ESCUELA SUPERIOR
POLITECNICA DEL LITORAL
FACULTAD DE INGENIERIA EN ELECTRICIDAD

SIMULACION DE SISTEMAS DINAMICOS CONTINUOS (CSMP versión para Windows 3.1)

TESIS DE GRADO
Previo a la obtención del Título de
INGENIERO EN COMPUTACION

Presentada por:
VICENTE ROBERTO CHANG NAN

GUAYAQUIL - ECUADOR
1995

AGRADECIMIENTO

Al Señor, por darme paciencia y entusiasmo.

Al Ing. Villavicencio, por creer en mi capacidad y ayudarme cuando más lo necesitaba.

A mis tios, por coolaborar en la elaboración de mi tesis.

A mis padres, por incentivarme continuamente.

DEDICATORIA

A MIS PADRES

A MIS COMPAÑEROS

A MIS MAESTROS

ING. ARMANDO ALTAMIRANO
SUB-DECANO DE LA FACULTAD
DE INGENIERIA ELECTRICA

ING. HUGO VILLAVICENCIO
DIRECTOR DE TESIS

ING. CARLOS JORDAN
MIEMBRO PRINCIPAL
DEL TRIBUNAL

ING. JAVIER URQUIZO
MIEMBRO PRINCIPAL
DEL TRIBUNAL

DECLARACION EXPRESA

"LA RESPONSABILIDAD POR LOS HECHOS, IDEAS Y DOCTRINAS EXPUESTOS EN ESTA TESIS, ME CORRESPONDEN EXCLUSIVAMENTE; Y, EL PATRIMONIO INTELECTUAL DE LA MISMA, A LA ESCUELA SUPERIOR POLITECNICA DEL LITORAL."

(REGLAMENTO DE EXAMENES Y TITULOS PROFESIONALES DE LA ESPOL).

Roberto Chang Nan

VICENTE ROBERTO CHANG NAN

RESUMEN

Nuestro enfoque va dirigido a todos aquellos sistemas físicos, contínuos y dinámicos que pueden ser expresados con Ecuaciones Diferenciales. A su vez estas ecuaciones pueden ser convertidas por medio de Transformadas de La Place en un Diagrama de Bloques. Cada bloque sufre una transformación adicional a Elementos de Computador Analógico (Integrador, Sumador, Multiplicador, etc). Todo este proceso es realizado manualmente por el investigador.

Luego de esto, cada Elemento de Computador Analógico es ingresado como dato al programa de Simulación. La finalidad del CSMP es ofrecer los valores de todas aquellas salidas escogidas por el usuario, a lo largo del período de tiempo seleccionado. Los resultados pueden darse tanto en forma tabular como en gráficos. El trabajo se facilita gracias al ambiente amigable de Windows.

INDICE GENERAL

INTRODUCCION

El presente trabajo ha sido realizado con el fin de permitir la simulación de sistemas dinámicos continuos en microcomputadores que posean ambiente Windows. No pretende ser más que una herramienta para la investigación de estos sistemas, supliendo en algo, la falta de medios (especialmente facilidad) a la que se enfrenta el investigador, sea docente o alumno, para la simulación de problemas específicos.

Una gran dificultad normalmente presente en simulación de sistemas reales, es la falta de un programa capaz de graficar y ejecutar en el tiempo un circuito de Computador Analógico dado. Aun más, se espera que dicho programa sea de fácil uso y corra en un ambiente gráfico. Un programa donde el usuario posea ayudas suficientes y explicaciones de cómo manejarlo, paso a paso.

Esta basado en un programa para la IBM 1130; el 1130 CSMP (Continuous System Modeling Program) fue creado con el objetivo de simular sistemas continuos complejos. Este programa fue de gran utilidad en las investigaciones y estudios de este tipo de sistemas, pese al inconveniente de trabajar en el IBM 1130, que tenía un tiempo de respuesta muy largo para la ejecución del programa.

La version actual del CSMP puede correr en cualquier microcomputador compatible con los personales de IBM, y además que posea el ambiente Windows. Por las caracteristicas de este tipo de equipo, la velocidad de repuesta se ha incrementado enormemente. Además por la presencia del ambiente gráfico, se ha facilitado el uso del programa llegando a una audiencia mayor que incluye estudiantes principiantes.

Gracias a la difusión de los microcomputadores personales se ha logrado que este programa pueda llegar a un mayor repertorio de sistemas dinámicos continuos. Uno de los beneficiados será los futuros estudiantes del Laboratorio del Control Automático, quienes incluirán el uso del computador dentro de sus simulaciones.

Este programa permite el acceso de información bien desde el teclado del computador así como de algún circuito previamente grabado (en forma de un archivo). Tanto el listado de los elementos de Computador Analógico como las curvas de Salidas, pueden visualizarse bien en pantalla o bien en papel.

El programa ha sido implementado íntegramente en el lenguaje *Visual C++* (orientado a objetos); produciendose un sistema idóneo para ambiente Windows.

Las pruebas (problema ilustrativo) se realizaron en base a un ejemplo tomado del libro *Introduction to Computer Simulation* por *A. Wayne Bennett.* Este ejemplo se muestra en el capitulo 4.

CAPITULO 1
GENERALIDADES

1.1 MODELAJE DE SISTEMAS DINAMICOS CONTINUOS

1.1.1 Definición de Sistema Dinámico Continuo

Un sistema es una combinación de componentes que actúan conjuntamente y cumplen determinado objetivo. Estos objetivos pueden ser físicos o abstractos, así tenemos sistemas físicos, biológicos, económicos, etc. El centro de nuestro estudio son los sistemas físicos, es decir, aquellos sistemas que involucran una variedad de cantidades físicas, como voltaje y corriente eléctrica, fuerza y desplazamiento, volúmenes, flujos, temperaturas, etc.

Un sistema esta formado por una colección de componentes interconectados, en los cuales están especificados dos juegos de variables dinámicas, las primeras llamadas entradas o excitaciones, y las segundas, salidas o respuestas.

Si las entradas y salidas de un sistema cambian en cualquier instante de tiempo, el sistema es llamado *Sistema Continuo*. Para sobresalir más este hecho, las variables de entrada y salida son representadas como *Variables en Función de Tiempo*.

Entonces, las excitaciones y respuestas son funciones que dependen de variables contínuas (variantes en el tiempo). Un sistema cuyas salidas no dependen únicamente del valor actual de las entradas, sino que las respuestas en un instante dado dependen de valores pasados o futuros de las excitaciones, es un *Sistema Dinámico*.

Entonces, un *Sistema Dinámico Continuo*, es un sistema formado por un conjunto de componentes cuyas entradas y salidas son funciones del tiempo y cuyas salidas dependen de valores presentes, pasados y/o futuros de las entradas.

La mayoría de los sistemas dinámicos ya sean mecánicos, eléctricos, térmicos, etc, pueden ser caracterizados por ecuaciones diferenciales. Se puede obtener la respuesta de un sistema a una excitación dada, si se resuelven estas ecuaciones.
Para obtener las ecuaciones se utilizan las leyes físicas que gobiernan el sistema, por ejemplo, las *Leyes de Kirchoff* para sistemas eléctricos o las de *Newton* para sistemas mecánicos.

Se denomina modelo matemático a la descripción matemática de las características dinámicas del sistema. El primer paso (el más importante) que se realiza en el análisis de un sistema es la *Elaboración de su Modelo*. Se pueden realizar un sinnumero de modelos, sin embargo, para cada caso en particular, será más ventajoso la utilización de un modelo específico en vez de otro.

Luego de obtener el modelo matemático de un sistema, se podrán usar diversas técnicas analíticas y computacionales para realizar el análisis y síntesis del sistema. En la elaboración del modelo se debe tener presente que se necesita un compromiso entre la exactitud y la simplicidad, evitando caer en cualquiera de los dos extremos. El modelo a elaborarse debe ajustarse al sistema físico, siendo conveniente, a veces, ignorar ciertas linealidades y parámetros distribuidos que originan derivadas parciales. Si los efectos de estas características en la respuesta son pequeños, se logra un buen resultado en el análisis del sistema.

Se debe tener presente que los parámetros distribuidos producen poco efecto en operaciones de baja frecuencia, pero estos parámetros pueden influenciar enormemente en la respuesta a alta frecuencia.

Se aconseja elaborar primero un modelo simplificado para tener una idea global del mismo, y posteriormente, elaborar un modelo más complicado, que nos dé una respuesta más exacta del sistema.

En los sistemas dinámicos continuos podemos establecer una diferencia en base a la linealidad de los mismos. Un sistema que satisfaga el *Principio de Superposición* es lineal, esto es, si y solo si:

$$H(ax_1 + bx_2) = aHx_1 + b\,Hx_2$$

donde **a** y **b** son constantes y $\mathbf{x_1}$ y $\mathbf{x_2}$ son señales de entrada.

Aunque muchas relaciones físicas son representadas por ecuaciones lineales, en la mayoría de los casos, estas relaciones son no-lineales. En estos sistemas, no es aplicable el *Principio de Superposición* y el proceso para hallar la solución de este tipo de problema, se vuelve muy complicado. Es por esto, que frecuentemente se crean sistemas lineales equivalentes en reemplazo de los no-lineales. Estos sistemas son válidos en un rango restringido de operación.

Para análisis de sistemas con una sola entrada y una sola salida, el modelo matemático más conveniente es la utilización de la *Función de Transferencia* del mismo. Se denominan *Funciones de Transferencia* a todas aquellas funciones que caracterizan las relaciones de entrada - salida de sistemas lineales invariantes en el tiempo. La *Función de Transferencia* de un sistema esta definida como la relación de la *Transformada de LaPlace* de la respuesta a la *Transformada de LaPlace* de la entrada. Esta *función de transferencia* es una expresión en términos de los parámetros del sistema y es independiente de la *función excitatriz* o de entrada. Pero no provee ninguna información respecto de la estructura física del sistema.

Por ejemplo un sistema invariante en el tiempo esta definido por la ecuación diferencial:

$$a_0y_n + a_1y_{n-1} + \ldots + a_ny = b_0x_n + b_1x_{n-1} + \ldots + b_{n-1}x_1 + b_nx$$

donde **x** es la entrada y la salida es **y**, se tendrá que:

$$\textit{Función de Transferencia} = G(s) = Y(s) / X(s)$$

$$G(s) = [b_0s^n + b_1s^{n-1} + \ldots + b_{n-1}s^1 + b_n] / [a_0s^n + a_1s^{n-1} + \ldots + a_{n-1}s^1 + a_n]$$

1.1.2 Diagrama de Bloques (Transformada de LaPlace)

Habíamos analizado anteriormente, que un sistema es un conjunto de componentes y que cada uno de ellos cumple una función. Estas funciones pueden ser mostradas por medio de un *Diagrama de Bloques*. Un *Diagrama de Bloques* no es más que una representación gráfica de

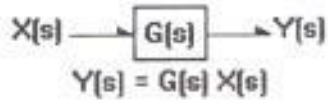
las funciones de cada componente de un sistema y del flujo de las señales. El *Diagrama de Bloques* permite visualizar de una manera real el sistema, a diferencia de una representación matemática, en la cual el sistema es visto en forma abastracta.

Incluso es más fácil comprender el funcionamiento de un sistema por medio del diagrama de bloques que examinando el sistema físico en sí. Todas las variables del sistema se enlazan entre sí por medio de los bloques funcionales. Cada uno de estos bloques es un símbolo que representa la operación matemática realizada sobre la señal de entrada para obtener la señal de salida. En el bloque se coloca la *Función de Transferencia* correspondiente. Los bloques se unen unos con otros por medio de *señales unidireccionales*, que muestran el flujo de información.

Por ejemplo:

$$G(s) = Y(s) / X(s)$$

se representará:



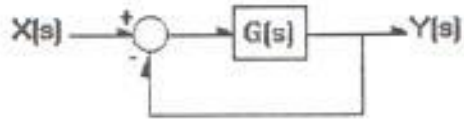
$$Y[s] = G[s]\,X[s]$$

La flecha que apunta al bloque es la señal de entrada y la que se aleja es la señal de salida.

Los bloques funcionales pueden estar relacionados por medio de un detector de errores, que emite una señal que es la diferencia entre la referencia de entrada y la señal de retro-alimentación del sistema de control.

Por ejemplo:

$$G(s) / [1 + G(s)] = Y(s) / X(s)$$



Para elaborar el diagrama de bloques de un sistema, primero se escriben las ecuaciones del sistema, las mismas que deberán describir el comportamiento dinámico de cada componente. Luego se toman las *Transformadas de LaPlace* de esas ecuaciones, suponiendo en algunas ocasiones condiciones iniciales 0. Se representa cada ecuación de LaPlace como un bloque. Se juntan los elementos y se obtiene un *Diagrama de Bloques*.

Para representar un sistema con varias señales de entrada y salida, se usa una interconexión de bloques.

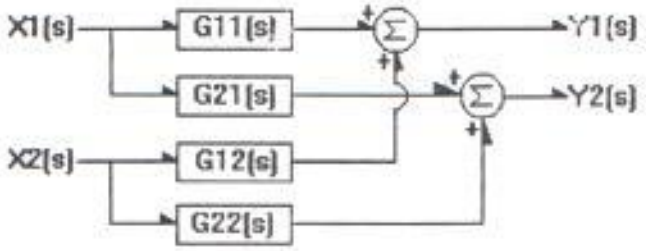
Por ejemplo, las ecuaciones:

$$Y_1(s) = G_{11}(s)X_1(s) + G_{12}(s)X_2(s)$$

$$Y_2(s) = G_{21}(s)X_1(s) + G_{22}(s)X_2(s)$$

representa el sistema:

En donde $G_{ij}(s)$ representa la *Función de Transferencia* que relaciona la variable de Salida i, con la Entrada j.



En general, para J entradas e I salidas, escribimos las ecuaciones simultáneas en forma matricial, así:

$$Y_1(s) = [G_{11}(s)\ \ G_{12}(s) \ldots G_{1J}(s)]\ \ [X_1(s)]$$

$$Y_2(s) = [G_{21}(s)\ \ G_{22}(s) \ldots G_{2J}(s)]\ \ [X_2(s)]$$

$$\cdot \quad \cdot \quad \cdot \quad \cdot \quad \cdot$$

$$\cdot \quad \cdot \quad \cdot \quad \cdot \quad \cdot$$

$$Y_I(s) = [G_{I1}(s)\ \ G_{I2}(s) \ldots G_{IJ}(s)]\ \ [X_J(s)]$$

Una vez que tenemos un modelo en diagrama de bloques para un sistema determinado, se puede utilizar un computador analógico o digital para simular el sistema. De esta forma, para investigar un diseño no es necesario construirlo, sino que con las condiciones reales del sistema y señales de entrada reales, se puede investigar las señales de salida. Con un computador analógico, se simula el sistema usando la analogía entre el voltaje del amplificador electrónico y la variable del sistema que se modela.

Por lo general, un computador analógico tiene las funciones de *integración*, de *multiplicación entre 2 variables*, *sumadores de varias entradas*, *amplificación* y *constantes*, funciones con las

que ya puede realizarse una simulación. En un computador digital, se crea un programa en el cual se introducen funciones tales como *integración, suma de señales, multiplicación entre señales*, etc. Se tienen las mismas funciones del analógico. La ventaja del uso del computador digital es la de poder utilizar *funciones no lineales* y *generadores de números al azar*, con lo que la simulación puede realizarse de un modo más eficaz. Además, la simulación del sistema se la hace en función del tiempo, con lo cual, la simulación es más exacta. Se trabaja con las variables reales del sistema.

1.2 DEFINICION DEL PROBLEMA

Hasta la fecha se disponía de un Computador Analógico como único medio de simulación para sistemas dinámicos continuos. Esto quiere decir que el diseñador no solamente tenía que plantearse las ecuaciones diferenciales del modelo, además tenía que realizar el diagrama de bloques, posteriormente armar el circuito de Computador Analógico, y *conseguir dichos elementos electrónicos* colocándolos en un proto-board energizado. No se tenía noción del tiempo total de simulación, a menos que se utilizase un cronómetro o reloj. Quizás uno de los aspectos más trágicos era tener que cambiar los elementos del circuito, si es que se requería modificar el valor de alguno de los parámetros (por ejemplo coeficientes de amortiguamientos, frecuencias naturales de oscilación, etc). Para graficar las salidas era necesario tener un osciloscopio.

Posteriormente el Computador IBM 1130 introdujo una cierta ventaja: usar un computador como medio de simulación, sustituyendo al proto-board energizado. En dicha ocasión se abrevió el tener que conseguir los elementos (que inclusive eran en cantidades limitadas). Sólo era necesario tener el circuito de elementos de Computador Analógico, para luego introducirlos en el computador digital IBM. Lastimosamente tanto el programa como el método de introducción de dichos elementos, eran rudos; la palabra correcta sería *poco amigables*. Recordemos que dicho programa fue desarrollado en un lenguaje muy mecánico, el FORTRAN. Además no había tanta difusión de dicho programa, ya que solo corría en los terminales de la Universidad

Politécnica (solo personas con acceso a terminales y pertenecientes a Control Automático podían usarlo).

Fue en aquella situación que la actual Ing. Ruth Santana se propuso diseñar el mismo programa CSMP, pero en esta ocasión que corriera en computadores personales. Para evitarse mayores complicaciones, decidió utilizar el mismo lenguaje de programación pero con mayores limitaciones (variables de no tan alta precisión como las del IBM). Sin embargo a pesar de haber tenido éxito en su tesis, ella nunca consideró el punto de vista del usuario, *hacerle fácil el aprendizaje*. El lenguaje FORTRAN seguía presentando el ambiente rudo y poco atractivo para ingresar los valores. Una y otra vez el diseñador tenía que verificar la validez de lo que ingresaba y corregir sus propios errores (el programa no estaba al tanto de los errores involuntarios cometidos por el usuario). El gran éxito logrado fue la portabilidad del software a una mayor audiencia dentro de la facultad. Quizás una gran limitación fue la restringida cantidad de ciertos elementos (máximo 25 integradores, por ejemplo) y el hecho de solo poder graficar una sola salida cada vez que se corría la simulación.

En mi presente trabajo desarrollé un nuevo CSMP capaz de ser fácil y amigable al usuario. No solo ofrece interfases gráficas, sino una amplia ayuda en línea. En el ambiente actual donde Windows reina, este es justo un miembro más de dicha gran familia de productos. El programa anterior de CSMP lamentablemente no creció con la tecnología. Tenía la gran limitante de solo visualizarse en monitores CGA y EGA. La actual versión para Windows permite sacar provecho de la resolución de un VGA. No tiene límites para la cantidad de elementos que se puede graficar, inclusive todos los del circuito. Tampoco existen limitantes para el número de elementos dentro de un circuito. Sin embargo el actual máximo de elementos es de 100 (en total). Además ofrecemos la facilidad de usar una impresora tanto para imprimir los gráficos como el listado de los elementos. Se pueden considerar varios circuitos en una misma sesión de CSMP. Lo más resaltable es que se pueden alterar los parámetros de cualquier elemento, y ver la nueva curva lograda. Podemos añadir y eliminar elementos ya no deseados, y tomar hasta un máximo de 50 muestras para lograr una curva más suave. El programa es de fácil uso y

entendimiento a cualquier usuario (claro está, que posea conocimientos de Control Automático). El lenguaje de programación fue en esta ocasión Visual C++, considerado el *más moderno lenguaje orientado a objetos.*

Este programa quizás traspase las barrreras de la Escuela Politécnica, y sea distribuido comercialmente.

1.3 OBJETIVOS

Mis objetivos para el presente trabajo son:

- Deseo proporcionar a la Comunidad Investigadora de un programa para PC capaz de simular un sistema físico dinámico y continuo.
- Mejorar las versiones anteriores del CSMP incluyendo en esta ocasión la facilidad del manejo y comprensión típico del ambiente Windows.
- Proporcionar al Laboratorio de Control Automático, una nueva herramienta de simulación-investigación que resulte fácil de difundirse entre el alumnado.
- Este programa dará un nuevo enfoque al Control Automático. Posiblemente hará realidad proyectos que anteriormente no tuvieron la oportunidad de ser construidos o al menos simulados.
- Reducir los costos dentro de la simulación, ya que no será necesario conseguir los elementos empleados en el Computador Analógico.
- Generar reportes y gráficas de salidas más precisas. En esta ocasión visualizando la escala de valores y del tiempo total de simulación.

1.4 DESCRIPCION GENERAL DEL PROGRAMA DE SIMULACION PARA SISTEMAS CONTINUOS (CSMP versión Windows 3.1)

CSMP es un programa de simulación de sistemas continuos y dinámicos. Inicialmente cada dato ingresado corresponde a un elemento de Computador Analógico (por lo general a un amplificador operacional y diodos). Típicamente se ingresan un identificador y/o numero que marca al elemento, los identificadores de los otros elementos que sirven como entradas al actual, y parámetros que normalmente son ganancias o condiciones iniciales para dicho elemento. El

programa posee un maximo de 25 elementos de Computador Analógico diferentes; desde los totalmente lineales, constantes y los no-lineales. El usuario esta libre de escoger cualquier combinación para formar su circuito. La unica condición a cumplirse siempre, es escoger por lo menos un integrador. Existe un máximo de 100 elementos por circuito. Se espera que con el avance de la tecnología dicho circuito sea ampliado a un mayor rango de elementos al igual que a una mayor cantidad. Se escogió dicha cifra (100 elementos por circuito como máximo) como protección a memorias RAM pequeñas. No deseamos que el proyecto deje de funcionar por falta de ella!!

Cada uno de los elementos involucra una relación funcional que incluye hasta máximo 3 bloques como entradas, y máximo 3 parámetros (pueden ser condiciones iniciales o bien ganancias). La salida de un elemento es un escalar de tipo real que viene dado por la relación funcional que cumple el mismo. Cada elemento es representado por un símbolo único, que aparece en la Barra de Herramientas de la Interfase Principal.

Si acaso el usuario no recuerda el uso o definición de algún elemento, o quizás no conoce el método para llenar cada casilla dentro de una Caja de Diálogo, se le ofrece una amplia ayuda en línea dentro del programa. Esta ayuda no solo informa de la forma representativa del elemento (sea el gráfico esquemático para Computador Analógico) sino también como trabaja en la vida práctica, y el orden y significado de cada casilla en las Cajas de Diálogo.

Como se dijo anteriormente cada elemento puede generar su propia salida en función del tiempo. Tanto gráficos como listados de elementos pueden ser presentados en la pantalla y opcionalmente en la impresora del usuario. Un diseñador puede ingresar los datos directamente desde el teclado o puede usar algún circuito grabado previamente como archivo.

CAPITULO 2
ANALISIS Y DISEÑO DEL PROGRAMA

2.1 REQUERIMIENTOS GENERALES DEL PROGRAMA

Obtener los requerimientos generales del CSMP no fue una tarea de ardua investigación. Bastaba ejecutar una copia de la versión anterior de CSMP para darse cuenta de las innumerables fallas, de diseño como de toque artístico, que sufría. Sirvieron comentarios de los mismos usuarios del CSMP, para fundar mis propios criterios sobre los defectos de las versiones anteriores. Opiniones de lo poco práctico que resultaba emplear el Computador Analógico, y de lo limitado de su alcance (además de lo costoso de cada elemento). Con el tiempo ambas versiones del CSMP fueron cayendo en desuso. Quizás por lo poco amigable, muy probablemente por la tecnología arcaica que se empleaba entonces.

En base a todos estos puntos decidí proponerme como nuevos requerimientos al ya existente CSMP, los siguientes:

- Ofrecer un sistema capaz de correr a la altura de la tecnología moderna y que sea capaz de ejecutarse en el ambiente más popular gráfico, Windows.
- Romper con el tabú creado por la Programación Orientada a Objetos que ronda entre las mentes de cada estudiante de Computación.
- Lograr interfases y ayudas muy amigables. No solo que lleguen al diseñador experimentado sino también al novato de Control Automático.
- Gráficos precisos, con datos de gran precisión y exactitud. Toda simulación espera generar resultados exactos y prácticos.
- Romper las limitaciones de un Computador Análogico, evitar los costos y precauciones al elaborar un circuito. Es humano que el diseñador se equivoque en algún dato.
- Poder comparar varios circuitos al mismo tiempo, y así decidirse por el más adecuado.
- Graficar cualquier elemento que el usuario desee, sin tener que usar un osciloscopio o un monitor de tipo antiguo.

- Poder alterar parámetros en cualquier momento, sin tener que volver a construir todo el circuito.
- Guardar datos en papel o bien en archivos, y poder usarlos una y otra vez.
- Un gráfico suficientemente exacto, con varias muestras que represente fielmente el comportamiento.

Como propósito personal, deseo probar que el Visual C++ es la mejor herramienta de programación orientada a objetos que existe en el mercado. Deseo probar la superioridad sobre el Borland C++.

2.2 ESPECIFICACIONES

Para el buen desempeño de la nueva versión del CSMP debemos tomar en cuenta ciertas especificaciones (mas no limitantes) que necesariamente se deben cumplir:

- A excepción del Sumador la mayoría de los elementos no pueden tener como entradas un valor negativo. Recordemos que la entrada de a un elemento simboliza el número o identificador de otro elemento cuya salida ingresa al actual. Todo identificador debe ser un número mayor que 0 y menor o igual que el máximo 100 por circuito.
El signo negativo para una entrada al sumador simboliza la operación de resta para dicho valor que ingresa al elemento. Esto quiere decir que el valor entrante se restara de los valores de las restantes entradas. La mayoría de los elementos no utilizan dicha operación.
- Siempre será necesario incluir por lo menos un integrador al circuito total. Se ha dicho que el integrador es el *alma* que moviliza a todo el circuito. Este es el elemento más dinámico que existe.
- La mayoría de los elementos que utilicen entradas, deben utilizar por lo menos la #1. Las excepciones son los sumadores, integradores y sumadores con pesos. Es lógico pensar que un elemento con entradas, debe usar por lo menos una entrada (mínimo la primera).

- Es indispensable ingresar un tiempo total de simulación para que el circuito corra. Además se necesita de una cantidad de intervalos/muestras (lo máximo posible son 50 muestras) para tener una curva suave y precisa.

- Toda entrada a un elemento necesariamente debe existir su identificador.

- No se asuste en caso de un desborde de algún valor calculado. El programa ha sido diseñado para mostrar mensajes de error en caso que ocurra en el calculo. Existen ciertos computadores PC que poseen menor rango para los desbordes, y emiten instantáneamente mensajes de error provenientes del Windows. Máquinas de procesador 386 en adelante, tienen un mayor rango y el programa alcanza a evitarlos.

- Una entrada no puede usar el mismo identificador de su propio elemento. Quizás esta sea una limitante al CSMP actual, ya que no permite que un elemento se retroalimente directamente. Será necesario incluir un elemento adicional en medio de dicha entrada.

- Trate de no ingresar valores de tiempo total demasiado altos. Si conoce con anticipación el tiempo en que la señal se estabiliza, ingréselo como parámetro. Valores muy altos de tiempo logran que el calculo se distorsione y altere la veracidad del gráfico.

- Recuerde que la capacidad de su memoria RAM es el mayor limitante para la cantidad de circuitos que se pueden abrir al mismo tiempo.

- A pesar de que se pueden tomar hasta máximo 50 muestras, la tabla mostrada a lado del gráfico solo incluirá las primeras 21 muestras. Las restantes se pueden apreciar al hacer *Revisión* y un *ZOOM In* del gráfico justo antes de imprimirse.

- Se han sugerido incluir otros tipos de elemento al repertorio actual del CSMP. Lastimosamente esto requerirá un rediseño total de las interfases, ya que existe poco espacio en pantalla para los nuevos.

- Sea consciente de la magnitud de los parámetros, o utilice escalas de magnitud para números demasiado grandes. Recuerde que esto puede llevarlo a futuros desbordes (pero el programa ha previsto dicha situación).

2.3 DISEÑO DEL PROGRAMA

2.3.1 Ingreso y Clasificación de Datos

Una vez que se tiene el modelo a simular, el siguiente paso es necesariamente la definición de las ecuaciones diferenciales. Cada ecuación describe el comportamiento de una porción integrante de todo el modelo. Para muchos diseñadores un paso que proporciona mayor facilidad previo a obtenerse el circuito de Elementos de Computador Analógico, es utilizar las ecuaciones convertidas a Diagrama de Bloques (usando la Transformada de LaPlace). Recordemos que cada bloque en el diagrama corresponde a un elemento de Computador Analógico. Tome en cuenta que no pueden existir más de 100 elementos entre los 25 tipos diferentes.

Ya que el computador es una máquina (de mentalidad mecánica, sin razonamiento) es indispensable nombrar a cada elemento del circuito con un número Identificador. Dicho número es único en todo el circuito, no puede existir otro gemelo del elemento. Para cada bloque debe especificarse también los identificadores de los bloques que conforman las entradas para el mencionado. Recuerde que solo el Sumador puede tener una entrada de signo negativo, ya que el menos implica una operación de resta. El tipo del elemento queda automáticamente determinado en el momento que el usuario escoge el botón con el símbolo representante de aquel. Por lo general este es un símbolo perteneciente a los 25 tipos diferentes existentes para esta versión de CSMP.

Además de las entradas (aunque algunos no las necesitan, por lo que no aparecen en la caja de diálogo) se debe ingresar los parámetros o condiciones iniciales que afectarán al elemento. Estos son valores reales sin ninguna restricción. Pocos son los casos de elementos que necesariamente deben tener un cierto parámetro (ya que por defecto sus valores son 0).

Existe el elemento *Generador de Funciones* que contiene una cantidad extra de datos a ingresarse: los 11 interceptos para interpolar. Estos también son valores reales sin restricción que servirán para dar la forma a una función ya existente, y con una entrada dada poder obtenerle un valor correspondiente dentro del rango de dichos interceptos.

En versiones anteriores del CSMP se habla de clasificar los datos ingresados en cada caja de diálogo, según su uso. Los diseñadores previos consideraron 3 grandes grupos, que quizás ayuden a familiarizarse al fiel usuario del CSMP:

-*Datos de Configuración*

-*Condiciones Iniciales y Parametros*

-*Generador de Funciones*

Datos de Configuración.-

Los autores anteriores consideraron como datos de configuración los siguientes valores:

- NOMBRE DE LA SALIDA: para la versión actual del CSMP no se consideró importante asignarle un nombre en especial a la salida de un elemento. Las curvas mencionan la numeración del bloque al cual pertenecen.
- NUMERO DEL BLOQUE: este es el clásico identificador asociado con el bloque del Diagrama de Bloques, o con el Elemento de Computador Analógico. Recordemos que este es único y que está entre el rango de 1 a 100.
- TIPO: el usuario no debe preocuparse por ingresar este dato, ya que basta la acción de seleccionar el botón representativo del tipo de elemento, para que dicho campo se llene automáticamente (tanto en memoria o en archivo, si el usuario desea grabar). Recuerde que incluimos 25 versiones diferentes de elemento.
- ENTRADA 1,2,3: típico identificador del bloque que proporciona el valor de entrada al bloque analizado. La salida de este bloque previo sirve de entrada al actual. Solo el Sumador es el único que puede incluir un identificador negativo como entrada. El elemento no necesariamente debe tener 3 entradas, inclusive según el tipo de elemento este puede que tenga ninguna.

Condiciones Iniciales y Parámetros.-

No todos los elementos poseen condiciones iniciales y/o parámetros (ganancias a sus entradas). Pero para los que sí los tienen debemos incluir los siguientes:

- NOMBRE DEL PARAMETRO: en la versión actual del CSMP no consideramos necesario nombrar en forma especial a algún parámetro o condición inicial; basta mencionar que forma parte del elemento analizado.
- NUMERO DEL BLOQUE: gracias a que todos los datos previamente mencionados y los pertenecientes a este grupo se llenan en la misma Caja de Diálogo, se entiende que los parámetros y condiciones iniciales ingresados pertenecen al identificador del elemento actual.
- CONDICION INICIAL O PRIMER PARAMETRO: muchos bloques utilizan una condicion inicial, este dato debe llenarse aquí. Claro está que existen otros que utilizan esta casilla para ingresar ganancias o valores de potenciómetros, asociados a la entrada 1.
- SEGUNDO Y TERCER PARAMETROS: si acaso el elemento los utiliza, estos valores son definitivamente ganancias asociadas a sus respectivas entradas.

Generador de Funciones.-

Recordemos que se consideraba esta clasificación para elementos Generadores de Funciones. Debemos registrar los siguientes valores:

- NUMERO DE BLOQUE: en la versión actual de CSMP no es necesario ingresar este dato en forma separada, ya que se considera el mismo identificador que se coloca para el elemento de la Caja de Diálogo.
- PUNTOS DE INTERCEPCION: ingresamos los 11 puntos que ayudaran a mapear una función, y a su vez ayudaran a interpolar la entrada con un valor de salida respectivo.

Finalmente recordemos que podemos grabar cualquier circuito en un archivo. El ingreso de datos se logra no solamente a través del teclado sino también a través de un archivo/circuito existente.

2.3.2 Técnicas para el Procesamiento de Información

2.3.2.1 *Método de Integración*

Recordemos que el circuito para Computador Analógico tiene sus orígenes en Ecuaciones Diferenciales. Además recordemos que el elemento considerado *dinámico* dentro de un circuito es el *Integrador*. Es muy importante contar con un método de integración efectivo, exacto y de un calculo corto.

Si la ecuacion diferencial es:

$$y' = f(x,y)$$

y se conoce la solución numérica en un punto (X_n, Y_n) podemos calcular un valor aproximado Y_{n+1} resolviendo para Y (X_{n+1}) donde

$$X_{n+1} - X_n = h$$

Podemos resolverlo usando 2 técnicas:

- La Expansión de la Serie de Taylor
- La Integral Definida

La Expansión de la Serie de Taylor se puede expresar en forma:

$$y(x_{n+1}) = y(x_n) + hy'(x_n) + [h^2/2]y''(x_n) + \ldots + [h^p/p!]y^p(x_n) + T_{n+1}$$

donde,

$$T_{n+1} = [h^{p+1}y^{p+1}(xx_{n+1})] / [p+1]! \qquad x_n < xx < x_{n+1}$$

y denota el error por truncación de la serie de Taylor después del término p. La Expansión de la serie de Taylor puede ser representada en la forma:

$$y(x_{n+1}) = y(x_n) + hf(x_n) + [h^2/2]f'(x_n) + \ldots + [h^p/p!]f^p(x_n)$$

y se desecha el término T_{n+1}.

La evaluación de las derivadas de orden p muy alto puede resultar un proceso lento y de muchos pasos. Hemos considerado un método que utiliza en forma indirecta la Expansión de Taylor, el de Runge - Kutta.

En general los métodos de Runge - Kutta evalúan f (x,y) en más de un punto de las cercanías de (x_n,y_n) en vez de evaluar las derivadas de f (x,y). El método de Runge - Kutta de orden p es equivalente al desarrollo de la serie de Taylor truncada en el término p.

Runge - Kutta de Segundo Orden.-

Este método ha sido desarrollado asumiendo que y_{n+1} se puede calcular como:

$$y_{n+1} = y_{n+h} [K_1 f(x_n,y_n) + K2f(x_{n+A}h,y_{n+B}h\, f(x_n,y_n))]$$

que es equivalente a la Expansión Cuadrática de Taylor:

$$y_{n+1} = y_n + h\, [f(x_n,y_n) + [h/2]\, f'(x_n,y_n)]$$

Relacionando estas dos expresiones, podemos obtener los valores de las constantes K_1, K_2, A y B.

$$K_1 + K_2 = 1$$

$$K_2A = 1/2$$

$$K_2B = 1/2$$

De donde:

$$K_1 = 1 - K_2$$

$$A = 1 / [2K_2]$$

$$B = 1 / [2K_2]$$

Si escogemos $K_2 = 0.5$, entonces:

$$K_1 = 0.5$$

$$A = 1$$

$$B = 1$$

Obtenemos:

$$y_{n+1} = y_n + [h / 2][f(x_n,y_n) + f(x_{n+h},y_{n+h}\, f(x_n,y_n))]$$

Este caso especial se conoce como el *Método de Euler Mejorado*. Usaremos este método para representar la operación de Integración en nuestras simulaciones.

2.3.2.2 *Generador de Numeros Aleatorios*

El elemento *Generador de Numeros Aleatorios* utiliza un *Método de Congruencias* para generar diversos números entre -1 y +1. Estos números son realmente pseudoaleatorios pues son deterministicos, debido a que los procesos aritméticos que se incluyen en el cálculo determinan cada término de la sucesión.

El método aplicado es el *Multiplicativo de Congruencias* en el cual se parte de un valor inicial el cual es multiplicado por un factor. Se toman los cuatro últimos dígitos del producto y se lo divide para 10000 y ese es el valor aleatorio. El número aleatorio se lo multiplica por 10000 y este resultado se lo multiplica por el factor y el ciclo se repite. El proceso asegura que puede generar hasta 500 numeros sin repetir la serie.

2.3.2.3 *Generador de Funciones*

El elemento *Generador de Funciones* utiliza los 11 interceptos en el plano X vs. Y para interpolar un nuevo valor que corresponda como Ordenada a la Entrada (considerada como Abscisa). En otras palabras, la entrada e_i al elemento generador de funciones servirá como una coordenada X, a la cual se la asociará con una coordenada Y. Si dicha coordenada X se encuentra en el rango entre p_2 y p_1 se le podrá mapear un valor en Y. Si es menor que p_2 se le asignará el valor del primer intercepto. Si es mayor que p_1 en cambio se le asignará el último intercepto.

2.3.3 Presentación de los Resultados

2.3.3.1 *Presentación por Pantalla*

Una vez que el usuario ha ingresado un circuito de mínimo un integrador, los parámetros de tiempo (tiempo total y cantidad de intervalos) y además ha verificado que todas las entradas a cada uno de sus elementos existen, podrá escoger cualquiera de los elementos integrantes para visualizar su salida en función del tiempo. Al instante en que el usuario hace doble click con el botón derecho de su ratón, automáticamente se presentará una ventana mostrando una tabla con los valores calculados para cada intervalo de la simulación y a su derecha el gráfico correspondiente al elemento escogido. Recordemos que la curva es Salida del Elemento vs.

Tiempo. La escala de Tiempo esta apropiadamente segmentada con tiempos pertinentes y claves de la simulación. Algo parecido se ha realizado con la escala de las Ordenadas.

Debido a que la cantidad total de intervalos puede llegar hasta 50, no es posible mostrar la tabla con las 50 muestras tomadas. En la pantalla mostraremos apenas las 21 primeras muestras. Las restantes podrán visualizarse en modo *Revisión* haciendo un *ZOOM In.*

2.3.3.2 ***Presentación por Impresora***

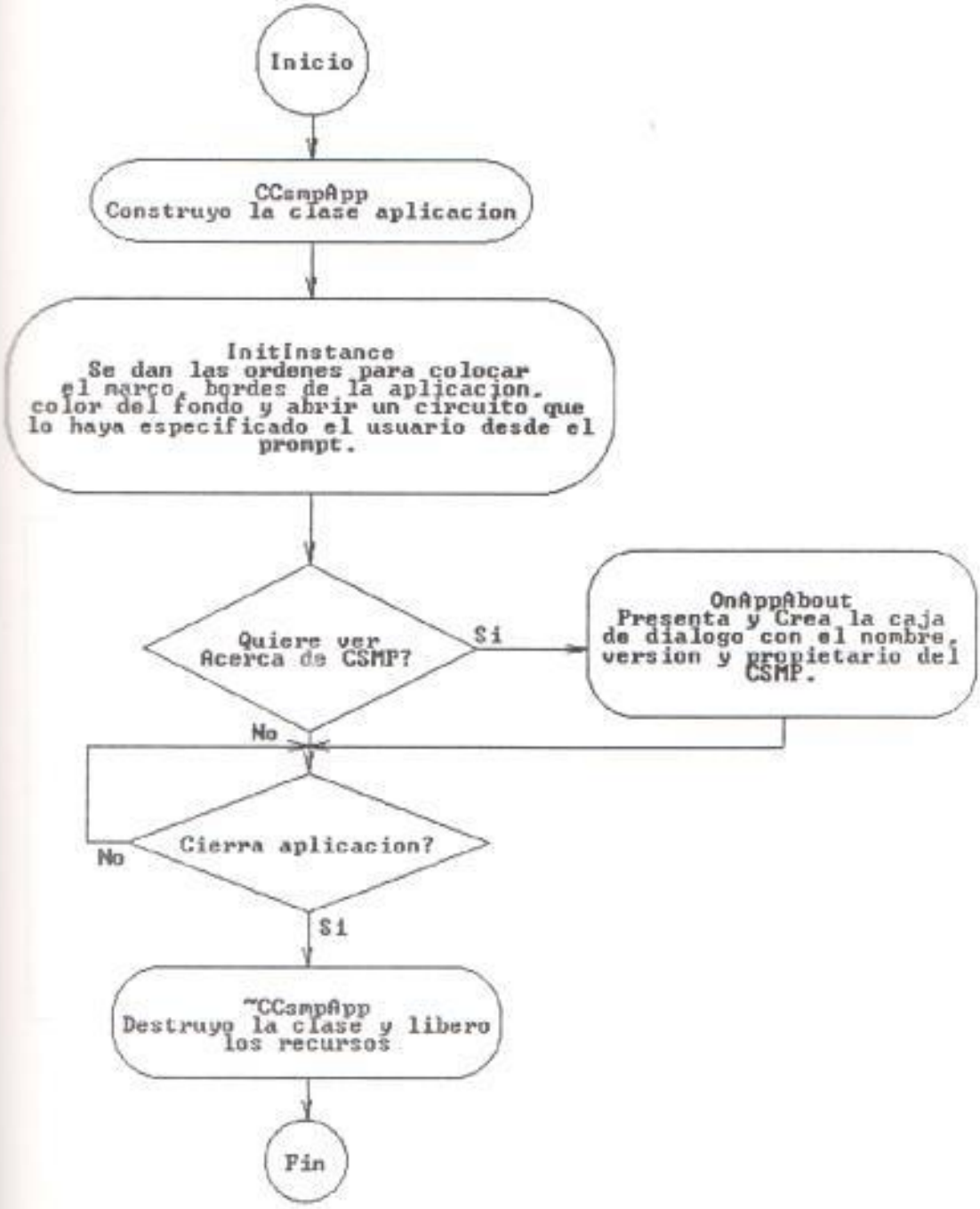
No existe una diferencia notable con respecto a la gráfica por pantalla. Sin embargo debe destacarse el uso de la opción *Revisión* dentro del Menu *Archivo.* Es aquí donde el usuario podrá ver la tabla completa con todas las muestras tomadas durante la simulación (ademas del gráfico). En la hoja de papel se incluirán tanto la tabla de muestras (completa) junto al gráfico de la curva.

CAPITULO 3

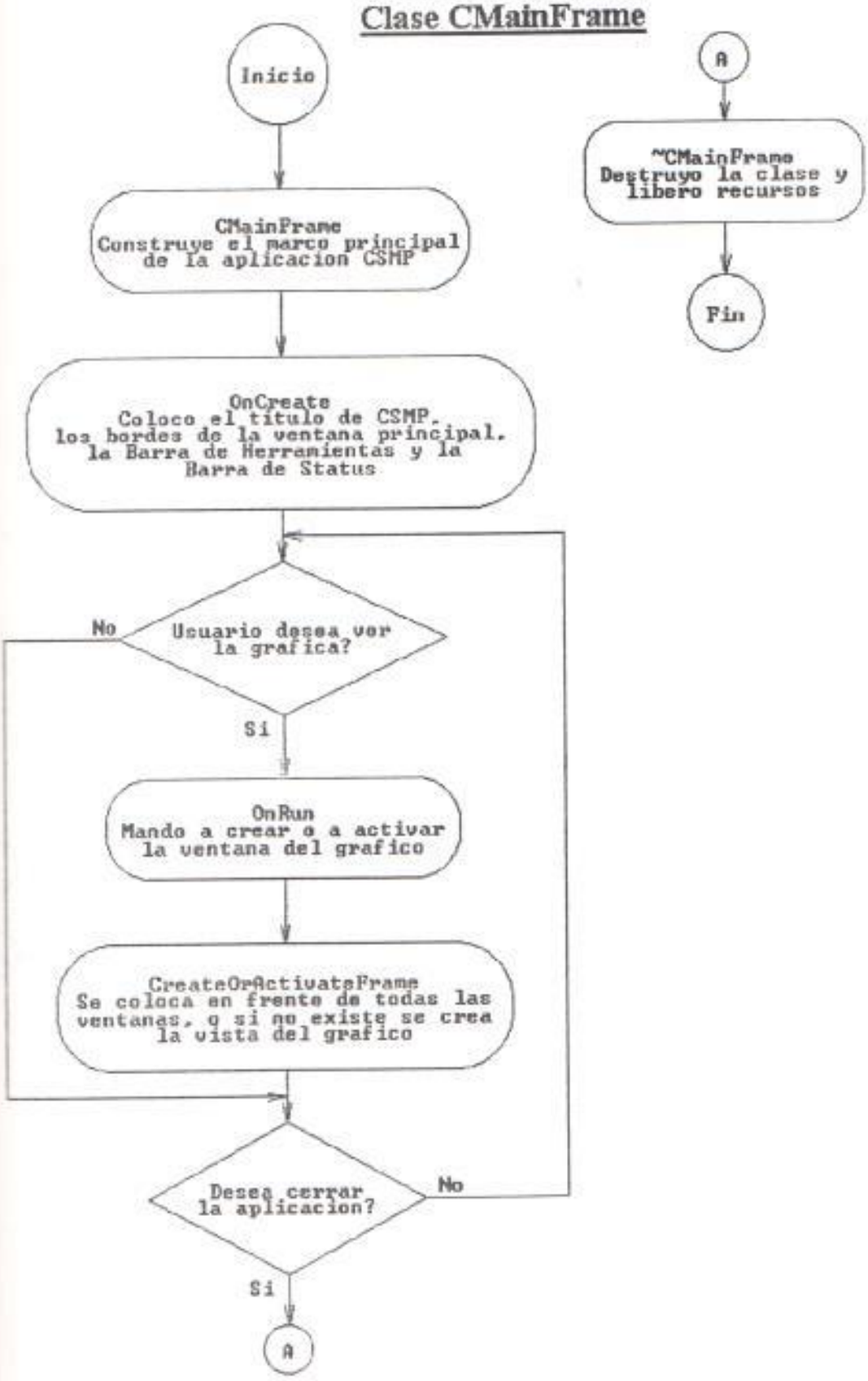
IMPLEMENTACION DEL PROGRAMA

3.1 DIAGRAMA DE FLUJOS

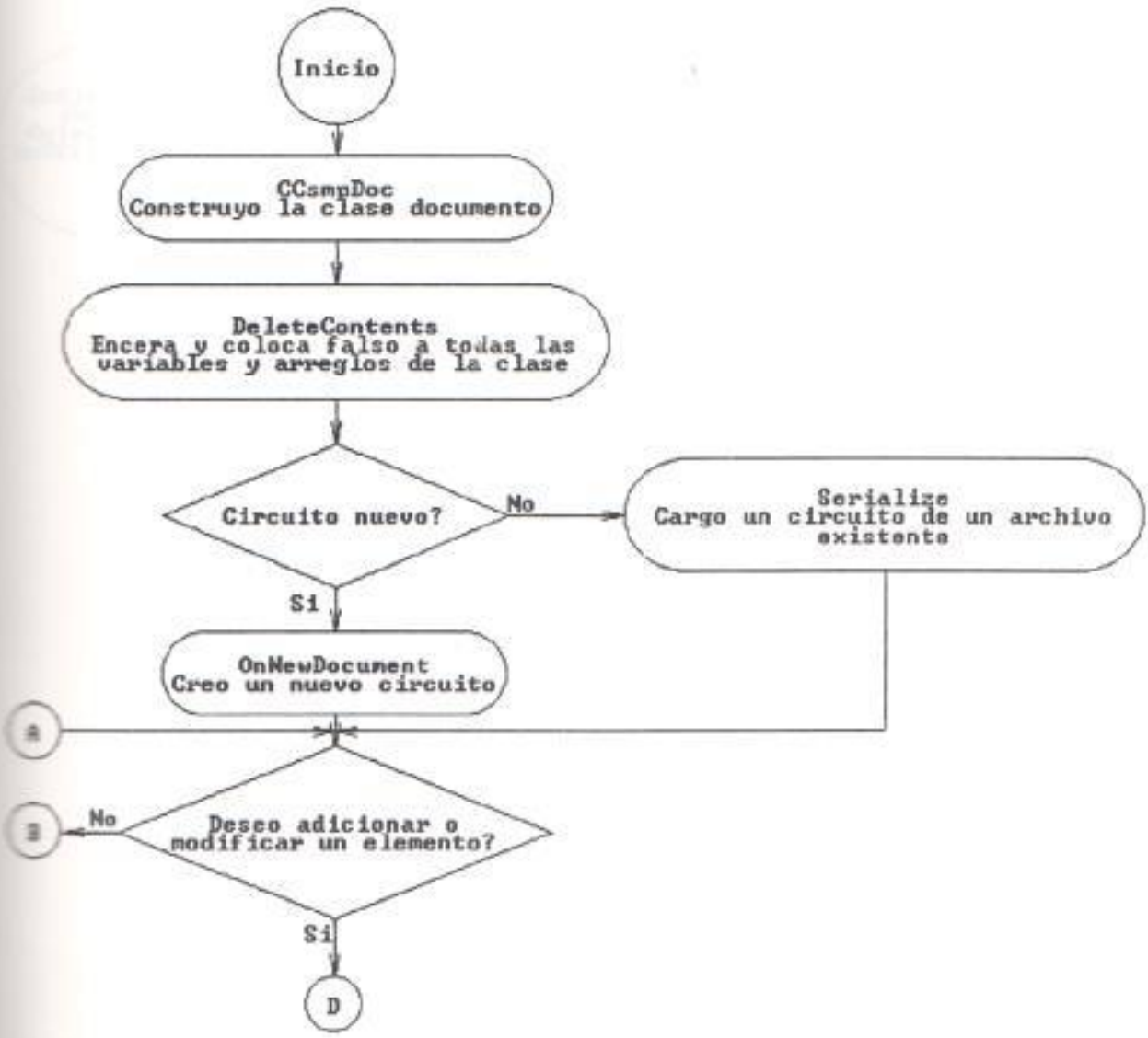
Clase CCsmpApp



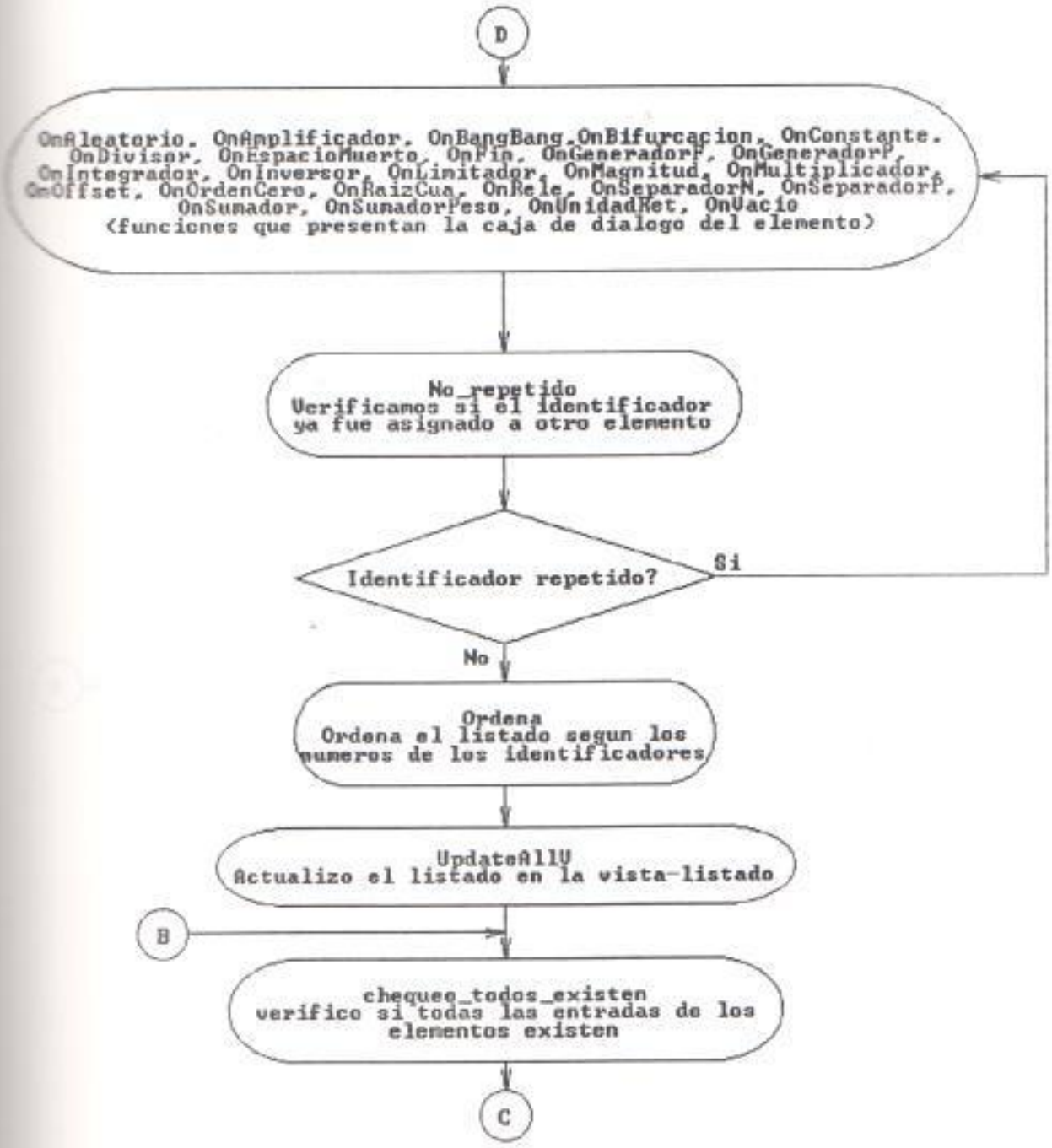
Clase CMainFrame



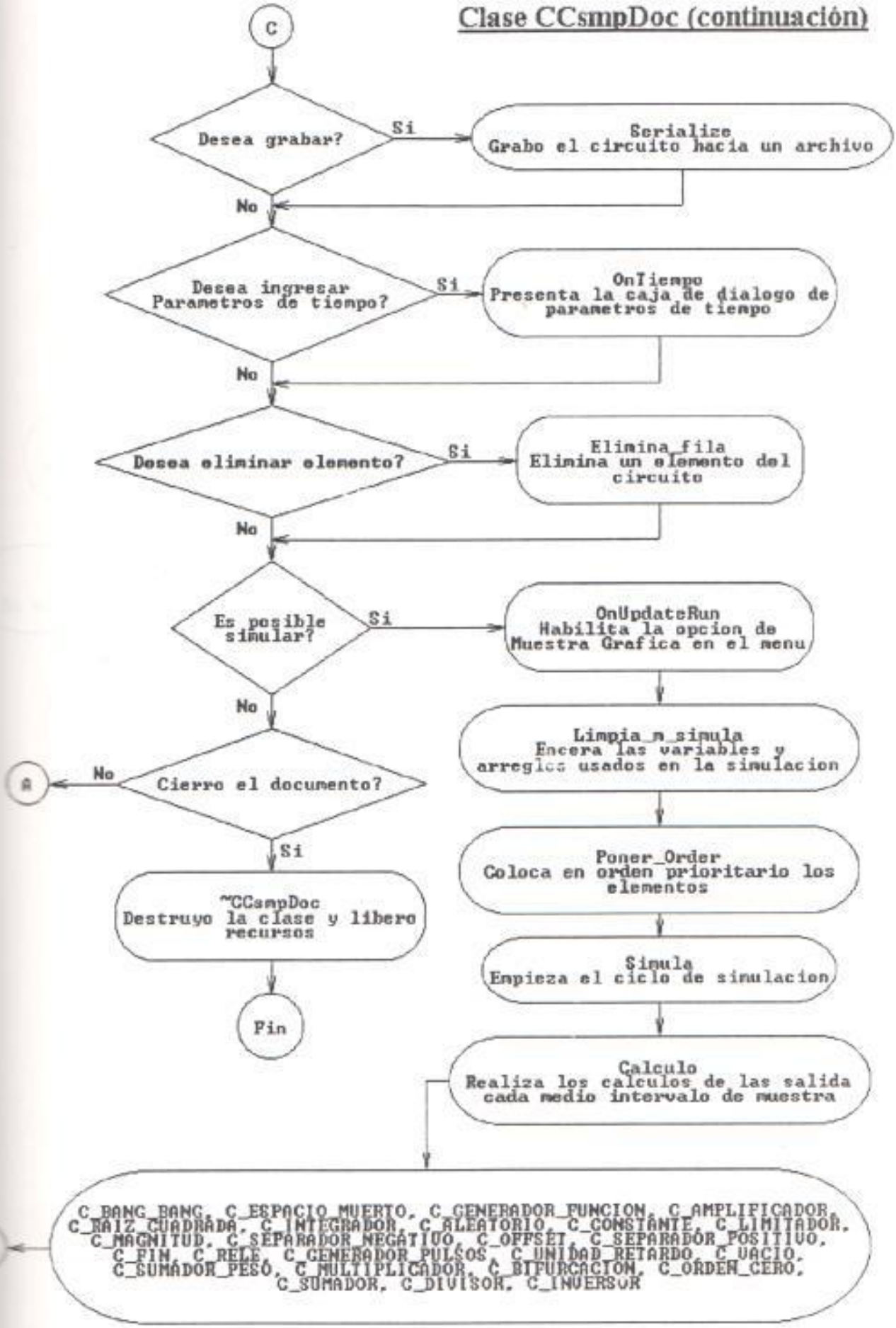
Clase CCsmpDoc



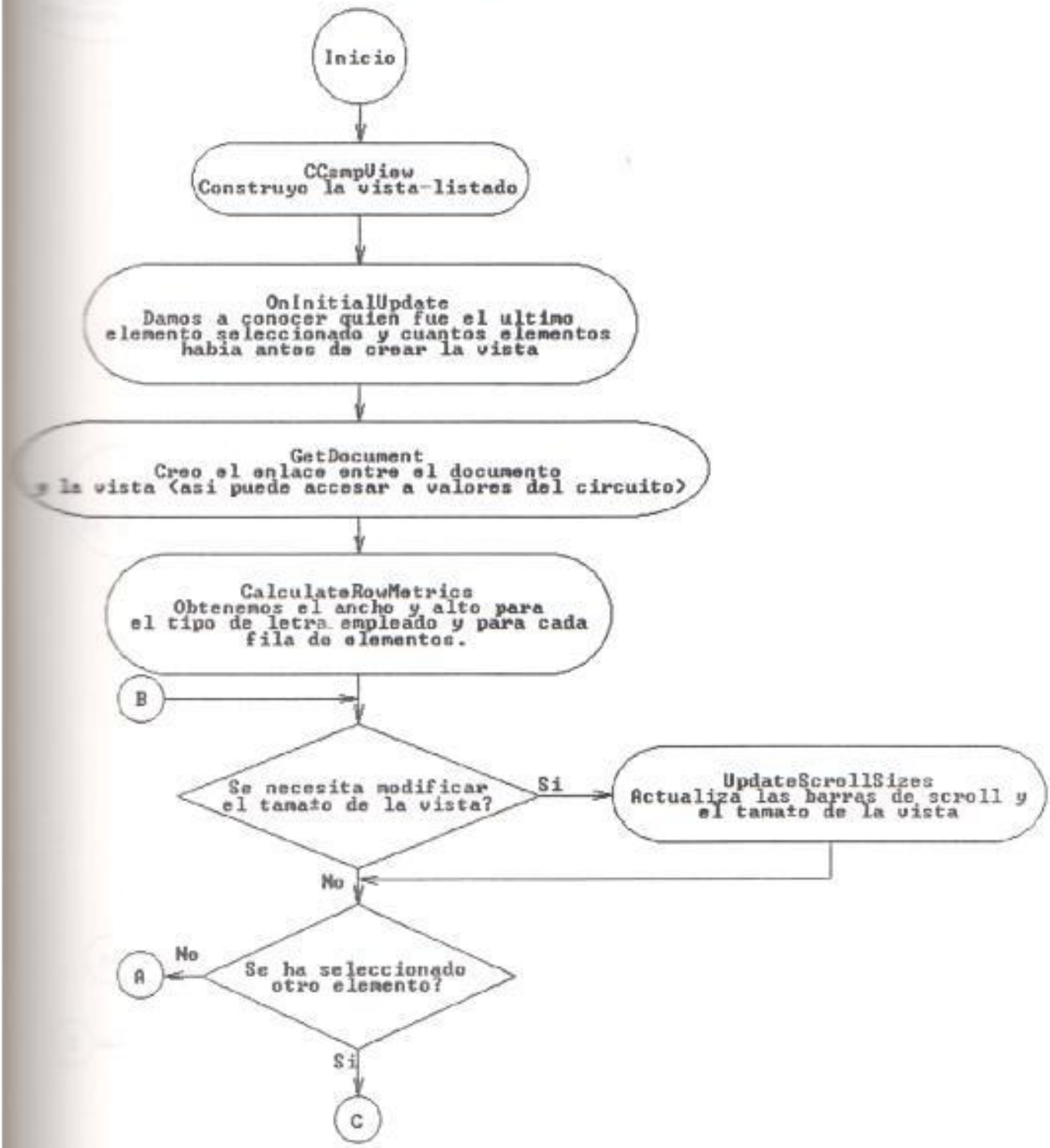
Clase CCsmpDoc (continuación)



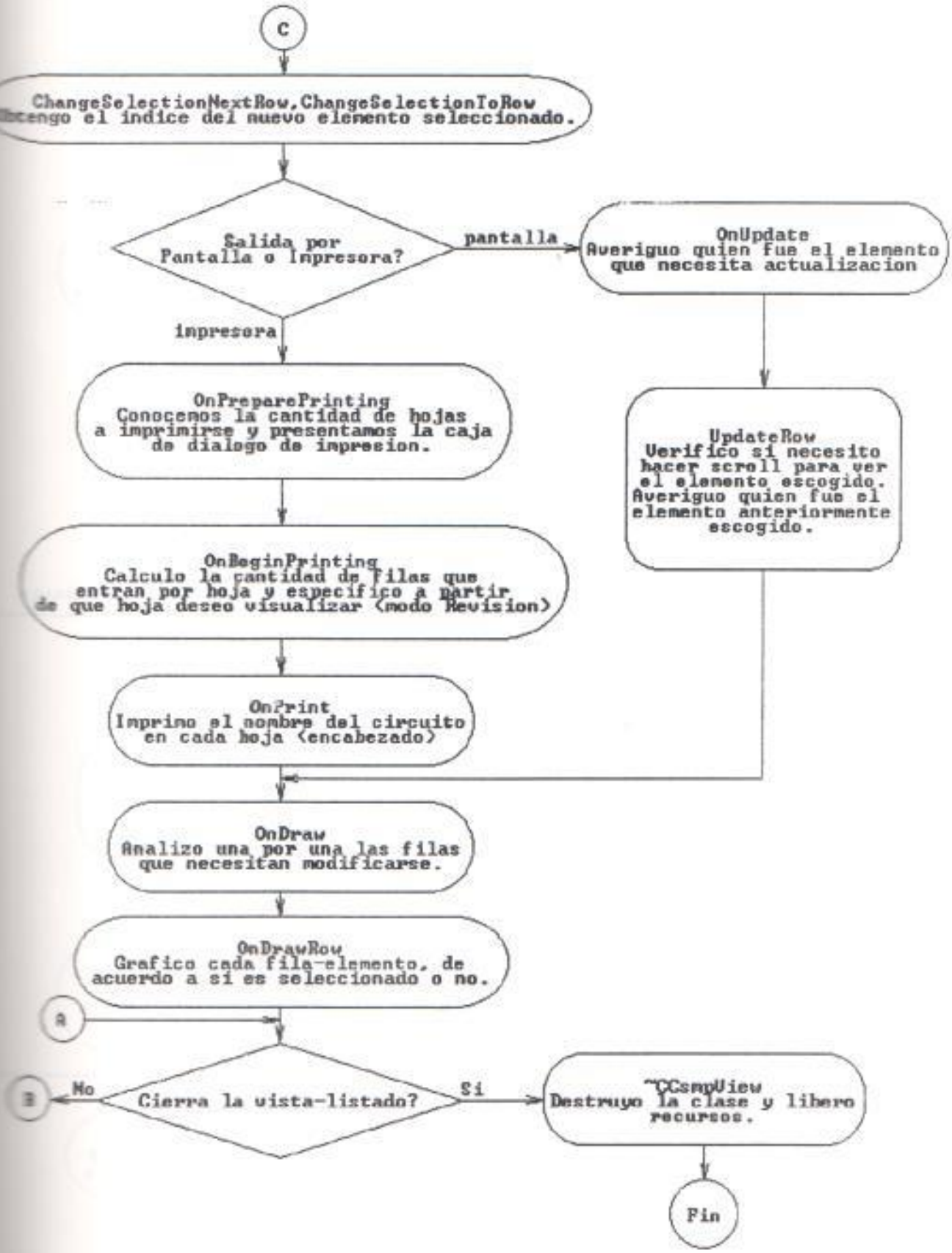
Clase CCsmpDoc (continuación)



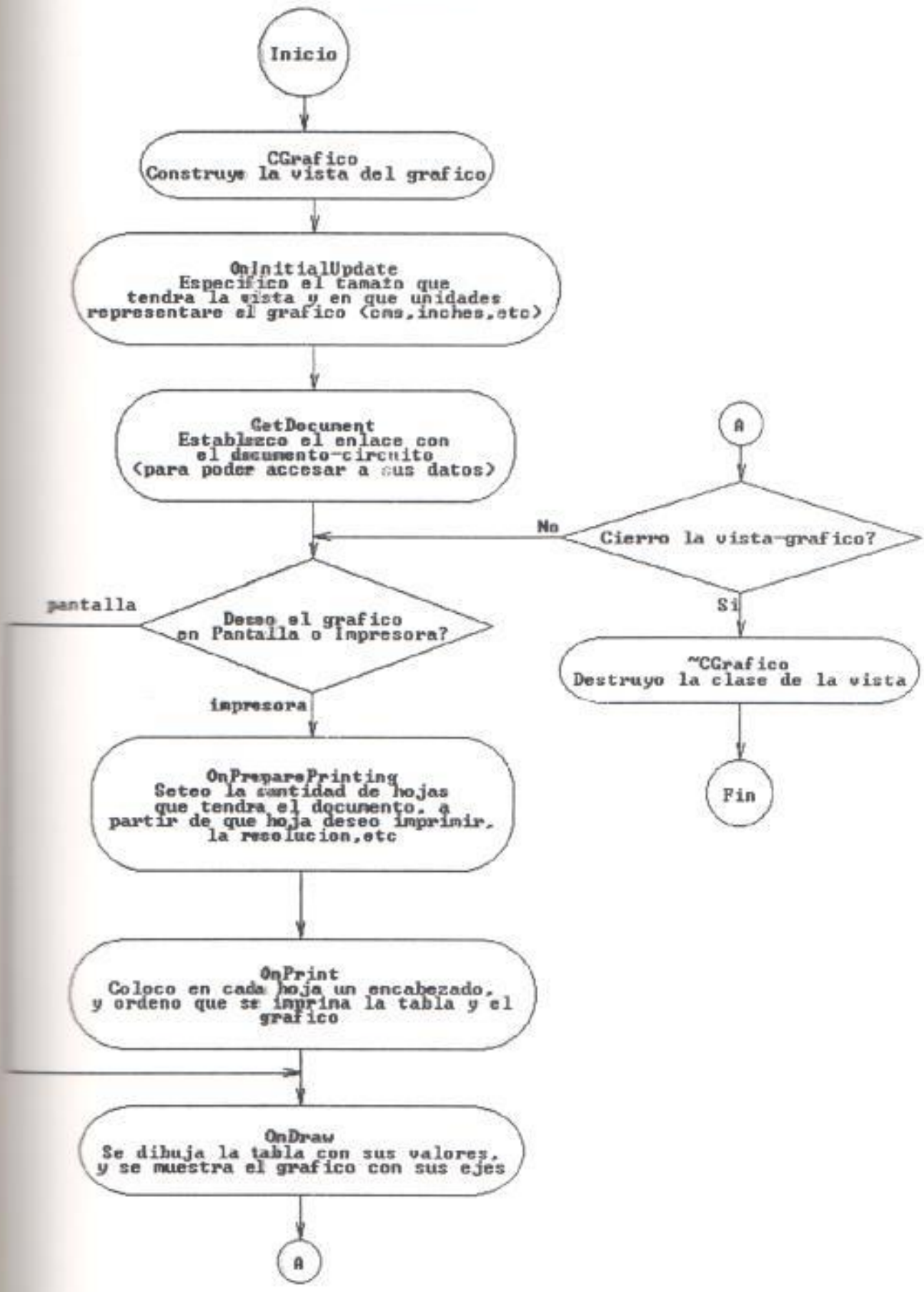
Clase CCsmpView



Clase CCsmpView (continuación)

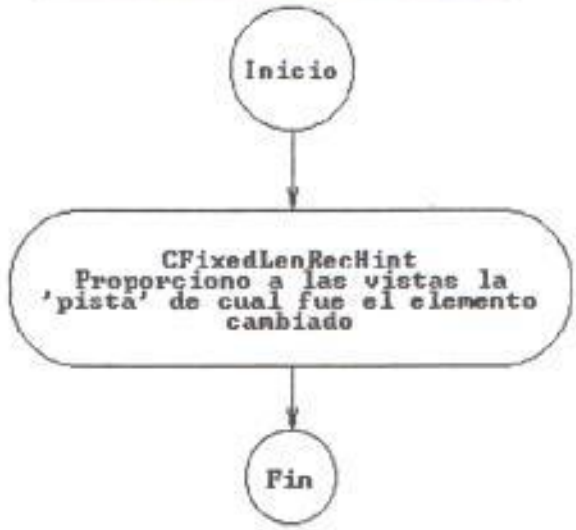


Clase CGrafico

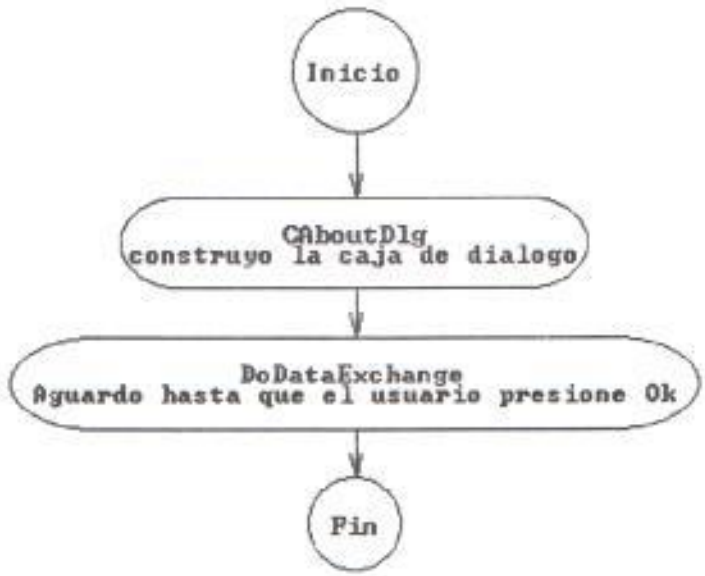


Clase CGFrame

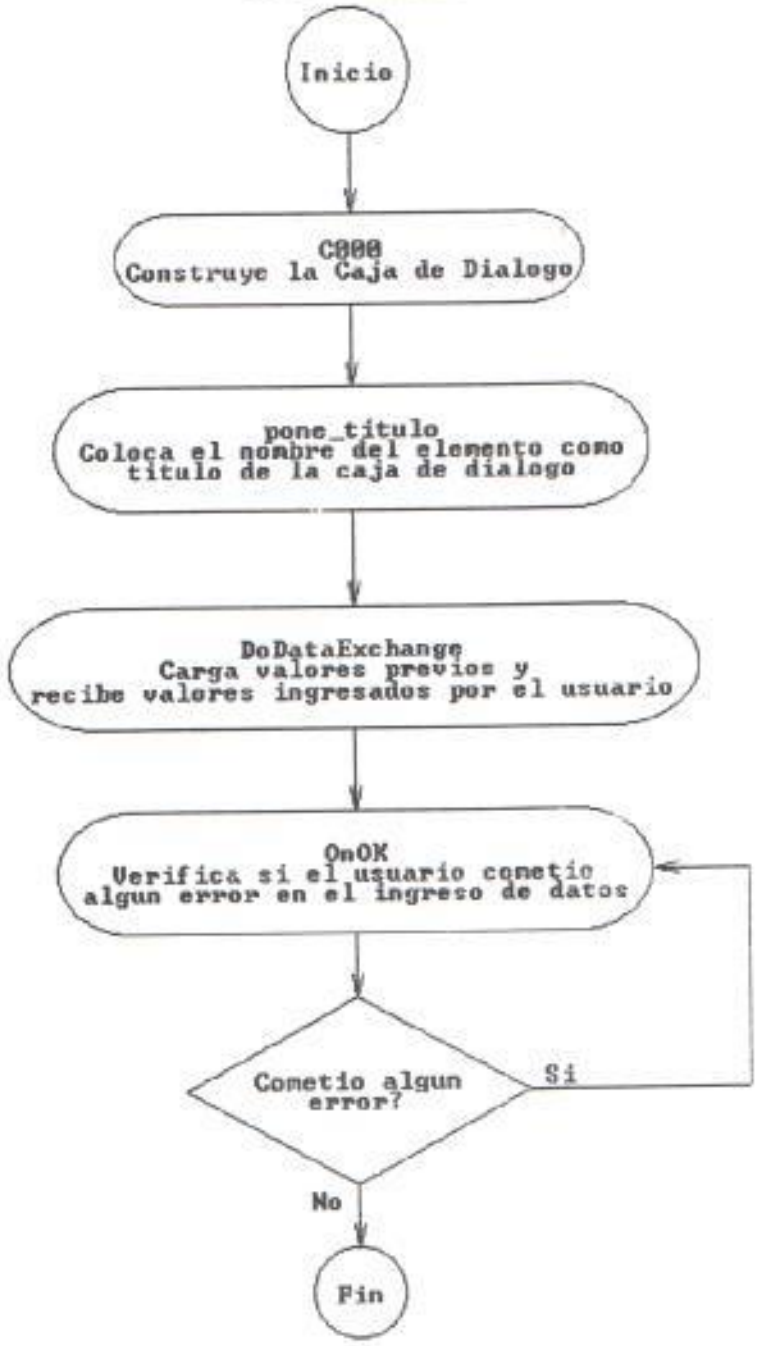
Clase CFixedLenRecHint

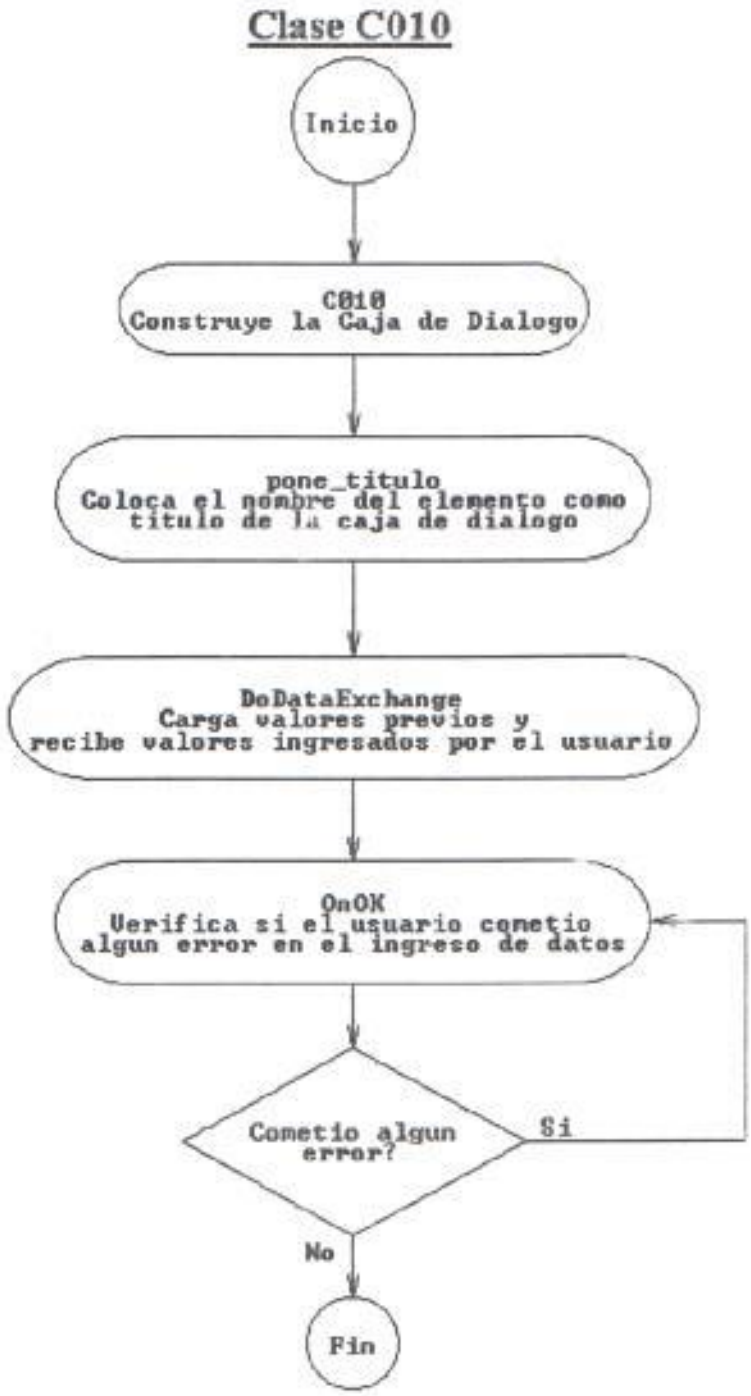


Clase CAboutDlg

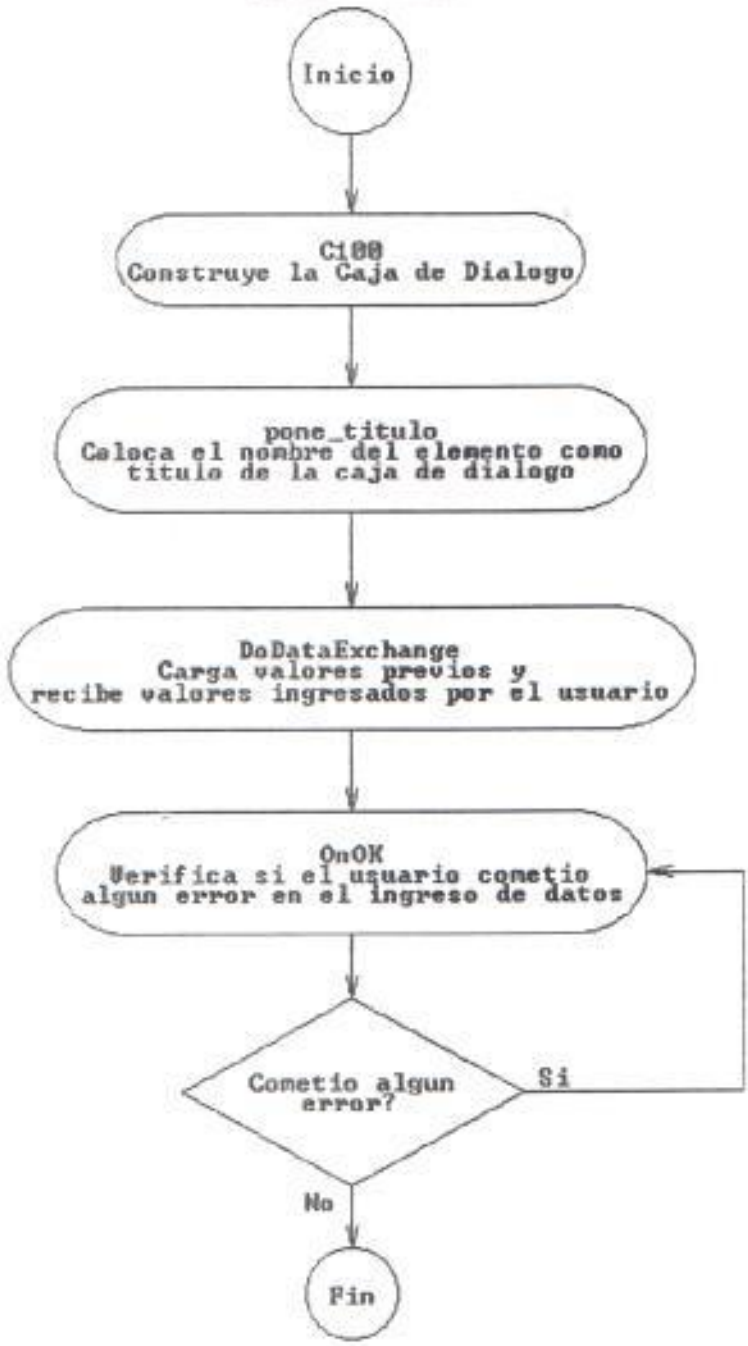


Clase C000

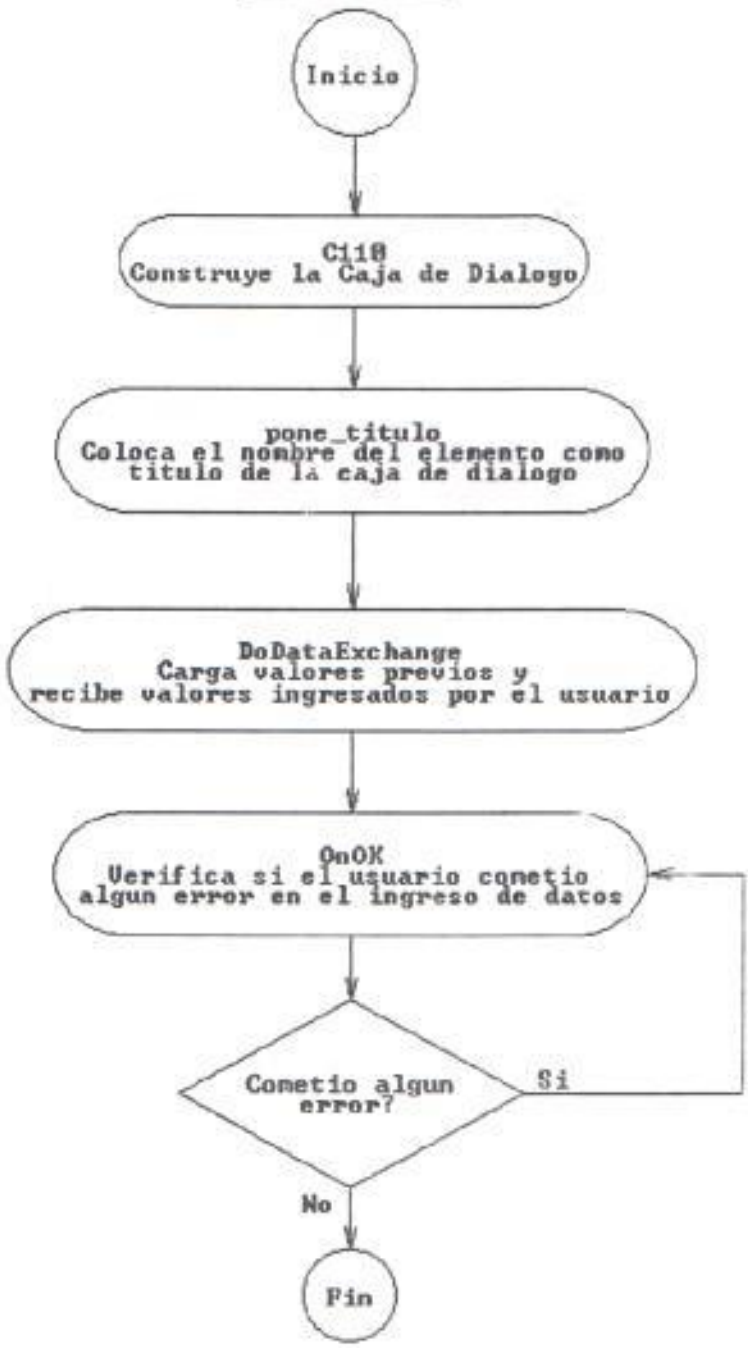

Inicio
C000
Construye la Caja de Dialogo
pone_titulo
Coloca el nombre del elemento como titulo de la caja de dialogo
DoDataExchange
Carga valores previos y recibe valores ingresados por el usuario
OnOK
Verifica si el usuario cometio algun error en el ingreso de datos
Cometio algun error?
Si
No
Fin


Clase C010
Inicio
C010
Construye la Caja de Dialogo
pone_titulo
Coloca el nombre del elemento como
titulo de la caja de dialogo
DoDataExchange
Carga valores previos y
recibe valores ingresados por el usuario
OnOK
Verifica si el usuario cometio
algun error en el ingreso de datos
Cometio algun
error?
Si
No
Fin

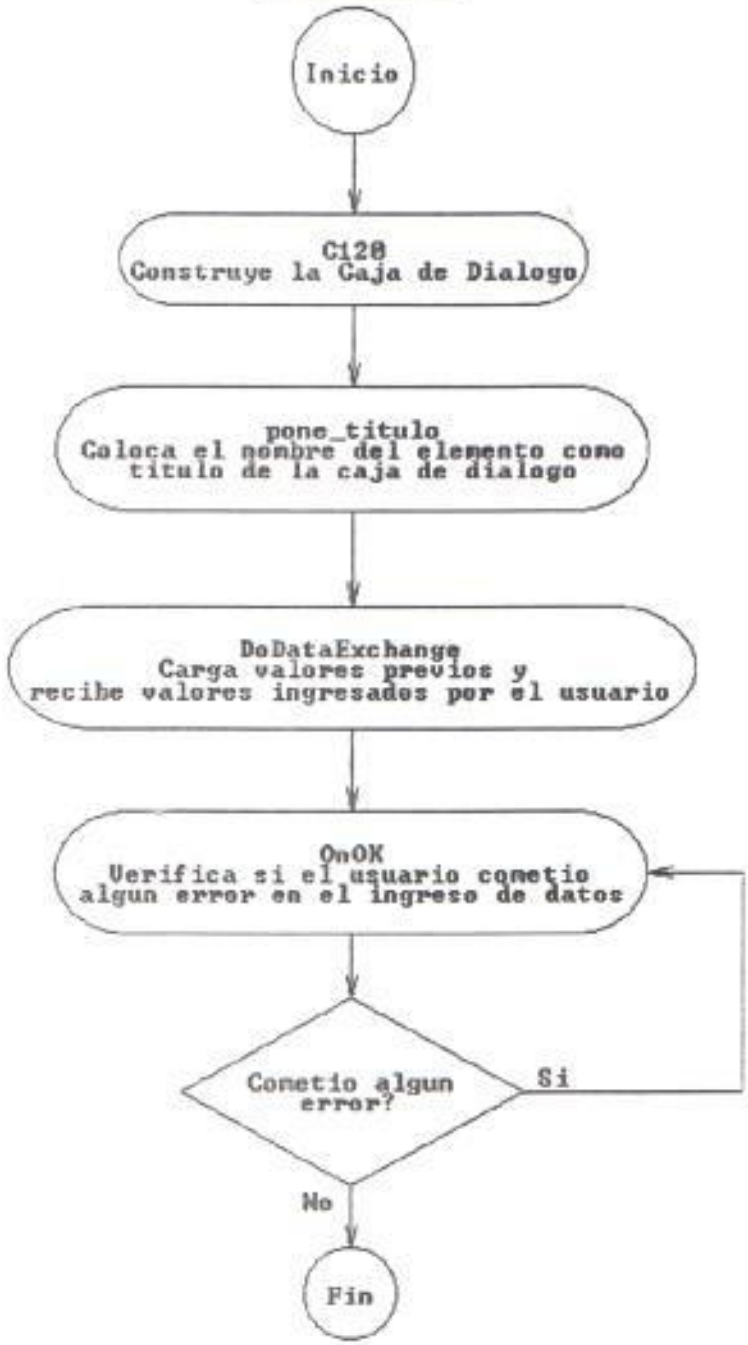
Clase C100


Inicio
C100
Construye la Caja de Dialogo
pone_titulo
Coloca el nombre del elemento como
titulo de la caja de dialogo
DoDataExchange
Carga valores previos y
recibe valores ingresados por el usuario
OnOK
Verifica si el usuario cometio
algun error en el ingreso de datos
Cometio algun
error?
Si
No
Fin

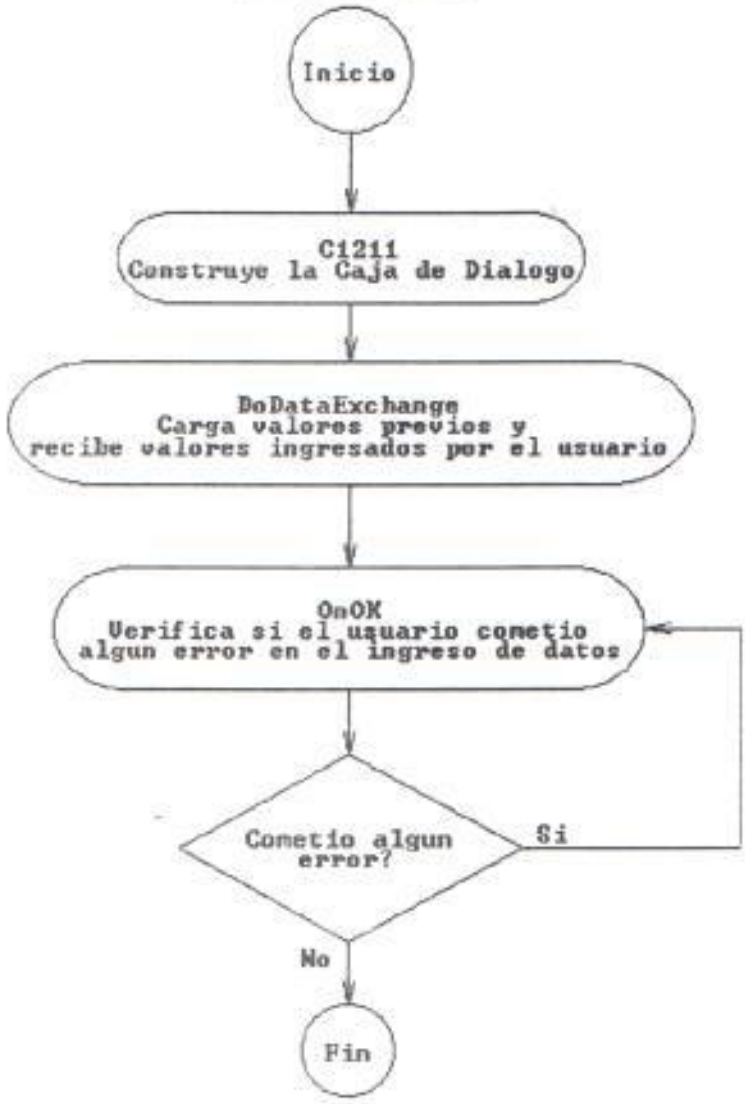
Clase C110


Inicio
Ci10
Construye la Caja de Dialogo
pone_titulo
Coloca el nombre del elemento como titulo de la caja de dialogo
DoDataExchange
Carga valores previos y recibe valores ingresados por el usuario
OnOK
Verifica si el usuario cometio algun error en el ingreso de datos
Cometio algun error?
Si
No
Fin

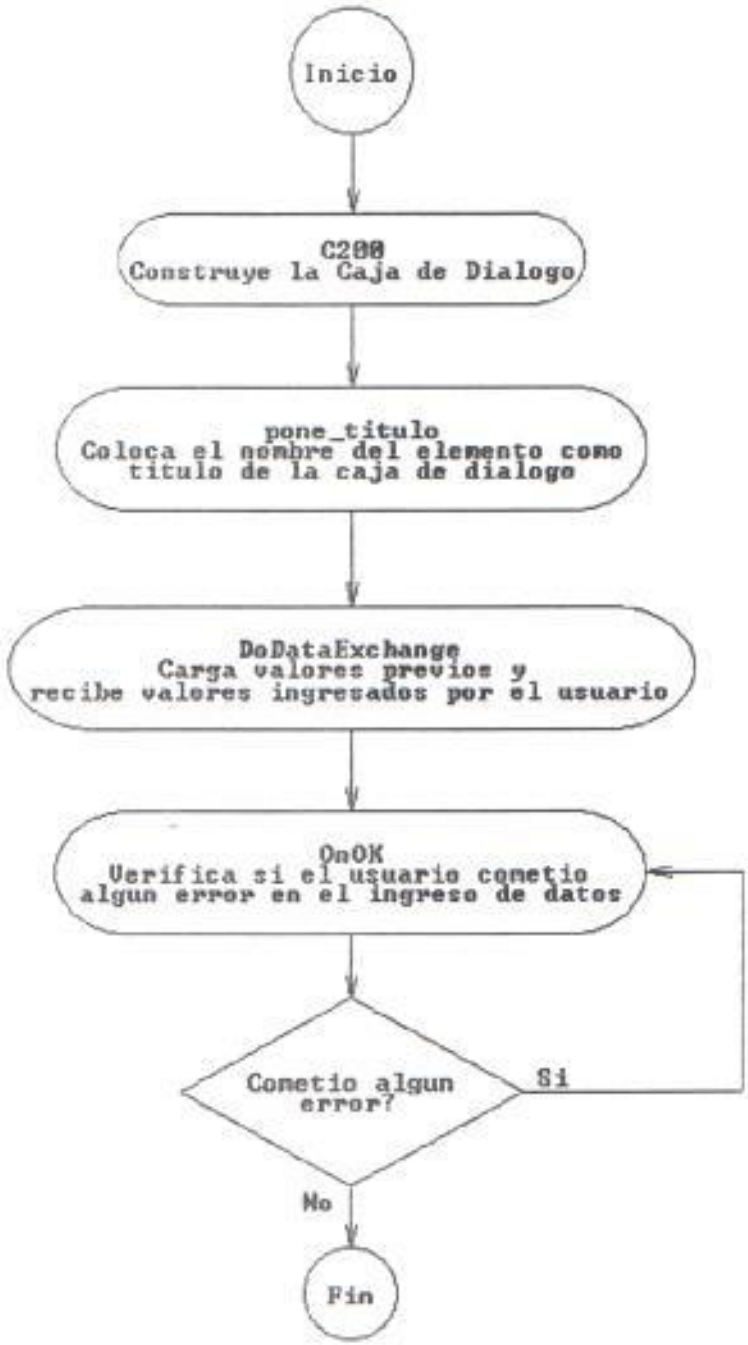
Clase C120


Inicio
C120
Construye la Caja de Dialogo
pone_titulo
Coloca el nombre del elemento como
titulo de la caja de dialogo
DoDataExchange
Carga valores previos y
recibe valores ingresados por el usuario
OnOK
Verifica si el usuario cometio
algun error en el ingreso de datos
Cometio algun
error?
Si
No
Fin

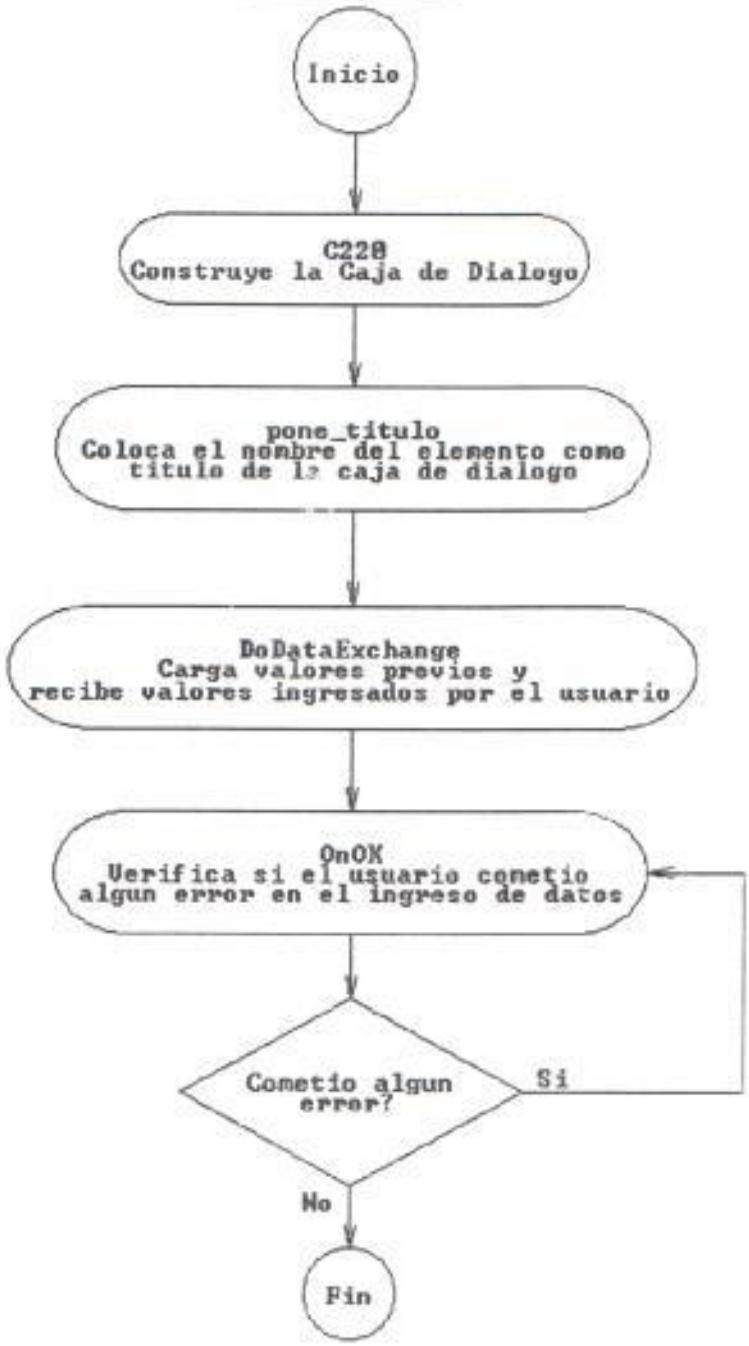
Clase C1211


Inicio
C1211
Construye la Caja de Dialogo
DoDataExchange
Carga valores previos y
recibe valores ingresados por el usuario
OnOK
Verifica si el usuario cometio
algun error en el ingreso de datos
Cometio algun
error?
Si
No
Fin

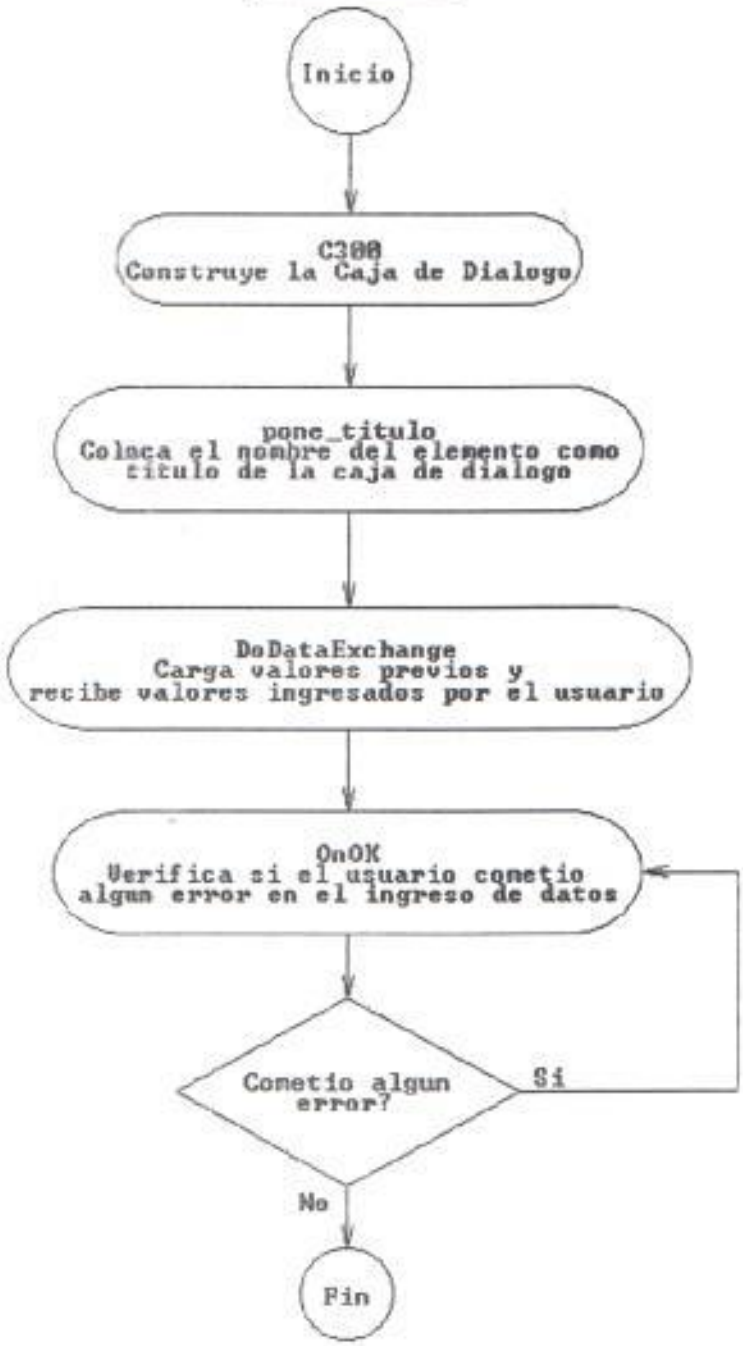
Clase C200


Inicio
C200
Construye la Caja de Dialogo
pone_titulo
Coloca el nombre del elemento como titulo de la caja de dialogo
DoDataExchange
Carga valores previos y recibe valores ingresados por el usuario
OnOK
Verifica si el usuario cometio algun error en el ingreso de datos
Cometio algun error?
Si
No
Fin

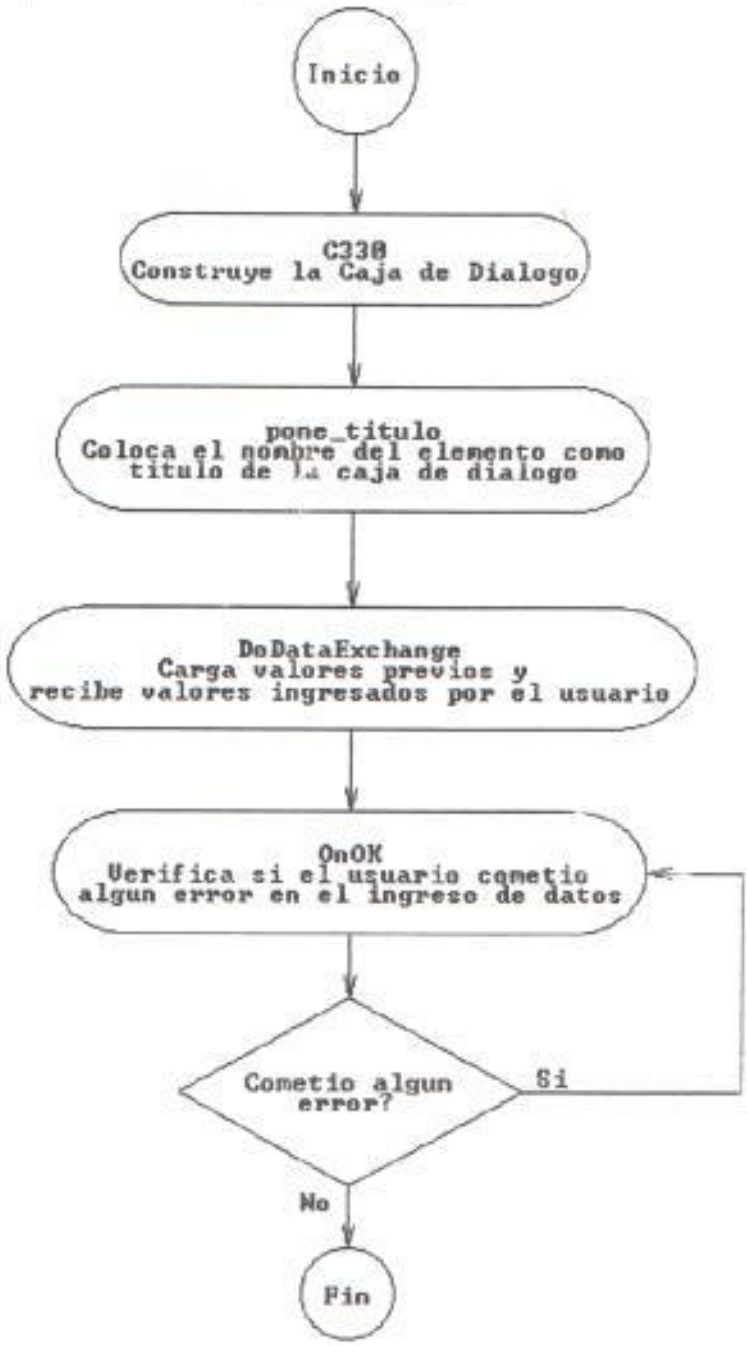
Clase C220


Inicio
C220
Construye la Caja de Dialogo
pone_titulo
Coloca el nombre del elemento como
titulo de la caja de dialogo
DoDataExchange
Carga valores previos y
recibe valores ingresados por el usuario
OnOK
Verifica si el usuario cometio
algun error en el ingreso de datos
Cometio algun
error?
Si
No
Fin

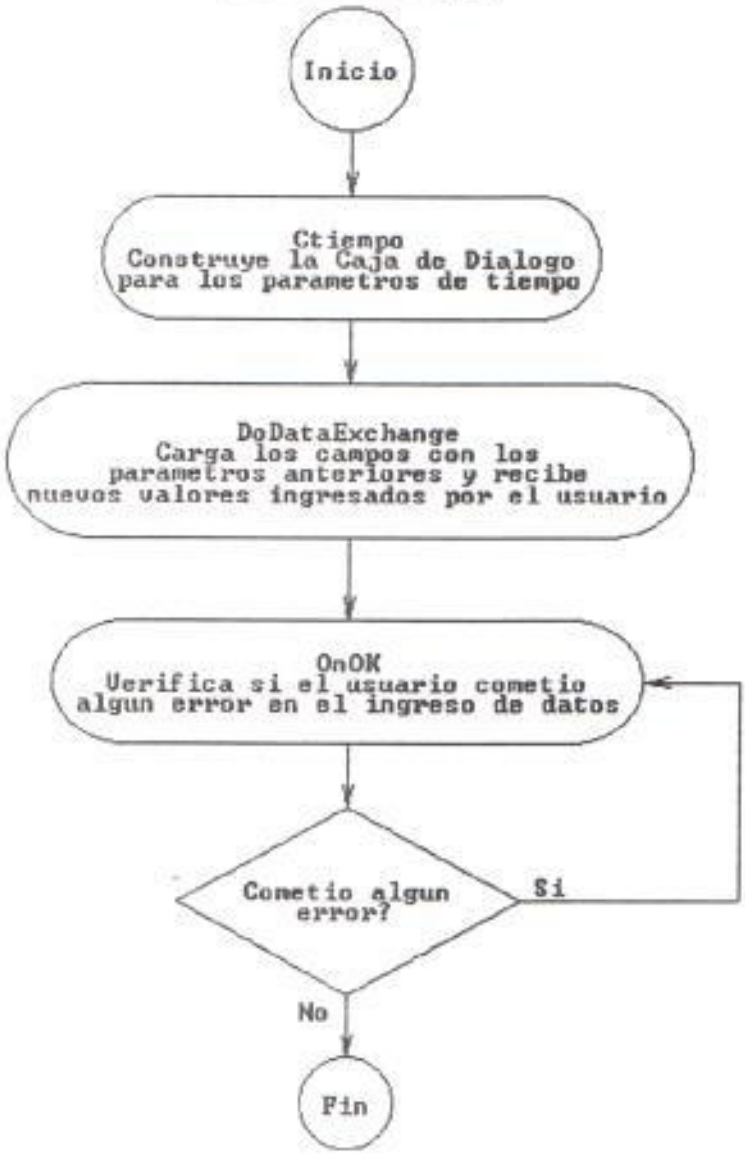
Clase C300


Inicio
C300
Construye la Caja de Dialogo
ponc_titulo
Coloca el nombre del elemento como titulo de la caja de dialogo
DoDataExchange
Carga valores previos y recibe valores ingresados por el usuario
OnOK
Verifica si el usuario cometio algun error en el ingreso de datos
Cometio algun error?
Si
No
Fin

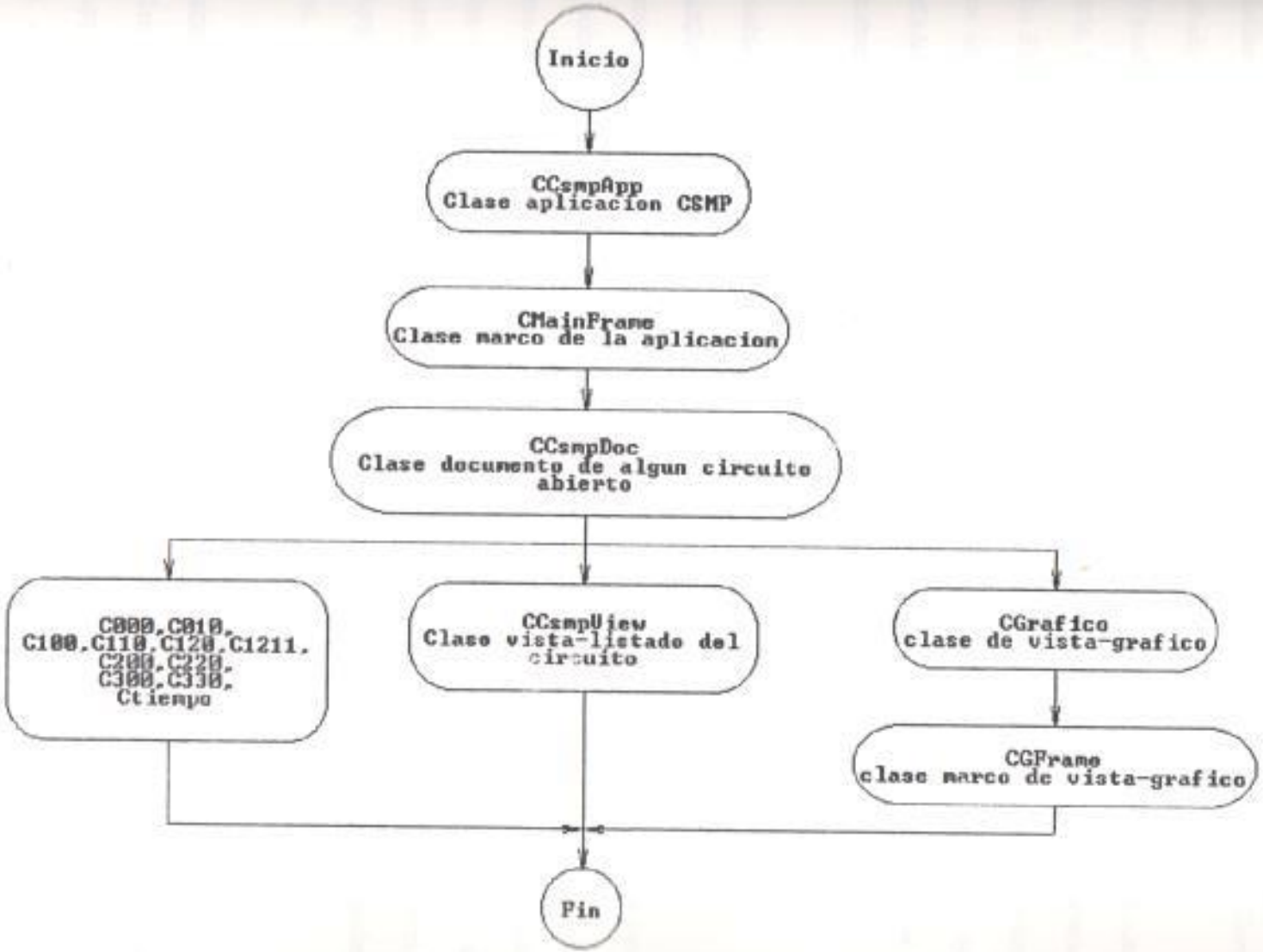
Clase C330


Inicio
C330
Construye la Caja de Dialogo
pone_titulo
Coloca el nombre del elemento como
titulo de la caja de dialogo
DoDataExchange
Carga valores previos y
recibe valores ingresados por el usuario
OnOK
Verifica si el usuario cometio
algun error en el ingreso de datos
Cometio algun
error?
Si
No
Fin

Clase Ctiempo



Aplicación CSMP (versión para Windows)



3.2 DEFINICIONES DE VARIABLES Y CONSTANTES USADAS

Debido a que este trabajo fue desarrollado en un lenguaje orientado a objetos, es indispensable mantener a capa y espada el concepto de *Encapsulado*. Cada clase derivada utiliza sus propias variables y funciones definidas dentro de ella. Muy poco se emplea el concepto de *Variables Globales* (pero si existen muy pocas excepciones).

El Visual C++ dispone de la ayuda de un *Estudio* para la creación de Cajas de Diálogo como de Menús, Iconos, Aceleradores, Tablas de Strings, etc. A su vez cada objeto que se va creando en dicho *Estudio* se le va asignando un Identificador Unico (un nombre junto con un número). Estos últimos son las constantes más utilizadas a lo largo del programa. Claro está que el diseñador de este trabajo incluyó constantes adicionales para mayor facilidad al programar. Nos referiremos a todos estos valores, a continuación.

Variables Globales.-

CCsmpApp NEAR theApp

Debemos tener en claro que cada vez que se abre la Aplicación CSMP se crea una y solamente una sola aplicación. Por supuesto que el usuario puede abrir muchas Aplicaciones CSMP al mismo tiempo, pero en cada sesión solo existe una *sola* aplicación que desconoce la existencia de las otras abiertas. La variable global **theApp** representa la instancia de aquella única aplicación por sesión. Vemos que es de tipo CCsmpApp, la clase derivada de la base CWinApp. theApp será utilizada a lo largo y ancho del programa, por eso la necesidad de considerarla global.

CMainFrame* pMainFrame

pMainFrame en cambio representa la clase quien podrá el marco principal de la aplicación theApp, colocará la Barra de Herramientas, la Barra de Status, el Menú Principal, etc. La clase a la cual pertenece **pMainFrame** (CMainFrame) es a su vez derivada de la base CMDIFrameWnd, quién es la encargada de adornar la interfase (marco, ventana principal, barras, menús iniciales,etc) para la aplicación recientemente creada. **pMainFrame** fue definida global debido a que su clase CMainFrame contiene algunas funciones muy solicitadas por la

clase que generará la pantalla del gráfico. CMainFrame por su naturaleza de tipo CMDIFrameWnd era la única que podía contener dichas funciones tan solicitadas, imposible incluirlas en la propia clase graficadora.

Constantes.-

Cada vez que el *Estudio* incluye un nuevo objeto a la interfase de la aplicación, se añade una línea a un archivo tipo include, llamado *resource.h*. Las constantes que examinaremos a continuación provienen de dicho archivo; cada nombre/número representa más que un simple valor, una *macro*. Estas macros sirven para relacionar un objeto de la interfase con cierto tipo de mensajes provenientes del Windows; de acuerdo al nombre y número de la macro, la aplicación sabrá qué tipo de acción cumplir en caso de que se presente un determinado *mensaje* (básicamente un mensaje es una interrupción que envía Windows, como reacción a una determinada acción que ejecuta el usuario o la misma aplicación).

CONSTANTES DE MULTIPLES OBJETOS.-

IDR_MAINFRAME 2

Esta constante asocia 3 objetos diferentes: el menú inicial(el que se presenta cuando no se abre ninguna ventana), el ícono CSMP de la aplicación y un string que representa el título de la aplicación (título sobre la ventana principal).

IDR_CSMPTYPE 3

Esta constante en cambio asocia el menú principal (se presenta cuando el usuario visualiza el listado de algún circuito), el ícono cuando se minimiza un listado de elementos y un string que contiene el nombre del circuito abierto junto con el tipo de archivo que se pueden abrir y la extensión típica (csm).

IDR_GRAFICO 127

Esta constante en cambio asocia el menú de solo la ventana del gráfico(la que presenta la barra), el ícono cuando se minimiza una ventana de gráfico y un string con el título *Gráfico*.

CONSTANTES DE CAJAS DE DIALOGO.-

IDD_ABOUTBOX 100

Esta es la típica caja de diálogo que muestra el nombre de la Aplicación CSMP, la versión de esta, el propietario del programa y el CopyRight del mismo. También muestra el ícono representativo de CSMP.

IDD_000 123

Recordemos que existe un grupo de elementos de Computador Analógico que no posee ni entradas ni parámetros; el único dato necesario es apenas el identificador del elemento. En este grupo tenemos al Aleatorio y al elemento Vacío.

IDC_000_NO 210

Este identificador representa el campo donde debe ingresarse el número que represente al elemento.

IDC_000_AYUDA 211

Este identificador en cambio se asocia con el botón marcado como *Ayuda*. Tiene el mismo efecto que presionar la tecla funcional F1.

IDD_010 122

En esta ocasión esta caja de diálogo incluye a todos los elementos que no poseen entradas pero si poseen 1 solo parámetro, además de incluir siempre el número de identificación. En este grupo consideramos a la Constante

IDC_010_NO 207

Este identificador representa el campo donde debe ingresarse el número que represente al elemento.

IDC_010_PAR1 208

En este campo es necesario ingresar el valor de condicion inicial o bien el de ganancia asociada a la entrada 1.

IDC_010_AYUDA 209

Este identificador en cambio se asocia con el botón marcado como *Ayuda*. Tiene el mismo efecto que presionar la tecla funcional F1.

DD_100 117

Existe un gran grupo de elementos que utilizan esta caja de diálogo. Recordemos que se ingresan los campos tanto de Identificador como el de Entrada 1. En este grupo consideraremos a Bang-Bang, Raíz Cuadrada, Magnitud, Separador Negativo, Separador Positivo e Inversor de Signo.

IDC_100_NO 189

Este identificador representa el campo donde debe ingresarse el número que represente al elemento.

IDC_100_ENT1 190

En este campo ingresaremos el número de identificación de aquel elemento cuya salida representa un valor de entrada al actual. La mayoría de estos valores son + exceptuando el caso del Sumador que puede tener un valor negativo en dicho campo (operación de resta). Este identificador se asocia con la entrada 1.

IDC_100_AYUDA 191

Este identificador en cambio se asocia con el botón marcado como *Ayuda*. Tiene el mismo efecto que presionar la tecla funcional F1.

DD_110 114

En este grupo de elementos consideraremos aquellos con el campo de identificación, un campo de entrada y un campo de parámetros. Entre aquellos elementos tenemos: Amplificador, Offset, Generador de Pulsos de Tiempo, Unidad de Retardo.

IDC_110_NO 177

Este identificador representa el campo donde debe ingresarse el número que represente al elemento.

IDC_110_ENT1 178

En este campo ingresaremos el número de identificación de aquel elemento cuya salida representa un valor de entrada al actual. La mayoría de estos valores son + exceptuando el caso del Sumador que puede tener un valor negativo en dicho campo (operación de resta). Este identificador se asocia con la entrada 1.

IDC_110_PAR1 179

En este campo es necesario ingresar el valor de condición inicial o bien el de ganancia asociada a la entrada 1.

IDC_110_AYUDA 180

Este identificador en cambio se asocia con el botón marcado como *Ayuda*. Tiene el mismo efecto que presionar la tecla funcional F1.

IDD_120 111

Representando a la caja de diálogo para elementos de una sola entrada y de 2 parámetros. En este grupo encontraremos a: Espacio Muerto y Limitador. Por supuesto siempre contamos con el campo de identificador.

IDC_120_NO 201

Este identificador representa el campo donde debe ingresarse el número que represente al elemento.

IDC_120_ENT1 202

En este campo ingresaremos el número de identificación de aquel elemento cuya salida representa un valor de entrada al actual. La mayoría de estos valores son + exceptuando el caso del Sumador que puede tener un valor negativo en dicho campo (operación de resta). Este identificador se asocia con la entrada 1.

IDC_120_PAR1 203

En este campo es necesario ingresar el valor correspondiente a un límite superior o bien dentro del Eje de Abscisas o del Eje de Ordenadas. Este campo sirve de límite para un cierto rango que el usuario desea imponer en el funcionamiento de este elemento.

IDC_120_PAR2 204

En este campo es necesario ingresar el valor correspondiente a un límite inferior o bien dentro del Eje de Abscisas o del Eje de Ordenadas. Este campo sirve de límite para un cierto rango que el usuario desea imponer en el funcionamiento de este elemento.

IDC_120_AYUDA 205

Este identificador en cambio se asocia con el botón marcado como *Ayuda*. Tiene el mismo efecto que presionar la tecla funcional F1.

IDD_1211 113

Esta caja de diálogo esta dedicada un solo tipo de elemento, el Generador de Funciones. Para este elemento encontramos a más del campo identificador, un campo entrada, 2 campos de parámetros, y 11 campos de interceptos para el eje de Ordenadas.

IDC_1211_NO 165

Este identificador representa el campo donde debe ingresarse el número que represente al elemento.

IDC_1211_ENT1 166

En este campo ingresaremos el número de identificación de aquel elemento cuya salida representa un valor de entrada al actual. La mayoría de estos valores son + exceptuando el caso del Sumador que puede tener un valor negativo en dicho campo (operación de resta). Este identificador se asocia con la entrada 1.

IDC_1211_PAR1 167

Este campo representa el límite superior en el Eje de Abscisas para el rango de interceptos del Eje Ordenadas. En esta ocasión este campo no va asociado a ningún tipo de ganancia.

IDC_1211_PAR2 175

Este campo representa el límite inferior en el Eje de Abscisas para el rango de interceptos del Eje Ordenadas. En esta ocasión este campo no va asociado a ningún tipo de ganancia.

IDC_1211_INT1 161

Este campo representa el valor del primer intercepto en el Eje de Ordenadas para la coordenada de Parámetro2 en el eje de Abscisas.

IDC_1211_INT2 162

El segundo intercepto en el Eje de Ordenadas.

IDC_1211_INT3 163

El tercer intercepto en el Eje de Ordenadas.

IDC_1211_INT4 164

El cuarto intercepto en el Eje de Ordenadas.

IDC_1211_INT5 168

El quinto intercepto en el Eje de Ordenadas.

IDC_1211_INT6 169

El sexto intercepto en el Eje de Ordenadas.

IDC_1211_INT7 170

El séptimo intercepto en el Eje de Ordenadas.

IDC_1211_INT8 171

El octavo intercepto en el Eje de Ordenadas.

IDC_1211_INT9 172

El noveno intercepto en el Eje de Ordenadas.

IDC_1211_INT10 173

El décimo intercepto en el Eje de Ordenadas.

IDC_1211_INT11 177

Este campo representa el valor del último intercepto en el Eje de Ordenadas para la ordenada de Parámetro1 en el eje de Abscisas.

IDC_1211_AYUDA 176

Este identificador en cambio se asocia con el botón marcado como *Ayuda*. Tiene el mismo efecto que presionar la tecla funcional F1.

IDD_200 107

Este tipo de Caja de Diálogo esta dedicada a todos aquellos elementos que a más de su campo identificador, tienen 2 campos de entrada. Entre ellos encontramos: Fin, Multiplicador, Mantener Orden Cero, Divisor.

IDC_200_NO 135

Este identificador representa el campo donde debe ingresarse el número que represente al elemento.

IDC_200_ENT1 136

En este campo ingresaremos el número de identificación de aquel elemento cuya salida representa un valor de entrada al actual. La mayoría de estos valores son + exceptuando el caso del Sumador que puede tener un valor negativo en dicho campo (operación de resta). Este identificador se asocia con la entrada 1.

DC_200_ENT2 137

A excepción del Sumador, en este campo se puede ingresar el identificador (siempre +) de aquel elemento cuya salida sirve de entrada 2 al elemento actual.

DC_200_AYUDA 138

Este identificador en cambio se asocia con el botón marcado como *Ayuda*. Tiene el mismo efecto que presionar la tecla funcional F1.

DC_220 106

Esta caja de diálogo incluye elementos de 2 entradas y 2 parámetros. Entre aquel grupo encontramos los siguientes: Bifurcación.

DC_220_NO 129

Este identificador representa el campo donde debe ingresarse el número que represente al elemento.

DC_220_ENT1 130

En este campo ingresaremos el número de identificación de aquel elemento cuya salida representa un valor de entrada al actual. La mayoría de estos valores son + exceptuando el caso del Sumador que puede tener un valor negativo en dicho campo (operación de resta). Este identificador se asocia con la entrada 1.

DC_220_ENT2 131

A excepción del Sumador, en este campo se puede ingresar el identificador (siempre +) de aquel elemento cuya salida sirve de entrada 2 al elemento actual.

DC_220_PAR1 132

En este campo es necesario ingresar el valor de condición inicial o bien el de ganancia asociada a la entrada 1.

DC_220_PAR2 133

Por lo general este campo esta asociado a una ganancia de entrada para el valor proveniente del elemento en la entrada 2.

DC_220_AYUDA 134

Este identificador en cambio se asocia con el botón marcado como *Ayuda*. Tiene el mismo efecto que presionar la tecla funcional F1.

IDD_300 104

Esta caja de diálogo incluye todos aquellos elementos de 3 entradas y el campo identificador. En aquel grupo encontramos a: Relé, Sumador.

IDC_300_NO 119

Este identificador representa el campo donde debe ingresarse el número que represente al elemento.

IDC_300_ENT1 120

En este campo ingresaremos el número de identificación de aquel elemento cuya salida representa un valor de entrada al actual. La mayoría de estos valores son + exceptuando el caso del Sumador que puede tener un valor negativo en dicho campo (operación de resta). Este identificador se asocia con la entrada 1.

IDC_300_ENT2 121

A excepción del Sumador, en este campo se puede ingresar el identificador (siempre +) de aquel elemento cuya salida sirve de entrada 2 al elemento actual.

IDC_300_ENT3 122

A excepción del Sumador, en este campo se puede ingresar el identificador (siempre +) de aquel elemento cuya salida sirve de entrada 3 al elemento actual.

IDC_300_AYUDA 123

Este identificador en cambio se asocia con el botón marcado como *Ayuda*. Tiene el mismo efecto que presionar la tecla funcional F1.

IDD_330 101

Esta caja de diálogo representa a todos los elementos de 3 entradas y 3 parámetros, excluyendo la presencia de su campo identificador.

IDC_330_NO 206

Este identificador representa el campo donde debe ingresarse el número que represente al elemento.

IDC_330_ENT1 102

En este campo ingresaremos el número de identificación de aquel elemento cuya salida representa un valor de entrada al actual. La mayoría de estos valores son + exceptuando el caso

del Sumador que puede tener un valor negativo en dicho campo (operación de resta). Este identificador se asocia con la entrada 1.

DC_330_ENT2 103

A excepción del Sumador, en este campo se puede ingresar el identificador (siempre +) de aquel elemento cuya salida sirve de entrada 2 al elemento actual.

DC_330_ENT3 104

A excepción del Sumador, en este campo se puede ingresar el identificador (siempre +) de aquel elemento cuya salida sirve de entrada 3 al elemento actual.

DC_330_PAR1 107

En este campo es necesario ingresar el valor de condición inicial o bien el de ganancia asociada a la entrada 1.

DC_330_PAR2 108

Por lo general este campo esta asociado a una ganancia de entrada para el valor proveniente del elemento en la entrada 2.

DC_330_PAR3 110

Por lo general este campo esta asociado a una ganancia de entrada para el valor proveniente del elemento en la entrada 3.

DC_330_AYUDA 109

Este identificador en cambio se asocia con el botón marcado como *Ayuda*. Tiene el mismo efecto que presionar la tecla funcional F1.

DC_TIEMPO 124

Esta caja de diálogo le pregunta al usuario por los parámetros de tiempo. En los campos se debe ingresar tanto el tiempo total de simulación, como la cantidad de intervalos o muestras a tomarse.

DC_TIEMPO_TOTAL 214

Este es propiamente el campo donde debe ingresarse el valor de tiempo total. Este es un real positivo >=0.

IDC_TAMANO_INTERVALO 215

Aquí debe ingresarse el tamaño de lo que será el intervalo de integración. Recuerde que este es un valor fracción de 1 segundo (mayor que 0 y no mayor que todo el tiempo de simulación).

IDC_INTERVALOS 216

Aparecerá en la pantalla una frase señalando la cantidad de muestras calculadas a partir del tiempo total de simulación, y del tamaño del intervalo de integración.

IDC_A_PARTIR 219

Acepta el tiempo inicial a partir del cual se comenzará a presentar las muestras en la tabla. De cualquier forma la simulación se da para todo el tiempo total.

IDC_TIEMPO_AYUDA 212

Este identificador en cambio se asocia con el botón marcado como *Ayuda*. Tiene el mismo efecto que presionar la tecla funcional F1.

IDC_TITULO 217

Ud habrá notado que en la parte superior de cada caja de diálogo se presenta el nombre del elemento que Ud. ha escogido. Ese nombre está asociado con un identificador, IDC_TITULO. Basta configurar el contenido de este identificador, y mostrarlo en el sitio adecuado.

IDC_P1 218

Este identificador solo tiene aplicación para la caja de diálogo IDD_110. Recordemos que en caso de escogerse el elemento Unidad de Retardo, el usuario debe llenar un campo de parámetro. En esta ocasión, Unidad de Retardo utiliza el parámetro #2 en lugar del parámetro #1. IDC_P1 cambia el nombre de #1 a #2.

CONSTANTES DE OPCIONES DE MENU.-

Recordemos que uno de los menús presentes es el de *Elementos*. Aquí encontramos la lista de todos los elementos incluidos para esta versión de CSMP. Basta escoger una de las opciones de este menú, para que se presente la respectiva caja de diálogo. Cada una de estas opciones a su vez está directamente relacionada con uno de los botones de la Barra de Herramientas. Da lo mismo escoger el elemento Bang-Bang en el menú, o presionar el botón de la *B*.

ID_TIEMPO 32768

Representa la opción de Parámetros de Tiempo incluida en el menú de Opciones.

ID_ALEATORIO 32804

Representa al elemento Aleatorio.

ID_AMPLIFICADOR 32801

Representa al elemento Amplificador.

ID_BANG_BANG 32798

Representa al elemento Bang-Bang.

ID_BIFURCACION 32818

Representa al elemento Bifurcación.

ID_CONSTANTE 32805

Representa al elemento Constante.

ID_DIVISOR 32821

Representa al elemento Divisor.

ID_ELIMINA 32823

Representa la acción de presionar la tecla *Delete* y eliminar de la lista del circuito, al elemento previamente seleccionado.

ID_ESPACIO_MUERTO 32799

Representa al elemento Espacio Muerto.

ID_FIN 32811

Representa al elemento Fin.

ID_GENERADOR_FUNCION 32800

Representa al elemento Generador de Funciones.

ID_GENERADOR_PULSOS 32813

Representa al elemento Generador de Pulsos.

ID_INTEGRADOR 32803

Representa al elemento Integrador.

ID_INVERSOR 32822

Representa al elemento Inversor.

ID_LIMITADOR 32806

Representa al elemento Limitador.

ID_MAGNITUD 32807

Representa al elemento Magnitud.

ID_MULTIPLICADOR 32817

Representa al elemento Multiplicador.

ID_OFFSET 32809

Representa al elemento Offset.

ID_ORDEN_CERO 32819

Representa al elemento de Orden Cero.

ID_RAIZ_CUADRADA 32802

Representa al elemento Raíz Cuadrada.

ID_RELE 32812

Representa al elemento Relé.

ID_RUN 32829

Representa la opción de Mostrar el Gráfico dentro del menú de Opciones. Esta presentará inmediatamente la ventana con la curva y la tabla respectiva.

ID_SEPARADOR_NEGATIVO 32808

Representa al elemento Separador Negativo.

ID_SEPARADOR_POSITIVO 32810

Representa al elemento Separador Positivo.

ID_SUMADOR 32820

Representa al elemento Sumador.

ID_SUMADOR_PESO 32816

Representa al elemento Sumador con Pesos.

ID_UNIDAD_RETARDO 32814

Representa al elemento Unidad de Retardo.

ID_VACIO 32815

Representa al elemento Vacío.

CONSTANTES DE TABLAS DE STRING.-

IDS_NUEVO 57609

Muestra un string en la Barra de Status, en caso que el usuario cree un nuevo circuito.

IDS_NO_PUEDE_ABRIR 57610

Si el programa tiene problemas en abrir un circuito dado, mostrará este string en una ventanita, mencionando que no es posible abrir dicho circuito.

IDS_TOO_MANY_ROWS 4

Si en el momento de mostrar en pantalla o en impresora el listado de elementos, se encuentra que la cantidad de filas (cada fila describiendo a un elemento) son excesivas, se exhibe este string en una ventanita indicando que hay demasiadas filas para ser mostradas.

CONSTANTES DE ICONOS.-

IDI_TIEMPO 125

Este es el ícono que aparece en una de las esquinas de la caja de diálogo para Parámetros de Tiempo.

IDI_ELEMENTO 126

Este es el ícono que aparece en una de las esquinas en cada caja de diálogo correspondiente a cualquiera de los elementos que se quiere adicionar al circuito.

Las constantes que examinaremos a continuación en cambio especifican una posición de columna en la pantalla. Esto es muy útil en caso de ubicar en qué columna se presentarán los datos de Identificador, tipo, entradas y parámetros en el listado de elementos. También es útil para situar los valores en la tabla junto a la curva de salida.

CONSTANTES PARA SITUAR COLUMNAS EN EL LISTADO DE ELEMENTOS.-

Recordemos que al listar los elementos correspondientes a un circuito dado, no solo se muestran los valores de cada elemento sino además se le antepone una descripción del significado del mismo. A esta descripción más el valor del dato, es lo que llamaremos *string*.

ID_COL 0

Esta constante sirve para identificar en qué columna se desea mostrar el Identificador del elemento, al visualizar en pantalla o impresora el listado.

ID_LEN 7

Esta constante proporciona la longitud total del string que representa al número de identificador.

TIPO_COL (ID_COL + ID_LEN + 4)

Esta constante representa la columna a partir de la cual se sitúa el tipo del elemento dentro del listado.

TIPO_LEN 4

Esta constante proporciona la longitud total del string que representa al tipo del elemento.

ENTRADA1_COL (TIPO_COL + TIPO_LEN + 5)

Esta constante representa la columna donde se sitúa el identificador de la entrada 1. Tómese en cuenta que usamos la columna del tipo, su longitud y un adicional de 5 columnas.

ENTRADA1_LEN 8

Esta constante proporciona la longitud total del string que representa la entrada 1.

ENTRADA2_COL (ENTRADA1_COL + ENTRADA1_LEN + 8)

Esta constante representa la columna donde se sitúa el identificador de la entrada 2. Tómese en cuenta que utilizamos la columna de la entrada 1, la longitud del string de entrada 1 y le sumamos un adicional de 8 columnas.

ENTRADA2_LEN 8

Esta constante proporciona la longitud total del string que representa la entrada 2.

ENTRADA3_COL (ENTRADA2_COL + ENTRADA2_LEN + 8)

Esta constante representa la columna donde se sitúa el identificador de la entrada 3. Tómese en cuenta que utilizamos la columna de la entrada 2, la longitud del string de entrada 2 y le sumamos un adicional de 8 columnas.

ENTRADA3_LEN 8

Esta constante proporciona la longitud total del string que representa la entrada 3.

[illegible]_WIDTH (ENTRADA3_COL+11)

Esta constante especifica la longitud total o el ancho que tendrá cada fila equivalente a un [illegible] del circuito. Tómese en cuenta que se utiliza la columna de la tercera entrada y se le [illegible] 11.

[illegible]AMETRO1_COL (TIPO_COL + TIPO_LEN + 5)

Recordemos que cada elemento utiliza 2 filas para mostrar todos sus datos. En la segunda fila [illegible] los valores de cada parámetro. En este caso esta constante proporciona la columna [illegible] se sitúa el valor correspondiente al parametro 1.

[illegible]AMETRO1_LEN 11

Esta constante nos entrega la longitud del string correspondiente al parámetro 1.

[illegible]AMETRO2_COL (ENTRADA1_COL + ENTRADA1_LEN + 8)

Esta constante proporciona la columna donde se sitúa el parámetro 2. Tómese en cuenta que se [illegible] la columna de la entrada 1 junto con la longitud de dicha entrada y se le adiciona 8.

[illegible]AMETRO2_LEN 11

Esta constante nos entrega la longitud del string correspondiente al parámetro 2.

[illegible]AMETRO3_COL (ENTRADA2_COL + ENTRADA2_LEN + 8)

La columna del parámetro 3 esta dada por la columna de la entrada 2 más la longitud de dicha [illegible] y adicionándole 8.

[illegible]AMETRO3_LEN 11

Esta constante nos proporciona la longitud del string para el parámetro 3.

[illegible]STANTES PARA SITUAR COLUMNAS EN LA TABLA DEL GRAFICO.-

En el momento que se presenta la curva de salida de algún elemento, también se muestra en el [illegible] izquierdo de la pantalla o del papel una tabla con las muestras tomadas a lo largo del [illegible] de simulación. Para situar mejor en la pantalla estos datos, usamos las siguientes [illegible] que especifican las columnas donde se situarán los valores de tiempo y de salida.

[illegible]_COL 1

Esta constante nos proporciona la columna donde irá el título de los Tiempos tomados.

TP_COL 12

Esta constante en cambio, nos da la columna donde se posesionará el título de Salidas.

TV_COL (TP_COL + 5)

Este es el valor usado como columna para las muestras de tiempo tomadas. Tómese en cuenta que se usa el valor de la columna del título de Tiempos y se le adiciona 5. Esta columna es diferente a la del título, debido a que justificamos cada muestra hacia su derecha.

TE_COL 19

Esta columna tambien proporciona la posición para todos los valores de salida tomados. Al igual que TV_COL también justificamos hacia la derecha.

CONSTANTES EXTRAS (MISCELANEAS) USADAS EN PROGRAMACION.-

Aparte de todas las constantes previamente mencionadas el programa utiliza otros valores que es poder agruparlos bajo un mismo tipo, ayudan y benefician el trabajo del programador. Debemos mencionar los siguientes:

MAXIMO 100

Recordemos que una de las limitaciones del programa es el ingreso de circuitos de Computador Analógico, de un máximo de 100 elementos. En el futuro si el programador de mantenimiento desea modificar dicho límite, lo puede hacer en MAXIMO (que contiene dicho valor).

BANG_BANG 66

Dentro de la programación de este sistema fue práctico asociar el nombre de cada elemento con su valor ASCII. Asi Bang-Bang se asocia con el ASCII de *B*.

ESPACIO_MUERTO 68

El valor ASCII de la letra *D*.

GENERADOR_FUNCION 70

El valor ASCII de la letra *F*.

AMPLIFICADOR 71

El valor ASCII de la letra *G*.

[illegible]MADOR_PESO 87

[illegible]lor ASCII de la letra *W*.

[illegible]LTIPLICADOR 88

[illegible]lor ASCII de la letra *X*.

[illegible]RCACION 89

[illegible]lor ASCII de la letra *Y*.

[illegible]EN_CERO 90

[illegible]lor ASCII de la letra *Z*.

[illegible]MADOR 43

[illegible]lor ASCII de la letra +.

[illegible]SOR 47

[illegible]lor ASCII de la letra /.

[illegible]ERSOR 45

[illegible]lor ASCII de la letra -.

[illegible]pesar de que existían las constantes de tipo ID_BANG_BANG, estos valores representaban [illegible] y eran números considerados grandes (mayor que la capacidad de un valor entero). [illegible] fue muy útil en el momento de almacenar, grabar el ASCII del tipo del elemento. [illegible] convertir dicho valor entero a caracter, para obtener el símbolo del elemento.

[illegible]DESCRIPCION DE CLASES CREADAS

[illegible]ntinuación mencionaremos todas las clases que integran CSMP. Debido a que el lenguaje [illegible]gramación empleado en este trabajo fue *orientado a objetos*, la programación se dividió [illegible] diferentes archivos de tipo .CPP y de tipo .H que contenían las definiciones y las [illegible]mentaciones de funciones y variables, dentro de cada clase. Toda la programación [illegible] en dividir el proyecto total en módulos (.H) y a su vez cada módulo que contenga una [illegible] clases. Cada clase a su vez guardaba sus propias variables y sus propias funciones. El [illegible]llo de cada función se incluye en los archivos .CPP o fuentes.

Desde el mismo inicio del programa Visual C++ nos obliga a seguir el patrón de clases. Clases que a su vez llaman funciones y valores de otras clases. Con esto se mantiene la idea de *Encapsulamiento*. Cada quien administra solo sus componentes. La información que se podía compartir entre ellos se declaraban como *Públicas*. Las que solo se podían utilizar por la misma clase y sus derivadas, era o bien *privadas* o bien *protegidas*. Analicemos cada una:

CLASE C000 (derivada de CDialog).-

Esta clase es la asociada y encargada de manejar la Caja de Diálogo para IDD_000. Recuerde que dicha caja de diálogo es útil para los elementos que solo poseen el número de Identificación.

Funciones

C000.- Esta es la función constructora de la Caja de Diálogo. Se encarga de crear la Caja de Diálogo, y cargar cada campo que se presente con valores nulos. Sin embargo no grafica la caja en pantalla, sera necesario otra función.

DoDataExchange.- Esta función en cambio se encarga de tomar los valores ingresados por el usuario en cada campo de la Caja y asociarlos con variables definidas dentro de esta clase. Así el programador utiliza directamente las variables y no los identificadores de los objetos en pantalla.

Nombre_titulo.- Recordemos que la misma Caja de Diálogo puede servir a más de un elemento que posean iguales características. La gran diferencia está en colocar el nombre del Elemento en la parte superior de la caja. Esta función se encarga de poner dicho nombre, de acuerdo al elemento específicamente escogido.

OnOK.- Esta función tiene la misión de revisar cualquier falta o error cometido por el usuario al llenar los campos de la caja de diálogo. Envía ventanitas en el centro de la pantalla en caso de haberse cometido el error.

m_numero (int)	m_id (int)

Variables

m_id.- La función pone_titulo necesita conocer el identificador específico del elemento, para así colocarle el nombre a la caja de diálogo. Esta variable es la encargada de proporcionarle dicho nombre.

m_numero.- El valor que el usuario ingresa como identificador del elemento a su vez es recibido por esta variable. Así no es necesario que el usuario utilice el identificador del campo en pantalla, basta manejar el contenido de dicha variable.

CLASE C010 (derivada de CDialog).-

Esta clase es la asociada y encargada de manejar la Caja de Diálogo para IDD_010. Recuerde que dicha caja de diálogo es útil para los elementos que poseen el número de Identificación, y un solo parámetro.

Funciones

C010.- Esta es la función constructora de la Caja de Diálogo. Se encarga de crear la Caja de Diálogo, y cargar cada campo que se presente con valores nulos. Sin embargo no grafica la caja en pantalla, será necesario otra función.

DoDataExchange.- Esta función en cambio se encarga de tomar los valores ingresados por el usuario en cada campo de la Caja y asociarlos con variables definidas dentro de esta clase. Así el programador utiliza directamente las variables y no los identificadores de los objetos en pantalla.

pone_titulo.- Recordemos que la misma Caja de Diálogo puede servir a más de un elemento que posean iguales características. La gran diferencia está en colocar el nombre del Elemento en la parte superior de la caja. Esta función se encarga de poner dicho nombre, de acuerdo al elemento específicamente escogido.

OnOK.- Esta función tiene la misión de revisar cualquier falta o error cometido por el usuario al llenar los campos de la caja de diálogo. Envía ventanitas en el centro de la pantalla en caso de haberse cometido el error.

m_numero [int]	m_id [int]
m_parametro1[float]	

Variables

m_id.- La función pone_titulo necesita conocer el identificador específico del elemento, para así colocarle el nombre a la caja de diálogo. Esta variable es la encargada de proporcionarle dicho nombre.

m_numero.- El valor que el usuario ingresa como identificador del elemento a su vez es recibido por esta variable. Así no es necesario que el usuario utilice el identificador del campo en pantalla, basta manejar el contenido de dicha variable.

m_parametro1.- El valor ingresado por el usuario en el campo de parámetro 1, es recibido directamente por esta variable. Basta manejar y cargar el contenido de esta variable.

CLASE C100 (derivada de CDialog).-

Esta clase es la asociada y encargada de manejar la Caja de Diálogo para IDD_100. Recuerde que dicha caja de diálogo es útil para los elementos que poseen el número de Identificación, y una sola entrada.

Funciones

C100.- Esta es la función constructora de la Caja de Diálogo. Se encarga de crear la Caja de Diálogo, y cargar cada campo que se presente con valores nulos. Sin embargo no grafica la caja en pantalla, será necesario otra función.

DoDataExchange.- Esta función en cambio se encarga de tomar los valores ingresados por el usuario en cada campo de la Caja y asociarlos con variables definidas dentro de esta clase. Así el programador utiliza directamente las variables y no los identificadores de los objetos en pantalla.

pone_titulo.- Recordemos que la misma Caja de Diálogo puede servir a más de un elemento que posean iguales características. La gran diferencia está en colocar el nombre del Elemento en la parte superior de la caja. Esta función se encarga de poner dicho nombre, de acuerdo al elemento específicamente escogido.

OnOK.- Esta función tiene la misión de revisar cualquier falta o error cometido por el usuario al llenar los campos de la caja de diálogo. Envía ventanitas en el centro de la pantalla en caso de haberse cometido el error.

m_numero (int)	m_id (int)
m_entrada1 (int)	

Variables

m_id.- La función pone_titulo necesita conocer el identificador específico del elemento, para así colocarle el nombre a la caja de diálogo. Esta variable es la encargada de proporcionarle dicho nombre.

m_numero.- El valor que el usuario ingresa como identificador del elemento a su vez es recibido por esta variable. Así no es necesario que el usuario utilice el identificador del campo en pantalla, basta manejar el contenido de dicha variable.

m_entrada1.- El valor ingresado por el usuario representando el Identificador de otro elemento que sirve como entrada al actual, es recibido diréctamente por esta variable. Basta solo manejar el contenido de esta, sin referirse al objeto en pantalla.

CLASE C110 (derivada de CDialog).-

Esta clase es la asociada y encargada de manejar la Caja de Diálogo para IDD_110. Recuerde que dicha caja de diálogo es útil para los elementos que poseen el número de Identificación, una sola entrada y un solo parámetro.

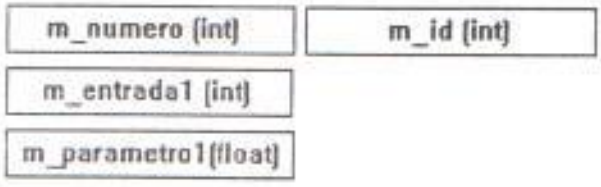
Funciones

C110.- Esta es la función constructora de la Caja de Diálogo. Se encarga de crear la Caja de Diálogo, y cargar cada campo que se presente con valores nulos. Sin embargo no grafica la caja en pantalla, será necesario otra función.

DoDataExchange.- Esta función en cambio se encarga de tomar los valores ingresados por el usuario en cada campo de la Caja y asociarlos con variables definidas dentro de esta clase. Así el programador utiliza directamente las variables y no los identificadores de los objetos en pantalla.

pone_titulo.- Recordemos que la misma Caja de Diálogo puede servir a más de un elemento que posean iguales características. La gran diferencia está en colocar el nombre del Elemento en la parte superior de la caja. Esta función se encarga de poner dicho nombre, de acuerdo al elemento especificamente escogido. Además si el elemento resulta ser *Unidad de Retardo*, se colocará en lugar de la palabra *Parámetro No. 1*, la palabra *Parámetro No. 2*.

OnOK.- Esta función tiene la misión de revisar cualquier falta o error cometido por el usuario al llenar los campos de la caja de diálogo. Envía ventanitas en el centro de la pantalla en caso de haberse cometido el error.



Variables

m_id.- La función pone_titulo necesita conocer el identificador específico del elemento, para así colocarle el nombre a la caja de diálogo. Esta variable es la encargada de proporcionarle dicho nombre.

m_numero.- El valor que el usuario ingresa como identificador del elemento a su vez es recibido por esta variable. Así no es necesario que el usuario utilice el identificador del campo en pantalla, basta manejar el contenido de dicha variable.

m_entrada1.- El valor ingresado por el usuario representando el Identificador de otro elemento que sirve como entrada al actual, es recibido diréctamente por esta variable. Basta solo manejar el contenido de esta, sin referirse al objeto en pantalla.

m_parametro1.- El valor ingresado por el usuario en el campo de parámetro 1, es recibido diréctamente por esta variable. Basta manejar y cargar el contenido de esta variable.

CLASE C120 (derivada de CDialog).-

Esta clase es la asociada y encargada de manejar la Caja de Diálogo para IDD_120. Recuerde que dicha caja de diálogo es útil para los elementos que poseen el número de Identificación, una sola entrada y 2 parámetros.

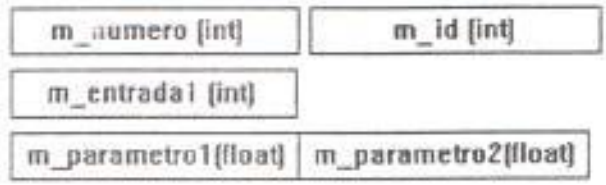
Funciones

CDB.- Esta es la función constructora de la Caja de Diálogo. Se encarga de crear la Caja de Diálogo, y cargar cada campo que se presente con valores nulos. Sin embargo no grafica la caja en pantalla, será necesario otra función.

DoDataExchange.- Esta función en cambio se encarga de tomar los valores ingresados por el usuario en cada campo de la Caja y asociarlos con variables definidas dentro de esta clase. Así el programador utiliza directamente las variables y no los identificadores de los objetos en pantalla.

pone_titulo.- Recordemos que la misma Caja de Diálogo puede servir a más de un elemento que posean iguales características. La gran diferencia está en colocar el nombre del Elemento en la parte superior de la caja. Esta función se encarga de poner dicho nombre, de acuerdo al elemento específicamente escogido.

OnOK.- Esta función tiene la misión de revisar cualquier falta o error cometido por el usuario al llenar los campos de la caja de diálogo. Envia ventanitas en el centro de la pantalla en caso de haberse cometido el error.



Variables

m_id.- La función pone_titulo necesita conocer el identificador específico del elemento, para así colocarle el nombre a la caja de diálogo. Esta variable es la encargada de proporcionarle dicho nombre.

m_numero.- El valor que el usuario ingresa como identificador del elemento a su vez es recibido por esta variable. Así no es necesario que el usuario utilice el identificador del campo en pantalla, basta manejar el contenido de dicha variable.

m_entrada1.- El valor ingresado por el usuario representando el Identificador de otro elemento que sirve como entrada al actual, es recibido directamente por esta variable. Basta solo manejar el contenido de esta, sin referirse al objeto en pantalla.

m_parametro1.- El valor ingresado por el usuario en el campo de parámetro 1, es recibido directamente por esta variable. Basta manejar y cargar el contenido de esta variable.

m_parametro2.- El valor ingresado por el usuario en el campo de parámetro 2, es recibido directamente por esta variable. Basta manejar y cargar el contenido de esta variable.

CLASE C1211 (derivada de CDialog).-

Esta clase es la asociada y encargada de manejar la Caja de Diálogo para IDD_1211. Recuerde que dicha caja de diálogo es útil para los elementos que poseen el número de Identificación, una sola entrada, 2 parámetros y 11 interceptos. El único elemento de ese tipo es el Generador de Funciones.

Funciones

C1211.- Esta es la función constructora de la Caja de Diálogo. Se encarga de crear la Caja de Diálogo, y cargar cada campo que se presente con valores nulos. Sin embargo no grafica la caja en pantalla, será necesario otra función.

DoDataExchange.- Esta función en cambio se encarga de tomar los valores ingresados por el usuario en cada campo de la Caja y asociarlos con variables definidas dentro de esta clase. Así el programador utiliza directamente las variables y no los identificadores de los objetos en pantalla.

OnOK.- Esta función tiene la misión de revisar cualquier falta o error cometido por el usuario al llenar los campos de la caja de diálogo. Envía ventanitas en el centro de la pantalla en caso de haberse cometido el error.

m_numero (int)	
m_entrada1 (int)	
m_parametro1(float)	m_parametro2(float)
m_intercepto1	m_intercepto7
m_intercepto2	m_intercepto8
m_intercepto3	m_intercepto9
m_intercepto4	m_intercepto10
m_intercepto5	m_intercepto11
m_intercepto6	

Variables

m_numero.- El valor que el usuario ingresa como identificador del elemento a su vez es recibido por esta variable. Así no es necesario que el usuario utilice el identificador del campo en pantalla, basta manejar el contenido de dicha variable.

m_entrada1.- El valor ingresado por el usuario representando el Identificador de otro elemento que sirve como entrada al actual, es recibido diréctamente por esta variable. Basta solo manejar el contenido de esta, sin referirse al objeto en pantalla.

m_parametro1.- El valor ingresado por el usuario en el campo de parámetro 1, es recibido diréctamente por esta variable. Basta manejar y cargar el contenido de esta variable.

m_parametro2.- El valor ingresado por el usuario en el campo de parámetro 2, es recibido diréctamente por esta variable. Basta manejar y cargar el contenido de esta variable.

m_intercepto1 ... m_intercepto11.- El valor ingresado por el usuario en el campo de interceptos del 1 al 11 es recibido diréctamente por estas variables. Recordemos que sirven para interpolar la entrada y generarle un valor en el eje de las Ordenadas, y por lo tanto una salida a un elemento.

CLASE C200 (derivada de CDialog).-

Esta clase es la asociada y encargada de manejar la Caja de Diálogo para IDD_200. Recuerde que dicha caja de diálogo es útil para los elementos que poseen el número de Identificación, y dos entradas.

Funciones

CDD.- Esta es la función constructora de la Caja de Diálogo. Se encarga de crear la Caja de Diálogo, y cargar cada campo que se presente con valores nulos. Sin embargo no grafica la caja en pantalla, será necesario otra función.

DoDataExchange.- Esta función en cambio se encarga de tomar los valores ingresados por el usuario en cada campo de la Caja y asociarlos con variables definidas dentro de esta clase. Así el programador utiliza directamente las variables y no los identificadores de los objetos en pantalla.

pone_titulo.- Recordemos que la misma Caja de Diálogo puede servir a más de un elemento que posean iguales características. La gran diferencia está en colocar el nombre del Elemento en la parte superior de la caja. Esta función se encarga de poner dicho nombre, de acuerdo al elemento específicamente escogido.

OnOK.- Esta función tiene la misión de revisar cualquier falta o error cometido por el usuario al llenar los campos de la caja de diálogo. Envía ventanitas en el centro de la pantalla en caso de haberse cometido el error.

m_numero (int)	m_id (int)
m_entrada1 (int)	m_entrada2 (int)

Variables

m_id.- La función pone_titulo necesita conocer el identificador específico del elemento, para así colocarle el nombre a la caja de diálogo. Esta variable es la encargada de proporcionarle dicho nombre.

m_numero.- El valor que el usuario ingresa como identificador del elemento a su vez es recibido por esta variable. Así no es necesario que el usuario utilice el identificador del campo en pantalla, basta manejar el contenido de dicha variable.

m_entrada1.- El valor ingresado por el usuario representando el Identificador de otro elemento que sirve como entrada al actual, es recibido directamente por esta variable. Basta solo manejar el contenido de esta, sin referirse al objeto en pantalla.

m_entrada2.- El valor ingresado por el usuario representando el Identificador de otro elemento que sirve como entrada al actual, es recibido diréctamente por esta variable. Basta solo manejar el contenido de esta, sin referirse al objeto en pantalla.

CLASE C220 (derivada de CDialog).-

Esta clase es la asociada y encargada de manejar la Caja de Diálogo para IDD_220. Recuerde que dicha caja de diálogo es útil para los elementos que poseen el número de Identificación, dos entradas y 2 parámetros.

Funciones

C220.- Esta es la función constructora de la Caja de Diálogo. Se encarga de crear la Caja de Diálogo, y cargar cada campo que se presente con valores nulos. Sin embargo no grafica la caja en pantalla, será necesario otra función.

DoDataExchange.- Esta función en cambio se encarga de tomar los valores ingresados por el usuario en cada campo de la Caja y asociarlos con variables definidas dentro de esta clase. Así el programador utiliza directamente las variables y no los identificadores de los objetos en pantalla.

pone_titulo.- Recordemos que la misma Caja de Diálogo puede servir a más de un elemento que posean iguales características. La gran diferencia está en colocar el nombre del Elemento en la parte superior de la caja. Esta función se encarga de poner dicho nombre, de acuerdo al elemento específicamente escogido.

OnOK.- Esta función tiene la misión de revisar cualquier falta o error cometido por el usuario al llenar los campos de la caja de diálogo. Envía ventanitas en el centro de la pantalla en caso de haberse cometido el error.

m_numero (int)	m_id (int)
m_entrada1 (int)	m_entrada2 (int)
m_parametro1(float)	m_parametro2(float)

Variables

m_id.- La función pone_titulo necesita conocer el identificador específico del elemento, para así colocarle el nombre a la caja de diálogo. Esta variable es la encargada de proporcionarle dicho nombre.

m_numero.- El valor que el usuario ingresa como identificador del elemento a su vez es recibido por esta variable. Así no es necesario que el usuario utilice el identificador del campo en pantalla, basta manejar el contenido de dicha variable.

m_entrada1.- El valor ingresado por el usuario representando el Identificador de otro elemento que sirve como entrada al actual, es recibido diréctamente por esta variable. Basta solo manejar el contenido de esta, sin referirse al objeto en pantalla.

m_entrada2.- El valor ingresado por el usuario representando el Identificador de otro elemento que sirve como entrada al actual, es recibido diréctamente por esta variable. Basta solo manejar el contenido de esta, sin referirse al objeto en pantalla.

m_parametro1.- El valor ingresado por el usuario en el campo de parámetro 1, es recibido diréctamente por esta variable. Basta manejar y cargar el contenido de esta variable.

m_parametro2.- El valor ingresado por el usuario en el campo de parámetro 2, es recibido diréctamente por esta variable. Basta manejar y cargar el contenido de esta variable.

CLASE C300 (derivada de CDialog).-

Esta clase es la asociada y encargada de manejar la Caja de Diálogo para IDD_300. Recuerde que dicha caja de diálogo es útil para los elementos que poseen el número de Identificación, y dos entradas.

Funciones

CDD.- Esta es la función constructora de la Caja de Diálogo. Se encarga de crear la Caja de Diálogo, y cargar cada campo que se presente con valores nulos. Sin embargo no grafica la caja en pantalla, será necesario otra función.

DoDataExchange.- Esta función en cambio se encarga de tomar los valores ingresados por el usuario en cada campo de la Caja y asociarlos con variables definidas dentro de esta clase. Así el programador utiliza diréctamente las variables y no los identificadores de los objetos en pantalla.

pone_titulo.- Recordemos que la misma Caja de Diálogo puede servir a más de un elemento que posean iguales características. La gran diferencia está en colocar el nombre del Elemento en la parte superior de la caja. Esta función se encarga de poner dicho nombre, de acuerdo al elemento específicamente escogido.

OnOK.- Esta función tiene la misión de revisar cualquier falta o error cometido por el usuario al llenar los campos de la caja de diálogo. Envía ventanitas en el centro de la pantalla en caso de haberse cometido el error.

m_numero (int)	m_id (int)	
m_entrada1 (int)	m_entrada2 (int)	m_entrada3 (int)

Variables

m_id.- La función pone_titulo necesita conocer el identificador específico del elemento, para así colocarle el nombre a la caja de diálogo. Esta variable es la encargada de proporcionarle dicho nombre.

m_numero.- El valor que el usuario ingresa como identificador del elemento a su vez es recibido por esta variable. Así no es necesario que el usuario utilice el identificador del campo en pantalla, basta manejar el contenido de dicha variable.

m_entrada1.- El valor ingresado por el usuario representando el Identificador de otro elemento que sirve como entrada al actual, es recibido diréctamente por esta variable. Basta solo manejar el contenido de esta, sin referirse al objeto en pantalla.

m_entrada2.- El valor ingresado por el usuario representando el Identificador de otro elemento que sirve como entrada al actual, es recibido directamente por esta variable. Basta solo manejar el contenido de esta, sin referirse al objeto en pantalla.

m_entrada3.- El valor ingresado por el usuario representando el Identificador de otro elemento que sirve como entrada al actual, es recibido directamente por esta variable. Basta solo manejar el contenido de esta, sin referirse al objeto en pantalla.

CLASE C330 (derivada de CDialog).-

Esta clase es la asociada y encargada de manejar la Caja de Diálogo para IDD_330. Recuerde que dicha caja de diálogo es útil para los elementos que poseen el número de Identificación, tres entradas y tres parámetros.

Funciones

C330.- Esta es la función constructora de la Caja de Diálogo. Se encarga de crear la Caja de Diálogo, y cargar cada campo que se presente con valores nulos. Sin embargo no grafica la caja en pantalla, será necesario otra función.

DoDataExchange.- Esta función en cambio se encarga de tomar los valores ingresados por el usuario en cada campo de la Caja y asociarlos con variables definidas dentro de esta clase. Así el programador utiliza directamente las variables y no los identificadores de los objetos en pantalla.

pone_titulo.- Recordemos que la misma Caja de Diálogo puede servir a más de un elemento que posean iguales características. La gran diferencia está en colocar el nombre del Elemento en la parte superior de la caja. Esta función se encarga de poner dicho nombre, de acuerdo al elemento específicamente escogido.

OnOK.- Esta función tiene la misión de revisar cualquier falta o error cometido por el usuario al llenar los campos de la caja de diálogo. Envía ventanitas en el centro de la pantalla en caso de haberse cometido el error.

m_numero (int)	m_id (int)	
m_entrada1 (int)	m_entrada2 (int)	m_entrada3 (int)
m_parametro1(float)	m_parametro2(float)	m_parametro3(float)

Variables

m_id.- La función pone_titulo necesita conocer el identificador específico del elemento, para así colocarle el nombre a la caja de diálogo. Esta variable es la encargada de proporcionarle dicho nombre.

m_numero.- El valor que el usuario ingresa como identificador del elemento a su vez es recibido por esta variable. Así no es necesario que el usuario utilice el identificador del campo en pantalla, basta manejar el contenido de dicha variable.

m_entrada1.- El valor ingresado por el usuario representando el Identificador de otro elemento que sirve como entrada al actual, es recibido diréctamente por esta variable. Basta solo manejar el contenido de esta, sin referirse al objeto en pantalla.

m_entrada2.- El valor ingresado por el usuario representando el Identificador de otro elemento que sirve como entrada al actual, es recibido diréctamente por esta variable. Basta solo manejar el contenido de esta, sin referirse al objeto en pantalla.

m_entrada3.- El valor ingresado por el usuario representando el Identificador de otro elemento que sirve como entrada al actual, es recibido diréctamente por esta variable. Basta solo manejar el contenido de esta, sin referirse al objeto en pantalla.

m_parametro1.- El valor ingresado por el usuario en el campo de parámetro 1, es recibido diréctamente por esta variable. Basta manejar y cargar el contenido de esta variable.

m_parametro2.- El valor ingresado por el usuario en el campo de parámetro 2, es recibido diréctamente por esta variable. Basta manejar y cargar el contenido de esta variable.

m_parametro3.- El valor ingresado por el usuario en el campo de parámetro 3, es recibido diréctamente por esta variable. Basta manejar y cargar el contenido de esta variable.

Clase CGrafico (derivada de CScrollView).-

Esta clase es la encargada de obtener los valores de las salidas, los tiempos de cada muestra y generar una vista (una ventana) en la pantalla que muestre la curva representativa de la salida

...gida, vs. tiempo. Además muestra en la parte izquierda de dicha ventana la tabla con todas ...muestras obtenidas, tanto sus valores como sus tiempos. Recuerde que en la pantalla solo se ...den observar las 21 primeras muestras (limitación impuesta por el tamaño de la pantalla). ...en la hoja de papel o bien en modo *Revisión* se logra visualizar toda la tabla con un ...mo de hasta 51 muestras (en este caso la limitante es el tamaño *Carta* que puede tener una ... de papel).

...ciones

...rafico.- Esta es la función constructora de la clase. Su única función es la creación de esta ...na.

...Document.- Es el enlace que existe entre esta vista (ventana del gráfico) con la información ...lada dentro del documento (circuito). La clase CGrafico solo se encarga de recoger datos y ...trarlos en pantalla o en papel. Pero los datos fueron calculados y manipulados a su vez en ... clase de tipo *Documento*. GetDocument devuelve un puntero a dicha clase-documento. ... es la única forma de accesar a los datos calculados en documento.

...nitialUpdate.- En esta función se especifica el tamaño que requiere para la ventana del ...fico (la vista). También es importante especificar en qué modo (unidades métricas) quiero ...bujar, o especificar el tamaño de los objetos que pienso insertar en esta ventana. Recuerde ... un mismo objeto puede especificarse bien en pixels, en cms, o en pulgadas; esto queda ...nido por el modo que yo desee.

~...rafico.- Una vez que deseo cerrar (desaparecer) la vista-gráfico de la pantalla, se invoca su ...ción destructora. Ella es la encargada de liberar memoria y recursos para que otra clase los ...da usar.

...Draw.- Ella se encarga de graficar cada punto de la curva de salida. Además coloca los ...res y ejes del gráfico. También es responsable por crear y llenar la tabla de las muestras.

...PreparePrinting.- Marca la cantidad de hojas que deberán imprimirse (o mostrarse en ...do *Revisión*), a partir de qué hoja se visualizará en pantalla y cuántas tiene en total el gráfico. ...lógico pensar que cada gráfico solo tendrá una sola hoja. Además esta función se encarga de

presentar en pantalla las cajas de diálogos y ventanas típicas de cualquier impresión (desde qué hoja desea imprimir, todo el documento, tipo de papel, tipo de impresora, etc).

OnPrint.- Se invoca cada vez que se imprimirá una hoja del gráfico. Está diseñada para incluir un encabezado con el nombre del circuito analizado en cada hoja. Además ella se encarga de llamar a OnDraw quien será la responsable de dibujar la curva y la tabla. OnPrint está únicamente dirigida al uso de la impresora, y no a presentar datos en pantalla.

Clase CGFrame (derivada de CMDIChildWnd).-

Cada vez que se crea la vista del gráfico (clase CGrafico) la clase CGFrame es la responsable de ponerle el *marco* alrededor de dicha vista. La vista es el interior o contenido de la ventana; pero es el marco quien encierra con los bordes y el título a dicha vista. CGFrame nos servirá únicamente para colocarle dicho marco solo a la vista del gráfico (mas no al listado, quien usa el marco por defecto) y le cambia el título, en lugar de poner el nombre del Circuito le coloca la palabra *Gráfico.*

Funciones

CGFrame.- Esta es la función constructora o creadora del marco. En el momento que se crea la vista también se invoca la creación del marco.

PreCreateWindow.- Su misión es cambiarle el nombre al título del marco. Ya no será el nombre del circuito pero si la palabra *Gráfico.*

~CGFrame.- Una vez que se deja de usar la vista del gráfico, tanto la vista como el marco se autodestruyen. Esta es la función destructora del marco.

Clase Ctiempo (derivada de CDialog).-

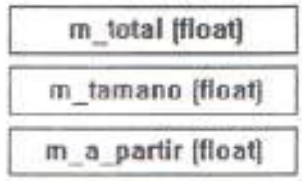
Esta clase está asociada directamente a la Caja de Diálogo IDD_TIEMPO. Ella es la encargada de crear y mostrar esta opción donde el usuario deberá ingresar los parámetros de tiempo para la simulación.

Funciones

Ctiempo.- Esta es la función constructora de la clase. Sin embargo no la muestra en pantalla; para esto será necesario usar una función perteneciente a la clase CDialog.

DoDataExchange.- Esta función es la encargada de colocar y recibir los valores ingresados y que se presentarán al usuario en cada campo. También se encarga de tomar dichos valores y guardarlos en variables definidas en la clase Ctiempo.

OnOK.- Esta es la función encargada de verificar que el usuario no ha cometido ningún error al llenar los campos. Coloca en pantalla pequeñas ventanas donde se menciona el tipo de error cometido.



Variables

m_total.- En esta variable real se guarda el valor del tiempo total de simulación (obtenido a partir de la caja de diálogo de Ctiempo). Este dato lo ingresa el usuario.

m_tamano.- En esta variable se guarda el tamaño que tendrá el intervalo de integración (ingresado por el usuario).

m_a_partir.- En esta variable se guarda el tiempo inicial a partir del cual se presentarán las muestras de salida en la tabla. A pesar de que la simulación dura todo el tiempo total, se le permite al usuario especificar qué porción de las muestras desea ver.

Clase CMainFrame (derivada de CMDIFrameWnd).-

Esta clase coloca el marco, la Barra de Herramientas y la Barra de Status para la aplicación SIMP. Además contiene funciones que crean o activan la vista del gráfico (en caso que aún no se haya creado, o si se creó la pone al frente).

Funciones

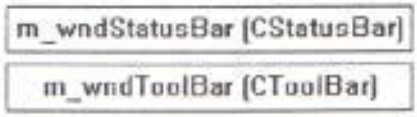
CMainFrame.- Esta es la función constructora. Ella se encarga de crear el marco con el título y los bordes para la aplicación CSMP.

CreateOrActivateFrame.- Llamada por la función OnRun. Esta es la encargada de crear o si acaso ya está presente volverla activa a la vista que muestra la curva de la salida.

~CMainFrame.- Cuando se cierra la aplicación, al mismo tiempo que desaparece la aplicación también desaparece su marco principal; esta función es la destructora del marco, encargada de liberar los recursos asociados a él.

OnCreate.- Es en esta función donde se da la orden de crearse el marco. También se da la orden de crear la Barra de Herramientas y la Barra de Status. Además se verifica la posibilidad de que alguno de ellos no se cree (mostrando el respectivo mensaje de error).

OnRun.- Recordemos que hay una opción en el menú de Opciones llamada *Mostrar la Gráfica*. Esta opción se encarga de crear o de volver activa la ventana donde se muestra el gráfico de la salida. La función OnRun es quien realiza dicha acción, a su vez llamando a la función CreateOrActivateFrame.



Variables

m_wndStatusBar.- Esta es a su vez una clase de tipo CStatusBar. Ella representará a la Barra de Status de la aplicación CSMP. OnCreate invoca a funciones de creación dentro de CStatusBar, así logra colocar dicha barra en pantalla.

m_wndToolBar.- Esta es a su vez una clase de tipo CToolBar. Ella representará a la Barra de Herramientas de la aplicación CSMP. OnCreate invoca a funciones de creación dentro de CToolBar, así logra colocar dicha barra en pantalla.

Clase CCsmpApp (derivada de CWinApp).-

Esta es la clase más importante de todo el sistema. Aquí es donde se crea la única instancia de tipo Aplicación. En otras palabras, para iniciar, todo lo básico es crear la aplicación y de allí en adelante el resto del programa. En esta clase se define el tipo de ventanas que se presentarán y el tipo de documento que se podrían abrir (en nuestro caso circuitos). También incluimos qué marcos usarán las ventanas y qué vistas existirán.

Funciones

CCsmpApp.- Esta es la función constructora de la aplicación. Su misión es crearla.

InitInstance.- Aquí se dan las órdenes para colocar el marco principal de la aplicación, ponerle color al fondo de la ventana principal y mencionar los tipos de documentos, vistas y marcos que podremos abrir dentro de la aplicación. También se permite abrir algún circuito, en caso de que el usuario lo haya especificado en el momento de la ejecución de la aplicación (desde la línea de comandos).

OnAppAbout.- Recordemos que en el Menú de Ayudas existe la opción *Acerca de CSMP*. Esta función está asociada con dicha opción, y su misión es mostrar en pantalla la Caja de Diálogo con el nombre del programa, su versión y propietario (además del CopyRight).

m_pElementoTemplate (CMultiDocTemplate)
m_pGraficoTemplate (CMultiDocTemplate)

Variables

m_pElementoTemplate.- En el momento de creación de la aplicación se le debe mencionar los tipos de vistas, marcos y documentos que se podrían abrir. Esta variable especifica un conjunto de valores que incluyen: documento tipo circuito, vista del listado de elementos y marco de la vista (el por defecto que muestra como título el nombre del circuito abierto).

m_pGraficoTemplate.- En cambio esta variable también le informa a la aplicación que podría abrir un segundo conjunto de datos: documento tipo circuito, vista de un gráfico, y marco que muestre la palabra *Gráfico* como título.

Clase CAboutDlg (derivada de CDialog).-

Esta es la clase asociada a la caja de diálogo de la opción *Acerca de CSMP*. Muestra el nombre del sistema, su versión, su ícono, el propietario y el CopyRight.

Funciones

CAboutDlg.- Esta es la función constructora de la caja de diálogo. Su misión es crearla pero no necesariamente mostrarla.

DoDataExchange.- Usada para recolectar los valores ingresados en los campos de la caja de diálogo y asignárselos a variables dentro de la clase CAboutDlg. Para esta clase no existe ninguna variable ni valor que deba llenarse en algun campo.

Clase CCsmpView (derivada de CScrollView).-

Esta es la clase que nos muestra una vista con el listado de los elementos que conforman el circuito. Aquí es donde podemos adicionar, eliminar y modificar alguno de los elementos que aparecen en dicho listado. Por supuesto los datos y valores de cada elemento se encuentran en una clase de tipo documento. Si la lista es larga tambien nos permite movilizarnos usando las barras de scroll.

Funciones

CCsmpView.- Esta es la función constructora. Su misión es crear la vista y mostrarla.

GetDocument.- Para poder usar los datos de cada elemento del circuito es necesario tener un vínculo con la clase que los contiene. En este caso la función GetDocument devuelve ese vínculo (de tipo puntero) para que a través de él podamos accesar a información del circuito y poderla mostrar en nuestra vista.

UpdateRow.- Cada vez que uno de los elementos es modificado en sus valores, o se añade un nuevo elemento, o se elimina alguno de los existentes, es necesario reflejar dichas acciones en la vista (osea en la lista de elementos mostrada). Cada vez que ocurre ese tipo de cambio se llama a la función OnUpdate quien tiene la labor de saber qué elemento fue el modificado. En UpdateRow está la implementación de cómo reflejar dicho cambio en la vista (para el elemento

una vez localizado). Si son muchos elementos los alterados, entonces esta función refleja todos los cambios necesarios. Ella se encarga de hacer scroll si es necesario, o de darse cuenta si el elemento que modifiqué está presente actualmente en la pantalla o no. También actualiza el elemento que el usuario últimamente escogió, y desactualiza el que previamente había escogido.

OnUpdate.- Si hubo algún cambio en el circuito (osea en la clase documento) ese cambio es inmediátamente anunciado a la función OnUpdate. Esta función es la encargada de descifrar cual es el elemento que se modificó o añadió o eliminó en la clase documento. A su vez OnUpdate llamará a UpdateRow.

OnInitialUpdate.- Luego de crear la vista la siguiente función a llamarse es OnInitialUpdate. En esta función cargamos quién fue el último elemento escogido/resaltado por el usuario, y cuántos elementos hubieron la última vez que se invocó a esta vista.

GetRowWidthHeight.- Esta función calcula el ancho y alto de una fila (osea de un elemento) en el listado. Recordemos que cada elemento usa 2 hileras de información, la altura total es la suma de las alturas de cada hilera.

GetActiveRow.- Obtenemos el índice que representa el elemento que últimamente seleccionó el usuario.

GetRowCount.- Nos devuelve la cantidad de elementos que hasta el momento ha ingresado el usuario en el circuito.

OnDrawRow.- Esta es la función que se encarga de dibujar en pantalla o bien en papel o en modo *Revisión* las filas correspondientes a los elementos del circuito. Además resalta la última escogida por el usuario y desactiva la que escogió antes.

ChangeSelectionNextRow.- Si acaso el usuario utilizó las teclas de flechas hacia arriba o hacia abajo, entonces se debe resaltar la nueva fila escogida y desactivar la previa.

ChangeSelectionToRow.- El usuario puede escoger una fila totalmente lejana a la actualmente seleccionada (usando el ratón). Esta función cambia el resaltado hacia la nueva fila que escogió.

~CCsmpView.- Esta es la función destructora de la vista. Se invoca una vez que el usuario decide cerrar esta vista/listado de elementos. Libera los recursos empleados por dicha vista.

OnDraw.- Ejecuta un lazo que invoca OnDrawRow para cada fila donde se detectó un cambio.

OnPrepareDC.- Dependiendo del dispositivo destino (quien recibirá la información) se debe calcular las dimensiones del texto. Esto lo realiza OnPrepareDC.

OnPreparePrinting.- Antes de imprimir el listado en papel o en modo *Revisión*, es necesario conocer la cantidad de páginas (largo del documento). También se debería mostrar la caja de diálogo de Impresión, para que el usuario escoja qué páginas desea imprimir.

OnBeginPrinting.- Según el tamaño de la hoja de papel se pueden calcular cuántas filas de elementos pueden entrar por página. En esta función también se puede especificar a partir de qué página podría comenzar el modo *Revisión*.

OnPrint.- Dedicada solo a la impresora o al modo *Revisión*. OnPrint coloca un encabezado al principio de cada hoja y a su vez invoca a OnDraw para que imprima el listado de elementos en cada hoja (según cuántas entren por hoja).

CalculateRowMetrics.- Esta función calcula el ancho y alto del tipo de letra que se emplea. En base a aquello puede obtenerse el ancho y alto de cada fila (elemento).

UpdateScrollSizes.- Si el usuario ingresa más elementos de los que pueden entrar en el tamaño actual de la ventana, será necesario redimensionar las barras de scroll. Recuerde que cada vez que hace scroll, se mueven una cierta cantidad de filas hacia arriba o hacia abajo (también izquierda o derecha). Esta función es la encargada de calcular cuánto se debe desplazar.

RowToWndRect.- Si conozco el índice de algún elemento, puedo convertir dicho índice en coordenadas pixels dentro de la pantalla.

RowToYPos.- Si conozco el índice de algún elemento, puedo convertir dicho índice en coordenadas lógicas dentro del listado total. La diferencia entre coordenadas lógicas y físicas, es que un listado puede ser mucho más grande que el tamaño actual de la vista. Una coordenada física solo está dentro de las dimensiones de la vista. Pero una coordenada lógica puede sobrepasar las dimensiones de la vista, y son con respecto al tamaño completo del listado.

RectLPtoRowRange.- Se conoce un rectángulo en coordenadas lógicas. Por medio de esta función dicho rectángulo se puede transformar en un rango de índices, correspondientes a varios elementos.

LastViewableRow.- La vista a pesar de que es tipo Scroll tiene un límite máximo. Esta función calcula el índice del posible último elemento que entraría en dicha vista.

OnLButtonDown.- Si el usuario presiona el botón izquierdo del ratón se selecciona automáticamente dicho elemento apuntado.

OnSize.- Esta función ajustará los limites de las barras de scroll, si acaso el usuario decide modificar el tamaño del marco que encierra esta vista.

OnKeyDown.- El usuario puede utilizar las teclas de flechas y de PageUp y PageDown. Esta función reconoce que se han presionado dichas teclas y cambia el resaltado al nuevo elemento escogido.

OnLButtonDblClk.- En caso de que el usuario ejecuta un doble click con el botón izquierdo del ratón, se mostrará la caja de diálogo correspondiente al tipo del elemento seleccionado. Esta caja aparecerá con los campos llenos de los valores previamente grabados para dicho elemento.

OnElimina.- Esta función reconoce la tecla DELETE en caso de que la presione el usuario. El objetivo es eliminar el elemento resaltado y quitarlo de la lista.

OnRButtonDblClk.- Si el usuario desea ver la gráfica de salida de algún elemento, puede hacer doble click con el botón derecho del ratón. Esta función reconoce que el usuario ha presionado el botón del ratón 2 veces.

m_nRowWidth (int)	m_nPrevRowCount (int)
m_nRowHeight (int)	m_nRowsPerPrintedPage (int)
m_nPrevSelectedRow (int)	bNeedToScroll (BOOL)

Variables

m_nRowWidth.- Se almacena el ancho de cada fila correspondiente a un elemento.

m_nRowHeight.- Se almacena el alto de las 2 hileras que corresponden a un elemento.

m_nPrevSelectedRow.- Contiene el índice de la fila últimamente seleccionada.

m_nPrevRowCount.- Contiene la cantidad de elementos que anteriormente contenía el circuito.

m_nRowsPerPrintedPage.- Contiene la cantidad de filas de elementos que pueden entrar por hoja de papel (depende del tamaño de la hoja).

NeedToScroll.- Es verdadero en caso de que el elemento que se escogió no está presente en la vista. Necesariamente deberá realizarse un scroll de la vista, para que este elemento aparezca. Dicho elemento pudo estar más abajo o más arriba que los actualmente visibles (recuerde que el circuito puede ser muy grande, y no todo entra en la vista al mismo tiempo).

Clase CFixedLenRecHint (derivada de CObject).-

Esta clase sirve para informar a cualquier vista qué porción del documento ha sido modificado o aumentado. Pueden ser los índices de todos aquellos elementos que se modificaron. Esta clase actúa como una *pista* que servirá para que la clase tipo Vista, se dé cuenta de quién ha sido modificado y lo actualice en pantalla.

Funciones

CFixedLenRecHint.- Esta es la constructora de la clase. Aquí se crea la *pista* que servirá para que las vistas que reciben el mensaje de actualización (OnUpdate) sepan a quién deben actualizar dentro del documento.

Clase CCsmpDoc (derivada de CDocument).-

Esta clase se crea en el momento que el usuario abre un circuito de Computador Analógico. Recordemos que todas las operaciones de adición, eliminación y modificación de los valores de cualquier elemento se logran a través de esta clase. Ella es quién contiene las funciones que modifican o cambian valores de elementos. Además esta clase también posee funciones que se invocarán justo para el momento de la simulación. Debemos dar a conocer que todos los elementos de un circuito deben colocarse en un orden prioritario antes de comenzar la simulación; esta función también está incluida aquí. Por supuesto tendremos funciones que realizan los cálculos específicos para cada tipo de elemento. El usuario podrá observar que por más que ingrese los elementos desordenadamente (de acuerdo al número de sus identificadores respectivos), la clase automáticamente los ordena en pantalla. Esta clase también se encarga de verificar los requerimientos mínimos para una correcta simulación (cantidad mínima de integradores, si acaso ingresó los parámetros de tiempo, y si todas las entradas existen).

Funciones

CCsmpDoc.- Esta es la función constructora de la clase. No solo está encargada de encerar y poner falso a todas las variables que se emplean en CCsmpDoc, sino que también crea la clase documento.

Poner_Order.- Recordemos que justo antes de la simulación será necesario ordenar los elementos del circuito de acuerdo a su tipo, quienes son sus entradas y quienes son constantes, unidades de retardo o integradores. A pesar de que el listado permanece en el orden original con que lo ingresó el usuario, se requiere este segundo ordenamiento para asegurarse de que se ejecutarán los elementos de mayor prioridad antes que los de menor.

Simula.- Esta es la función que comienza todo el proceso de simulación. Cada intervalo de tiempo (inclusive cada medio intervalo de tiempo) se van calculando las salidas de los elementos en orden prioritario. A su vez ella llamará a la función encargada de calcular los valores de salida para cada tipo de elemento. También está atenta en caso de cualquier error en los cálculos, y detendrá inmediátamente el proceso de simulación.

UpdateAllV.- Recordemos que cada vez que se realiza una modificación en el circuito dichos cambios se deben reflejar en pantalla (específicamente en la vista-listado). La función encargada de comunicar a todas las vistas existentes de que se ha logrado un cambio es UpdateAllV. Ella es la encargada de proporcionar la *pista* que posteriormente le informará a la vista respectiva de que cierto elemento fue alterado.

GetDocSize.- Esta función es típicamente llamada por la vista del gráfico. Esta le proporciona el tamaño idóneo que debería tener la vista (de tipo scroll) para que allí se muestre la curva de salida del elemento escogido.

Elimina_fila.- Recordemos que el usuario puede presionar la tecla DELETE y así eliminar un elemento dentro del listado. La función que captaba el hecho de presionar dicha tecla se encuentra en la clase vista-listado. Pero es con Elimina_fila que realmente se logra borrar dicho elemento del circuito.

DeleteContents.- La función constructora invoca a esta función con la finalidad de blanquear o encerar todas las variables que están involucradas en la clase. También coloca falsos en todas aquellas variables de tipo Booleano.

Chequeo_todos_existen.- Cada cierto tiempo es necesario que el programa vaya chequeando si todas las entradas para cada elemento existen. Esto es una medida preventiva para que cuando el usuario decida empezar la simulación, todos los elementos estén correctamente conectados.

Ordena.- Recuerde que el usuario puede ingresar los elementos en forma desordenada pero es el programa que automáticamente los ordena de acuerdo a los valores de sus identificadores. Esto se refleja inmediátamente en la vista-listado.

Limpia_m_simula.- Su único objetivo es encerar la matriz m_simula. Recuerde que en m_simula cargaremos los valores más usados de cada elemento. Esta es una manera práctica de acceder a los datos de algún elemento y de grabar las salidas y vectores de integración. Debo aclarar que en m_arreglo es donde el usuario almacena los valores que va ingresando; sin embargo es en m_simula donde se vuelven a cargar parte de dichos valores y es aquí donde se almacena las salidas.

Calculo.- Calculo es la función invocada por Simula. Aquí se ingresa en un lazo donde cada elemento según su orden prioritario, se le calcula su salida (o también su vector integrativo). Calculo a su vez llama a otras subfunciones que son las que realmente ejecutan la operación que generará la salida del elemento. Estas subfunciones marchan de acuerdo al tipo de elemento. Recuerde que la simulación examina *todos* los elementos del circuito y a cada uno le calcula su valor de salida.

C_BANG_BANG.- Esta subfunción es la que proporciona el valor de salida, para este elemento.

C_ESPACIO_MUERTO.- Esta subfunción es la que proporciona el valor de salida, para este elemento.

C_GENERADOR_FUNCION.- Esta subfunción es la que proporciona el valor de salida, para este elemento.

C_AMPLIFICADOR.- Esta subfunción es la que proporciona el valor de salida, para este elemento.

C_RAIZ_CUADRADA.- Esta subfunción es la que proporciona el valor de salida, para este elemento.

C_INTEGRADOR.- Esta subfunción es la que proporciona el valor de salida, para este elemento.

C_ALEATORIO.- Esta subfunción es la que proporciona el valor de salida, para este elemento.

C_CONSTANTE.- Esta subfunción es la que proporciona el valor de salida, para este elemento.

C_LIMITADOR.- Esta subfunción es la que proporciona el valor de salida, para este elemento.

C_MAGNITUD.- Esta subfunción es la que proporciona el valor de salida, para este elemento.

C_SEPARADOR_NEGATIVO.- Esta subfunción es la que proporciona el valor de salida, para este elemento.

C_OFFSET.- Esta subfunción es la que proporciona el valor de salida, para este elemento.

C_SEPARADOR_POSITIVO.- Esta subfunción es la que proporciona el valor de salida, para este elemento.

C_FIN.- Esta subfunción es la que proporciona el valor de salida, para este elemento.

C_RELE.- Esta subfunción es la que proporciona el valor de salida, para este elemento.

C_GENERADOR_PULSOS.- Esta subfunción es la que proporciona el valor de salida, para este elemento.

C_UNIDAD_RETARDO.- Esta subfunción es la que proporciona el valor de salida, para este elemento.

C_VACIO.- Esta subfunción es la que proporciona el valor de salida, para este elemento.

C_SUMADOR_PESO.- Esta subfunción es la que proporciona el valor de salida, para este elemento.

C_MULTIPLICADOR.- Esta subfunción es la que proporciona el valor de salida, para este elemento.

C_BIFURCACION.- Esta subfunción es la que proporciona el valor de salida, para este elemento.

C_ORDEN_CERO.- Esta subfunción es la que proporciona el valor de salida, para este elemento.

C_SUMADOR.- Esta subfunción es la que proporciona el valor de salida, para este elemento.

C_DIVISOR.- Esta subfunción es la que proporciona el valor de salida, para este elemento.

C_INVERSOR.- Esta subfunción es la que proporciona el valor de salida, para este elemento.

OnNewDocument.- Esta función es invocada cada vez que se desea generar un circuito totalmente nuevo (nunca antes escrito). La función retornará verdadero en caso de que sea nuevo, o falso en caso de ya existir.

No_repetido.- Recordemos que cada elemento debe tener un identificador único y que este no puede concidir con otro ya existente (dentro de un mismo circuito). Esta función verifica que cada vez que un elemento se crea o se modifica el numero de identificación no coincida con otro ya existente.

~CCsmpDoc.- Si el usuario decide cerrar un circuito (no desea analizarlo más) se invoca automáticamente la función destructora de la clase documento. Esta función libera los recursos que utilizaba la clase mientras existia.

Serialize.- Cada vez que el usuario decide cargar de algún archivo o grabar hacia algún archivo un circuito especifico, invoca indiréctamente a esta función. Serialize se encarga de traer los datos desde un archivo especificado hacia las variables en memoria de la clase documento; también se ejecuta en sentido inverso.

OnAleatorio.- Si el usuario decide presionar el botón de la Barra de Herramientas para este elemento, o bien lo escoge dentro del menú de Elementos, esta función se encarga de cargar los campos de la caja de diálogo asociada a este elemento, crear la caja, presentar la caja en pantalla, recibir los nuevos valores ingresados por el usuario, chequear por posibles errores de ingreso y finalmente cargar dichos datos en las variables de memoria. Tómese en cuenta que esta función aún no calcula la salida del elemento.

OnAmplificador.- Si el usuario decide presionar el botón de la Barra de Herramientas para este elemento, o bien lo escoge dentro del menú de Elementos, esta función se encarga de cargar los campos de la caja de diálogo asociada a este elemento, crear la caja, presentar la caja en pantalla, recibir los nuevos valores ingresados por el usuario, chequear por posibles errores de

greso y finalmente cargar dichos datos en las variables de memoria. Tómese en cuenta que esta función aún no calcula la salida del elemento.

BangBang.- Si el usuario decide presionar el botón de la Barra de Herramientas para este elemento, o bien lo escoge dentro del menú de Elementos, esta función se encarga de cargar los campos de la caja de diálogo asociada a este elemento, crear la caja, presentar la caja en pantalla, recibir los nuevos valores ingresados por el usuario, chequear por posibles errores de ingreso y finalmente cargar dichos datos en las variables de memoria. Tómese en cuenta que esta función aún no calcula la salida del elemento.

Bifurcacion.- Si el usuario decide presionar el botón de la Barra de Herramientas para este elemento, o bien lo escoge dentro del menú de Elementos, esta función se encarga de cargar los campos de la caja de diálogo asociada a este elemento, crear la caja, presentar la caja en pantalla, recibir los nuevos valores ingresados por el usuario, chequear por posibles errores de ingreso y finalmente cargar dichos datos en las variables de memoria. Tómese en cuenta que esta función aún no calcula la salida del elemento.

Constante.- Si el usuario decide presionar el botón de la Barra de Herramientas para este elemento, o bien lo escoge dentro del menú de Elementos, esta función se encarga de cargar los campos de la caja de diálogo asociada a este elemento, crear la caja, presentar la caja en pantalla, recibir los nuevos valores ingresados por el usuario, chequear por posibles errores de ingreso y finalmente cargar dichos datos en las variables de memoria. Tómese en cuenta que esta función aún no calcula la salida del elemento.

Divisor.- Si el usuario decide presionar el botón de la Barra de Herramientas para este elemento, o bien lo escoge dentro del menú de Elementos, esta función se encarga de cargar los campos de la caja de diálogo asociada a este elemento, crear la caja, presentar la caja en pantalla, recibir los nuevos valores ingresados por el usuario, chequear por posibles errores de ingreso y finalmente cargar dichos datos en las variables de memoria. Tómese en cuenta que esta función aún no calcula la salida del elemento.

EspacioMuerto.- Si el usuario decide presionar el botón de la Barra de Herramientas para este elemento, o bien lo escoge dentro del menú de Elementos, esta función se encarga de cargar los campos de la caja de diálogo asociada a este elemento, crear la caja, presentar la caja en

pantalla, recibir los nuevos valores ingresados por el usuario, chequear por posibles errores de ingreso y finalmente cargar dichos datos en las variables de memoria. Tómese en cuenta que esta función aún no calcula la salida del elemento.

OnFin.- Si el usuario decide presionar el botón de la Barra de Herramientas para este elemento, o bien lo escoge dentro del menú de Elementos, esta función se encarga de cargar los campos de la caja de diálogo asociada a este elemento, crear la caja, presentar la caja en pantalla, recibir los nuevos valores ingresados por el usuario, chequear por posibles errores de ingreso y finalmente cargar dichos datos en las variables de memoria. Tómese en cuenta que esta función aún no calcula la salida del elemento.

OnGeneradorF.- Si el usuario decide presionar el botón de la Barra de Herramientas para este elemento, o bien lo escoge dentro del menú de Elementos, esta función se encarga de cargar los campos de la caja de diálogo asociada a este elemento, crear la caja, presentar la caja en pantalla, recibir los nuevos valores ingresados por el usuario, chequear por posibles errores de ingreso y finalmente cargar dichos datos en las variables de memoria. Tómese en cuenta que esta función aún no calcula la salida del elemento.

OnGeneradorP.- Si el usuario decide presionar el botón de la Barra de Herramientas para este elemento, o bien lo escoge dentro del menú de Elementos, esta función se encarga de cargar los campos de la caja de diálogo asociada a este elemento, crear la caja, presentar la caja en pantalla, recibir los nuevos valores ingresados por el usuario, chequear por posibles errores de ingreso y finalmente cargar dichos datos en las variables de memoria. Tómese en cuenta que esta función aún no calcula la salida del elemento.

OnIntegrador.- Si el usuario decide presionar el botón de la Barra de Herramientas para este elemento, o bien lo escoge dentro del menú de Elementos, esta función se encarga de cargar los campos de la caja de diálogo asociada a este elemento, crear la caja, presentar la caja en pantalla, recibir los nuevos valores ingresados por el usuario, chequear por posibles errores de ingreso y finalmente cargar dichos datos en las variables de memoria. Tómese en cuenta que esta función aún no calcula la salida del elemento.

OnInversor.- Si el usuario decide presionar el botón de la Barra de Herramientas para este elemento, o bien lo escoge dentro del menú de Elementos, esta función se encarga de cargar los

mpos de la caja de diálogo asociada a este elemento, crear la caja, presentar la caja en ntalla, recibir los nuevos valores ingresados por el usuario, chequear por posibles errores de greso y finalmente cargar dichos datos en las variables de memoria. Tómese en cuenta que ta función aún no calcula la salida del elemento.

nLimitador.- Si el usuario decide presionar el botón de la Barra de Herramientas para este mento, o bien lo escoge dentro del menú de Elementos, esta función se encarga de cargar los mpos de la caja de diálogo asociada a este elemento, crear la caja, presentar la caja en ntalla, recibir los nuevos valores ingresados por el usuario, chequear por posibles errores de greso y finalmente cargar dichos datos en las variables de memoria. Tómese en cuenta que ta función aún no calcula la salida del elemento.

nMagnitud.- Si el usuario decide presionar el botón de la Barra de Herramientas para este mento, o bien lo escoge dentro del menú de Elementos, esta función se encarga de cargar los mpos de la caja de diálogo asociada a este elemento, crear la caja, presentar la caja en ntalla, recibir los nuevos valores ingresados por el usuario, chequear por posibles errores de greso y finalmente cargar dichos datos en las variables de memoria. Tómese en cuenta que ta función aún no calcula la salida del elemento.

nMultiplicador.- Si el usuario decide presionar el botón de la Barra de Herramientas para e elemento, o bien lo escoge dentro del menú de Elementos, esta función se encarga de cargar s campos de la caja de diálogo asociada a este elemento, crear la caja, presentar la caja en ntalla, recibir los nuevos valores ingresados por el usuario, chequear por posibles errores de greso y finalmente cargar dichos datos en las variables de memoria. Tómese en cuenta que ta función aún no calcula la salida del elemento.

nOffset.- Si el usuario decide presionar el botón de la Barra de Herramientas para este mento, o bien lo escoge dentro del menú de Elementos, esta función se encarga de cargar los mpos de la caja de diálogo asociada a este elemento, crear la caja, presentar la caja en ntalla, recibir los nuevos valores ingresados por el usuario, chequear por posibles errores de greso y finalmente cargar dichos datos en las variables de memoria. Tómese en cuenta que ta función aún no calcula la salida del elemento.

OnOrdenCero.- Si el usuario decide presionar el botón de la Barra de Herramientas para este elemento, o bien lo escoge dentro del menú de Elementos, esta función se encarga de cargar los campos de la caja de diálogo asociada a este elemento, crear la caja, presentar la caja en pantalla, recibir los nuevos valores ingresados por el usuario, chequear por posibles errores de ingreso y finalmente cargar dichos datos en las variables de memoria. Tómese en cuenta que esta función aún no calcula la salida del elemento.

OnRaizCua.- Si el usuario decide presionar el botón de la Barra de Herramientas para este elemento, o bien lo escoge dentro del menú de Elementos, esta función se encarga de cargar los campos de la caja de diálogo asociada a este elemento, crear la caja, presentar la caja en pantalla, recibir los nuevos valores ingresados por el usuario, chequear por posibles errores de ingreso y finalmente cargar dichos datos en las variables de memoria. Tómese en cuenta que esta función aún no calcula la salida del elemento.

OnRete.- Si el usuario decide presionar el botón de la Barra de Herramientas para este elemento, o bien lo escoge dentro del menú de Elementos, esta función se encarga de cargar los campos de la caja de diálogo asociada a este elemento, crear la caja, presentar la caja en pantalla, recibir los nuevos valores ingresados por el usuario, chequear por posibles errores de ingreso y finalmente cargar dichos datos en las variables de memoria. Tómese en cuenta que esta función aún no calcula la salida del elemento.

OnSeparadorN.- Si el usuario decide presionar el botón de la Barra de Herramientas para este elemento, o bien lo escoge dentro del menú de Elementos, esta función se encarga de cargar los campos de la caja de diálogo asociada a este elemento, crear la caja, presentar la caja en pantalla, recibir los nuevos valores ingresados por el usuario, chequear por posibles errores de ingreso y finalmente cargar dichos datos en las variables de memoria. Tómese en cuenta que esta función aún no calcula la salida del elemento.

OnSeparadorP.- Si el usuario decide presionar el botón de la Barra de Herramientas para este elemento, o bien lo escoge dentro del menú de Elementos, esta función se encarga de cargar los campos de la caja de diálogo asociada a este elemento, crear la caja, presentar la caja en pantalla, recibir los nuevos valores ingresados por el usuario, chequear por posibles errores de

greso y finalmente cargar dichos datos en las variables de memoria. Tómese en cuenta que sta función aún no calcula la salida del elemento.

OnSumador.- Si el usuario decide presionar el botón de la Barra de Herramientas para este lemento, o bien lo escoge dentro del menú de Elementos, esta función se encarga de cargar los ampos de la caja de diálogo asociada a este elemento, crear la caja, presentar la caja en antalla, recibir los nuevos valores ingresados por el usuario, chequear por posibles errores de greso y finalmente cargar dichos datos en las variables de memoria. Tómese en cuenta que sta función aún no calcula la salida del elemento.

OnSumadorPeso.- Si el usuario decide presionar el botón de la Barra de Herramientas para ste elemento, o bien lo escoge dentro del menú de Elementos, esta función se encarga de cargar os campos de la caja de diálogo asociada a este elemento, crear la caja, presentar la caja en antalla, recibir los nuevos valores ingresados por el usuario, chequear por posibles errores de greso y finalmente cargar dichos datos en las variables de memoria. Tómese en cuenta que sta función aún no calcula la salida del elemento.

OnTiempo.- Recuerde que el usuario puede escoger la opcion de Parámetros de Tiempo para gresar tanto tiempo total de simulación como cantidad de intervalos a tomarse. Esta función es encargada de crear, presentar y cargar los campos de la caja de diálogo asociada a dicha pción de menú. También recibe los valores ingresados por el usuario, verifica la existencia de gún error al ingresarse y finalmente los guarda en las variables respectivas de memoria.

OnUnidadRet.- Si el usuario decide presionar el botón de la Barra de Herramientas para este lemento, o bien lo escoge dentro del menú de Elementos, esta función se encarga de cargar los ampos de la caja de diálogo asociada a este elemento, crear la caja, presentar la caja en antalla, recibir los nuevos valores ingresados por el usuario, chequear por posibles errores de greso y finalmente cargar dichos datos en las variables de memoria. Tómese en cuenta que sta función aún no calcula la salida del elemento.

OnVacio.- Si el usuario decide presionar el botón de la Barra de Herramientas para este emento, o bien lo escoge dentro del menú de Elementos, esta función se encarga de cargar los ampos de la caja de diálogo asociada a este elemento, crear la caja, presentar la caja en antalla, recibir los nuevos valores ingresados por el usuario, chequear por posibles errores de

greso y finalmente cargar dichos datos en las variables de memoria. Tómese en cuenta que a función aún no calcula la salida del elemento.

UpdateRun.- Si el usuario cumplió con las condiciones mínimas para una correcta mulación (por lo menos un integrador, todas las entradas existen y además ingresó los rámetros de simulación), entonces la opción Mostrar Gráfica del menú de Opciones se abilitará. Por lo general esto ocurre la primera vez que el usuario decide visualizar alguna curva salida para algún elemento; de allí en adelante esta vista queda activada hasta que el usuario cide cerrarla. OnUpdateRun permite que dicha opción del menú se habilite (siempre que mplan las condiciones mínimas de simulación).

no_hay_cambios [BOOL]	m_order[MAXIMO] [int]
borro_linea [BOOL]	m_resto [int]
m_actual [int]	ir [float]
m_doc [int]	m_Y[MAXIMO+1] [float]
quien_esta_en_view [int]	m_YK[MAXIMO+1] [float]
tiempo_listo [BOOL]	m_DYDT[MAXIMO+1] [float]
cantidad_suficiente [BOOL]	m_INTG[MAXIMO+1] [int]
todos_existen [BOOL]	m_NOFG[MAXIMO+1] [int]
tiempos[51] [float]	m_NEQ [int]
resultados[51] [float]	m_NFG [int]
delta [float]	tiempo_total [float]
m_abortar [BOOL]	intervalo [int]
m_delay [BOOL]	acceso_permitido [BOOL]
a_partir [float]	

riables

_hay_cambios.- Desde la última vez que el usuario corrió la simulación hasta la vez actual, si usuario alteró en algo el circuito será necesario recalcular el orden prioritario antes de ejecutar simulación. No_hay_cambios es verdadero en caso de no haberse alterado para nada el cuito desde la última vez que se simuló; caso contrario, se coloca en falso.

rro_linea.- En caso de que el usuario decida eliminar algún elemento, esta variable se coloca o verdadera. Esta variable es útil cada vez que se desee pintar el listado de elementos en la

ista-listado. Gracias a ella, la vista sabrá si resaltar, borrar , o volver a escribir los datos de cada emento en la vista.

_actual.- Esta variable contiene el índice del elemento que actualmente está resaltado (el timo elemento que el usuario seleccionó).

_partir.- Esta variable contiene el tiempo inicial a partir del cual se presentarán en la tabla de uestras los valores de las salidas. Esto le permite al usuario especificar el instante y qué porción todas las muestras desea ver.

_doc.- esta variable actúa de bandera. Si m_doc=1 quiere decir que el usuario ha escogido gún elemento y recién lo está creando (osea que previamente no tenía almacenado ningún lor en sus campos de la caja de diálogo). Si m_doc=2 quiere decir que el elemento ya existe y e posee valores previamente grabados (estos se cargarán en los campos de la caja de diálogo) nto antes de presentarla.

ien_esta_en_view.- Esta variable contiene el índice del elemento que actualmente está siendo strado en la vista-gráfico. Es muy útil para la vista del gráfico, ya que le indica que a pesar de e el usuario seleccione otro elemento en el listado, mientras no haga doble click con el botón echo del ratón, el gráfico no debe alterarse.

mpo_listo.- Esta variable se convierte en verdadera cuando el usuario finalmente ingresa a la ja de diálogo de parámetros de Tiempo. Mientras él no ingrese a dicha caja (y por supuesto ne los campos de la misma) esta variable se mantendra en falso.

ntidad_suficiente.- Si el circuito no tiene por lo menos un integrador (o si el usuario decidió rrar todos los integradores del circuito) esta variable se mantendrá en falso.

dos_existen.- Esta variable pasa a ser verdadero bajo la condición de que todas las entradas cada elemento existan.

mpos[51].- Este es un arreglo unidimensional que guarda hasta 51 muestras de tiempo, rrespondientes a la ejecución de la simulación.

sultados[51].- Este arreglo unidimensional guarda hasta 51 valores de salida asociados al emento que aparece en la vista-gráfico (el elemento que el usuario escogió para ver sus idas).

delta.- Es una variable que contiene el tiempo total de simulación dividido para la cantidad de intervalos a tomarse. Es una diferencia de tiempo entre muestra y muestra.

m_abortar.- Si por alguna razón uno de los elementos se desborda en sus cálculos, o si acaso existe un elemento Fin dentro del circuito y se cumplió su condición, m_abortar será verdadera bajo la condición de que la simulación debe detenerse.

m_delay.- Pasa a ser verdadera si el circuito contiene por lo menos un elemento de Unidad de Retardo.

m_order [MAXIMO].- Recordemos que los elementos del circuito deben colocarse en un orden prioritario justo antes de comenzar la simulación. El arreglo m_order contiene los identificadores de todos los elementos del circuito pero ya ordenados prioritariamente.

m_resto.- Contiene un índice del arreglo m_order a partir del cual comienzan los elementos que no son constantes. Al ordenar los elementos, siempre se consideran de máxima prioridad a las constantes. El resto de elementos siguen después. M_resto contiene el índice a partir del cual empiezan los identificadores de los otros elementos que no son tipo constante.

ir.- Recuerde que el método empleado para generar números aleatorios entre -1 y +1 utiliza siempre un factor para multiplicar los valores que se van generando. Ir es dicho factor.

m_Y [MAXIMO+1].- Este arreglo contiene los valores de los vectores de integración en el momento que culmina un intervalo de muestreo.

m_YK [MAXIMO+1].- Habíamos mencionado anteriormente que no solo es necesario muestrear cada intervalo, sino también para cada medio intervalo (muy importante para los integradores). Este arreglo contiene los valores de los vectores de integración luego de haber pasado medio intervalo de muestreo.

m_DYDT [MAXIMO+1].- Para el inicio de un nuevo intervalo de muestreo, los valores asociados de los vectores de integración se almacenan en este arreglo (solo al comienzo de cada intervalo).

m_INTG [MAXIMO+1].- Este arreglo contiene los identificadores de todos aquellos elementos que son integradores en el circuito.

m_NOFG [MAXIMO+1].- Este arreglo contiene los identificadores de todos aquellos elementos que son generadores de funciones dentro del circuito.

m_NEQ.- Esta variable contiene la cantidad de elementos integradores que tiene actualmente el circuito.

m_NFG.- Esta variable contiene la cantidad de elementos generadores de funciones que actualmente tiene el circuito.

acceso_permitido.- Se coloca verdadero siempre y cuando el usuario haya cumplido con las condiciones mínimas para una simulación y haya escogido visualizar alguno de los elementos del circuito (haciendo doble click). A partir de aquel instante en adelante acceso_permitido se mantiene como verdadero.

tiempo_total.- En esta variable se guarda el valor ingresado por el usuario como tiempo total de simulación del sistema.

intervalo.- Aquí se guardan la cantidad de muestras que desea tomar el usuario durante la simulación.

struct m_arreglo[MAXIMO+1]

tipo [int]
numero [int]
entrada1 [int]
entrada2 [int]
entrada3 [int]
parametro1 [float]
parametro2 [float]
parametro3 [float]
intercepto[11] [float]

struct m_header

Signature [DWORD]
indice_nuevo [int]

struct m_simula[MAXIMO+1]

indice [int]
e1 [int]
e2 [int]
e3 [int]
p1 [float]
p2 [float]
p3 [float]
c [float]
mtrx5 [int]

struct

indice.- Contiene el índice de m_arreglo correspondiente a este elemento.

e1.- Contiene el identificador del elemento que sirve como entrada 1 a este elemento.

e2.- Contiene el identificador del elemento que sirve como entrada 2 a este elemento.

e3.- Contiene el identificador del elemento que sirve como entrada 3 a este elemento.

p1.- Contiene el valor de la condición inicial o ganancia asociada a la entrada 1.

p2.- Contiene el valor de la ganancia asociada a la entrada 2.

p3.- Contiene el valor de la ganancia asociada a la entrada 3.

c.- Contiene el valor de la salida del elemento en el instante actual.

mtrx5.- Para integradores y generadores de funciones, contiene el índice de los arreglos m_INTG o m_NOFG donde se guardan los identificadores del elemento, vector integrativo Y o YK o DYDT.

m_simula[MAXIMO+1].- Recordemos que los datos originales para cada elemento que el usuario va ingresando se almacenan en m_arreglo; sin embargo usamos m_simula como una forma abreviada para accesar a los valores más utilizados de cada elemento. Además el índice que sirve para localizar un elemento dentro de m_simula es realmente el identificador del elemento (mas no el índice original de la matriz m_arreglo).

struct

{

Signature.- Para evitar que el usuario abra algún otro archivo que no sea del tipo (csm), cada circuito se graba con un número en hexadecimal único para los (csm). Si el archivo no contiene dicho valor a su inicio, no podemos abrirlo.

indice_nuevo.- Esta variable contiene la cantidad total de elementos que el circuito posee.

m_header.- Siempre es necesario incluir esta estructura al inicio de cada archivo (justo antes de guardar los datos del circuito). Son muy necesarios para reconocer que el archivo es del tipo (csm) y además saber anticipadamente la cantidad de elemento que tendrá (así podremos separar suficiente espacio en memoria para los elementos).

struct

{

tipo.- Esta variable contiene el ASCII correspondiente al tipo del elemento.

numero.- Esta variable contiene el identificador único del elemento (entre 1 y 100).

entrada1.- Esta variable contiene el identificador del elemento que sirve como entrada 1.

entrada2.- Esta variable contiene el identificador del elemento que sirve como entrada 2.

entrada3.- Esta variable contiene el identificador del elemento que sirve como entrada 3.

parametro1.- Esta variable contiene el valor de condición inicial o de ganancia asociada a la entrada 1.

parametro2.- Esta variable contiene el valor de ganancia de la entrada 2.

parametro3.- Esta variable contiene el valor de ganancia de la entrada 3.

intercepto[11].- En este arreglo se almacenan los 11 interceptos necesarios para un generador de funciones.

_arreglo[MAXIMO+1].- Cada vez que el usuario crea o lee de algún archivo un elemento, sus valores respectivos se guardan en memoria dentro de este arreglo. Cualquier modificación a algún elemento se lo hace diréctamente a este arreglo. El índice asociado a cada elemento (dentro del arreglo) sigue el orden de sus identificadores (mas no es exáctamente el mismo número).

3.4 DESCRIPCION DE MODULOS USADOS

Recordemos que para este proyecto por lo general a cada clase se le ha creado un archivo .H y otro .CPP. El trabajo esta realizado de tal forma que cada archivo .H actúa como un módulo que se interconectará con el resto para el buen desarrollo del proyecto. Basta examinar el contenido de cada archivo .H para darnos cuenta de la modularidad del proyecto. Si revisamos el punto 3.3 (*Descripción de Clases Creadas*) encontraremos toda la información referente al *Encapsulamiento* de cada clase y a la independencia de su correspodiente archivo .H. Existen algunos archivos .H que definen a más de una clase. Examinemos todos a continuación:

-***Grafico.H.***- Este archivo solo contiene la especificación de una sola clase la CGrafico.

-***GFrame.H.***- Este archivo solo contiene la especificación de una sola clase la CGFrame.

-***Csmp.H.***- Este archivo en cambio contiene la especificación de 2 clases: CCsmpApp y CAboutDlg.

-***CsmpView.H.***- Este archivo contiene la especificación de una sola clase la CCsmpView.

-***CsmpDoc.H.***- Este archivo en cambio contiene la especificación de 2 clases: CFixedLenRecHint y CCsmpDoc.

-***Tiempo.H.***- Este archivo contiene la especificación de una sola clase la Ctiempo.

- ***MainFrm.H.***- Este archivo contiene la especificación de una sola clase la CMainFrame.
- ***000.H.***- Este archivo contiene la especificación de una sola clase la C000.
- ***010.H.***- Este archivo contiene la especificación de una sola clase la C010.
- ***100.H.***- Este archivo contiene la especificación de una sola clase la C100.
- ***110.H.***- Este archivo contiene la especificación de una sola clase la C110.
- ***120.H.***- Este archivo contiene la especificación de una sola clase la C120.
- ***1211.H.***- Este archivo contiene la especificación de una sola clase la C1211.
- ***200.H.***- Este archivo contiene la especificación de una sola clase la C200.
- ***220.H.***- Este archivo contiene la especificación de una sola clase la C220.
- ***300.H.***- Este archivo contiene la especificación de una sola clase la C300.
- ***330.H*** .- Este archivo contiene la especificación de una sola clase la C330.

CAPITULO 4

PRUEBAS DEL PROGRAMA (EJEMPLO PRACTICO)

4.1 PLANTEAMIENTO DEL PROBLEMA

Para demostrar la eficacia de la versión actual del CSMP, hemos escogido un ejemplo típico conveniente del libro *Introduction to Computer Simulation* por A. Wayne Bennett. Este ejemplo mostraró la facilidad de conversión a partir de un diagrama de bloques (o para aquellos diseñadores que usan directamente las ecuaciones diferenciales) a un circuito de Computador Analógico. A partir de este momento el CSMP se encargará de recibir los datos del circuito de Computador Analógico generado y simularlo para un cierto intervalo de tiempo.

Nuestro ejemplo utiliza típicamente elementos *lineales* y hasta un máximo de *Segundas Derivadas*. Recuerde que esto es apenas un granito de arena para el gran potencial que posee el CSMP, con circuitos muchísimo más complejos. El circuito utilizará elementos inversores, integradores y constantes. Consideraremos una alimentación inicial de +0 (unidades gráficas) con condiciones iniciales de $x'(0) = 0$ y $x(0) = 2$; posteriormente obtendremos respuestas diferentes para la variable x usando una alimentación de +4 y ambas condiciones iniciales a 0. Recuerde que el circuito no posee ninguna unidad de medida específica. Lo interpretaremos en unidades gráficas.

4.2 DEFINICION DE ECUACIONES DIFERENCIALES

Trabajaremos con la típica ecuación de segundo orden (hasta segundas derivadas):

$$x'' + 2x' + 4x = f(t)$$

donde $x'(0) = 0$ y $x(0) = 2$. Consideraremos también que f(t) es una función escalón unitario de amplitud igual a 0.

Si resolvemos para la derivada de mayor orden la ecuación quedaría así:

$$x'' = - [2x' + 4x - f(t)]$$

El siguiente paso es convertir la ecuación diferencial por medio de la Transformada de LaPlace a diagrama de bloques.

4.3 DIAGRAMA DE BLOQUES (LAPLACE)

Usando el operador de LaPlace podemos convertir la ecuación de segundo orden anterior a otra equivalente, que será útil para graficar el diagrama de bloques.

$$L[x''] + L[2x'] + L[4x] = L[f(t)]$$

Reemplazando los términos por las correspondientes transformadas tenemos:

$$\{s^2X(s) - 2s - 0\} + 2\{sX(s) - 2\} + 4X(s) = F(s)$$

Reduciendo términos obtendremos:

$$X(s)[s^2 + 2s + 4] = 2s + 4 + F(s)$$

Despejemos la variable X(s):

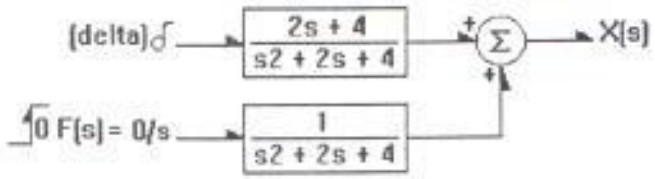
$$X(s) = [2s + 4 + F(s)] / [s^2 + 2s + 4]$$

Separando en 2 términos tenemos:

$$X(s) = [2s + 4] / [s^2 + 2s + 4] + F(s) / [s^2 + 2s + 4]$$

Esta última ecuación podemos representarla como diagrama de bloques:

Para f(t)=0 y x'(0)=0 y x(0)=2.-



Para el segundo set de condiciones iniciales podemos repetir el proceso de integración y obtendremos:

$$L[x''] + L[2x'] + L[4x] = L[f(t)]$$

Reemplazando los términos por las correspondientes transformadas tenemos:

$$\{s^2X(s)\} + 2\{sX(s)\} + 4X(s) = F(s)$$

Reduciendo términos obtendremos:

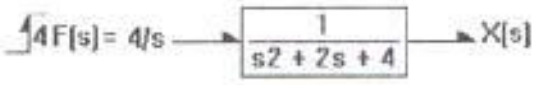
$$X(s)[s^2 + 2s + 4] = F(s)$$

Despejemos la variable X(s):

$$X(s) = F(s) / [s^2 + 2s + 4]$$

Esta última ecuación la podemos representar como diagrama de bloques:

Para f(t)=4 y x'(0)=0 y x(0)=0.-



4.4 CONVERSION A ELEMENTOS DE COMPUTADOR ANALOGICO

Para mejor comprensión desarrollemos paso a paso la integración de la ecuación diferencial. Mostremos nuevamente la ecuación despejada para x":

$$x'' = - [2x' + 4x - f(t)]$$

Si integramos a ambos lados obtendremos:

$$x' = \int [-2x' - 4x + f(t)]dt + x'(0)$$

A su vez si usamos esta nueva expresión y la integramos, obtendremos la variable x:

$$x = \int [x']dt + x(0)$$

Podemos apreciar que este circuito utilizará 2 integradores (las 2 expresiones integrales anteriores) y un elemento constante quien representará f(t). Debido a las limitaciones del diseño del CSMP será necesario incluir un elemento adicional; oportuno para nuestro circuito sería un inversor. El CSMP no permite retroalimentación directa del mismo elemento (esto lo podemos apreciar en la integral para x', donde existe el termino -2x' en la misma expresión). Por eso justificamos la presencia de otro elemento como es el inversor (para eliminar la retroalimentación directa).

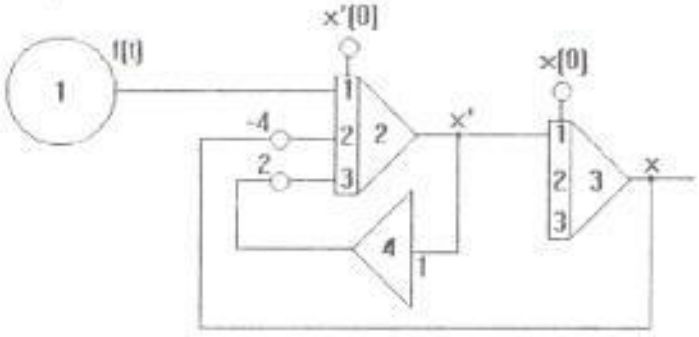


Fig. 1 .- Circuito de Elementos de Computador Analógico

5 DATOS PARA LA SIMULACION

Los únicos datos restantes para empezar la simulación es incluir los parámetros de tiempo. Especificaremos en esta ocasión un tiempo total de simulación de 6 segundos (unidades de tiempo). Para mayor resolución y suavidad de la curva tomaremos el máximo de muestras posibles osea 50. Nos interesa conocer las salidas para x(t) y x'(t).

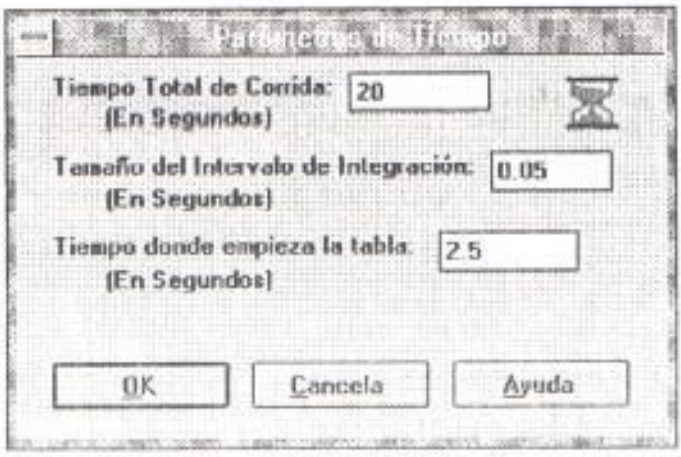


Fig. 2.- Caja de Diálogo para Parámetros de Tiempo

4.6 PRESENTACION DE RESULTADOS

4.6.1 Tabular

Los valores de las salidas los obtendremos tanto del elemento #3 como del elemento #2, para x(t) y x'(t) respectivamente.

Para f(t)=0 y x'(0)=0 y x(0)=2.-

Tabla I.- Salidas vs. Tiempo para x(t) y x'(t)

Para X[t]		Para X'[t]	
TIEMPOS	SALIDAS	TIEMPOS	SALIDAS
0.000	2.000	0.000	0.000
0.120	1.942	0.120	-0.960
0.240	1.776	0.240	-1.634
0.360	1.551	0.360	-2.049
0.480	1.287	0.480	-2.242
0.600	1.013	0.600	-2.257
0.720	0.746	0.720	-2.137
0.840	0.500	0.840	-1.920
0.960	0.285	0.960	-1.644
1.080	0.105	1.080	-1.339
1.200	-0.037	1.200	-1.029
1.320	-0.143	1.320	-0.735
1.440	-0.214	1.440	-0.469
1.560	-0.256	1.560	-0.240
1.680	-0.272	1.680	-0.053
1.800	-0.269	1.800	0.092
1.920	-0.250	1.920	0.196
2.040	-0.222	2.040	0.263
2.160	-0.187	2.160	0.299
2.280	-0.150	2.280	0.308
2.400	-0.113	2.400	0.298
2.520	-0.079	2.520	0.272
2.640	-0.048	2.640	0.236
2.760	-0.022	2.760	0.196
2.880	-0.001	2.880	0.154
3.000	0.015	3.000	0.113
3.120	0.026	3.120	0.075
3.240	0.033	3.240	0.042
3.360	0.037	3.360	0.015
3.480	0.037	3.480	-0.007
3.600	0.035	3.600	-0.023
3.720	0.032	3.720	-0.033
3.840	0.027	3.840	-0.039
3.960	0.022	3.960	-0.042
4.080	0.017	4.080	-0.041
4.200	0.012	4.200	-0.038
4.320	0.008	4.320	-0.034
4.440	0.004	4.440	-0.028
4.560	0.001	4.560	-0.023
4.680	-0.001	4.680	-0.017
4.800	-0.003	4.800	-0.012
4.920	-0.004	4.920	-0.007
5.040	-0.005	5.040	-0.003
5.160	-0.005	5.160	0.000
5.280	-0.005	5.280	0.002
5.400	-0.004	5.400	0.004
5.520	-0.004	5.520	0.005
5.640	-0.003	5.640	0.006
5.760	-0.003	5.760	0.006
5.880	-0.002	5.880	0.005
6.000	-0.001	6.000	0.005

Para $f(t)=4$ y $x'(0)=0$ y $x(0)=0$.-

Tabla II.- Salidas vs. Tiempo para x(t) y x'(t)

Para X[t]		Para X'[t]	
TIEMPOS	SALIDAS	TIEMPOS	SALIDAS
0.000	0.000	0.000	0.000
0.120	0.029	0.120	0.480
0.240	0.111	0.240	0.817
0.360	0.225	0.360	1.024
0.480	0.355	0.480	1.121
0.600	0.494	0.600	1.129
0.720	0.627	0.720	1.068
0.840	0.750	0.840	0.960
0.960	0.858	0.960	0.822
1.080	0.948	1.080	0.669
1.200	1.019	1.200	0.515
1.320	1.071	1.320	0.367
1.440	1.107	1.440	0.234
1.560	1.128	1.560	0.120
1.680	1.136	1.680	0.026
1.800	1.134	1.800	-0.046
1.920	1.125	1.920	-0.098
2.040	1.111	2.040	-0.132
2.160	1.094	2.160	-0.150
2.280	1.075	2.280	-0.154
2.400	1.057	2.400	-0.149
2.520	1.039	2.520	-0.136
2.640	1.024	2.640	-0.118
2.760	1.011	2.760	-0.098
2.880	1.000	2.880	-0.077
3.000	0.992	3.000	-0.056
3.120	0.987	3.120	-0.038
3.240	0.983	3.240	-0.021
3.360	0.982	3.360	-0.007
3.480	0.982	3.480	0.003
3.600	0.982	3.600	0.011
3.720	0.984	3.720	0.017
3.840	0.986	3.840	0.020
3.960	0.989	3.960	0.021
4.080	0.992	4.080	0.021
4.200	0.994	4.200	0.019
4.320	0.996	4.320	0.017
4.440	0.998	4.440	0.014
4.560	1.000	4.560	0.011
4.680	1.001	4.680	0.009
4.800	1.002	4.800	0.006
4.920	1.002	4.920	0.004
5.040	1.002	5.040	0.002
5.160	1.003	5.160	-0.000
5.280	1.002	5.280	-0.001
5.400	1.002	5.400	-0.002
5.520	1.002	5.520	-0.003
5.640	1.002	5.640	-0.003
5.760	1.001	5.760	-0.003
5.880	1.001	5.880	-0.003
6.000	1.001	6.000	-0.002

4.6.2 Gráficos

Las curvas provienen de los elementos #3 y #2, ambos integradores.

Para f(t)=0 y x'(0)=0 y x(0)=2.-

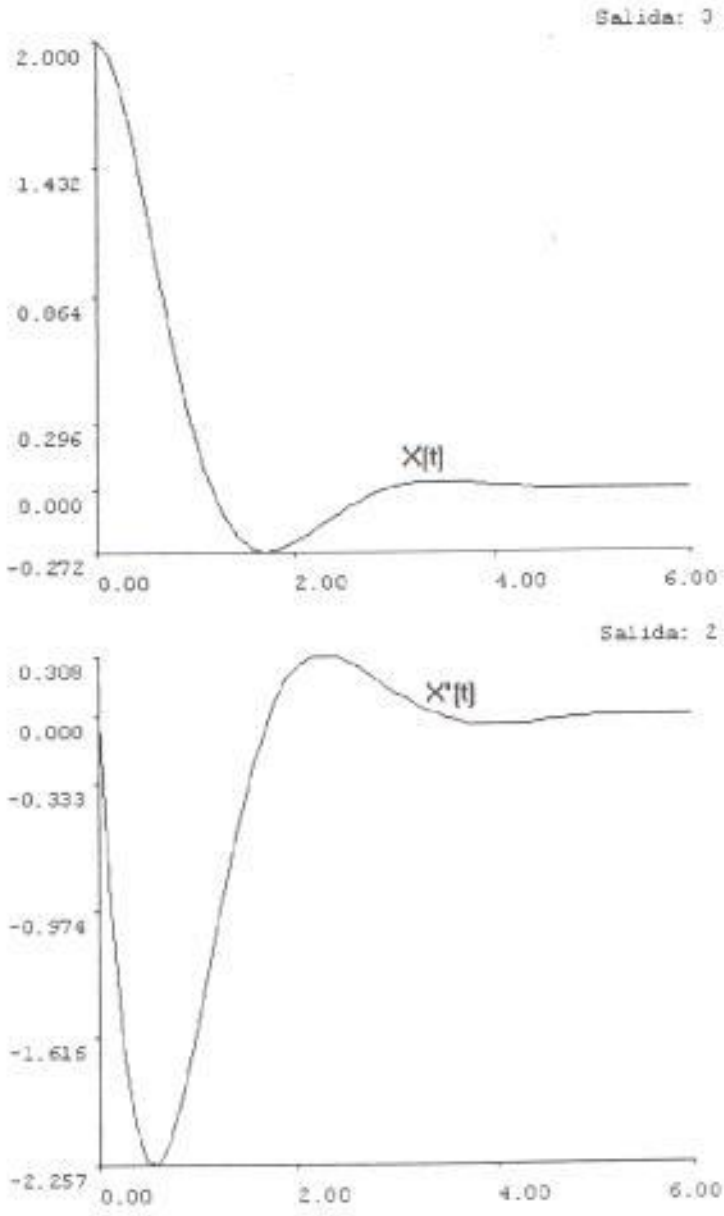


Fig. 3.- Salidas vs. Tiempo para x(t) y x'(t)

Para f(t)=4 y x'(0)=0 y x(0)=0.-

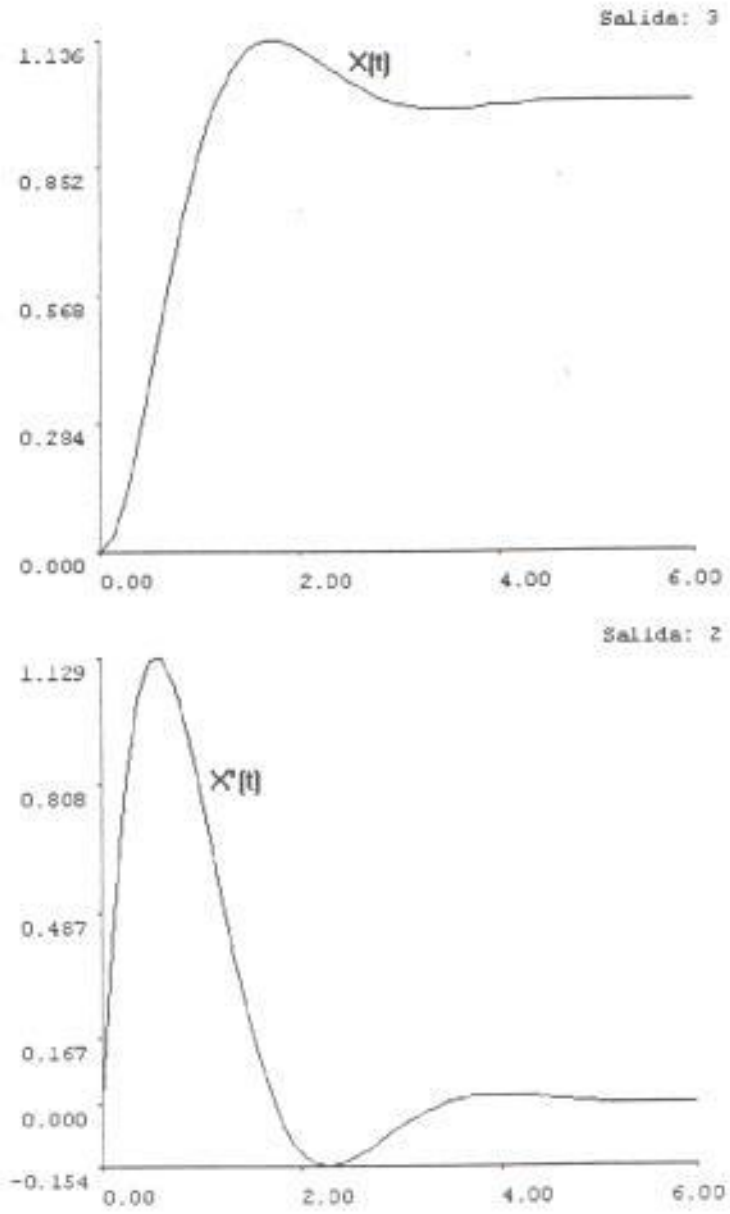


Fig. 4.- Salidas vs. Tiempo para x(t) y x'(t)

CONCLUSIONES

.a simulación de sistemas físicos continuos y dinámicos ha ido evolucionando a lo largo del mpo. En cada ocasión se ha deseado perfeccionar y adaptar las herramientas empleadas, al mo del avance tecnológico. Inicialmente la simulación solo se la lograba a través de un ›mputador Analógico. Posteriormente IBM desarrolló el primer CSMP que corría para el ›mputador 1130. Aquel programa fue totalmente implementado en FORTRAN. En la década : los 80, la actual Ing. Ruth Santana tomó la posta construyendo una versión del mismo CSMP :ro esta vez para PC. Aún se mantenía el mismo lenguaje de programación.

l último escalón lo ha presentado este trabajo. En la década de los 90, esta nueva versión del SMP repara muchas de las fallas que poseían los sistemas de simulación anteriores. Además el royecto actual ha sido implementado con lo último en herramientas de programación. Mientras : mantenga el ambiente Windows, la nueva versión de CSMP no morirá.

El nuevo CSMP ofrece al diseñador actual:

Ambiente gráfico y amigable de trabajo. Corre bajo uno de los ambientes más populares Windows).

Ayuda en línea tanto del manejo del CSMP como de elementos de simulación en Computador Analógico.

Alta resolución en el momento de presentar las curvas en pantalla.

- Menús, íconos, ratón y botones para rápido acceso a la información.
- Circuitos sin limitaciones en las cantidades de cada tipo de elemento (lamentablemente el circuito si posee un tamaño máximo).
- Generar más de una curva de salida para varios elementos del circuito (tanto en pantalla como en impresora).
- Poder alterar los valores y parámetros de los elementos y de la simulación. Así se consigue tantear con diferentes valores hasta que el sistema se estabilice.
- Difusión a un número de usuarios mayor. Esta versión de CSMP fácilmente puede ser usada por el personal académico como el alumnado (inclusive novatos en Control Automático). Lo

nico que se requiere es tener conocimientos de conversión a Elementos de Computador nalógico.

Requiere de pocos recursos (basta con los necesarios para Windows).

Capacidad para expandirse, si alguna vez se requiere aumentar la cantidad de elementos o los pos.

Gracias al presente trabajo se ha dado una renovación a la simulación digital. Ahora existe un uevo enfoque a la par con la tecnologia actual. Con este proyecto logré demostrar la mayor ficacia del Visual C++ sobre el Borland C++. No solo se aprovecharon los recursos (librerías al náximo) sino también el tiempo se acortó. Recomiendo para futuros trabajos el empleo de esta erramienta.

BIBLIOGRAFIA

- ...ESLANDER, TAKAHASHI, RABINES. Introducción a Sistemas y Control. ...Graw-Hill, México, 1976.

- ...ENAHAN, LUTHER, WILKES. Applied Numerical Methods. ... Wiley & Sons Inc., USA, 1969.

- ...YLOR, BALINTFY, BURDICK, CHU. Técnicas de Simulación en Computadoras. ...musa, México, 1980.

- ...SANTANA VILLAMAR, RUTH. Tesis de Graduación de Ingeniería Electrónica. ...spol, Guayaquil-Ecuador, 1980.

- ...SMITH M. JON. Mathematical Modeling and Digital Simulation for Engineers and ...Scientists. Jotn Wiley & Sons Inc., USA, 1977.

- ...TORRES, CZITROM. Métodos para la Solución de Problemas con Computadora ...Digital. Representaciones y Servicios de Ingeniería S.A., México, 1980.

- ...WAYNE BENNETT, A. Introduction to Computer Simulation. ...West Publishing Company, USA, 1974.