if ConexionP^ is Tmemoria then
```

```
begin
  if salidaY = salidaA then
  begin
  borralinea(ConexionP^.left+40,ConexionP^.top-1,left+40,top+42);
  for s:=1 to ConexionP.N do
  if (ConexionP.puntos[s,1]=left+40)and (ConexionP.puntos[s,2]=top+42) then
  begin
  ConexionP.puntos[s,1]:=-1;
  ConexionP.puntos[s,2]:=-1;
  end;
  end
  else
  begin
  borralinea(ConexionP^.left+60,ConexionP^.top-4,left+40,top+42);
  for s:=1 to (ConexionP^ as Tmemoria).NB do
  if ((ConexionP^ as Tmemoria).puntosB[s,1]=left+40)and((ConexionP^ as
Tmemoria).puntosB[s,2]=top+42) then
  begin
 (ConexionP^as tmemoria).puntosB[s,1]:=-1;
 (ConexionP^ as tmemoria).puntosB[s,2]:=-1;
  end;
  end;
end //si es memoria
else
begin
borralinea(ConexionP^.Xe,ConexionP^.Ye,left+40,top+42);
for s:=1 to ConexionP.N do
if (ConexionP.puntos[s,1]=left+40)and(ConexionP.puntos[s,2]=top+42) then
begin
ConexionP.puntos[s,1]:=-1;
ConexionP.puntos[s,2]:=-1;
end;
end;
end;
//
form2.NuevaRodillo[i].destroy;NuevaRodillo[i]:=nil; ConCf:=ConCf+1; Cf[ConCf]:=i; f:=f-1;
 break;  end;
//***************PULSADOR*********************
if (form2.activecontrol is TVPulsador) then
for i:=1 to 10 do
if  form2.activecontrol = NuevaPulsador[i] then
with NuevaPulsador[i] do
begin
for s:=1 to N do
borralinea(Xe,Ye,puntos[s,1],puntos[s,2]);
//conexion es P
if ConexionP <> nil then
if conexionP^<>nil then
begin
if ConexionP^ is Tmemoria then
begin
```

```
 if salidaY = salidaA then
 begin
 borralinea(ConexionP^.left+40,ConexionP^.top-1,left+40,top+42);
 for s:=1 to ConexionP.N do
 if (ConexionP.puntos[s,1]=left+40)and (ConexionP.puntos[s,2]=top+42) then
 begin
 ConexionP.puntos[s,1]:=-1;
 ConexionP.puntos[s,2]:=-1;
 end;
 end
 else
 begin
 borralinea(ConexionP^.left+60,ConexionP^.top-4,left+40,top+42);
 for s:=1 to (ConexionP^ as Tmemoria).NB do
 if ((ConexionP^ as Tmemoria).puntosB[s,1]=left+40)and((ConexionP^ as
Tmemoria).puntosB[s,2]=top+42) then
 begin
 (ConexionP^as tmemoria).puntosB[s,1]:=-1;
 (ConexionP^ as tmemoria).puntosB[s,2]:=-1;
 end;
 end;
end //si es memoria
else
begin
borralinea(ConexionP^.Xe,ConexionP^.Ye,left+40,top+42);
for s:=1 to ConexionP.N do
if (ConexionP.puntos[s,1]=left+40)and(ConexionP.puntos[s,2]=top+42) then
begin
ConexionP.puntos[s,1]:=-1;
ConexionP.puntos[s,2]:=-1;
end;
end;
end;
form2.NuevaPulsador[i].destroy;NuevaPulsador[i]:=nil;  ConCg:=ConCg+1;  Cg[ConCg]:=i;
g:=g-1;
break;  end;
actualizalineas;
end;//si activecontrol es TcustomElemento
end;//fin del with
end;
```

DESCONECTAR

El usuario determina que salida y que entrada se van a desconectar, el procedimiento borra la línea de conexión y actualiza las líneas del circuito. La

entrada del elemento desconectado se habilita para que se pueda volver a conectar.

CODIGO DE DESCONECTAR

```
procedure TForm1.botondesconectarClick(Sender: TObject);
var
i,j,Xp,Yp,s:integer;
begin
with form2 do
begin
 for j:=1 to  5  do
 begin
  if (NuevoCilindro[j]<>nil)  then
   if NuevoCilindro[j].conector then
     begin
     conectora:=@NuevoCilindro[j];
     Nconectora:=j;
     end
   else
    if NuevoCilindro[j].conectado then
    begin
    conectado:=@NuevoCilindro[j];
    Nconectado:=j;
   end;

 if (NuevoCResorte[j]<>nil)  then
  if form2.NuevoCResorte[j].conector then
  begin
    conectora:=@NuevoCResorte[j];
    Nconectora:=j;
  end
  else
   if NuevoCResorte[j].conectado then
   begin
    conectado:=@NuevoCResorte[j];
    Nconectado:=j;
   end;
end; // del for

 for j:=1 to  10  do
 begin
 if (NuevaMemoria[j]<>nil)  then
  if NuevaMemoria[j].conector then
   begin
   conectora:=@NuevaMemoria[j];
   Nconectora:=j;
```

```
 end
 else
 if NuevaMemoria[j].conectado then
 begin
  conectado:=@NuevaMemoria[j];
  Nconectado:=j;
 end;

if (NuevaSimultaneidad[j]<>nil)  then
 if NuevaSimultaneidad[j].conector then
 begin
 conectora:=@NuevaSimultaneidad[j];
 Nconectora:=j;
 end
 else
 if NuevaSimultaneidad[j].conectado then
 begin
 conectado:=@NuevaSimultaneidad[j];
 Nconectado:=j;
 end;

if (NuevaSelectora[j]<>nil)  then
 if NuevaSelectora[j].conector then
 begin
 conectora:=@NuevaSelectora[j];
 Nconectora:=j;
 end
else
 if NuevaSelectora[j].conectado then
 begin
 conectado:=@NuevaSelectora[j];
 Nconectado:=j;
 end;


if (NuevaRodillo[j]<>nil)  then
 if NuevaRodillo[j].conector then
 begin
  conectora:=@NuevaRodillo[j] ;
  Nconectora:=j;
  end
  else
   if NuevaRodillo[j].conectado then
   begin
   conectado:=@NuevaRodillo[j];
   Nconectado:=j;
   end;

if (NuevaPulsador[j]<>nil)  then
 if NuevaPulsador[j].conector then
 begin
```

```
    conectora:=@NuevaPulsador[j];
    Nconectora:=j;
    end
  else
   if NuevaPulsador[j].conectado then
   begin
   conectado:=@NuevaPulsador[j];
   Nconectado:=j;
   end;
end;// del for

  if (NuevaFuente<>nil)  then
   if NuevaFuente.conector then
   conectora:=@NuevaFuente;

//*********************************************************************
if (conectora <> nil) and (conectado<>nil) then
begin
if conectado^.Entrada = 1 then
begin
// si se activa a un cilindro
if (conectado^ is Tcilindro) then
begin
NuevoCilindro[NConectado].conexionX:=nil;
NuevoCilindro[NConectado].salidaX:=salidaA;
NuevoCilindro[NConectado].activacionIzquierda:=false;
NuevoCilindro[NConectado].refresh;
if (conectora^ is Tfuente) then
if Nuevafuente.TipoConexion=Cindirecta then
begin
// aqui se borra la fuente de conexion
nohacerlinea:=true;
form2.image1.canvas.Brush.style:=bsSolid;
form2.image1.canvas.Brush.color:=clblue;
form2.image1.Canvas.ellipse(conectado.Xs-
5,conectado.Ys,conectado.Xs+5,conectado.Ys+10);
end;
end;

// si se activa a un cilindro con muelle
if (conectado^ is TCResorte) then
begin
NuevoCresorte[NConectado].conexionX:=Nil;
NuevoCresorte[NConectado].salidaX:=salidaA;
if (conectora^ is Tfuente) then
if Nuevafuente.TipoConexion=Cindirecta then
begin
nohacerlinea:=true;
form2.image1.canvas.Brush.style:=bsSolid;
form2.image1.canvas.Brush.color:=clblue;
```

```
form2.image1.Canvas.ellipse(conectado.Xs-
5,conectado.Ys,conectado.Xs+5,conectado.Ys+10);
end;
end;

// si se activa a un memoria
if (conectado^ is Tmemoria)then
begin
Nuevamemoria[NConectado].conexionX:=nil;
Nuevamemoria[NConectado].salidaX:=salidaA;
Nuevamemoria[NConectado].activacionIzquierda:=false;
Nuevamemoria[NConectado].refresh;
if (conectora^ is Tfuente) then
if Nuevafuente.TipoConexion=Cindirecta then
begin
nohacerlinea:=true;
form2.image1.canvas.Brush.style:=bsSolid;
form2.image1.canvas.Brush.color:=clblue;
form2.image1.Canvas.ellipse(conectado.Xs,conectado.Ys-5,conectado.Xs-
10,conectado.Ys+5);
end;
end;
// si se activa a una selectora

if (conectado^ is Tconecc)then
begin
NuevaSelectora[NConectado].conexionX:=nil;//conectora;
NuevaSelectora[NConectado].salidaX:=salidaA;
NuevaSelectora[NConectado].ActivacionIzquierda:=false;
NuevaSelectora[NConectado].refresh;
if(conectora^ is Tfuente) then
if Nuevafuente.TipoConexion=Cindirecta then
begin
nohacerlinea:=true;
form2.image1.canvas.Brush.style:=bsSolid;
form2.image1.canvas.Brush.color:=clblue;
form2.image1.Canvas.ellipse(conectado.Xs,conectado.Ys-5,conectado.Xs-
10,conectado.Ys+5);
end;
end;

// si se activa a una simultaneidad
if (conectado^ is Tsimul)then
begin
Nuevasimultaneidad[NConectado].conexionX:=nil;//conectora;
Nuevasimultaneidad[NConectado].salidaX:=salidaA;
Nuevasimultaneidad[NConectado].ActivacionIzquierda:=false;
Nuevasimultaneidad[NConectado].refresh;
if (conectora^ is Tfuente) then
if Nuevafuente.TipoConexion=Cindirecta then
begin
```

```
nohacerlinea:=true;
form2.image1.canvas.Brush.style:=bsSolid;
form2.image1.canvas.Brush.color:=clblue;
form2.image1.Canvas.ellipse(conectado.Xs,conectado.Ys-5,conectado.Xs-
10,conectado.Ys+5);
end;
end;

// si se activa a un rodillo
if (conectado^ is TVrodillos) then
if (conectora^ is Tcilindro) or (conectora^ is TCresorte) then
begin
NuevaRodillo[NConectado].conexionX:=Nil;
NuevaRodillo[NConectado].salidaX:=salidaA;
NuevaRodillo[NConectado].Nombre.caption:='0.0';
end;
end;// entrada es 1


if conectado^.entrada = 2 then
begin
// si se activa a un cilindro
if (conectado^ is Tcilindro) then
begin
NuevoCilindro[NConectado].conexionY:=nil;//conectora;
NuevoCilindro[NConectado].salidaY:=salidaA;
NuevoCilindro[NConectado].activacionDerecha:=false;
NuevoCilindro[NConectado].refresh;
if (conectora^ is Tfuente) then
if Nuevafuente.TipoConexion=Cindirecta then
begin
nohacerlinea:=true;
form2.image1.canvas.Brush.style:=bsSolid;
form2.image1.canvas.Brush.color:=clblue;
form2.image1.Canvas.ellipse(conectado.Xs-
5,conectado.Ys,conectado.Xs+5,conectado.Ys+10);
end;
end;

// si se activa a un memoria
if (conectado^ is Tmemoria) then
begin
Nuevamemoria[NConectado].conexionY:=nil;//conectora;
Nuevamemoria[NConectado].salidaY:=salidaA;
Nuevamemoria[NConectado].activacionDerecha:=false;
Nuevamemoria[NConectado].refresh;
if (conectora^ is Tfuente) then
if Nuevafuente.TipoConexion=Cindirecta then
begin
nohacerlinea:=true;
form2.image1.canvas.Brush.style:=bsSolid;
```

```
form2.image1.canvas.Brush.color:=clblue;
form2.image1.Canvas.ellipse(conectado.Xs,conectado.Ys-
5,conectado.Xs+10,conectado.Ys+5);
end;
end;

// si se activa a una selectora
if (conectado^ is Tconecc) then
begin
NuevaSelectora[NConectado].conexionY:=nil;
NuevaSelectora[NConectado].salidaY:=salidaA;
NuevaSelectora[NConectado].ActivacionDerecha:=false;
NuevaSelectora[NConectado].refresh;
if (conectora^ is Tfuente) then
if Nuevafuente.TipoConexion=Cindirecta then
begin
nohacerlinea:=true;
form2.image1.canvas.Brush.style:=bsSolid;
form2.image1.canvas.Brush.color:=clblue;
form2.image1.Canvas.ellipse(conectado.Xs,conectado.Ys-
5,conectado.Xs+10,conectado.Ys+5);
end;
end;


// si se activa a una simultaneidad
if (conectado^ is Tsimul) then
begin
Nuevasimultaneidad[NConectado].conexionY:=nil;//conectora;
Nuevasimultaneidad[NConectado].salidaY:=SalidaA;
Nuevasimultaneidad[NConectado].ActivacionDerecha:=false;
Nuevasimultaneidad[NConectado].refresh;
if (conectora^ is Tfuente) then
if Nuevafuente.TipoConexion=Cindirecta then
begin
nohacerlinea:=true;
form2.image1.canvas.Brush.style:=bsSolid;
form2.image1.canvas.Brush.color:=clblue;
form2.image1.Canvas.ellipse(conectado.Xs,conectado.Ys-
5,conectado.Xs+10,conectado.Ys+5);
end;
end;

end;//  fin si la entrada es 2


// si  la entrada es 3
if conectado^.entrada = 3 then
begin
// si se activa a un memoria
if (conectado^ is Tmemoria) then
```

```
begin
Nuevamemoria[NConectado].conexionP:=nil;//conectora;
Nuevamemoria[NConectado].salidaP:=salidaA;
Nuevamemoria[NConectado].activacionPresion:=false;
Nuevamemoria[NConectado].refresh;
if (conectora^ is Tfuente) then
if Nuevafuente.TipoConexion=Cindirecta then
begin
nohacerlinea:=true;
form2.image1.canvas.Brush.style:=bsSolid;
form2.image1.canvas.Brush.color:=clblue;
form2.image1.Canvas.ellipse(conectado.Xs-
5,conectado.Ys,conectado.Xs+5,conectado.Ys+10);
end;
end;
// si se activa a un rodillo
if (conectado^ is TVrodillos) then
begin
NuevaRodillo[NConectado].conexionP:=nil;//conectora;
NuevaRodillo[NConectado].salidaP:=salidaA;
NuevaRodillo[NConectado].activacionPresion:=false;
NuevaRodillo[NConectado].refresh;
if (conectora^ is Tfuente) then
if Nuevafuente.TipoConexion=Cindirecta then
begin
nohacerlinea:=true;
form2.image1.canvas.Brush.style:=bsSolid;
form2.image1.canvas.Brush.color:=clblue;
form2.image1.Canvas.ellipse(conectado.Xs-
5,conectado.Ys,conectado.Xs+5,conectado.Ys+10);
end;
end;


// si se activa a un pulsador
if   (conectado^ is TVpulsador) then
begin
Nuevapulsador[NConectado].conexionP:=nil;//conectora;
Nuevapulsador[NConectado].salidaP:=salidaA;
Nuevapulsador[NConectado].activacionPresion:=false;
Nuevapulsador[NConectado].refresh;
if  (conectora^ is Tfuente) then
if Nuevafuente.TipoConexion=Cindirecta then
begin
nohacerlinea:=true;
form2.image1.canvas.pen.mode:=pmnot;
form2.image1.canvas.Brush.style:=bsSolid;
form2.image1.canvas.Brush.color:=clblue;
form2.image1.Canvas.ellipse(conectado.Xs-
5,conectado.Ys,conectado.Xs+5,conectado.Ys+10);
end;
```

```
end;
end;// fin si la entrada es 3
if not(nohacerlinea) then
begin
borralinea(conectora.Xe,conectora.Ye,conectado.Xs,conectado.Ys);

if (conectora^ is Tmemoria)  then
 if  (conectora^ as Tmemoria).salidas= salidaB then
  begin
  for i:=1 to (conectora^ as Tmemoria).NB do
  if (conectado^.Ys<>(conectora^ as Tmemoria).puntosB[i,2]) or
(conectado^.Xs<>(conectora^ as Tmemoria).puntosB[i,1]) then
  hacerlinea(conectora.Xe,conectora.Ye,(conectora^ as Tmemoria).puntosB[i,1],(conectora^
as  Tmemoria).puntosB[i,2])
 else
 begin
 (conectora^ as Tmemoria).puntosB[i,1]:=-1;
 (conectora^ as Tmemoria).puntosB[i,2]:=-1;
 end;
 end
else
 for i:=1 to conectora.N do
 if (conectado^.Ys<>conectora^.puntos[i,2]) or (conectado^.Xs<>conectora^.puntos[i,1])
then
 hacerlinea(conectora.Xe,conectora.Ye,conectora.puntos[i,1],conectora.puntos[i,2])
 else
 begin
 conectora.puntos[i,1]:=-1;
 conectora.puntos[i,2]:=-1;
 end
else
for i:=1 to conectora.N do
begin
if (conectado^.Ys<>conectora^.puntos[i,2]) or (conectado^.Xs<>conectora^.puntos[i,1]) then
 hacerlinea(conectora.Xe,conectora.Ye,conectora.puntos[i,1],conectora.puntos[i,2])
else
begin
 conectora.puntos[i,1]:=-1;
 conectora.puntos[i,2]:=-1;
end;
end;
nohacerlinea:=false;
end;//si no hacer linea
limpiaconexion;
conectora:=nil;
conectado:=nil;
end;// fin si conectora y conectado es verdadero
end; // fin del with
end;// fin del click
```

VALIDAR SECUENCIA

Es el encargado de leer la cadena capturada y validar que su entrada sea correcta.

CODIGO DE VALIDAR

```
procedure TForm3.SpeedButton1Click(Sender: TObject);
begin

o:=1;u:=1;CC:=0;w:=1;ma:=0;mb:=0;mc:=0;md:=0;me:=0;pa:=0;pc:=2;Ap:=0;Am:=0;cm:=0;b
m:=0;dm:=0;
  em:=0;an:=0;bn:=0;cn:=0;en:=0;dn:=0;Bp:=0;
  Cp:=0;Dp:=0;Ep:=0;sa:=0;sb:=0;sc:=0;sd:=0;
  se:=0;nj:=0;
  for i:=1 to 5 do
  begin
   CILINDROS[i]:='';
  end;
  cadena:=edit1.text;
  if(length(cadena)>20)or(length(cadena)<2)
   or(cadena[1]='+')or(cadena[1]='-')or(cadena[1]=')') then
    begin
     showmessage('corregir secuencia');
     w:=0;
     form3.edit1.setfocus;
    end;
  for i:=1 to length(cadena)do
   begin
    case(cadena[i]) of
     'A':begin
          Ap:=Ap+1;
          if(ap=1) then
           begin
            nj:=nj+1;
            cilindros[nj]:='A';
            Cadena2[u]:='A+';
            u:=u+1;
            Cadena2[u]:='A-';
            u:=u+1;
            cadena3[o]:='a0  a1';
            o:=o+1;
           end;
          if (ap=1) and(cadena[i+1]='-')then
           begin
```

```
  showmessage('el primer estado de A debe ser positivo');
  w:=0;
  break;
  form3.edit1.setfocus;
 end;
if (sa=1)and(ma=0)then
 begin
  sa:=0;
  showmessage('corregir secuencia');
  w:=0;
  break;
  form3.edit1.setfocus;
 end;
if(pa=1)and(sa=0) then
 begin
  sa:=1;
 end;
 if(Ap>2)then
 begin
  Ap:=0;
  showmessage('corregir secuencia');
  w:=0;
  break;
  form3.edit1.setfocus;
 end;
If(i=length(cadena)) then
 begin
  showmessage('falta signo de A');
  w:=0;
  break;
  form3.edit1.setfocus;
 end;
if(cadena[i+1]<>'+')and(cadena[i+1]<>'-') then
 begin
  showmessage('corregir secuencia');
  w:=0;
  break;
  form3.edit1.setfocus;
 end;
if(cadena[i+1]='-')then
 begin
  An:=An+1;
  if(An>1) then
   begin
    An:=0;
    showmessage('estado negativo  de A repetido');
    w:=0;
    break;
    form3.edit1.setfocus;
   end;
 end;
```

```
      if(cadena[i+1]='+')then
       begin
        Am:=Am+1;
        if(Am>1) then
         begin
          Am:=0;
          showmessage('estado positivo de A repetido');
          w:=0;
          break;
          form3.edit1.setfocus;
         end;
       end;
      end;
'B':begin
     Bp:=Bp+1;
      if(bp=1) then
       begin
        nj:=nj+1;
        cilindros[nj]:='B';
        Cadena2[u]:='B+';
        u:=u+1;
        Cadena2[u]:='B-';
        u:=u+1;
        cadena3[o]:='b0  b1';
        o:=o+1;
       end;
      if (bp=1) and(cadena[i+1]='-')then
       begin
        showmessage('el primer estado de B debe ser positivo');
        w:=0;
        break;
        form3.edit1.setfocus;
       end;
      if (sb=1)and(mb=0) then
       begin
        sb:=0;
        showmessage('corregir secuencia');
        w:=0;
        break;
        form3.edit1.setfocus;
       end;
      if(pa=1)and(sb=0) then
       begin
        sb:=1;
       end;
      if(Bp>2)then
       begin
        Bp:=0;
        showmessage('corregir secuencia');
        w:=0;
        break;
```

```
        form3.edit1.setfocus;
       end;
      If(i=length(cadena)) then
       begin
        showmessage('falta un signo de B');
        w:=0;
        break;
        form3.edit1.setfocus;
       end;
      if(cadena[i+1]<>'+')and(cadena[i+1]<>'-') then
       begin
        showmessage('corregir secuencia');
        w:=0;
        break;
        form3.edit1.setfocus;
       end;
      if(cadena[i+1]='-')then
       begin
        Bn:=bn+1;
        if(Bn>1) then
         begin
          bn:=0;
          showmessage('estado negativo  de B repetido');
          w:=0;
          break;
          form3.edit1.setfocus;
         end;
       end;
      if(cadena[i+1]='+')then
       begin
        Bm:=Bm+1;
        if(Bm>1) then
         begin
          Bm:=0;
          showmessage('estado positivo de B repetido');
          w:=0;
          break;
          form3.edit1.setfocus;
         end;
       end;
     end;
  'C':begin
     Cp:=Cp+1;
      if(cp=1) then
       begin
        nj:=nj+1;
        cilindros[nj]:='C';
        Cadena2[u]:='C+';
        u:=u+1;
        Cadena2[u]:='C-';
        u:=u+1;
```

```
  cadena3[o]:='c0  c1';
  o:=o+1;
 end;
if (cp=1) and(cadena[i+1]='-')then
 begin
  showmessage('el primer estado de C debe ser positivo');
  w:=0;
  break;
  form3.edit1.setfocus;
 end;
if (sc=1)and(mc=0) then
 begin
  sc:=0;
  showmessage('corregir secuencia');
  w:=0;
  break;
  form3.edit1.setfocus;
 end;
if(pc=1)and(sc=0) then
 begin
  sc:=1;
 end;
if(Cp>2)then
 begin
  Cp:=0;
  showmessage('corregir secuencia');
  w:=0;
  break;
  form3.edit1.setfocus;
 end;
If(i=length(cadena)) then
 begin
  showmessage('falta un signo de C');
  w:=0;
  break;
  form3.edit1.setfocus;
 end;
if(cadena[i+1]<>'+')and(cadena[i+1]<>'-') then
 begin
  showmessage('corregir secuencia');
  w:=0;
  break;
  form3.edit1.setfocus;
 end;
if(cadena[i+1]='-')then
 begin
  Cn:=Cn+1;
  if(cn>1) then
   begin
    cn:=0;
    showmessage('estado negativo  de C repetido');
```

```
          w:=0;
          break;
          form3.edit1.setfocus;
         end;
       end;
      if(cadena[i+1]='+')then
       begin
        Cm:=Cm+1;
        if(cm>1) then
         begin
          cm:=0;
          showmessage('estado positivo de C repetido');
          w:=0;
          break;
          form3.edit1.setfocus;
         end;
       end;
     end;
  'D':begin
     Dp:=Dp+1;
      if(dp=1) then
       begin
        nj:=nj+1;
        cilindros[nj]:='D';
        Cadena2[u]:='D+';
        u:=u+1;
        Cadena2[u]:='D-';
        u:=u+1;
        cadena3[o]:='d0  d1';
        o:=o+1;
       end;
      if (Dp=1) and(cadena[i+1]='-')then
       begin
        showmessage('el primer estado de D debe ser positivo');
        w:=0;
        break;
        form3.edit1.setfocus;
       end;
      if (sd=1)and(md=0) then
       begin
        sd:=0;
        showmessage('corregir secuencia');
        w:=0;
        break;
        form3.edit1.setfocus;
       end;
      if(pa=1)and(sd=0) then
       begin
        sd:=1;
       end;
      if(Dp>2)then
```

```
      begin
       Dp:=0;
       showmessage('corregir secuencia');
       w:=0;
       break;
       form3.edit1.setfocus;
      end;
     If(i=length(cadena)) then
      begin
       showmessage('falta un signo de D');
       w:=0;
       break;
       form3.edit1.setfocus;
      end;
     if(cadena[i+1]<>'+')and(cadena[i+1]<>'-') then
      begin
       showmessage('corregir secuencia');
       w:=0;
       break;
       form3.edit1.setfocus;
      end;
     if(cadena[i+1]='-')then
      begin
       Dn:=Dn+1;
       if(Dn>1) then
        begin
         Dn:=0;
         showmessage('estado negativo de D repetido');
         w:=0;
         break;
         form3.edit1.setfocus;
        end;
      end;
     if(cadena[i+1]='+')then
      begin
       Dm:=Dm+1;
       if(Dm>1) then
        begin
         Dm:=0;
         showmessage('estado positivo de D repetido');
         w:=0;
         break;
         form3.edit1.setfocus;
        end;
      end;
    end;
  'E':begin
     Ep:=Ep+1;
      if(ep=1) then
       begin
        nj:=nj+1;
```

```
  cilindros[nj]:='E';
  Cadena2[u]:='E+';
  u:=u+1;
  Cadena2[u]:='E-';
  u:=u+1;
  cadena3[o]:='e0  e1';
  o:=o+1;
 end;
if (Ep=1) and(cadena[i+1]='-')then
 begin
  showmessage('el primer estado de E debe ser positivo');
  w:=0;
  break;
  form3.edit1.setfocus;
 end;
if (se=1)and(me=0) then
 begin
  se:=0;
  showmessage('corregir secuencia');
  w:=0;
  break;
  form3.edit1.setfocus;
 end;
if(pa=1)and(se=0) then
 begin
  se:=1;
 end;
if(Ep>2)then
 begin
  Ep:=0;
  showmessage('corregir secuencia');
  w:=0;
  break;
  form3.edit1.setfocus;
 end;
If(i=length(cadena)) then
 begin
  showmessage('falta un signo de E');
  w:=0;
  break;
  form3.edit1.setfocus;
 end;
if(cadena[i+1]<>'+')and(cadena[i+1]<>'-') then
 begin
  showmessage('corregir secuencia');
  w:=0;
  break;
  form3.edit1.setfocus;
 end;
if(cadena[i+1]='-')then
 begin
```

```
       En:=En+1;
       if(En>1) then
        begin
         En:=0;
         showmessage('estado negativo  de E repetido');
         w:=0;
         break;
         form3.edit1.setfocus;
        end;
      end;
     if(cadena[i+1]='+')then
      begin
       Em:=Em+1;
       if(Em>1) then
        begin
         Em:=0;
         showmessage('estado positivo de E repetido');
         w:=0;
         break;
         form3.edit1.setfocus;
        end;
      end;
    end;
'+','-':begin
     if(i=length(cadena))and((pc=0)or(pc=2))and(pa=1)then
      begin
        showmessage('corregir secuencia');
        w:=0;
        break;
        form3.edit1.setfocus;
      end;
     if(cadena[i+1]='+')or(cadena[i+1]='-') then
      begin
        showmessage('corregir secuencia');
        w:=0;
        break;
        form3.edit1.setfocus;
      end;
     if(i=length(cadena))and((ap=1)or(bp=1)or(cp=1)or(dp=1)or(ep=1)) then
      begin
       showmessage('cada letra debe tener sus dos estados');
       w:=0;
       break;
       form3.edit1.setfocus;
      end;

    end;
'(':begin
     pa:=1;
     if(i=length(cadena))then
      begin
```

```
        showmessage('corregir secuencia');
        w:=0;
        break;
        form3.edit1.setfocus;
       end;
      if(pc=0)then
       begin
        pa:=0;
        showmessage('no anidar parentesis');
        w:=0;
        break;
        form3.edit1.setfocus;
       end;
      if(cadena[i+1]=')')or(cadena[i+1]='(')
      or(cadena[i+1]='+')or(cadena[i+1]='-')  then
       begin
        Pa:=0;
        showmessage('corregir secuencia');
        w:=0;
        break;
        form3.edit1.setfocus;
       end;
     pc:=0;
     end;
  ')':begin
      pc:=1;
      if(i=length(cadena))and((ap=1)or(bp=1)or(cp=1)or(dp=1)or(ep=1)) then
       begin
        showmessage('cada letra debe tener sus dos estados');
        w:=0;
        break;
        form3.edit1.setfocus;
       end;
      if(sa=1) then
        begin
         ma:=1;
        end;
      if(se=1) then
        begin
         me:=1;
        end;
      if(sb=1) then
        begin
         mb:=1;
        end;
      if(sc=1) then
        begin
         mc:=1;
        end;
      if(sd=1) then
        begin
```

```
          md:=1;
         end;
        if(pa=0) then
         begin
          pc:=2;
          showmessage('corregir secuencia');
          w:=0;
          break;
          form3.edit1.setfocus;
         end;
        if(cadena[i+1]=')')or(cadena[i+1]='+')
        or(cadena[i+1]='-') then
         begin
          pc:=2;
          showmessage('corregir secuencia');
          w:=0;
          break;
          form3.edit1.setfocus;
         end;
        pa:=0;
       end;
   end;{case}
  end;{for}
if(w=1)then
 begin
  showmessage('Secuencia correcta');
  cc:=(ap+bp+cp+dp+ep) div 2;
  if copiapantalla3.TipoCircuito=1 then
   begin
    stringgrid1.cells[0,0]:='Grupos';
    stringgrid1.cells[1,0]:='Secuencia';
    for i:=1 to 10 do
     begin
      stringgrid1.cells[0,i]:=inttostr(i);
     end;
    j:=1;
    for i:=1 to length(edit1.text) do
     begin
      if( i mod 2) = 1  then begin
       stringgrid1.cells[1,j]:=edit1.text[i]+edit1.text[i+1];
       j:=j+1;
      end;
     end;
   end;
   bitbtn2.click;
 end;
end;
```

GRAFICA_CILINDROS

Dibuja los cilindros, las memorias y sus conexiones, recibe el valor global cc de la unidad 3 que indica la cantidad de cilindros para el circuito, se ejecuta cuando es llamado para el diseño de cada una de las 4 clases de circuitos.

CODIGO DE GRAFICA_CILINDROS

```
procedure graficacilindros;
 begin
 form4.ClientWidth:=780;
 X:=60;j:=1;y:=30;
 for i:=1 to cc do
  begin
   with form4.IMAGE1.Canvas do begin
    pen.Width:=1;
    pen.style:=pssolid;
    pen.color:=clblue;
    Brush.Color:=clwhite;
    {textOut(x+18,y-15,unit3.cilindros[i]); }
    TextOut(x+44,y-5,unit3.cadena3[i] );
    TextOut(x-15,54,unit3.cadena2[J] );
    TextOut(x+40,54,unit3.cadena2[2*i] );
    j:=1+2*i;
    MoveTo(x,y);
    LineTo(x+40,y);
    MoveTo(x+40,y);
    LineTo(x+40,y+8);
    MoveTo(x+40,y+12);
    LineTo(x+40,y+20);
    MoveTo(x+40,y+20);
    LineTo(x,y+20);
    moveto(x,y+20);
    lineto(x,y);
    (*lineas medias*)
    MoveTo(x+10,y+8);
    LineTO(x+45,y+8);
    MoveTo(x+10,y+12);
    LineTo(x+45,y+12);
    MoveTo(x+6,y);
    LineTo(x+6,y+20);
    MoveTo(x+10,y);
    LineTo(x+10,y+8);
    MoveTo(x+10,y+12);
    LineTo(x+10,y+20);
    Moveto(x+45,y+8);
    LineTo(x+45,y+12);
    pen.color:=clred;

    {conexiones a memorias}
```

```
   Moveto(x+5,y+20);
   Lineto(x+5,y+30);
   Moveto(x+5,y+30);
   Lineto(x+25,y+30);
   Moveto(x+25,y+30);
   Lineto(x+25,y+40);
   Moveto(x+35,y+20);
   Lineto(x+35,y+40);
   {memorias}
   pen.color:=clblue;
   rectangle(x,70,x+40,90);
   Moveto(x+5,70);
   Lineto(x+5,90);
   Moveto(x+20,70);
   Lineto(x+20,90);
   Moveto(x+25,70);
   Lineto(x+35,90);
   Moveto(x+25,90);
   Lineto(x+35,70);
   Moveto(x+35,70);
   Lineto(x+31,73);
   Moveto(x+35,70);
   Lineto(x+35,75);
   Moveto(x+35,90);
   Lineto(x+31,87);
   Moveto(x+35,90);
   Lineto(x+35,85);
   Moveto(x+5,70);
   Lineto(x+2,74);
   Moveto(x+5,70);
   Lineto(x+8,74);
   Moveto(x+15,90);
   Lineto(x+12,86);
   Moveto(x+15,90);
   Lineto(x+18,86);
   Moveto(x+15,70);
   Lineto(x+15,90);
   Moveto(x+25,90);
   Lineto(x+25,93);
   ellipse(x+20,93,x+30,103);
   Moveto(x+24,98);
   Lineto(x+26,98);
  end;
   x:=x+120;
 end;
end;
```

INTEGRADOS

Realiza las gráficas de los circuitos integrados pequeños que contienen las válvulas selectora, simultaneidad y la 3/2 normalmente cerrada. Toma de la unidad 3 la variable ma-1 que corresponde al número de grupos para luego diseñar los integrados con la fórmula:

Nintegrados=#Grupos-1;

Se ejecuta cuando es llamado desde el procedimiento estado1.

CODIGO DE INTEGRADOS

```
Procedure integrados;
begin
  Y:=10;
  z:=15*(unit3.ma-1)-25;
  x1:=-2;
  FORM4.image1.Canvas.PEN.COLOR:=CLRED;
  {FORM4.image1.CANVAS.Moveto(85+x1,195+z);
  FORM4.CANVAS.LineTo(85+x1,215+z);}
  for i:=1 to unit3.ma-1  do begin
   with form4.image1.canvas do begin
   Pen.COLOR:=CLBLACK;
   {valvula selectora}
   if i<(unit3.ma-1) then
   begin
    if i = (length(unit3.form3.edit1.text)div 2) then
     BEGIN
       {Moveto(67+x1,183+z);
       Lineto(67+x1,180+z); }
     END;
    Moveto(40+x1,170+z);
    lineto(50+x1,170+z);
    moveto(50+x1,170+z);
    Lineto(50+x1,180+z);
    Moveto(50+x1,180+z);
    Lineto(40+x1,180+z);
    Moveto(40+x1,180+z);
    Lineto(40+x1,170+z);
    Moveto(45+x1,172+z);
    Lineto(45+x1,178+z);
    Moveto(42+x1,172+z);
    Lineto(48+x1,172+z);
    Moveto(42+x1,178+z);
    Lineto(48+x1,178+z);
```

```
 Moveto(40+x1,174+z);
 Lineto(43+x1,174+z);
 Moveto(40+x1,176+z);
 Lineto(43+x1,176+z);
 Moveto(47+x1,176+z);
 Lineto(50+x1,176+z);
 Moveto(47+x1,174+z);
 Lineto(50+x1,174+z);
 {flechas}
 Moveto(50+x1,175+z);
 Lineto(55+x1,175+z);
 Moveto(55+x1,175+z);
 lineto(52+x1,174+z);
 Moveto(55+x1,175+z);
 Lineto(52+x1,176+z);
 Moveto(75+x1,175+z);
 Lineto(80+x1,175+z);
 Moveto(75+x1,175+z);
 Lineto(78+x1,174+z);
 Moveto(75+x1,175+z);
 Lineto(78+x1,176+z);
{valvula2}
 ellipse(65+x1,183+z,70+x1,188+z);
 ellipse(67+x1,185+z,68+x1,186+z);
 Moveto(55+x1,170+z);
 Lineto(75+x1,170+z);
 Moveto(75+x1,170+z);
 Lineto(75+x1,180+z);
 Moveto(75+x1,180+z);
 Lineto(55+x1,180+z);
 Moveto(55+x1,180+z);
 Lineto(55+x1,170+z);
 Moveto(65+x1,170+z);
 Lineto(65+x1,180+z);
 {lineas intermedias}
 Moveto(57+x1,170+z);
 Lineto(57+x1,180+z);
 Moveto(67+x1,170+z);
 Lineto(73+x1,180+z);
  {flechas interm}
 Moveto(57+x1,170+z);
 Lineto(56+x1,173+z);
 Moveto(57+x1,170+z);
 Lineto(58+x1,173+z);
 {}
 Moveto(73+x1,180+z);
 Lineto(73+x1,177+z);
 Moveto(73+x1,180+z);
 Lineto(70+x1,176+z);
 {}
 Moveto(63+x1,180+z);
```

```
 Lineto(63+x1,177+z);
 Moveto(62+x1,177+z);
 Lineto(64+x1,177+z);
 Moveto(66+x1,177+z);
 Lineto(69+x1,177+z);
 Moveto(67+x1,180+z);
 Lineto(67+x1,177+z);
 {Valvula}
 Moveto(80+x1,170+z);
 lineto(90+x1,170+z);
 moveto(90+x1,170+z);
 Lineto(90+x1,180+z);
 Moveto(90+x1,180+z);
 Lineto(80+x1,180+z);
 Moveto(80+x1,180+z);
 Lineto(80+x1,170+z);
 {lineas intermedias}
 Moveto(85+x1,170+z);
 Lineto(85+x1,172+z);
 Moveto(85+x1,180+z);
 Lineto(85+x1,178+z);
 Moveto(85+x1,178+z);
 Lineto(84+x1,176+z);
 Moveto(85+x1,178+z);
 Lineto(86+x1,176+z);
 Moveto(85+x1,172+z);
 Lineto(84+x1,175+z);
 Moveto(85+x1,172+z);
 Lineto(86+x1,175+z);
 Moveto(85+x1,176+z);
 Lineto(86+x1,176+z);
 {conexiones de integrados}
 pen.color:=clred;
 Moveto(67+x1,135+y);
 Lineto(67+x1,170+z);
 MoveTo(85+x1,150+y);
 Lineto(85+x1,170+z);
 MoveTo(45+x1,155+z);
 Lineto(45+x1,170+z);
 Moveto(85+x1,195+z);
 LineTo(85+x1,180+z);
end;
if i>1 then begin
 pen.color:=clred;
 Moveto(45+x1,120+y);
 Lineto(45+x1,170+z);
 Moveto(85,195+z);
 LineTo(85+x1,195+z);
 if i<(unit3.ma-1) then
  begin
    pen.color:=clblack;
```

```
    Moveto(67+x1,183+z);
    Lineto(67+x1,180+z);
  end;
end;
if i=(unit3.ma-1) then
begin
 {selectora}
 pen.color:=clwhite;
 Moveto(60+x1,195+z);
 LineTo(85+x1,195+z);
 pen.color:=clblack;
 Moveto(40+x1,170+z);
 lineto(50+x1,170+z);
 moveto(50+x1,170+z);
 Lineto(50+x1,180+z);
 Moveto(50+x1,180+z);
 Lineto(40+x1,180+z);
 Moveto(40+x1,180+z);
 Lineto(40+x1,170+z);
 Moveto(45+x1,172+z);
 Lineto(45+x1,178+z);
 Moveto(42+x1,172+z);
 Lineto(48+x1,172+z);
 Moveto(42+x1,178+z);
 Lineto(48+x1,178+z);
 Moveto(40+x1,174+z);
 Lineto(43+x1,174+z);
 Moveto(40+x1,176+z);
 Lineto(43+x1,176+z);
 Moveto(47+x1,176+z);
 Lineto(50+x1,176+z);
 Moveto(47+x1,174+z);
 Lineto(50+x1,174+z);
 {Valvula}
 Moveto(55+x1,170+z);
 lineto(65+x1,170+z);
 moveto(65+x1,170+z);
 Lineto(65+x1,180+z);
 Moveto(65+x1,180+z);
 Lineto(55+x1,180+z);
 Moveto(55+x1,180+z);
 Lineto(55+x1,170+z);
 {lineas intermedias}
 Moveto(60+x1,170+z);
 Lineto(60+x1,172+z);
 Moveto(60+x1,180+z);
 Lineto(60+x1,178+z);
 Moveto(60+x1,178+z);
 Lineto(59+x1,176+z);
 Moveto(60+x1,178+z);
 Lineto(61+x1,176+z);
```

```
 Moveto(60+x1,172+z);
 Lineto(59+x1,175+z);
 Moveto(60+x1,172+z);
 Lineto(61+x1,175+z);
 Moveto(60+x1,176+z);
 Lineto(61+x1,176+z);
 {Flechas}
 Moveto(65+x1,175+z);
 Lineto(70+x1,175+z);
 Moveto(70+x1,175+z);
 lineto(67+x1,174+z);
 Moveto(70+x1,175+z);
 Lineto(67+x1,176+z);

 Moveto(50+x1,175+z);
 Lineto(53+x1,175+z);
 Moveto(53+x1,175+z);
 lineto(53+x1,165+z);
 Moveto(53+x1,165+z);
 Lineto(60+x1,165+z);
 Moveto(60+x1,165+z);
 Lineto(60+x1,170+z);
 Moveto(60+x1,170+z);
 Lineto(59+x1,167+z);
 Moveto(60+x1,170+z);
 Lineto(61+x1,167+z);
{valvula2}
 Moveto(70+x1,170+z);
 Lineto(90+x1,170+z);
 Moveto(90+x1,170+z);
 Lineto(90+x1,180+z);
 Moveto(90+x1,180+z);
 Lineto(70+x1,180+z);
 Moveto(70+x1,180+z);
 Lineto(70+x1,170+z);
 Moveto(80+x1,170+z);
 Lineto(80+x1,180+z);
 {lineas intermedias}
 Moveto(72+x1,170+z);
 Lineto(72+x1,180+z);
 Moveto(82+x1,170+z);
 Lineto(88+x1,180+z);
  {flechas interm}
 Moveto(72+x1,170+z);
 Lineto(71+x1,173+z);
 Moveto(72+x1,170+z);
 Lineto(73+x1,173+z);
 {}
 Moveto(88+x1,180+z);
 Lineto(88+x1,177+z);
 Moveto(88+x1,180+z);
```

```
   Lineto(85+x1,176+z);
   {}
   Moveto(78+x1,180+z);
   Lineto(78+x1,177+z);
   Moveto(77+x1,177+z);
   Lineto(79+x1,177+z);
   Moveto(81+x1,177+z);
   Lineto(84+x1,177+z);
   Moveto(82+x1,180+z);
   Lineto(82+x1,177+z);
  {conexiones de integrados}
   pen.color:=clred;
   Moveto(95+x1,145);
   Lineto(95+x1,175+z);
   Moveto(95+x1,175+z);
   Lineto(90+x1,175+z);
   Moveto(92+x1,173+z);
   Lineto(90+x1,175+z);
   Moveto(92+x1,177+z);
   Lineto(90+x1,175+z);
   Moveto(95+x1,135+y-15*unit3.cc);
   LineTo(90+x1,135+y-15*unit3.cc);
   Moveto(60+x1,195+z);
   LineTo(60+x1,180+z);
   pen.color:=clblack;
   Moveto(72+x1,180+z);
   Lineto(72+x1,183+z);
   Moveto(65,180+z);
   Lineto(65,183+z);
   ellipse(70+x1,183+z,75+x1,188+z);
  end;
  end;
  x1:=x1+105;
  y:=y+15;
  end;
end;
```

SIMULAR POR SECUENCIA

Este procedimiento realiza la simulación de la entrada y la salida de los cilindros, toma como referencia la secuencia de entrada capturada en edit1 de la unidad 3.Se ejecuta cuando se pulsa la opción simular del menú principal de la pantalla de diseño por secuencia.

CODIGO DE SIMULAR

```
procedure TForm4.Empezar1Click(Sender: TObject);
var
p,ax,bx,cx,dx,ex,I,J,X,Y:INTEGER;
JI,ci,r:longint;
begin

form4.refresh;
r:=1000000;
J:=1;I:=1;x:=60;y:=30;
 for i:=1 to length(unit3.form3.Edit1.text) do begin
  case (unit3.form3.edit1.text[i]) of
  'A': begin
       if (unit3.form3.edit1.text[i+1]='+') then
       begin
        p:=0;
        ax:=x;
        for ci:=1 to 30 do begin
        with form4.Canvas do begin
         for Ji:=1 to r do begin
         end;
         pen.Width:=1;
         pen.style:=pssolid;
         pen.color:=clblue;
         Brush.Color:=clwhite;
         {textOut(x+18,y-15,unit3.cilindros[i]);
         TextOut(x+44,y-5,unit3.cadena3[i] );
         TextOut(x-15,54,unit3.cadena2[J] );
         TextOut(x+40,54,unit3.cadena2[2*i] );  }
         MoveTo(ax,y);
         LineTo(ax+40,y);
         MoveTo(ax+40,y);
         LineTo(ax+40,y+8);
         MoveTo(ax+40,y+12);
         LineTo(ax+40,y+20);
         MoveTo(ax+40,y+20);
         LineTo(ax,y+20);
         moveto(ax,y+20);
         lineto(ax,y);
         {pen.color:=clpurple;}
         (*lineas medias*)
         MoveTo(ax+10+ci,y+8);
         LineTO(ax+45+ci,y+8);
         MoveTo(ax+10+ci,y+12);
         LineTo(ax+45+ci,y+12);
         MoveTo(ax+6+ci,y);
```

```
LineTo(ax+6+ci,y+20);
MoveTo(ax+10+ci,y);
LineTo(ax+10+ci,y+8);
MoveTo(ax+10+ci,y+12);
LineTo(ax+10+ci,y+20);
Moveto(ax+45+ci,y+8);
LineTo(ax+45+ci,y+12);
pen.color:=clwhite;
Moveto(ax+6+ci,y);
Lineto(ax+6+ci,y+20);
{conexiones a memorias}
pen.color:=cllime;
Moveto(ax+5,y+20);
Lineto(ax+5,y+30);
Moveto(ax+5,y+30);
Lineto(ax+25,y+30);
Moveto(ax+25,y+30);
Lineto(ax+25,y+40);
Moveto(ax+35,y+20);
Lineto(ax+35,y+40);
{memorias}
pen.color:=clblue;
rectangle(ax,70,ax+40,90);
Moveto(ax+5,70);
Lineto(ax+5,90);
Moveto(ax+20,70);
Lineto(ax+20,90);
Moveto(ax+25,70);
Lineto(ax+35,90);
Moveto(ax+25,90);
Lineto(ax+35,70);
Moveto(ax+35,70);
Lineto(ax+31,73);
Moveto(ax+35,70);
Lineto(ax+35,75);
Moveto(ax+35,90);
Lineto(ax+31,87);
Moveto(ax+35,90);
Lineto(ax+35,85);
Moveto(ax+5,70);
Lineto(ax+2,74);
Moveto(ax+5,70);
Lineto(ax+8,74);
Moveto(ax+15,90);
Lineto(ax+12,86);
Moveto(ax+15,90);
Lineto(ax+18,86);
Moveto(ax+15,70);
Lineto(ax+15,90);
Moveto(ax+25,90);
Lineto(ax+25,93);
```

```
  ellipse(ax+20,93,ax+30,103);
  Moveto(ax+24,98);
  Lineto(ax+26,98);
 end;
 end;
end;

if (unit3.form3.edit1.text[i+1]='-') then
begin
 p:=1;
 for ci:=30 DOWNTO 1 do begin
 with form4.Canvas do begin
  for Ji:=1 to r do begin
  end;
  pen.Width:=1;
  pen.style:=pssolid;
  pen.color:=clblue;
  Brush.Color:=clwhite;
  {textOut(x+18,y-15,unit3.cilindros[i]);
  TextOut(x+44,y-5,unit3.cadena3[i] );
  TextOut(x-15,54,unit3.cadena2[J] );
  TextOut(x+40,54,unit3.cadena2[2*i] );}
  MoveTo(ax,y);
  LineTo(ax+40,y);
  MoveTo(ax+40,y);
  LineTo(ax+40,y+8);
  MoveTo(ax+40,y+12);
  LineTo(ax+40,y+20);
  MoveTo(ax+40,y+20);
  LineTo(ax,y+20);
  moveto(ax,y+20);
  lineto(ax,y);
  {pen.color:=clpurple;}
  (*lineas medias*)
  MoveTo(ax+10+ci,y+8);
  LineTO(ax+45+ci,y+8);
  MoveTo(ax+10+ci,y+12);
  LineTo(ax+45+ci,y+12);
  MoveTo(ax+6+ci,y);
  LineTo(ax+6+ci,y+20);
  MoveTo(ax+10+ci,y);
  LineTo(ax+10+ci,y+8);
  MoveTo(ax+10+ci,y+12);
  LineTo(ax+10+ci,y+20);
  Moveto(ax+45+ci,y+8);
  LineTo(ax+45+ci,y+12);
  pen.color:=clwhite;
  Moveto(ax+50+ci,y+7);
  Lineto(ax+50+ci,y+13);
  Moveto(ax+10+ci,y);
  Lineto(ax+10+ci,y+8);
```

```
   MoveTo(ax+10+ci,y+12);
   LineTo(ax+10+ci,y+20);
   {conexiones a memorias}
   pen.color:=clred;
   Moveto(ax+5,y+20);
   Lineto(ax+5,y+30);
   Moveto(ax+5,y+30);
   Lineto(ax+25,y+30);
   Moveto(ax+25,y+30);
   Lineto(ax+25,y+40);
   Moveto(ax+35,y+20);
   Lineto(ax+35,y+40);
   {memorias}
   pen.color:=clblue;
   rectangle(ax,70,ax+40,90);
   Moveto(ax+5,70);
   Lineto(ax+5,90);
   Moveto(ax+20,70);
   Lineto(ax+20,90);
   Moveto(ax+25,70);
   Lineto(ax+35,90);
   Moveto(ax+25,90);
   Lineto(ax+35,70);
   Moveto(ax+35,70);
   Lineto(ax+31,73);
   Moveto(ax+35,70);
   Lineto(ax+35,75);
   Moveto(ax+35,90);
   Lineto(ax+31,87);
   Moveto(ax+35,90);
   Lineto(ax+35,85);
   Moveto(ax+5,70);
   Lineto(ax+2,74);
   Moveto(ax+5,70);
   Lineto(ax+8,74);
   Moveto(ax+15,90);
   Lineto(ax+12,86);
   Moveto(ax+15,90);
   Lineto(ax+18,86);
   Moveto(ax+15,70);
   Lineto(ax+15,90);
   Moveto(ax+25,90);
   Lineto(ax+25,93);
   ellipse(ax+20,93,ax+30,103);
   Moveto(ax+24,98);
   Lineto(ax+26,98);
  end;
  end;
 end;
if p=0 then
 x:=x+120;
```

```
    end;
'B': begin
     if (unit3.form3.edit1.text[i+1]='+') then
     begin
      p:=0;
      bx:=x;
      for ci:=1 to 30 do begin
      with form4.Canvas do begin
       for Ji:=1 to r do begin
       end;
       pen.Width:=1;
       pen.style:=pssolid;
       pen.color:=clblue;
       Brush.Color:=clwhite;
       {textOut(x+18,y-15,unit3.cilindros[i]);
       TextOut(x+44,y-5,unit3.cadena3[i] );
       TextOut(x-15,54,unit3.cadena2[J] );
       TextOut(x+40,54,unit3.cadena2[2*i] );}
       MoveTo(bx,y);
       LineTo(bx+40,y);
       MoveTo(bx+40,y);
       LineTo(bx+40,y+8);
       MoveTo(bx+40,y+12);
       LineTo(bx+40,y+20);
       MoveTo(bx+40,y+20);
       LineTo(bx,y+20);
       moveto(bx,y+20);
       lineto(bx,y);
       {pen.color:=clpurple;}
       (*lineas medias*)
       MoveTo(bx+10+ci,y+8);
       LineTO(bx+45+ci,y+8);
       MoveTo(bx+10+ci,y+12);
       LineTo(bx+45+ci,y+12);
       MoveTo(bx+6+ci,y);
       LineTo(bx+6+ci,y+20);
       MoveTo(bx+10+ci,y);
       LineTo(bx+10+ci,y+8);
       MoveTo(bx+10+ci,y+12);
       LineTo(bx+10+ci,y+20);
       Moveto(bx+45+ci,y+8);
       LineTo(bx+45+ci,y+12);
       pen.color:=clwhite;
       Moveto(bx+6+ci,y);
       Lineto(bx+6+ci,y+20);
       {conexiones a memorias}
       pen.color:=cllime;
       Moveto(bx+5,y+20);
       Lineto(bx+5,y+30);
       Moveto(bx+5,y+30);
       Lineto(bx+25,y+30);
```

```
  Moveto(bx+25,y+30);
  Lineto(bx+25,y+40);
  Moveto(bx+35,y+20);
  Lineto(bx+35,y+40);
  {memorias}
  pen.color:=clblue;
  rectangle(bx,70,bx+40,90);
  Moveto(bx+5,70);
  Lineto(bx+5,90);
  Moveto(bx+20,70);
  Lineto(bx+20,90);
  Moveto(bx+25,70);
  Lineto(bx+35,90);
  Moveto(bx+25,90);
  Lineto(bx+35,70);
  Moveto(bx+35,70);
  Lineto(bx+31,73);
  Moveto(bx+35,70);
  Lineto(bx+35,75);
  Moveto(bx+35,90);
  Lineto(bx+31,87);
  Moveto(bx+35,90);
  Lineto(bx+35,85);
  Moveto(bx+5,70);
  Lineto(bx+2,74);
  Moveto(bx+5,70);
  Lineto(bx+8,74);
  Moveto(bx+15,90);
  Lineto(bx+12,86);
  Moveto(bx+15,90);
  Lineto(bx+18,86);
  Moveto(bx+15,70);
  Lineto(bx+15,90);
  Moveto(bx+25,90);
  Lineto(bx+25,93);
  ellipse(bx+20,93,bx+30,103);
  Moveto(bx+24,98);
  Lineto(bx+26,98);
 end;
 end;
end;

if (unit3.form3.edit1.text[i+1]='-') then
begin
 p:=1;
 for ci:=30 DOWNTO 1 do begin
 with form4.Canvas do begin
  for Ji:=1 to r do begin
  end;
  pen.Width:=1;
  pen.style:=pssolid;
```

```
pen.color:=clblue;
Brush.Color:=clwhite;
{textOut(x+18,y-15,unit3.cilindros[i]);
TextOut(x+44,y-5,unit3.cadena3[i] );
TextOut(x-15,54,unit3.cadena2[J] );
TextOut(x+40,54,unit3.cadena2[2*i] ); }
MoveTo(bx,y);
LineTo(bx+40,y);
MoveTo(bx+40,y);
LineTo(bx+40,y+8);
MoveTo(bx+40,y+12);
LineTo(bx+40,y+20);
MoveTo(bx+40,y+20);
LineTo(bx,y+20);
moveto(bx,y+20);
lineto(bx,y);
{pen.color:=clpurple;}
(*lineas medias*)
MoveTo(bx+10+ci,y+8);
LineTO(bx+45+ci,y+8);
MoveTo(bx+10+ci,y+12);
LineTo(bx+45+ci,y+12);
MoveTo(bx+6+ci,y);
LineTo(bx+6+ci,y+20);
MoveTo(bx+10+ci,y);
LineTo(bx+10+ci,y+8);
MoveTo(bx+10+ci,y+12);
LineTo(bx+10+ci,y+20);
Moveto(bx+45+ci,y+8);
LineTo(bx+45+ci,y+12);
pen.color:=clwhite;
Moveto(bx+50+ci,y+7);
Lineto(bx+50+ci,y+13);
Moveto(bx+10+ci,y);
Lineto(bx+10+ci,y+8);
MoveTo(bx+10+ci,y+12);
LineTo(bx+10+ci,y+20);
{conexiones a memorias}
pen.color:=clred;
Moveto(bx+5,y+20);
Lineto(bx+5,y+30);
Moveto(bx+5,y+30);
Lineto(bx+25,y+30);
Moveto(bx+25,y+30);
Lineto(bx+25,y+40);
Moveto(bx+35,y+20);
Lineto(bx+35,y+40);
{memorias}
pen.color:=clblue;
rectangle(bx,70,bx+40,90);
Moveto(bx+5,70);
```

```
          Lineto(bx+5,90);
          Moveto(bx+20,70);
          Lineto(bx+20,90);
          Moveto(bx+25,70);
          Lineto(bx+35,90);
          Moveto(bx+25,90);
          Lineto(bx+35,70);
          Moveto(bx+35,70);
          Lineto(bx+31,73);
          Moveto(bx+35,70);
          Lineto(bx+35,75);
          Moveto(bx+35,90);
          Lineto(bx+31,87);
          Moveto(bx+35,90);
          Lineto(bx+35,85);
          Moveto(bx+5,70);
          Lineto(bx+2,74);
          Moveto(bx+5,70);
          Lineto(bx+8,74);
          Moveto(bx+15,90);
          Lineto(bx+12,86);
          Moveto(bx+15,90);
          Lineto(bx+18,86);
          Moveto(bx+15,70);
          Lineto(bx+15,90);
          Moveto(bx+25,90);
          Lineto(bx+25,93);
          ellipse(bx+20,93,bx+30,103);
          Moveto(bx+24,98);
          Lineto(bx+26,98);
         end;
         end;
        end;
       if p=0 then
       x:=x+120;
       end;
  'C': begin
        if (unit3.form3.edit1.text[i+1]='+') then
        begin
         p:=0;
         Cx:=x;
         for ci:=1 to 30 do begin
         with form4.Canvas do begin
          for Ji:=1 to r do begin
          end;
          pen.Width:=1;
          pen.style:=pssolid;
          pen.color:=clblue;
          Brush.Color:=clwhite;
          {textOut(x+18,y-15,unit3.cilindros[i]);
          TextOut(x+44,y-5,unit3.cadena3[i] );
```

```
TextOut(x-15,54,unit3.cadena2[J] );
TextOut(x+40,54,unit3.cadena2[2*i] ); }
MoveTo(Cx,y);
LineTo(cx+40,y);
MoveTo(cx+40,y);
LineTo(cx+40,y+8);
MoveTo(cx+40,y+12);
LineTo(cx+40,y+20);
MoveTo(cx+40,y+20);
LineTo(cx,y+20);
moveto(cx,y+20);
lineto(cx,y);
{pen.color:=clpurple;}
(*lineas medias*)
MoveTo(cx+10+ci,y+8);
LineTO(cx+45+ci,y+8);
MoveTo(cx+10+ci,y+12);
LineTo(cx+45+ci,y+12);
MoveTo(cx+6+ci,y);
LineTo(cx+6+ci,y+20);
MoveTo(cx+10+ci,y);
LineTo(cx+10+ci,y+8);
MoveTo(cx+10+ci,y+12);
LineTo(cx+10+ci,y+20);
Moveto(cx+45+ci,y+8);
LineTo(cx+45+ci,y+12);
pen.color:=clwhite;
Moveto(cx+6+ci,y);
Lineto(cx+6+ci,y+20);
{conexiones a memorias}
pen.color:=cllime;
Moveto(cx+5,y+20);
Lineto(cx+5,y+30);
Moveto(cx+5,y+30);
Lineto(cx+25,y+30);
Moveto(cx+25,y+30);
Lineto(cx+25,y+40);
Moveto(cx+35,y+20);
Lineto(cx+35,y+40);
{memorias}
pen.color:=clblue;
rectangle(cx,70,cx+40,90);
Moveto(cx+5,70);
Lineto(cx+5,90);
Moveto(cx+20,70);
Lineto(cx+20,90);
Moveto(cx+25,70);
Lineto(cx+35,90);
Moveto(cx+25,90);
Lineto(cx+35,70);
Moveto(cx+35,70);
```

```
  Lineto(cx+31,73);
  Moveto(cx+35,70);
  Lineto(cx+35,75);
  Moveto(cx+35,90);
  Lineto(cx+31,87);
  Moveto(cx+35,90);
  Lineto(cx+35,85);
  Moveto(cx+5,70);
  Lineto(cx+2,74);
  Moveto(cx+5,70);
  Lineto(cx+8,74);
  Moveto(cx+15,90);
  Lineto(cx+12,86);
  Moveto(cx+15,90);
  Lineto(cx+18,86);
  Moveto(cx+15,70);
  Lineto(cx+15,90);
  Moveto(cx+25,90);
  Lineto(cx+25,93);
  ellipse(cx+20,93,cx+30,103);
  Moveto(cx+24,98);
  Lineto(cx+26,98);
 end;
 end;
end;

if (unit3.form3.edit1.text[i+1]='-') then
begin
 p:=1;
 for ci:=30 DOWNTO 1 do begin
 with form4.Canvas do begin
  for Ji:=1 to r do begin
  end;
  pen.Width:=1;
  pen.style:=pssolid;
  pen.color:=clblue;
  Brush.Color:=clwhite;
  {textOut(x+18,y-15,unit3.cilindros[i]);
  TextOut(x+44,y-5,unit3.cadena3[i] );
  TextOut(x-15,54,unit3.cadena2[J] );
  TextOut(x+40,54,unit3.cadena2[2*i] ); }
  MoveTo(cx,y);
  LineTo(cx+40,y);
  MoveTo(cx+40,y);
  LineTo(cx+40,y+8);
  MoveTo(cx+40,y+12);
  LineTo(cx+40,y+20);
  MoveTo(cx+40,y+20);
  LineTo(cx,y+20);
  moveto(cx,y+20);
  lineto(cx,y);
```

```
{pen.color:=clpurple;}
(*lineas medias*)
MoveTo(cx+10+ci,y+8);
LineTO(cx+45+ci,y+8);
MoveTo(cx+10+ci,y+12);
LineTo(cx+45+ci,y+12);
MoveTo(cx+6+ci,y);
LineTo(cx+6+ci,y+20);
MoveTo(cx+10+ci,y);
LineTo(cx+10+ci,y+8);
MoveTo(cx+10+ci,y+12);
LineTo(cx+10+ci,y+20);
Moveto(cx+45+ci,y+8);
LineTo(cx+45+ci,y+12);
pen.color:=clwhite;
Moveto(cx+50+ci,y+7);
Lineto(cx+50+ci,y+13);
Moveto(cx+10+ci,y);
Lineto(cx+10+ci,y+8);
MoveTo(cx+10+ci,y+12);
LineTo(cx+10+ci,y+20);
{conexiones a memorias}
pen.color:=clred;
Moveto(cx+5,y+20);
Lineto(cx+5,y+30);
Moveto(cx+5,y+30);
Lineto(cx+25,y+30);
Moveto(cx+25,y+30);
Lineto(cx+25,y+40);
Moveto(cx+35,y+20);
Lineto(cx+35,y+40);
{memorias}
pen.color:=clblue;
rectangle(cx,70,cx+40,90);
Moveto(cx+5,70);
Lineto(cx+5,90);
Moveto(cx+20,70);
Lineto(cx+20,90);
Moveto(cx+25,70);
Lineto(cx+35,90);
Moveto(cx+25,90);
Lineto(cx+35,70);
Moveto(cx+35,70);
Lineto(cx+31,73);
Moveto(cx+35,70);
Lineto(cx+35,75);
Moveto(cx+35,90);
Lineto(cx+31,87);
Moveto(cx+35,90);
Lineto(cx+35,85);
Moveto(cx+5,70);
```

```
        Lineto(cx+2,74);
        Moveto(cx+5,70);
        Lineto(cx+8,74);
        Moveto(cx+15,90);
        Lineto(cx+12,86);
        Moveto(cx+15,90);
        Lineto(cx+18,86);
        Moveto(cx+15,70);
        Lineto(cx+15,90);
        Moveto(cx+25,90);
        Lineto(cx+25,93);
        ellipse(cx+20,93,cx+30,103);
        Moveto(cx+24,98);
        Lineto(cx+26,98);
       end;
       end;
      end;
     if p=0 then
     x:=x+120;
     end;
'D': begin
      if (unit3.form3.edit1.text[i+1]='+') then
      begin
       p:=0;
       Dx:=x;
       for ci:=1 to 30 do begin
       with form4.Canvas do begin
        for Ji:=1 to r do begin
        end;
        pen.Width:=1;
        pen.style:=pssolid;
        pen.color:=clblue;
        Brush.Color:=clwhite;
        {textOut(x+18,y-15,unit3.cilindros[i]);
        TextOut(x+44,y-5,unit3.cadena3[i] );
        TextOut(x-15,54,unit3.cadena2[J] );
        TextOut(x+40,54,unit3.cadena2[2*i] );  }
        MoveTo(dx,y);
        LineTo(dx+40,y);
        MoveTo(dx+40,y);
        LineTo(dx+40,y+8);
        MoveTo(dx+40,y+12);
        LineTo(dx+40,y+20);
        MoveTo(dx+40,y+20);
        LineTo(dx,y+20);
        moveto(dx,y+20);
        lineto(dx,y);
        {pen.color:=clpurple;}
        (*lineas medias*)
        MoveTo(dx+10+ci,y+8);
        LineTO(dx+45+ci,y+8);
```

```
MoveTo(dx+10+ci,y+12);
LineTo(dx+45+ci,y+12);
MoveTo(dx+6+ci,y);
LineTo(dx+6+ci,y+20);
MoveTo(dx+10+ci,y);
LineTo(dx+10+ci,y+8);
MoveTo(dx+10+ci,y+12);
LineTo(dx+10+ci,y+20);
Moveto(dx+45+ci,y+8);
LineTo(dx+45+ci,y+12);
pen.color:=clwhite;
Moveto(dx+6+ci,y);
Lineto(dx+6+ci,y+20);
{conexiones a memorias}
pen.color:=cllime;
Moveto(dx+5,y+20);
Lineto(dx+5,y+30);
Moveto(dx+5,y+30);
Lineto(dx+25,y+30);
Moveto(dx+25,y+30);
Lineto(dx+25,y+40);
Moveto(dx+35,y+20);
Lineto(dx+35,y+40);
{memorias}
pen.color:=clblue;
rectangle(dx,70,dx+40,90);
Moveto(dx+5,70);
Lineto(dx+5,90);
Moveto(dx+20,70);
Lineto(dx+20,90);
Moveto(dx+25,70);
Lineto(dx+35,90);
Moveto(dx+25,90);
Lineto(dx+35,70);
Moveto(dx+35,70);
Lineto(dx+31,73);
Moveto(dx+35,70);
Lineto(dx+35,75);
Moveto(dx+35,90);
Lineto(dx+31,87);
Moveto(dx+35,90);
Lineto(dx+35,85);
Moveto(dx+5,70);
Lineto(dx+2,74);
Moveto(dx+5,70);
Lineto(dx+8,74);
Moveto(dx+15,90);
Lineto(dx+12,86);
Moveto(dx+15,90);
Lineto(dx+18,86);
Moveto(dx+15,70);
```

```
  Lineto(dx+15,90);
  Moveto(dx+25,90);
  Lineto(dx+25,93);
  ellipse(dx+20,93,dx+30,103);
  Moveto(dx+24,98);
  Lineto(dx+26,98);
 end;
 end;
end;

if (unit3.form3.edit1.text[i+1]='-') then
begin
 p:=1;
 for ci:=30 DOWNTO 1 do begin
 with form4.Canvas do begin
  for Ji:=1 to r do begin
  end;
  pen.Width:=1;
  pen.style:=pssolid;
  pen.color:=clblue;
  Brush.Color:=clwhite;
  {textOut(x+18,y-15,unit3.cilindros[i]);
  TextOut(x+44,y-5,unit3.cadena3[i] );
  TextOut(x-15,54,unit3.cadena2[J] );
  TextOut(x+40,54,unit3.cadena2[2*i] );}
  MoveTo(dx,y);
  LineTo(dx+40,y);
  MoveTo(dx+40,y);
  LineTo(dx+40,y+8);
  MoveTo(dx+40,y+12);
  LineTo(dx+40,y+20);
  MoveTo(dx+40,y+20);
  LineTo(dx,y+20);
  moveto(dx,y+20);
  lineto(dx,y);
  {pen.color:=clpurple;}
  (*lineas medias*)
  MoveTo(dx+10+ci,y+8);
  LineTO(dx+45+ci,y+8);
  MoveTo(dx+10+ci,y+12);
  LineTo(dx+45+ci,y+12);
  MoveTo(dx+6+ci,y);
  LineTo(dx+6+ci,y+20);
  MoveTo(dx+10+ci,y);
  LineTo(dx+10+ci,y+8);
  MoveTo(dx+10+ci,y+12);
  LineTo(dx+10+ci,y+20);
  Moveto(dx+45+ci,y+8);
  LineTo(dx+45+ci,y+12);
  pen.color:=clwhite;
  Moveto(dx+50+ci,y+7);
```

```
 Lineto(dx+50+ci,y+13);
 Moveto(dx+10+ci,y);
 Lineto(dx+10+ci,y+8);
 MoveTo(dx+10+ci,y+12);
 LineTo(dx+10+ci,y+20);
 {conexiones a memorias}
 pen.color:=clred;
 Moveto(dx+5,y+20);
 Lineto(dx+5,y+30);
 Moveto(dx+5,y+30);
 Lineto(dx+25,y+30);
 Moveto(dx+25,y+30);
 Lineto(dx+25,y+40);
 Moveto(dx+35,y+20);
 Lineto(dx+35,y+40);
 {memorias}
 pen.color:=clblue;
 rectangle(dx,70,dx+40,90);
 Moveto(dx+5,70);
 Lineto(dx+5,90);
 Moveto(dx+20,70);
 Lineto(dx+20,90);
 Moveto(dx+25,70);
 Lineto(dx+35,90);
 Moveto(dx+25,90);
 Lineto(dx+35,70);
 Moveto(dx+35,70);
 Lineto(dx+31,73);
 Moveto(dx+35,70);
 Lineto(dx+35,75);
 Moveto(dx+35,90);
 Lineto(dx+31,87);
 Moveto(dx+35,90);
 Lineto(dx+35,85);
 Moveto(dx+5,70);
 Lineto(dx+2,74);
 Moveto(dx+5,70);
 Lineto(dx+8,74);
 Moveto(dx+15,90);
 Lineto(dx+12,86);
 Moveto(dx+15,90);
 Lineto(dx+18,86);
 Moveto(dx+15,70);
 Lineto(dx+15,90);
 Moveto(dx+25,90);
 Lineto(dx+25,93);
 ellipse(dx+20,93,dx+30,103);
 Moveto(dx+24,98);
 Lineto(dx+26,98);
end;
end;
```

```
      end;
     if p=0 then
     x:=x+120;
     end;
'E': begin
      if (unit3.form3.edit1.text[i+1]='+') then
      begin
       p:=0;
       Ex:=x;
       for ci:=1 to 30 do begin
       with form4.Canvas do begin
        for Ji:=1 to r do begin
        end;
        pen.Width:=1;
        pen.style:=pssolid;
        pen.color:=clblue;
        Brush.Color:=clwhite;
        {textOut(x+18,y-15,unit3.cilindros[i]);
        TextOut(x+44,y-5,unit3.cadena3[i] );
        TextOut(x-15,54,unit3.cadena2[J] );
        TextOut(x+40,54,unit3.cadena2[2*i] );  }
        MoveTo(ex,y);
        LineTo(ex+40,y);
        MoveTo(ex+40,y);
        LineTo(ex+40,y+8);
        MoveTo(ex+40,y+12);
        LineTo(ex+40,y+20);
        MoveTo(ex+40,y+20);
        LineTo(ex,y+20);
        moveto(ex,y+20);
        lineto(ex,y);
        {pen.color:=clpurple;}
        (*lineas medias*)
        MoveTo(ex+10+ci,y+8);
        LineTO(ex+45+ci,y+8);
        MoveTo(ex+10+ci,y+12);
        LineTo(ex+45+ci,y+12);
        MoveTo(ex+6+ci,y);
        LineTo(ex+6+ci,y+20);
        MoveTo(ex+10+ci,y);
        LineTo(ex+10+ci,y+8);
        MoveTo(ex+10+ci,y+12);
        LineTo(ex+10+ci,y+20);
        Moveto(ex+45+ci,y+8);
        LineTo(ex+45+ci,y+12);
        pen.color:=clwhite;
        Moveto(ex+6+ci,y);
        Lineto(ex+6+ci,y+20);
        {conexiones a memorias}
        pen.color:=cllime;
        Moveto(ex+5,y+20);
```

```
  Lineto(ex+5,y+30);
  Moveto(ex+5,y+30);
  Lineto(ex+25,y+30);
  Moveto(ex+25,y+30);
  Lineto(ex+25,y+40);
  Moveto(ex+35,y+20);
  Lineto(ex+35,y+40);
  {memorias}
  pen.color:=clblue;
  rectangle(ex,70,ex+40,90);
  Moveto(ex+5,70);
  Lineto(ex+5,90);
  Moveto(ex+20,70);
  Lineto(ex+20,90);
  Moveto(ex+25,70);
  Lineto(ex+35,90);
  Moveto(ex+25,90);
  Lineto(ex+35,70);
  Moveto(ex+35,70);
  Lineto(ex+31,73);
  Moveto(ex+35,70);
  Lineto(ex+35,75);
  Moveto(ex+35,90);
  Lineto(ex+31,87);
  Moveto(ex+35,90);
  Lineto(ex+35,85);
  Moveto(ex+5,70);
  Lineto(ex+2,74);
  Moveto(ex+5,70);
  Lineto(ex+8,74);
  Moveto(ex+15,90);
  Lineto(ex+12,86);
  Moveto(ex+15,90);
  Lineto(ex+18,86);
  Moveto(ex+15,70);
  Lineto(ex+15,90);
  Moveto(ex+25,90);
  Lineto(ex+25,93);
  ellipse(ex+20,93,ex+30,103);
  Moveto(ex+24,98);
  Lineto(ex+26,98);
 end;
 end;
end;

if (unit3.form3.edit1.text[i+1]='-') then
begin
 p:=1;
 for ci:=30 DOWNTO 1 do begin
 with form4.Canvas do begin
  for Ji:=1 to r do begin
```

```
end;
pen.Width:=1;
pen.style:=pssolid;
pen.color:=clblue;
Brush.Color:=clwhite;
{textOut(x+18,y-15,unit3.cilindros[i]);
TextOut(x+44,y-5,unit3.cadena3[i] );
TextOut(x-15,54,unit3.cadena2[J] );
TextOut(x+40,54,unit3.cadena2[2*i] );  }
MoveTo(ex,y);
LineTo(ex+40,y);
MoveTo(ex+40,y);
LineTo(ex+40,y+8);
MoveTo(ex+40,y+12);
LineTo(ex+40,y+20);
MoveTo(ex+40,y+20);
LineTo(ex,y+20);
moveto(ex,y+20);
lineto(ex,y);
{pen.color:=clpurple;}
(*lineas medias*)
MoveTo(ex+10+ci,y+8);
LineTO(ex+45+ci,y+8);
MoveTo(ex+10+ci,y+12);
LineTo(ex+45+ci,y+12);
MoveTo(ex+6+ci,y);
LineTo(ex+6+ci,y+20);
MoveTo(ex+10+ci,y);
LineTo(ex+10+ci,y+8);
MoveTo(ex+10+ci,y+12);
LineTo(ex+10+ci,y+20);
Moveto(ex+45+ci,y+8);
LineTo(ex+45+ci,y+12);
pen.color:=clwhite;
Moveto(ex+50+ci,y+7);
Lineto(ex+50+ci,y+13);
Moveto(ex+10+ci,y);
Lineto(ex+10+ci,y+8);
MoveTo(ex+10+ci,y+12);
LineTo(ex+10+ci,y+20);
{conexiones a memorias}
pen.color:=clred;
Moveto(ex+5,y+20);
Lineto(ex+5,y+30);
Moveto(ex+5,y+30);
Lineto(ex+25,y+30);
Moveto(ex+25,y+30);
Lineto(ex+25,y+40);
Moveto(ex+35,y+20);
Lineto(ex+35,y+40);
{memorias}
```

```
        pen.color:=clblue;
        rectangle(ex,70,ex+40,90);
        Moveto(ex+5,70);
        Lineto(ex+5,90);
        Moveto(ex+20,70);
        Lineto(ex+20,90);
        Moveto(ex+25,70);
        Lineto(ex+35,90);
        Moveto(ex+25,90);
        Lineto(ex+35,70);
        Moveto(ex+35,70);
        Lineto(ex+31,73);
        Moveto(ex+35,70);
        Lineto(ex+35,75);
        Moveto(ex+35,90);
        Lineto(ex+31,87);
        Moveto(ex+35,90);
        Lineto(ex+35,85);
        Moveto(ex+5,70);
        Lineto(ex+2,74);
        Moveto(ex+5,70);
        Lineto(ex+8,74);
        Moveto(ex+15,90);
        Lineto(ex+12,86);
        Moveto(ex+15,90);
        Lineto(ex+18,86);
        Moveto(ex+15,70);
        Lineto(ex+15,90);
        Moveto(ex+25,90);
        Lineto(ex+25,93);
        ellipse(ex+20,93,ex+30,103);
        Moveto(ex+24,98);
        Lineto(ex+26,98);
       end;
       end;
      end;

     if p=0 then
      x:=x+120;
     end;
  end;
  j:=1+2*i;
 end;
end;
```

ELIMINACION_DE_SEÑALES

Este Procedimiento realiza el diagrama de eliminación de señales y muestra las gráficamente las señales que se deben eliminar colocando válvulas de rodillo abatible. Se ejecuta cuando se pulsa la opción eliminación de señales de la pantalla principal de diseño de circuitos por secuencia.

CODIGO DE ELIMINACION_DE_SEÑALES

```
procedure TForm4.Eliminaciondeseales1Click(Sender: TObject);
var
aux:string[20];
vector:array[1..20] of integer;
a,b,c,d,e,i,j,ik,lax,iayn,ibx,ibyn,icx,icyn,idx,idyn,iex,ieyn,iayp,ibyp,icyp,idyp,ieyp:integer;
ij,R:longint;
begin
 form4.Empezar1.enabled:=false;
 form4.speedButton2.enabled:=false;
 button1.click;
 form4.refresh;
 form4.Caption:='Diagrama de Eliminación de señales';
 j:=1;
 a:=1;b:=0;
 lax:=100;
 lbx:=100;
 lcx:=100;
 ldx:=100;
 lex:=100;
 with canvas do begin
  pen.Width:=3;
  pen.color:=clRed;
  brush.color:=clwhite;
  rectangle(30,30,360,360);
  pen.color:=clblack;
  textout(160,45,'FASES');
  textout(130,330,'Secuencia: '+unit3.form3.edit1.text);
  pen.color:=clred;
  pen.Width:=3;
  moveto(98,78);
  lineto(98,302);
  pen.color:=clblue;
  moveto(98,100);
  lineto(104,100);
  textout(70,95,'Start');
  pen.color:=clred;
  moveto(98,78);
  lineto(95,82);
```

```
 moveto(98,78);
 lineto(101,82);
 moveto(98,302);
 lineto(300,302);
 moveto(300,302);
 lineto(296,305);
 moveto(300,302);
 lineto(296,299);
 pen.style:=psdot;
 for i:=1 to 23 do begin
  pen.Width:=1;
  moveto(100,70+10*i);
  lineto(100+10*(length(unit3.form3.Edit1.text)),70+10*i);
 end;
 for i:=1 to (1+length(unit3.form3.edit1.text)) do begin
  moveto(90+10*i,80);
  lineto(90+10*i,300);
  if (i mod 2)=0 then
   begin
    brush.color:=clwhite;
    textout(93+10*i,62,'F'+inttostr(j));
    j:=j+1;
   end;
 end;
end;
ik:=length(unit3.form3.edit1.text);
for i:=1 to 20 do begin
 aux[i]:=' ';
 vector[i]:=0;
end;
aux[1]:=unit3.form3.edit1.text[ik-1];
aux[2]:=unit3.form3.edit1.text[ik];
for i:=1 to ik-2 do begin
  aux[i+2]:=unit3.form3.Edit1.text[i];
end;
for i:=3 to ik do begin
 if aux[i]='A' then
  begin
   b:=b+1;
   vector[i-2]:=a;
   a:=a+1;
  end;
end;
if b=1 then begin
  vector[ik-1]:=a;
  a:=a+1;
end;
b:=0;
for i:=3 to ik do begin
 if aux[i]='B' then
  begin
```

```
   b:=b+1;
   vector[i-2]:=a;
   a:=a+1;
  end;
end;
if b=1 then begin
  vector[ik-1]:=a;
  a:=a+1;
end;
b:=0;
for i:=3 to ik do begin
 if aux[i]='C' then
  begin
   b:=b+1;
   vector[i-2]:=a;
   a:=a+1;
  end;
end;
if b=1 then begin
  vector[ik-1]:=a;
  a:=a+1;
end;
b:=0;
for i:=3 to ik do begin
 if aux[i]='D' then
  begin
   b:=b+1;
   vector[i-2]:=a;
   a:=a+1;
  end;
end;
if b=1 then begin
  vector[ik-1]:=a;
  a:=a+1;
end;
b:=0;
for i:=3 to ik do begin
 if aux[i]='E' then
  begin
   b:=b+1;
   vector[i-2]:=a;
   a:=a+1;
  end;
end;
if b=1 then begin
  vector[ik-1]:=a;
  a:=a+1;
end;
b:=0;
i:=vector[1];
vector[ik]:=0;
```

```
for j:=3 to ik do begin
vector[j-2]:=vector[j];
end;
vector[ik-1]:=I;
vector[ik]:=0;
for i:=1 to ik do begin
 if unit3.form3.edit1.text[i]='A' then
  begin
   if unit3.form3.edit1.text[i+1]='-' then
    iayn:=100+20*vector[i];
   if unit3.form3.edit1.text[i+1]='+' then
    iayp:=100+20*vector[i];
  end;
 if unit3.form3.edit1.text[i]='B' then
  begin
   if unit3.form3.edit1.text[i+1]='-' then
    ibyn:=100+20*vector[i];
   if unit3.form3.edit1.text[i+1]='+' then
    ibyp:=100+20*vector[i];
  end;
 if unit3.form3.edit1.text[i]='C' then
  begin
   if unit3.form3.edit1.text[i+1]='-' then
    icyn:=100+20*vector[i];
   if unit3.form3.edit1.text[i+1]='+' then
    icyp:=100+20*vector[i];
  end;
 if unit3.form3.edit1.text[i]='D' then
  begin
   if unit3.form3.edit1.text[i+1]='-' then
    idyn:=100+20*vector[i];
   if unit3.form3.edit1.text[i+1]='+' then
    idyp:=100+20*vector[i];
  end;
 if unit3.form3.edit1.text[i]='E' then
  begin
   if unit3.form3.edit1.text[i+1]='-' then
    ieyn:=100+20*vector[i];
   if unit3.form3.edit1.text[i+1]='+' then
    ieyp:=100+20*vector[i];
  end;
end;
 r:=1000;
 A:=0;b:=0;c:=0;d:=0;e:=0;
 for i:=1 to length(unit3.form3.edit1.text) do begin
  canvas.pen.Width:=2;
  canvas.PEN.COLOR:=clblue;
  if (i mod 2)<>0 then begin
    if (unit3.form3.edit1.text[i]<>'A') THEN BEGIN
     if a=0 then begin
      for j:=1 to 20 do begin
```

```
   for ik:=1 to r do
   canvas.moveto(iax,iayn);
   canvas.lineto(iax+1,iayn);
   iax:=iax+1;
  end;
 end;
 if a=1 then begin
  for j:=1 to 20 do begin
   for ik:=1 to r do
   canvas.moveto(iax,iayp);
   canvas.lineto(iax+1,iayp);
   iax:=iax+1;
  end;
 end;
end;
if (unit3.form3.edit1.text[i]='A') THEN BEGIN
  brush.color:=clwhite;
  if (unit3.form3.edit1.text[i+1]='-')then
    if a=0 then
     canvas.textout(70,85+20*vector[i],unit3.Form3.edit1.text[i+2]);

   begin
     brush.color:=clwhite;
     canvas.textout(85,iayn-7,'a0');
   end;
  if (unit3.form3.edit1.text[i+1]='+')then
   begin
     brush.color:=clwhite;
     canvas.textout(85,iayp-7,'a1');
   end;
  IAX:=iax+20;
  if (unit3.form3.edit1.text[i+2]='A') then begin
   canvas.moveto(iax-2,iayp);
   canvas.lineto(iax+2,iayp);
  end;
  a:=a+1;
  if (a=2) then begin
   a:=0;
  end;
end;
if (unit3.form3.edit1.text[i]<>'B') THEN BEGIN
 if b=0 then begin
  for j:=1 to 20 do begin
   for ik:=1 to r do
   canvas.moveto(ibx,ibyn);
   canvas.lineto(ibx+1,ibyn);
   ibx:=ibx+1;
  end;
 end;
 if b=1 then begin
  for j:=1 to 20 do begin
```

```
   for ik:=1 to r do
   canvas.moveto(ibx,ibyp);
   canvas.lineto(ibx+1,ibyp);
   ibx:=ibx+1;
  end;
 end;
end;
if (unit3.form3.edit1.text[i]='B') THEN BEGIN
  brush.color:=clwhite;
    if b=0 then
     canvas.textout(70,85+20*vector[i],unit3.Form3.edit1.text[i+2]);
  if (unit3.form3.edit1.text[i+1]='-')then
   begin
    brush.color:=clwhite;
    canvas.textout(85,ibyn-7,'b0');
   end;
  if (unit3.form3.edit1.text[i+1]='+')then
   begin
    brush.color:=clwhite;
    canvas.textout(85,ibyp-7,'b1');
   end;
  IbX:=ibx+20;
  if (unit3.form3.edit1.text[i+2]='B') then begin
   canvas.moveto(ibx-2,ibyp);
   canvas.lineto(ibx+2,ibyp);
  end;
  b:=b+1;
  if (b=2) then begin
   b:=0;
  end;
end;
if (unit3.form3.edit1.text[i]<>'C') THEN BEGIN
 if c=0 then begin
  for j:=1 to 20 do begin
   for ik:=1 to r do
   canvas.moveto(icx,icyn);
   canvas.lineto(icx+1,icyn);
   icx:=icx+1;
  end;
 end;
 if c=1 then begin
  for j:=1 to 20 do begin
   for ik:=1 to r do
   canvas.moveto(icx,icyp);
   canvas.lineto(icx+1,icyp);
   icx:=icx+1;
  end;
 end;
end;
if (unit3.form3.edit1.text[i]='C') THEN BEGIN
  brush.color:=clwhite;
```

```
    if c=0 then
     canvas.textout(70,85+20*vector[i],unit3.Form3.edit1.text[i+2]);
  if (unit3.form3.edit1.text[i+1]='-')then
   begin
    brush.color:=clwhite;
    canvas.textout(85,icyn-7,'c0');
   end;
  if (unit3.form3.edit1.text[i+1]='+')then
   begin
    brush.color:=clwhite;
    canvas.textout(85,icyp-7,'c1');
   end;
  IcX:=icx+20;
  if (unit3.form3.edit1.text[i+2]='C') then begin
   canvas.moveto(icx-2,icyp);
   canvas.lineto(icx+2,icyp);
  end;
  c:=c+1;
  if (c=2) then begin
   c:=0;
  end;
 end;
 if (unit3.form3.edit1.text[i]<>'D') THEN BEGIN
  if d=0 then begin
   for j:=1 to 20 do begin
    for ik:=1 to r do
    canvas.moveto(idx,idyn);
    canvas.lineto(idx+1,idyn);
    idx:=idx+1;
   end;
  end;
  if d=1 then begin
   for j:=1 to 20 do begin
    for ik:=1 to r do
    canvas.moveto(idx,idyp);
    canvas.lineto(idx+1,idyp);
    idx:=idx+1;
   end;
  end;
 end;
 if (unit3.form3.edit1.text[i]='D') THEN BEGIN
   brush.color:=clwhite;
    if d=0 then
     canvas.textout(70,85+20*vector[i],unit3.Form3.edit1.text[i+2]);
  if (unit3.form3.edit1.text[i+1]='-')then
   begin
    brush.color:=clwhite;
    canvas.textout(85,idyn-7,'d0');
   end;
  if (unit3.form3.edit1.text[i+1]='+')then
   begin
```

```
      brush.color:=clwhite;
      canvas.textout(85,idyp-7,'d1');
     end;
    IdX:=idx+20;
    if (unit3.form3.edit1.text[i+2]='D') then begin
     canvas.moveto(idx-2,idyp);
     canvas.lineto(idx+2,idyp);
    end;
    d:=d+1;
    if (d=2) then begin
     d:=0;
    end;
   end;
  if (unit3.form3.edit1.text[i]<>'E') THEN BEGIN
   if e=0 then begin
    for j:=1 to 20 do begin
     for ik:=1 to r do
     canvas.moveto(iex,ieyn);
     canvas.lineto(iex+1,ieyn);
     iex:=iex+1;
    end;
   end;
   if e=1 then begin
    for j:=1 to 20 do begin
     for ik:=1 to r do
     canvas.moveto(iex,ieyp);
     canvas.lineto(iex+1,ieyp);
     iex:=iex+1;
    end;
   end;
  end;
  if (unit3.form3.edit1.text[i]='E') THEN BEGIN
    brush.color:=clwhite;
      if e=0 then
      canvas.textout(70,85+20*vector[i],unit3.Form3.edit1.text[i+2]);
    if (unit3.form3.edit1.text[i+1]='-')then
     begin
       brush.color:=clwhite;
       canvas.textout(85,ieyn-7,'e0');
     end;
    if (unit3.form3.edit1.text[i+1]='+')then
     begin
       brush.color:=clwhite;
       canvas.textout(85,ieyp-7,'e1');
     end;
    IeX:=iex+20;
    if (unit3.form3.edit1.text[i+2]='E') then begin
     canvas.moveto(iex-2,ieyp);
     canvas.lineto(iex+2,ieyp);
    end;
    e:=e+1;
```

```
      if (e=2) then begin
       e:=0;
      end;
     end;
   end;
  end;
Elimsles;
end;
```

ESPACIO_FASE

Este procedimiento realiza el diagrama del comportamiento de los cilindros en cada una de las fases de la simulación del circuito. Se ejecuta cuando se pulsa la opción espacio fase del menú principal o pulsando el botón etiquetado con esta opción.

CODIGO DE ESPACIO_FASE

```
procedure TForm4.Mando1Click(Sender: TObject);
var
i,j,ik,Iax,iay,ibx,iby,icx,icy,idx,idy,iex,Iey:integer;
ij,R:longint;
begin
 form4.Empezar1.enabled:=false;
 form4.speedButton2.enabled:=false;
 button1.click;
 form4.refresh;
 form4.Caption:='Diagrama Espacio fase';
 j:=1;
 Iax:=100;iay:=iax;
 Ibx:=100;iby:=150;
 Icx:=100;icy:=200;
 Idx:=100;idy:=250;
 Iex:=100;iey:=300;
 with canvas do begin
  pen.Width:=3;
  pen.color:=clRed;
  brush.color:=clwhite;
  rectangle(30,30,360,360);
  pen.color:=clblack;
```

```
 textout(160,45,'FASES');
 textout(130,330,'Secuencia: '+unit3.form3.edit1.text);
 pen.color:=clred;
 pen.Width:=3;
 moveto(98,78);
 lineto(98,302);
 moveto(98,78);
 lineto(95,82);
 moveto(98,78);
 lineto(101,82);
 moveto(98,302);
 lineto(300,302);
 moveto(300,302);
 lineto(296,305);
 moveto(300,302);
 lineto(296,299);
 pen.style:=psdot;
 for i:=1 to 23 do begin
  pen.Width:=1;
  moveto(100,70+10*i);
  lineto(100+10*(length(unit3.form3.Edit1.text)),70+10*i);
 end;
 for i:=1 to (1+length(unit3.form3.edit1.text)) do begin
  moveto(90+10*i,80);
  lineto(90+10*i,300);
  if (i mod 2)=0 then
   begin
    brush.color:=clwhite;
    textout(93+10*i,62,'F'+inttostr(j));
    j:=j+1;
   end;
 end;
end;
for i:=1 to length(unit3.form3.edit1.text)do
 begin
  case(unit3.form3.edit1.text[i]) of
   'A':begin
       if(unit3.form3.edit1.text[i+1]='+') then
        begin
          canvas.Brush.Color:=clWHITE;
          canvas.TextOut(75,iay-7,'a0');
          for ik:=1 to 20 do begin
           for iJ:=1 to r do begin
           end;
            with form4.canvas do begin
             pen.Width:=2;
             pen.color:=clBLUE;
             Moveto(iax,iay);
             Lineto(iax+1,iay-1);
             { cilindros en reposo  }
             Moveto(ibx,iby);
```

```
          Lineto(ibx+1,iby);
          Moveto(icx,icy);
          Lineto(icx+1,icy);
          Moveto(idx,idy);
          Lineto(idx+1,idy);
          Moveto(iex,iey);
          Lineto(iex+1,iey);
          iax:=iax+1;
          iay:=iay-1;
          ibx:=ibx+1;
          icx:=icx+1;
          idx:=idx+1;
          iex:=iex+1;
         end;
       end;
      end;
{}
    if(unit3.form3.edit1.text[i+1]='-') then
     begin
       canvas.TextOut(75,iay-7,'a1');
       for ik:=1 to 20 do begin
        for iJ:=1 to r do begin
        end;
         with form4.canvas do begin
          pen.Width:=2;
          pen.color:=clBLUE;
          Moveto(iax,iay);
          Lineto(iax+1,iay+1);
          {cilindros en reposo  }
          Moveto(ibx,iby);
          Lineto(ibx+1,iby);
          Moveto(icx,icy);
          Lineto(icx+1,icy);
          Moveto(idx,idy);
          Lineto(idx+1,idy);
          Moveto(iex,iey);
          Lineto(iex+1,iey);
          iax:=iax+1;
          iay:=iay+1;
          ibx:=ibx+1;
          icx:=icx+1;
          idx:=idx+1;
          iex:=iex+1;
         end;
      end;
     end;
    end;
 'B':begin
    if(unit3.form3.edit1.text[i+1]='+') then
     begin
       canvas.TextOut(75,iby-7,'b0');
```

```
        for ik:=1 to 20 do begin
         for iJ:=1 to r do begin
         end;
          with form4.canvas do begin
           pen.Width:=2;
           pen.color:=clBLUE;
           Moveto(ibx,iby);
           Lineto(ibx+1,iby-1);
           { cilindros en reposo  }
           Moveto(iax,iay);
           Lineto(iax+1,iay);
           Moveto(icx,icy);
           Lineto(icx+1,icy);
           Moveto(idx,idy);
           Lineto(idx+1,idy);
           Moveto(iex,iey);
           Lineto(iex+1,iey);
           ibx:=ibx+1;
           iby:=iby-1;
           iax:=iax+1;
           icx:=icx+1;
           idx:=idx+1;
           iex:=iex+1;
          end;
       end;
      end;
{}
    if(unit3.form3.edit1.text[i+1]='-') then
     begin
       canvas.TextOut(75,iby-7,'b1');
       for ik:=1 to 20 do begin
        for iJ:=1 to r do begin
        end;
         with form4.canvas do begin
          pen.Width:=2;
          pen.color:=clBLUE;
          Moveto(ibx,iby);
          Lineto(ibx+1,iby+1);
          { cilindros en reposo  }
          Moveto(iax,iay);
          Lineto(iax+1,iay);
          Moveto(icx,icy);
          Lineto(icx+1,icy);
          Moveto(idx,idy);
          Lineto(idx+1,idy);
          Moveto(iex,iey);
          Lineto(iex+1,iey);
          ibx:=ibx+1;
          iby:=iby+1;
          iax:=iax+1;
          icx:=icx+1;
```

```
          idx:=idx+1;
          iex:=iex+1;
         end;
       end;
      end;
     end;
  'C':begin
     if(unit3.form3.edit1.text[i+1]='+') then
      begin
        canvas.TextOut(75,icy-7,'c0');
       for ik:=1 to 20 do begin
        for iJ:=1 to r do begin
        end;
         with form4.canvas do begin
          pen.Width:=2;
          pen.color:=clBLUE;
          Moveto(icx,icy);
          Lineto(icx+1,icy-1);
          { cilindros en reposo  }
          Moveto(ibx,iby);
          Lineto(ibx+1,iby);
          Moveto(iax,iay);
          Lineto(iax+1,iay);
          Moveto(idx,idy);
          Lineto(idx+1,idy);
          Moveto(iex,iey);
          Lineto(iex+1,iey);
          icx:=icx+1;
          icy:=icy-1;
          ibx:=ibx+1;
          iax:=iax+1;
          idx:=idx+1;
          iex:=iex+1;
         end;
       end;
      end;
{}
     if(unit3.form3.edit1.text[i+1]='-') then
      begin
        canvas.TextOut(75,icy-7,'c1');
       for ik:=1 to 20 do begin
        for iJ:=1 to r do begin
        end;
         with form4.canvas do begin
          pen.Width:=2;
          pen.color:=clBLUE;
          Moveto(icx,icy);
          Lineto(icx+1,icy+1);
          { cilindros en reposo  }
          Moveto(ibx,iby);
          Lineto(ibx+1,iby);
```

```
          Moveto(iax,iay);
          Lineto(iax+1,iay);
          Moveto(idx,idy);
          Lineto(idx+1,idy);
          Moveto(iex,iey);
          Lineto(iex+1,iey);
          icx:=icx+1;
          icy:=icy+1;
          ibx:=ibx+1;
          iax:=iax+1;
          idx:=idx+1;
          iex:=iex+1;
         end;
       end;
     end;
    end;
 'D':begin
    if(unit3.form3.edit1.text[i+1]='+') then
     begin
       canvas.TextOut(75,idy-7,'d0');
       for ik:=1 to 20 do begin
        for iJ:=1 to r do begin
        end;
         with form4.canvas do begin
          pen.Width:=2;
          pen.color:=clBLUE;
          Moveto(idx,idy);
          Lineto(idx+1,idy-1);
          { cilindros en reposo  }
          Moveto(ibx,iby);
          Lineto(ibx+1,iby);
          Moveto(icx,icy);
          Lineto(icx+1,icy);
          Moveto(iax,iay);
          Lineto(iax+1,iay);
          Moveto(iex,iey);
          Lineto(iex+1,iey);
          idx:=idx+1;
          idy:=idy-1;
          ibx:=ibx+1;
          icx:=icx+1;
          iax:=iax+1;
          iex:=iex+1;
         end;
       end;
     end;
{}
    if(unit3.form3.edit1.text[i+1]='-') then
     begin
       canvas.TextOut(75,idy-7,'d1');
       for ik:=1 to 20 do begin
```

```
        for iJ:=1 to r do begin
        end;
         with form4.canvas do begin
          pen.Width:=2;
          pen.color:=clBLUE;
          Moveto(idx,idy);
          Lineto(idx+1,idy+1);
          { cilindros en reposo  }
          Moveto(ibx,iby);
          Lineto(ibx+1,iby);
          Moveto(icx,icy);
          Lineto(icx+1,icy);
          Moveto(iax,iay);
          Lineto(iax+1,iay);
          Moveto(iex,iey);
          Lineto(iex+1,iey);
          idx:=idx+1;
          idy:=idy+1;
          ibx:=ibx+1;
          icx:=icx+1;
          iax:=iax+1;
          iex:=iex+1;
         end;
       end;
     end;
    end;
  'E':begin
     if(unit3.form3.edit1.text[i+1]='+') then
      begin
        canvas.TextOut(75,iey-7,'e0');
        for ik:=1 to 20 do begin
         for iJ:=1 to r do begin
         end;
          with form4.canvas do begin
           pen.Width:=2;
           pen.color:=clBLUE;
           Moveto(iex,iey);
           Lineto(iex+1,iey-1);
           { cilindros en reposo  }
           Moveto(ibx,iby);
           Lineto(ibx+1,iby);
           Moveto(icx,icy);
           Lineto(icx+1,icy);
           Moveto(idx,idy);
           Lineto(idx+1,idy);
           Moveto(iax,iay);
           Lineto(iax+1,iay);
           iex:=iex+1;
           iey:=iey-1;
           ibx:=ibx+1;
           icx:=icx+1;
```

```
          idx:=idx+1;
          iax:=iax+1;
         end;
        end;
       end;
       if(unit3.form3.edit1.text[i+1]='-') then
       begin
        canvas.TextOut(75,iey-7,'e1');
        for ik:=1 to 20 do begin
         for iJ:=1 to r do begin
         end;
          with form4.canvas do begin
           pen.Width:=2;
           pen.color:=clBLUE;
           Moveto(iex,iey);
           Lineto(iex+1,iey+1);
           { cilindros en reposo  }
           Moveto(ibx,iby);
           Lineto(ibx+1,iby);
           Moveto(icx,icy);
           Lineto(icx+1,icy);
           Moveto(idx,idy);
           Lineto(idx+1,idy);
           Moveto(iax,iay);
           Lineto(iax+1,iay);
           iex:=iex+1;
           iey:=iey+1;
           ibx:=ibx+1;
           icx:=icx+1;
           idx:=idx+1;
           iax:=iax+1;
          end;
         end;
        end;
       end;
   end;
  end;
  espaciofase;
end;
```

MANUAL DE USUARIO

INTRODUCCION

1. REQUERIMIENTOS

Para la instalación y ejecución optima de este programa se requiere un equipo compatible con Windows con una memoria RAM de 16 MB. Un procesador de 133 Mhz o más, Mouse, impresora, unidad de CD-ROM, espacio en disco para su ejecución de 200 Mb. ó más y tarjeta de videos.

1.1 PAQUETES USADOS

- **Windows 95/98**
- **Delphi 3.0**
- **Gif Animator 2.0**
- **Word 97**
- **Photo Editor y otros editores gráficos**
- **Internet Explorer**

2. PROCESO DE INSTALACION

2.1 INSTALACION

2.2 QUITAR INSTALACION

2.3 EJECUTAR PROGRAMA

3. ICONOS MAS USUALES

4. MANEJO DEL PROGRAMA

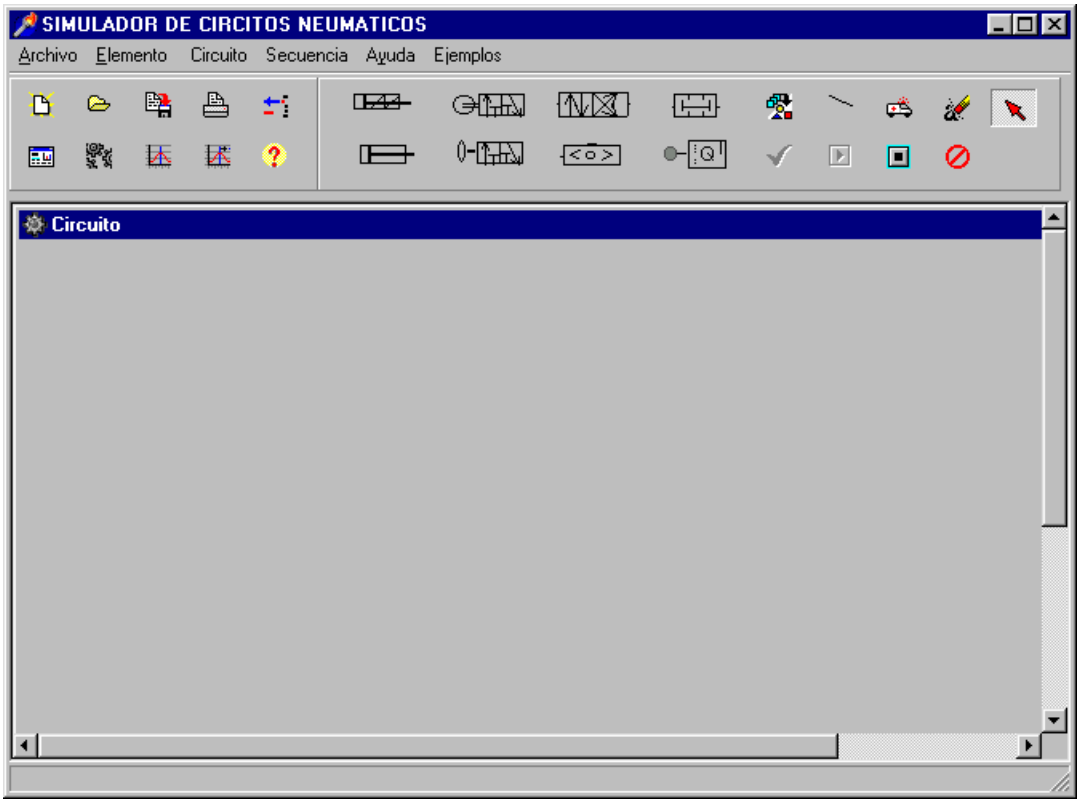
4.1 DISEÑO DE UN CIRCUITO MANUAL

El diseño de un circuito por usuario se puede hacer en la pantalla de diseño manual que se muestra en la siguiente figura.

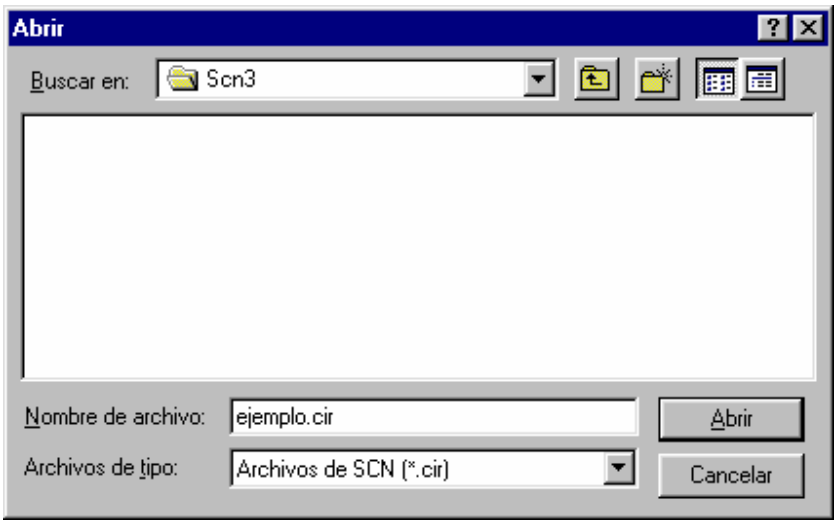
En esta pantalla observamos que tiene muchas opciones entre estas están las opciones de Archivo, Elemento, Circuito, Secuencia, Ayuda y Ejemplos.

- La opción **Archivo** contiene un submenú con las siguientes divisiones:

Nuevo, esta opción limpia la pantalla y la prepara para realizar el diseño de un nuevo circuito.

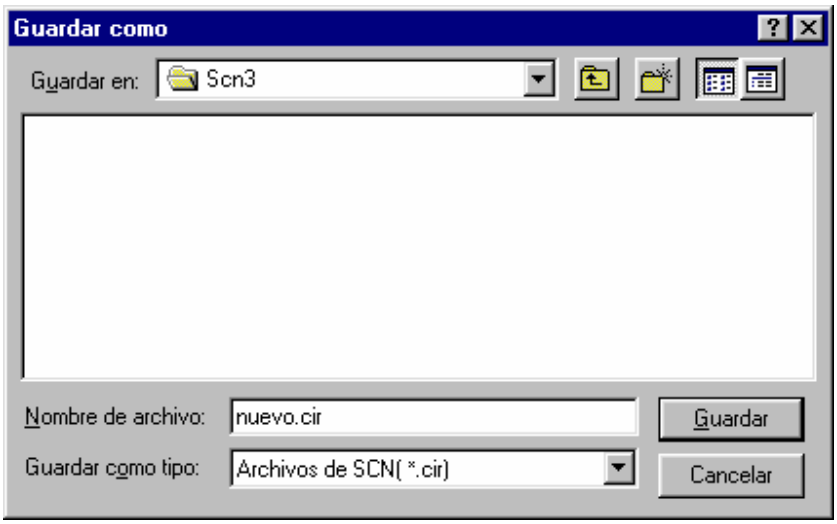


Abrir, abre un circuito de extensión (.CIR) que ya exista.



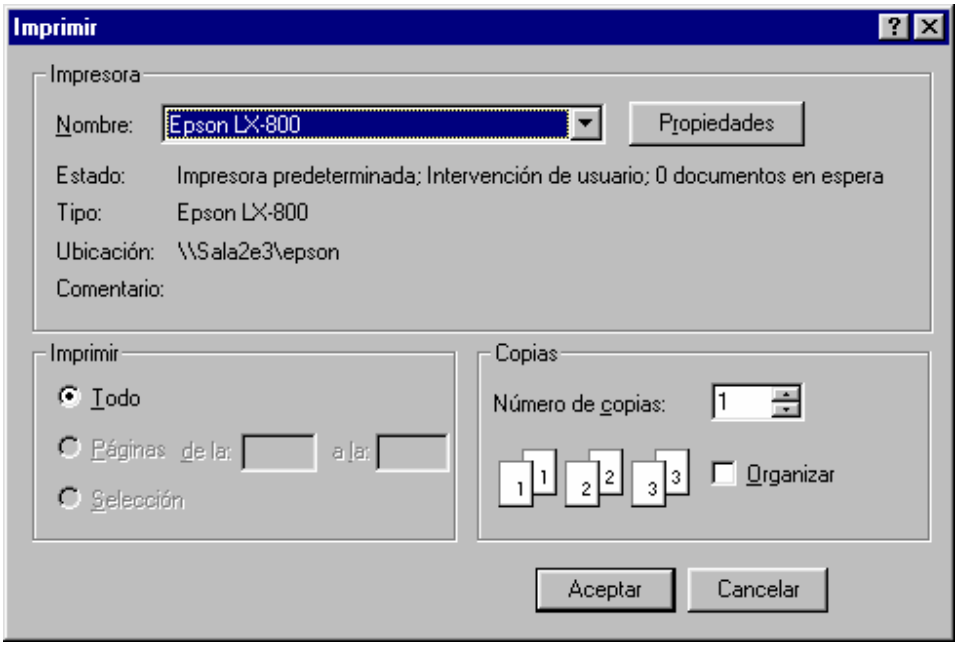
Guardar, permite guardar los cambios de un circuito que ya el circuito está creado.

Guardar como, esta opción presenta un caja de diálogo para realizar esta operación en donde se tiene que dar un nombre nuevo al circuito que se va a guardar, si el circuito ya existe este será sobrescrito.



Cerrar, cierra el circuito que esté abierto y limpia la pantalla de diseño.

Imprimir, esta opción permite que se imprima el circuito que esté diseñado y se le pueden ajustar las propiedades de impresión mediante una ventana de impresión.



Salir, con esta opción termina la aplicación y se cierran todas las ventanas activas.

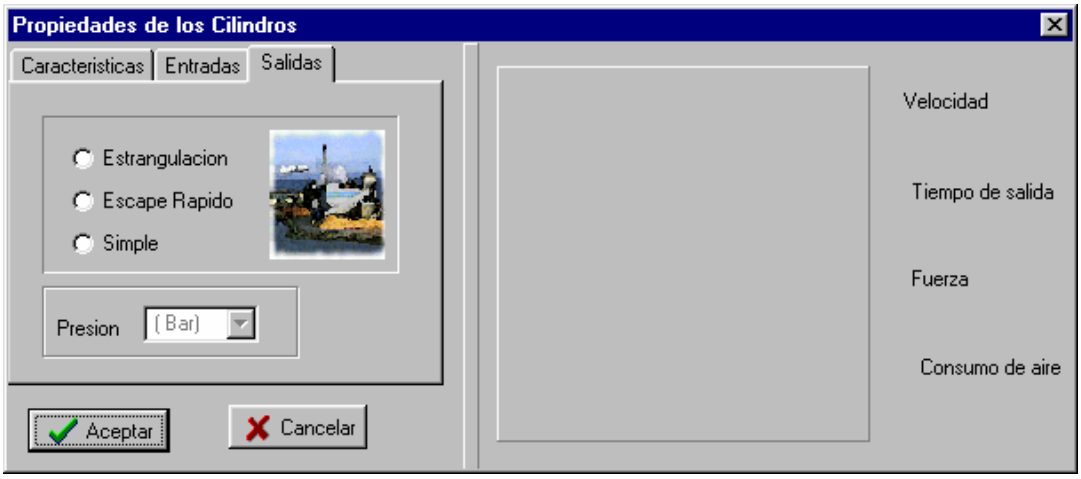
- La opción **Elemento** tiene tres divisiones que son desconectar, borrar y características de un elemento:

Desconectar, también existe un botón para activar esta opción y permite desconectar un circuito que ya ha sido conectado, se debe usar cuando se quiere realizar la corrección de una mala conexión.

Borrar, borra un elemento de la pantalla pero es necesario seleccionar el elemento antes de borrarlo.

Cilindro, esta opción sirve más que todo como ayuda didáctica para ver las características de un cilindro, su velocidad, fuerza, tiempo y consumo de aire.

Se muestra una ventana en donde se pueden modificar estas opciones.



4.1.2 COMO DISEÑAR UN CIRCUITO Y SIMULARLO

Ahora que ya conoce perfectamente todas las opciones de diseño de un circuito manual usted podrá diseñar sin ningún problema un circuito neumático y ejecutar su simulación.

Para diseñar un circuito neumático debemos seguir los siguientes pasos:

1. Podemos elegir los elementos que conforman el circuito Neumático del menú gráfico ubicado en la parte superior derecha de la pantalla, cuando pasamos el

indicador del ratón sobre estos botones se mostrara el nombre del elemento señalado.

2. Para seleccionar un elemento hacemos Click sobre el botón que contiene el elemento deseado, luego hacemos Click sobre la pantalla de diseño.

3. El elemento se muestra sobre la pantalla. De esta manera podemos pegar todos los elementos necesarios para diseñar el circuito.

4. Una vez tenemos los elementos sabré la pantalla, podemos moverlos haciendo un Click sostenido sobre el elemento y arrastrarlo sobre la pantalla de diseño.

5. Cuando los elementos están ubicados correctamente procedemos a conectarlos, para realizar una conexión de dos elementos se siguen los siguientes procedimientos:

5.1 Se hace Click sobre el botón de conexión que se encuentra en el menú gráfico. Una activado este botón los elementos pasan a un estado de conexión.

5.2 Si hago Click sobre la salida de un elemento esta cambia su forma a una línea, indicando que este es el elemento conector, luego hago Click sobre la entrada del elemento a conectar. Si se a conectado el triángulo

debe cerrase y este elemento entonces pasaría a ser el conectado.

5.3 Para completar la conexión se hace Click sobre el botón línea del menú gráfico, el cual dibuja una línea donde se representa la conexión que tiene los dos elementos

5.4 Si se desea borrar una conexión se hace el procedimiento anterior pero en vez de seleccionar el botón línea se selecciona el botón desconexión del menú gráfico.

Para eliminar un elemento sólo se selecciona y se hace Click sobre el botón borrar del menú gráfico.

Para simular el circuito se realizan los siguientes pasos:

1. Una vez realizadas todas las conexiones de los elementos se verifica que todos los elementos estén conectados, haciendo Click sobre el botón “verificar conexión” del menú gráfico. Si todos los elementos están conectados, este botón muestra un mensaje anunciando el éxito o no de las conexiones. Si no se tiene éxito se debe buscar el elemento no conectado y realizar su conexión.

2. Si se tiene éxito en las conexiones se habilitará el botón inicializar, el cual al hacer Click muestra los elementos inicializados. Si el circuito

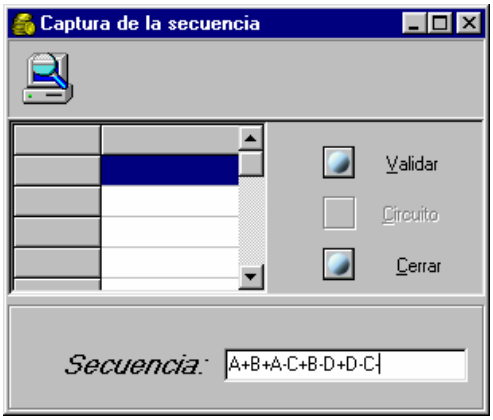
no es lógico, este botón lo detecta y envía un mensaje de error.

3. Si el circuito sé inicializa correctamente se habilita el botón de simulación. Al hacer click sobre este botón se mostrara la salida y entrada de los cilindros según sea el recorrido de la señal sobre el circuito.

4. Si se tiene una válvula pulsador en el circuito esta debe ser pulsada antes de hacer click sobre el botón de simulación. Si la simulación se detiene sin terminar el recorrido del circuito, seguro se debe a que se tiene una válvula pulsador, para este caso se hace click sobre el pulsador de la válvula y luego se hace click sobre el botón de simulación.

CAPTURA DE UNA SECUENCIA

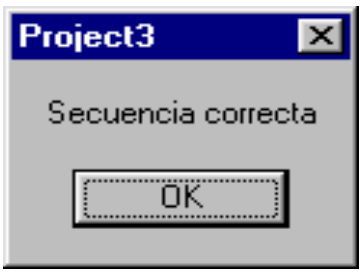
En la captura de una secuencia el usuario debe tener en cuenta que la cadena debe escribirse en mayúsculas y sólo debe contener letras de la 'A' hasta la 'E' y símbolos como '+' y '-'. Existe una pantalla de captura de la secuencia que contiene 3 botones.

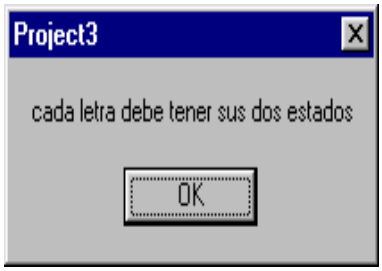


Al momento de que se ha validado la secuencia se puede observar el botón de circuito visible

Botón Validar permite comprobar si la secuencia digitada por el usuario es correcta. Cuando una secuencia está incorrecta aparece en pantalla una caja de diálogo que muestra el tipo de error que tiene la secuencia.

Cuadro de mensajes de error



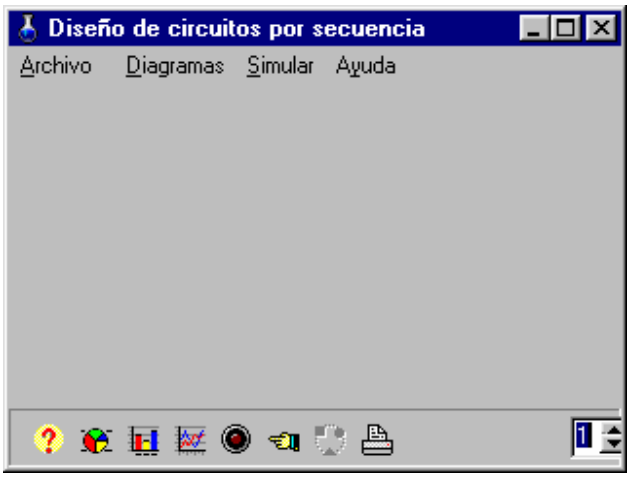


EL Botón Circuito se activa cuando una secuencia es correcta, con este botón el programa internamente procesa la secuencia y manda a diseñar el circuito.

EL Botón Cerrar cierra la ventana activa y regresa y permite que se active la pantalla de diseño manual, funciona como un botón de retroceso de ventanas.

DISEÑO DE UN CIRCUITO POR SECUENCIA

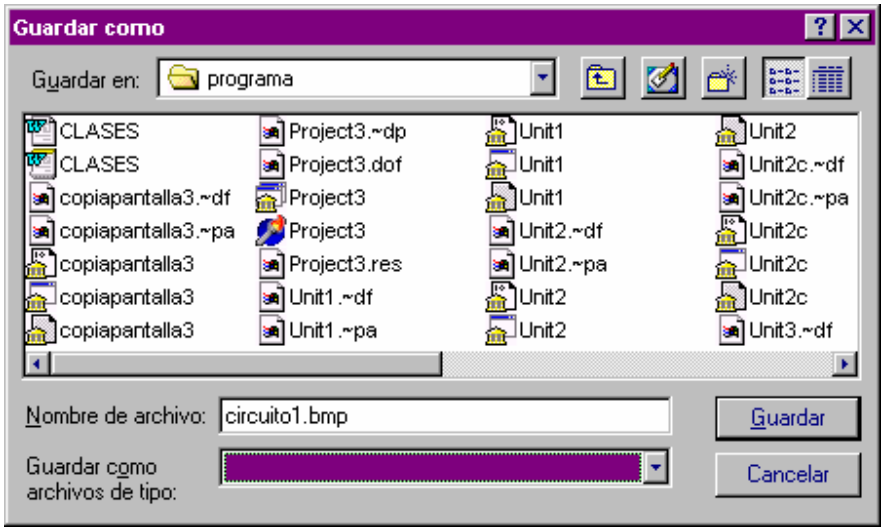
Esta pantalla se activa cuando se pulsa el botón circuito de la ventana de captura, tiene un menú principal con las opciones de: Archivo, diagramas, simular y ayuda.



En la opción **Archivo** encontrará las siguientes opciones:

Imprimir, esta opción se activa cuando es pulsada y muestra una ventana en donde se pueden modificar las propiedades de la impresión.

Guardar, esta opción permite guardar un diagrama como un archivo bitmap con extensión (.bmp). Al momento de guardar se la siguiente ventana.

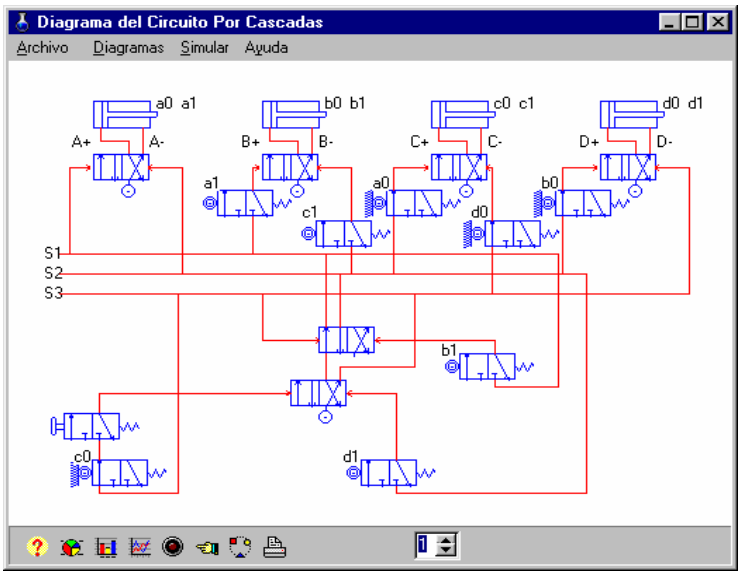


Cerrar, permite cerrar la ventana de diseño de circuitos por secuencia y regresa a la ventana de captura.

Salir, cierra todas las ventanas activas y termina la aplicación.

- La opción **diagramas** del menú principal es de tres tipos:

Diagrama de circuito, muestra el circuito diseñado con sus válvulas cilindros y conexiones. Se presenta un ejemplo de un circuito diseñado por el método de cascadas con la secuencia: **A+B+A-C+B-D+D-C-.**



Así como este diagrama también se puede obtener el diseño de un circuito por paso a paso mínimo, paso a paso máximo y por rodillo abatible.

Diagramas espacio fase, diagrama para la representación de las formas de estado de los elementos, muestra las fases de los movimientos de los cilindros por medio de un gráfico binario en donde un cero representa la entrada del vástago de un cilindro un uno es la salida del vástago de un cilindro.

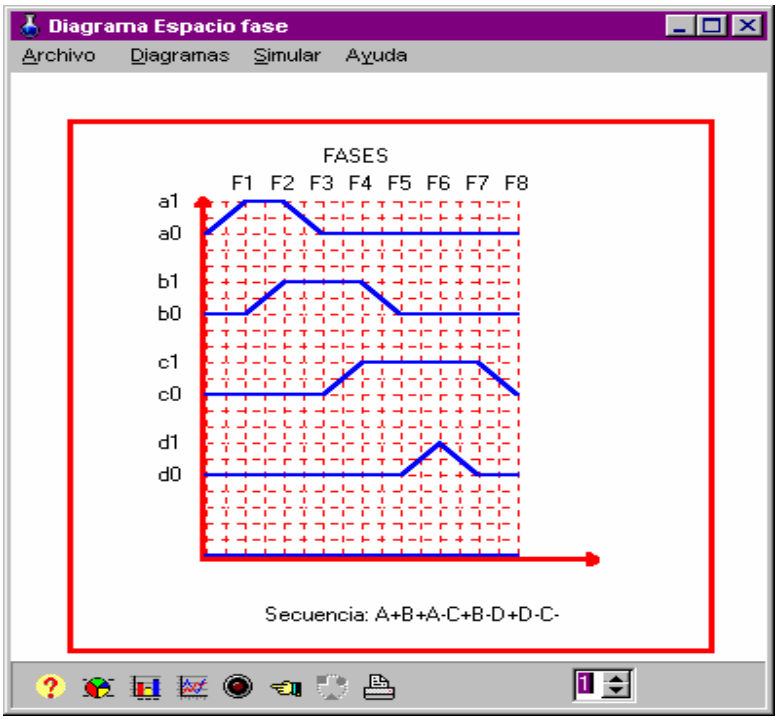
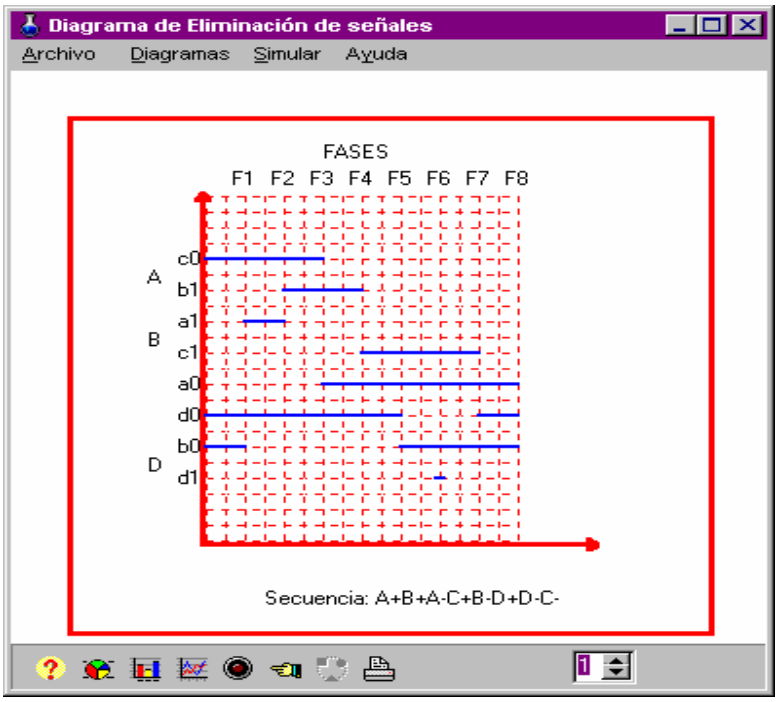


Diagrama de eliminación de señales, es un diagrama que muestra las señales que se deben eliminar en las válvulas para luego hacer una representación del circuito de rodillo abatible. En cada señal eliminada se debe colocar una válvula de rodillo abatible el circuito de rodillo abatible y en las señales que no se eliminan se colocan válvulas de rodillo normal.



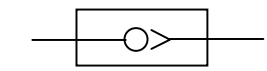
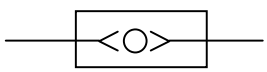
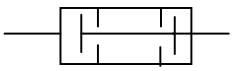
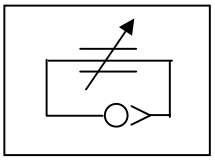
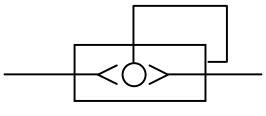
- La opción **simular** contiene un Item con la etiqueta empezar, cuando se pulsa, se simula el movimiento de la salida y entrada del vástago de los cilindros de acuerdo a la secuencia que generó dicho circuito, esta opción sólo está activa cuando hay un circuito diseñado, de otra forma siempre estará inhabilitada.

- La opción **ayuda** presenta dos ítems; Manejo del software y Acerca de….

En la opción de **manejo del Software** se presenta un sistema de ayuda como las que proporciona Windows, está dividida en varias secciones referentes a al manejo del software y algunos conceptos de Neumática.

La opción **acerca de** muestra el nombre del software, la versión del software, un pequeño logotipo de la CUTB. , el año de elaboración y el nombre de los autores.

ANEXO 1

ELEMENTOS	REPRESENTACION GRAFICA
• **Válvula Antiretorno.**	
• **Válvula Selectora.**	
	
	
	

- **Válvula de Simultaneidad.**

- **Válvula de Estrangulación.**

- **Válvula de Escape Rápido.**

- **Válvula 3/2 Normalmente Cerrada**
- **Válvula 4/2 Memoria.**

- **Cilindro de Simple Efecto**
  Retorno Por Muelle

- **Cilindro de Doble Efecto**
- **Fuente de Presión.**
- **Unidad de Mantenimiento**