1
1 ej.

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

"DISEÑO DE UN SISTEMA DIGITAL PARA EL ESTUDIO DE LINEAS DE TRANSMISION Y CORTO CIRCUITO EN S.E.P."

T E S I S

PARA OBTENER EL TITULO DE

INGENIERO ELECTRICO Y ELECTRONICO

P R E S E N T A :

LORENZO CABALLERO ROSAS

PARA OBTENER EL TITULO DE

DIRECTOR: ING. ARTURO MORALES COLLANTES.

CIUDAD UNIVERSITARIA AGOSTO 1999

DEDICATORIA: **Daniel**

Gracias **a la vida** por todo lo que soy, lo que se y lo que tengo.

A la UNAM :

En especial a la Facultad de Ingeniería por darme la Ciencia y la Tecnología, además de algo invaluable: las experiencias y consejos de mis profesores.

Al Ing. Arturo Morales Collantes:

Por su valiosa contribución en la creación de este proyecto, de una forma totalmente desinteresada; a pesar de sus múltiples ocupaciones.

A mi familia:

Por todo el apoyo recibido en el transcurso de los años.

Amigos y compañeros:

A todas los que estuvieron y están conmigo en todo momento.

Atte. Daniel

DEDICATORIA: **Lorenzo**

Gracias **a Dios** por todo lo que soy, lo que se y lo que tengo.

A **mi padre** por todo su legado. A mi padre por su "pequeña GRAN escalera", que nos ha hecho llegar a donde estamos y ¡hemos de subir más!.

A mi madre que se ha entregado de lleno a su familia. A mi madre formadora y forjadora de hombres y mujeres de bien.

A mis hermanos:

A Guadalupe por su apoyo

A Ignacio por su confianza

A Enriqueta por su ejemplo

A Carmen por su fortaleza

A la UNAM :

En especial a la Facultad de Ingeniería por darme la Ciencia y la Tecnología, además de algo invaluable: las experiencias y consejos de mis profesores.

A mis amigos:

A Roberto por su amistad consolidada
A Miriam por su amistad inquebrantable
A Omar por su amistad desinteresada
A Erika por su amistad sincera
A Sandra por su amistad confidente
A Gina por su amistad efervecente

A la generación 94:

Por todos los momentos que compartimos juntos muy en especial con los Eléctricos - Electrónicos: Hedilberto, Juan Manuel, Sergio, Gerardo, Gabriela, Leopoldo, Enrique, Yakhve, Carlos, Guillermo, Fernando, Skid, Francisco, José. También a mis amigos de otras carreras: Gustavo, Mercedes, Octavio, Ivan, Luis Enrique, Rodolfo entre muchos otros.

Al PTC:

Por todos los amigos y compañeros que he hecho. Por todo el conocimiento que me ha proporcionado. Por todos lo retos que me ha propuesto. Por la responsabilidad que me da el simple hecho de ser un PTCero.

Al personal del Laboratorio de Computadoras y Programación:

Lucía, Lucy, Yoatzin, Minerva, Armando, Guillermo, Daniel y al personal de Servicio Social por el apoyo que me han dado ya que junto hemos llegado a metas más altas en tiempos más breves.

Al Dr. Jesús Savage Carmona:

Por tener fe en mí para hacerme cargo del Laboratorio de Computadoras y Programación y el apoyo que me ha dado en este año y medio como miembro del Departamento de Computación.

Al Ing. Arturo Morales Collantes:

Por su valiosa contribución en la creación de este proyecto, de una forma totalmente desinteresada; a pesar de sus múltiples ocupaciones.

Con todo mi amor y cariño
para mi pequeña gran contadora:
Rosario.
Que la luz de su mirada
y la pureza de su alma permanezcan
conmigo.

Atte. Lorenzo

Lorenzo Caballero Rosas, nació en México D.F., el 12 de marzo de 1975. Realizó sus estudios en el Colegio México, Instituto México y Centro Universitario México; primaria, secundaria y preparatoria respectivamente.

En octubre 1993 ingresa a la Licenciatura en la carrera de Ingeniero Eléctrico Electrónico en la UNAM. De agosto de 1996 a diciembre de 1998 participo como becario del Programa de Tecnología en Computación (PTC). En febrero de 1998 ingresa como ayudante de profesor en la División de Ingeniería Eléctrica (DIE) en la propia Facultad de Ingeniería. Desde octubre de 1998 es el jefe del Laboratorio de Computadoras y Programación.

Sus intereses son el área de Eléctrica en general, la Electrónica de Potencia, las comunicaciones satelitales, las redes de computadoras, el desarrollo de software en ambientes visuales, el paradigma de Orientación a Objetos, entre otros.

Carlos Daniel Ramírez Briseño, nació en México D.F., el 4 de febrero de 1974. Realizó sus estudios en la escuela primaria "Maestros de México", Secundaria Diurna Num. 105, "José Guadalupe Posadas", y en la Escuela Nacional Preparatoria Num. 8, "Miguel E. Schultz" de la UNAM.

En octubre 1992 ingresa a la Licenciatura de Ingeniero Mecánico Electricista, en el área de Eléctrica Electrónica en la facultad de ingeniería de la UNAM.

En agosto de 1998 ingresa como ayudante de profesor en la División de Ingeniería Eléctrica (DIE), Facultad de Ingeniería.

Sus intereses principales son el área de las telecomunicaciones, la Eléctrica, la Electrónica y las redes de computadoras.

Índice

DESARROLLO DE UN SISTEMA DIGITAL
PARA EL ESTUDIO DE
LINEAS DE TRANSMISION Y
CORTO CIRCUITO EN
SISTEMAS ELECTRICOS DE POTENCIA

ÍNDICE

Introducción

La energía eléctrica tiene un lugar primordial en la vida moderna actual, no podemos concebir nuestro mundo sin la electricidad como elemento fundamental para que las industrias, los transportes eléctricos, las comunicaciones vía satélite, ó los diversos aparatos electrodomésticos nos permitan una vida cotidiana más fácil.

El desarrollo de la electrónica de estado sólido ha permitido un avance significativo en diversas áreas tecnológicas, una de las cuales es la energía eléctrica, en sus diferentes formas: generación, transmisión, distribución y consumo. Este desarrollo permitió automatizar los procesos que en tiempos anteriores eran realizados por el ser humano.

En un principio los sistemas eléctricos contaban con una capacidad limitada en donde solo unos cuantos operadores controlaban el sistema; al paso de los años estos sistemas crecieron en una forma espectacular, ocasionado por la interconexión entre sí de varias ciudades y de países enteros, propiciando la búsqueda de nuevas soluciones que brindaran un control automático ó semiautomático más estable para minimizar los errores y omisiones causadas por los diferentes tipos de fallas ó factores externos al sistema, logrando a su vez un sistema más eficiente y productivo.

La estabilidad, integridad y calidad de servicio en el sistema eléctrico son elementos claves para contar con un sistema confiable que permita hacer frente a los diferentes tipos de cargas que cada vez son más exigentes y mayores, propiciado esto por el incremento en la población y en las actividades industriales.

La estabilidad de los sistemas de potencia esta predeterminada por las condiciones de operación de estado estable y estado transitorio. Un sistema esta en una condición de estado estable si todas las cantidades físicas que se miden ó se calculan para describir las condiciones de operación del sistema se consideran constantes para propósitos de análisis, además de que un sistema es estable para una condición particular de operación, si después de que ocurre un disturbio pequeño el sistema regresa a la misma condición particular de estado estable.

Sin embargo, si después de un disturbio grande que puede ser ocasionado por fallas en los sistemas de transmisión, cambios repentinos de carga ó maniobras en líneas, se alcanza una condición de operación significativamente diferente, pero el estado estable es aceptable, se dice que el sistema es transitoriamente estable.

Para mantener la integridad del sistema se necesita tener un sistema de protecciones adecuado en función de la importancia de éste, no sin hacer a un lado el factor económico ya que para un elemento de alto valor económico y/o funcional el sistema de protecciones será de mayor calidad, en cambio para un elemento de valor reducido se tendrán protecciones mínimas que solo garanticen

el funcionamiento para ciertos rangos de operación; de esta forma se pretende seleccionar la protección adecuada para cada elemento del sistema, y por ende se deben de estudiar los diferentes parámetros eléctricos para que garanticen su seguridad.

El suministro de energía eléctrica debe de llevarse acabo con una calidad adecuada, de tal manera que las distintas cargas que se tengan en operación funcionen correctamente.

La calidad en el suministro de energía eléctrica queda definida por 3 factores principales:

1) Continuidad en el servicio, ya que si se presenta una interrupción en el suministro, se ocasiona en la mayoría de los casos pérdidas económicas importantes.

2) Regulación de voltaje, para que los aparatos funcionen correctamente con el voltaje determinado de diseño.

3) Control de frecuencia, para limitar variaciones en el consumo de potencia.

Se debe de mencionar también que partes del sistema eléctrico nacional tienen una prioridad máxima, como son: los hospitales, las industrias de proceso continuo, o algunos inmuebles federales; pero en términos generales las industrias suministradoras buscan dar servicio permanente y confiable a todos sus usuarios por pequeños que estos sean.

En los actuales sistemas eléctricos, las fuentes de generación se encuentran alejadas de los centros de consumo, la producción eléctrica es cara debido a que se necesita transformar diversas formas de energía a energía eléctrica pura, en procesos de baja eficiencia y alto costo, por ende no podemos darnos el lujo de perder potencia en una transmisión deficiente, lo cuál conduce a los sistemas eléctricos a tener líneas de transmisión eficientes que justifiquen la inversión.

El buen diseño de una línea de transmisión se basa en escoger las secciones y la topología adecuada para los conductores, lo cual implica elaborar un análisis exhaustivo ocasionado por el estudio de diversas secciones y topologías existentes, que relacionan una diversidad de parámetros propios de las líneas como su distribución geométrica, capacitancia, inductancia y resistencia, por solo mencionar a los más importantes.

Un punto importante a tratar son las fallas desequilibradas, las cuáles son aquellas que ocurren durante un periodo muy corto de tiempo. El principal elemento de análisis lo constituye la utilización de las componentes simétricas,

las cuales permiten representar un sistema desequilibrado en un sistema equilibrado por medio de componentes vectoriales de secuencia positiva, secuencia negativa y secuencia cero.

Con base en las componentes simétricas se utiliza el método de nodos, el cuál facilita de una manera importante el análisis de la red eléctrica para encontrar los valores de corto circuito.

El estudio de líneas de transmisión y corto circuito, genera un análisis muy tedioso y extenso por la cantidad de operaciones que se deben de realizar en la obtención de los resultados buscados, por eso es conveniente simplificar el trabajo para agilizar el análisis y el diseño de la instalación.

La computadora digital proporciona un gran apoyo para la reducción del trabajo, minimizando el tiempo de procesamiento de forma significativa y eludiendo errores en forma importante.

La Computadora ha sido una herramienta muy importante durante la segunda mitad del siglo XX, para diferentes campos de desarrollo, la ciencia, las comunicaciones, la política, la cultura, el deporte, la sociedad, etc., han sido afectados por su aplicación, ya que proporciona en forma general un ilimitado número de soluciones para los variados problemas que se vayan presentando.

Los sistemas eléctricos no han sido la excepción a estos cambios, y para apoyar a la automatización se ha buscado utilizar la rapidez de procesamiento de la computadora digital en el modelado de sistemas eléctricos de potencia.

El uso de grandes computadoras permite un económico y seguro control del estado del sistema en el momento en que se presentan dificultades, y una alternativa es la de subdividir el sistema en pequeñas redes de trabajo que proporcionen una fácil coordinación con la red central.

En los sistemas eléctricos de potencia una computadora de control tiene las siguientes funciones principales:

a) Predicción de demanda.
b) Seguridad en la generación y transmisión.
c) Controlar los voltajes del sistema y los flujos de potencia dentro de límites específicos.
d) Monitoreo de los cambios relativos ocurridos en la estación de generación.
e) Revisión de la configuración de la red.

Para obtener la información necesaria en el modelado computacional del sistema, se recurre a dos elementos importantes:

1) Manuales técnicos.
2) Libros de texto.

Los primeros proporcionan información sobre la estructura práctica del programa, así como de los variados procesos técnicos que se tengan para afrontar las modificaciones que se presenten durante la corrida y depuración del mismo.

Los libros de texto en cambio indican la información técnica incluida dentro del programa, además de que permiten obtener elementos de comparación para las alternativas que tenga el modelo eléctrico, y así elegir la más conveniente.

En el presente trabajo se desarrollan dos paquetes de computo, uno para el cálculo de parámetros en líneas de transmisión y el otro para calculo de fallas, desarrollados mediante el paquete de programación orientada a objetos C++, el cuál se caracteriza por su potencialidad total.

El poder llevar acabo desarrollo de software, da la posibilidad de prescindir a mediano plazo del software comercial que en estos momentos es casi la única posibilidad que tienen las facultades y escuelas para poder ocupar y aplicar las ventajas de la computadora digital, aparte de evitar un gasto económico se da la oportunidad de que los alumnos apliquen sus conocimientos de una manera práctica y rápida en beneficio de su institución, generando desarrollo y productividad en las diferentes áreas de la ingeniería.

De esta forma al combinar consideraciones teóricas y prácticas, se tendrá una aplicación de gran calidad, que propicie el descubrimiento de una nueva generación de programas para los sistemas eléctricos de potencia.

CAPÍTULO I

Elementos Constitutivos de un Sistema Eléctrico

1.1 Producción de energía eléctrica.

El proceso de generación de energía eléctrica consiste en la conversión de energía mecánica en energía eléctrica. Este proceso de generación de energía mecánica en eléctrica se lleva a cabo en sitios denominados "centrales eléctricas" ó "estaciones generadoras".

Por la fuente de energía que utiliza una central, estas se clasifican de diferentes tipos.

1.1.1 Termoeléctrica de Combustibles fósiles.

Estas centrales utilizan el poder calorífico de combustibles como el carbón ó algunos derivados del petróleo como el combustóleo y el gas natural, para calentar agua y producir el vapor que impulse a las turbinas, para que así de esta forma se lleve a cabo el proceso de conversión de energía mecánica en eléctrica.

1.1.2 Centrales nucleares.

En las centrales nucleares el calor necesario para calentar el agua y producir el vapor que mueva a las turbinas de gas se obtiene de un proceso conocido como fisión nuclear, que se lleva a cabo en un dispositivo denominado reactor nuclear.

De las principales características destacan su capacidad para generar grandes cantidades de energía y baja contaminación.

1.1.3 Termoeléctricas de turbina de gas.

Este tipo de centrales utiliza un proceso de conversión de energía, en donde gases a altas temperaturas y bajas presiones son expandidos en una turbina de potencia, la cuál esta acoplada directamente a un generador eléctrico. Estas se clasifican en Turbojet y del tipo industrial.

La Turbojet, es capaz de generar hasta 70 MW utilizando hasta 4 maquinas jet por separado para proporcionar las velocidades del flujo de gas necesarias.

La del tipo industrial, tiene tiempos de arranque mayores que pueden ir desde 15 hasta 20 min. , y tienen una capacidad de generación de 150 MW.

1.1.4 Centrales de Ciclo Combinado.

Estas centrales son una extensión de las centrales con turbina de gas, aprovechan el calor de los gases de escape de la turbina de gas para producir vapor y mover una turbina, la cual tiene acoplado un generador eléctrico como es el caso de las centrales de combustible fósiles.

1.1.5 Centrales Geotérmicas.

En este tipo de centrales la fuente de energía primaria es el vapor geotérmico extraído del subsuelo. La lava fundida atrapada a unos 5 km. Bajo el suelo libera calor y gases a altas temperaturas que calientan el agua existente en estratos porosos, que posteriormente se extrae mediante pozos profundos. El producto extraído es una mezcla de agua y vapor a alta presión, los cuales son separados por un separador centrífugo. El vapor obtenido se envía a una turbina de vapor del tipo convencional para obtener la energía eléctrica a través del proceso de conversión de energía electromecánica.

1.1.6 Centrales Hidroeléctricas.

Las centrales hidroeléctricas utilizan la energía potencial del agua como fuente primaria de energía. Estas centrales se localizan en sitios tales que exista una diferencia de altura entre la central eléctrica y el suministro de agua. De esta forma la energía potencial del agua se convierte en energía cinética la cuál es utilizada para impulsar el rodete de la turbina y hacerla girar produciendo energía mecánica. Acoplado a la flecha de la turbina se encuentra el generador síncrono que finalmente convierte la energía mecánica en energía eléctrica.

1.1.7 Generador Síncrono.

El generador síncrono es un dispositivo rotacional que transforma energía mecánica en energía eléctrica. La energía mecánica de entrada al generador se hace a través de una flecha que acopla el generador al elemento suministrador de energía mecánica, denominado generalmente turbina.

La turbina es un dispositivo compuesto de paletas que recibe un impulso. Este impulso hace que este se mueva rotacionalmente y por consecuencia mueva la parte rotatoria del generador.

El principio de operación de un generador síncrono se basa en la ley de inducción electromagnética de Faraday. en donde la generación de la fuerza electromotriz (fem.) se produce por un movimiento relativo entre conductores y líneas del flujo magnético.

Las líneas de flujo magnético se producen en la parte rotatoria del elemento conocido como rotor, mediante la excitación de un circuito alimentado por corriente directa, conocido como devanado de campo.

Los conductores encargados de cortar las líneas de flujo magnético producidas por el circuito de campo, se encuentran en la parte fija del generador conocida como estator ó armadura, esta sección se conecta al sistema al cuál se le va a suministrar la energía eléctrica. Estos conductores forman un devanado trifásico donde se genera la fem. de corriente alterna.

El devanado de CD de la estructura del campo se conecta a una fuente externa a través de anillos deslizantes y escobillas, aunque algunas estructuras del campo no tienen escobillas y son alimentadas por rectificadores rotatorios.

1.2 Transformación de energía.

1.2.1 Subestaciones.

Las subestaciones son los componentes del sistema en donde se modifican los parámetros de tensión y corriente, sirven además de punto de interconexión para facilitar la transmisión y distribución de la energía eléctrica.

Se clasifican de acuerdo a la función que desempeñan, por la forma de operar y por el arreglo de los buses.

1.2.1.1 Subestaciones Elevadoras.

En estas subestaciones se eleva la tensión suministrada por los generadores. Aquí las fuentes de energía alimentan el lado de baja tensión de los transformadores de potencia.

1.2.1.2 Subestaciones Reductoras.

Son aquellas en donde se reduce la tensión para subtransmitir a otras subestaciones ó para alimentar redes de distribución; en ellas la fuente de energía alimenta el lado de alta tensión de los transformadores de potencia, encontrándose conectada la carga en el lado de baja tensión.

1.2.1.3 Subestación tipo Intemperie.

Este tipo de subestaciones son construidas para operar expuestas a las condiciones atmosféricas (lluvia, nieve, viento, contaminación ambiental, etc.)

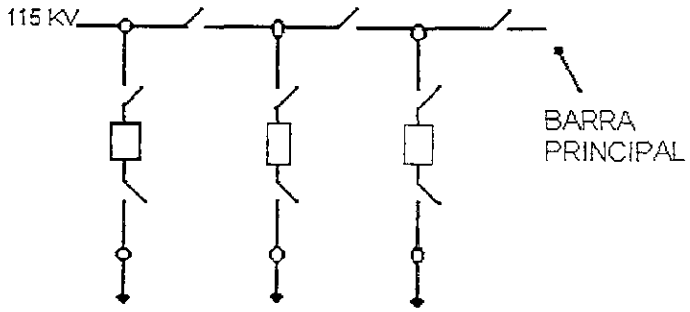
1.2.1.4 Subestación tipo interior.

Son las construidas en el interior de edificios, por su alto costo son utilizadas en lugares densamente poblados donde no hay posibilidad de contar con extensiones grandes de terreno.

1.2.1.5 Subestación de barra sencilla.

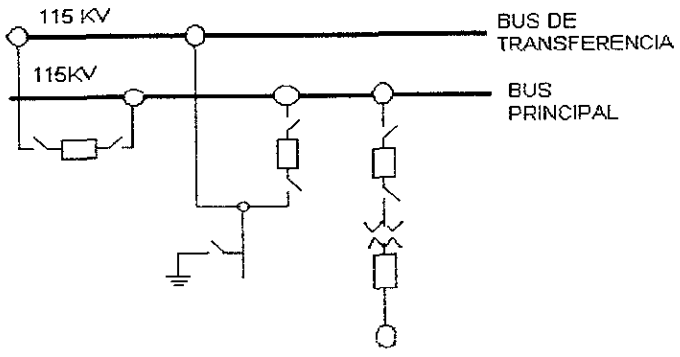
Estas subestaciones tienen solamente una barra para cada tensión, por lo que no ofrecen una gran flexibilidad, debido a que una falla en la barra produce la salida total del sistema.

En este tipo de arreglos se procura que tengan la capacidad de poder ser seccionadas mediante cuchillas. El mantenimiento se dificulta al no poder transferir el equipo.



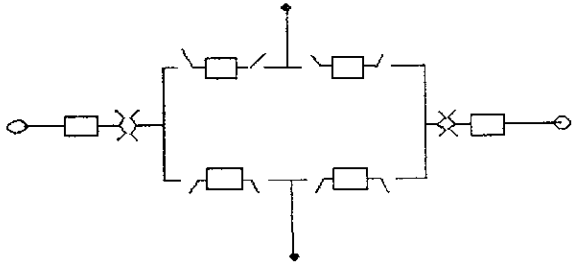
1.2.1.6 Subestación de barra principal y de transferencia.

Son subestaciones cuyo arreglo resulta más flexible ya que cuentan con interruptor de transferencia, lo que para darle mantenimiento no se requiere interrumpir el servicio, aquí la barra principal es la única permanentemente energizada y solo al librar algún interruptor se energiza la barra de transferencia.



1.2.1.7 Subestación de arreglo de barra en anillo.

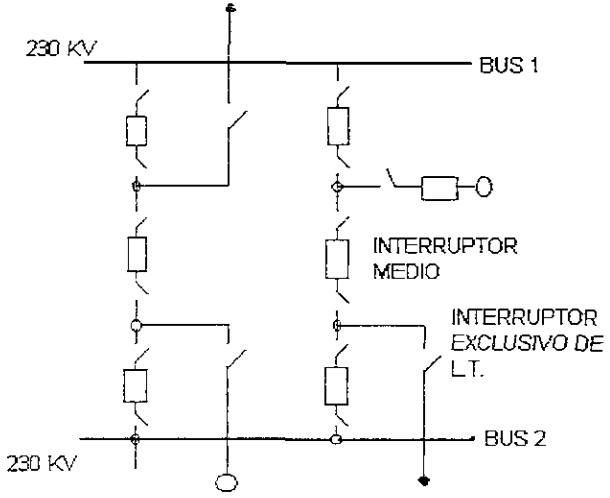
Este arreglo es una variante de la barra sencilla, dándole mayor flexibilidad al alimentarse los circuitos por dos caminos, ofreciendo la posibilidad de dar mantenimiento al equipo sin tener que dejar de proporcionar la energía eléctrica. Una desventaja que ofrece, es que al abrir un anillo por mantenimiento ó falla puede incrementarse la corriente que fluye por el resto de los interruptores conectados.



1.2.1.8 Subestación de arreglo de interruptor y medio.

Este arreglo ofrece facilidad de mantenimiento, flexibilidad y confiabilidad, ya que al perderse una barra no se deja de alimentar la totalidad de la carga ni se pierden las fuentes de energía. Toma su nombre del hecho de compartir un mismo interruptor para dos circuitos diferentes, contando además cada circuito con otro interruptor exclusivo. Estas subestaciones tienen dos barras principales

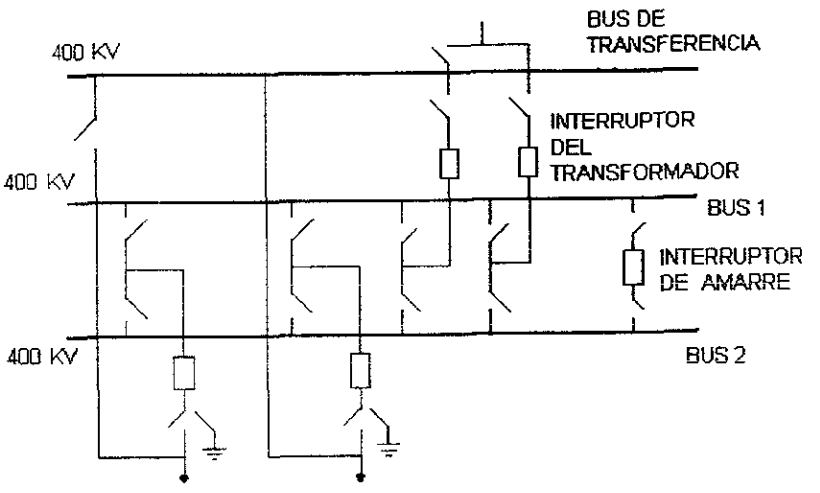
energizadas permanentemente, siendo más complejos los arreglos de protección, control y medición.



1.2.1.9 Arreglo de doble barra con un interruptor y barra de transferencia.

Este arreglo utiliza la flexibilidad de conexión por medio de cuchillas a la barra de preferencia, además de contar generalmente con un interruptor de amarre de barras.

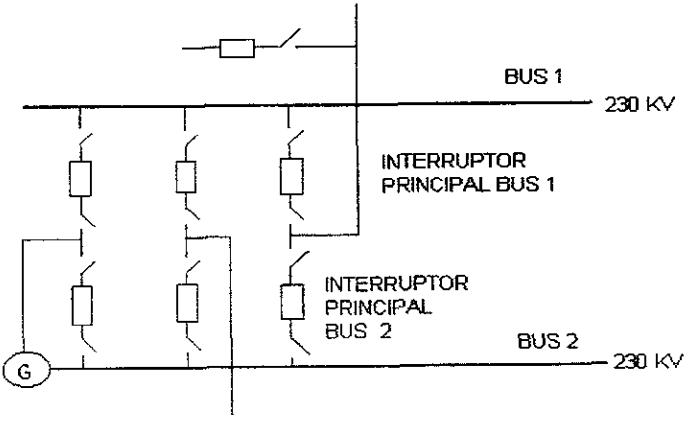
Adicionalmente se cuenta con otra barra para transferir el circuito que requiera de maniobras de mantenimiento.



1.2.1.10 Arreglo de doble barra con doble interruptor.

Este arreglo resulta la mejor opción en cuanto a flexibilidad y confiabilidad se utiliza en aquellos casos en donde la continuidad es muy importante, tanto en condiciones de falla como en mantenimiento.

Cada circuito cuenta con dos interruptores exclusivos permanentemente energizados y conectados a barras distintas. En estos equipos se tiene duplicidad de equipo (interruptores, cuchillas, transformadores de instrumentos, aisladores, barras, etc.)



1.2.2 Componentes de una subestación.

1.2.2.1 Transformadores de potencia.

Los transformadores cambian la tensión y corriente para altos niveles de potencia.

Los transformadores pueden ser trifásicos o monofásicos. En el caso de los monofásicos, se pueden conectar tres de ellos para formar un banco de transformación trifásico. Esta decisión se determina en función del grado de confiabilidad, de flexibilidad y del espacio disponible.

La impedancia en tanto por ciento de los transformadores es por definición, el porcentaje de caída del voltaje en el transformador. Es decir si Z=10 %, la caída de voltaje es del 10 %, *cuando circula la corriente nominal* en el devanado de alimentación.

Se tienen dos conexiones simétricas y balanceadas posibles en transformadores trifásicos: Conexión estrella y conexión delta.

1.2.2.1.1 Conexión Estrella – Estrella.

En este tipo de conexión los devanados de las 3 fases, se conectan a un punto común para formarla, y en este punto llamado neutro se conecta generalmente el sistema de tierras, ya sea directamente o bien a través de una resistencia limitadora de corriente.

1.2.2.1.2 Conexión Delta – Delta.

En este tipo de conexión ambos extremos de los devanados están conectados a la tensión de la línea directamente, lo cuál determina en forma precisa la tensión aplicada y desarrollada en los devanados. No existe una terminal llamada neutro, admitiéndose carga desequilibrada hasta el límite de la capacidad de cada transformador sin inconveniente.

1.2.2.1.3 Conexión Delta - Estrella.

En esta conexión la línea del lado estrella puede ser de 4 hilos, uno de ellos llamado neutro. Las tensiones principales de uno y otro devanado no están en fase. Las tensiones principales en el lado delta son 1.73 veces mayores que en la estrella por tratarse de tensiones entre fases. Las tensiones primarias están bien definidas por conexión directa y las tensiones secundarias (en la estrella) por

inducción. Esta conexión se emplea actualmente para elevar la tensión y en sistemas de distribución.

1.2.2.1.4 Conexión Estrella – Delta.

Sus características son similares a las de la conexión Delta – Estrella, solo que en este caso la estrella está del lado del primario y la delta del secundario. Su aplicación más frecuente está en subestaciones intermedias.

1.2.2.2 Interruptores.

Los interruptores son utilizados para interrumpir el flujo de la corriente y desconectar algún elemento del sistema eléctrico. Pueden interrumpir corrientes de cargas normales ó debidas a fallas eléctricas.

La posición de interruptor abierto ó cerrado se conoce en todo instante gracias a señaladores eléctricos, luminosos, neumáticos ó mecánicos, que están directamente unidos a los elementos de movimiento del interruptor.

Cuando la acción de apertura del interruptor es accionada por la ocurrencia de una falla eléctrica en el sistema eléctrico, la señal de disparo al interruptor es enviada desde el sistema de protección vía relevador de disparo final, que esta conectado a la bobina de disparo del interruptor.

1.2.2.3 Cuchillas Desconectadoras.

Su función es la de desconectar, su operación en alta tensión es sin carga y en ningún caso pueden desconectar corrientes de cortocircuito.

Las dimensiones y características de ellas dependen del circuito y de la subestación en donde serán colocadas En muchas instalaciones modernas su accionamiento se efectúa a distancia a través de motores ó en su caso su accionamiento es manual o individual con ayuda de pértigas

1.2.2.4 Transformadores de instrumentos.

Los dispositivos de protección y medición son accionados por corriente y tensión suministradas por los transformadores de instrumento (potencial y corriente) Estos transformadores proporcionan aislamiento a los equipos de protección y medición, alimentándose con magnitudes proporcionales a aquellas que circulan en el circuito de potencia, pero lo suficientemente reducidas en magnitud para que estos equipos de protección y medición sean fabricados pequeños y no costosos.

1.2.2.5 Apartarrayos.

Son aparatos automáticos conectados entre la fase y tierra, destinados a proteger las instalaciones contra las sobretensiones de origen atmosférico y las producidas por maniobras. Cuando operan conducen a tierra las ondas de sobretensión. Deben de ser instalados en la proximidad de los equipos e instalaciones a proteger.

La función del apartarrayos no es eliminar las ondas de sobretensión sino limitar su magnitud a valores que no sean perjudiciales al aislamiento del equipo. Todos tienen como base dos electrodos que al operar ponen en comunicación la línea con la tierra a través de una resistencia, así la tensión residual viene dada por el producto de la corriente de la onda de descarga por la resistencia.

1.2.2.6 Sistemas de protección.

Es el conjunto de relevadores instalados para monitorear magnitudes eléctricas como corriente, tensión impedancia, frecuencia, temperatura, potencia o las combinaciones entre ellas. En función de su valor envían diversos tipos de señales para disparar y cerrar los interruptores asociados con el equipo del cuál se están monitoreando los parámetros eléctricos.

1.2.2.7 Red de tierra.

La red de tierra en las subestaciones es una de las principales herramientas para la protección contra sobretensiones. A ellas se conectan los neutros de los equipos, las bayonetas, los cables de guarda, las estructuras metálicas y todas aquellas partes metálicas que deben de estar en potencial a tierra.

La red de tierra proporciona un circuito de muy baja impedancia para que circulen las corrientes de tierra, ya sean debidas a una falla de aislamiento ó a la operación de un apartarrayos.

1.2.2.8 Reactores.

Son bobinas que se utilizan para limitar una corriente de corto circuito y poder disminuir de esta forma la capacidad interruptiva de un interruptor. Otra función es la de corrección del factor de potencia en líneas muy largas. cuando circulan corrientes de carga muy bajas, en este caso los reactores se conectan en derivación.

En subestaciones los reactores se utilizan principalmente en el neutro de los bancos de transformadores, para limitar la corriente de corto circuito a tierra.

1.2.2.9 Capacitores.

Son dispositivos eléctricos formados por dos laminas conductoras, separadas por una lámina dieléctrica y que al aplicar una diferencia de tensión almacenan carga eléctrica.

Una de las aplicaciones más importantes del capacitor es la de corregir el factor de potencia en líneas de distribución y en instalaciones industriales, aumentando la capacidad de transmisión de las líneas, el aprovechamiento de la capacidad de los transformadores y la regulación del voltaje en los lugares de consumo.

1.2.3 Calidad en el servicio eléctrico.

La energía eléctrica debe de suministrarse con una calidad adecuada, buscando que los aparatos que la utilizan funcionen adecuadamente. La calidad en el servicio eléctrico queda definida por los siguientes 3 aspectos.

1.2.3.1 Continuidad.

Para mantener la continuidad de una manera adecuada principalmente se ha recurrido a la interconexión de las plantas generadoras de alta tensión, lo que origina operar el sistema de manera que las corrientes que circulan por los elementos no los sobrecarguen provocando colapsos importantes.

Se hace necesario también disponer de un equipo de protección automático que desconecte rápidamente la sección del sistema eléctrico afectado en algún momento por una falla, limitando los daños y manteniendo la sincronía de los generadores.

1.2.3.2 Control de frecuencia.

Una variación en la frecuencia con respecto a su valor nominal refleja un desequilibrio entre la potencia eléctrica total que se está generando y la potencia total que se esta demandando

Para restablecer la frecuencia del sistema a su valor nominal se requiere de un control centralizado, que establezca el error de frecuencia del sistema y actúe sobre las unidades generadoras para anularlo. El control centralizado se

puede dividir en regiones, si al error de frecuencia se le añade el error de intercambio de potencia entre sistemas.

1.2.3.3 Regulación de voltaje.

La regulación se define como la diferencia de tensión en el lado de carga del transformador, entre su valor en vacío y a plena carga, expresada en por ciento de voltaje de plena carga.

$$\% \text{ Regulación} = \frac{\text{V2 en vacio} - \text{V2 plena carga (valor nominal)} \times 100}{\text{V2 plena carga}}$$

Para que el funcionamiento de los distintos aparatos sea el satisfactorio se pretende que el voltaje aplicado no varíe mas allá de ciertos límites; una variación de ± 5% en los puntos de utilización con respecto al voltaje nominal se considera adecuada

1.3 Transporte de energía eléctrica.

Se denomina sistema de transporte de energía eléctrica al sistema responsable de llevar a los centros de consumo la energía eléctrica producida por el sistema de generación.

La capacidad de transmisión de potencia activa esta dada por:

$$P = (V^2 / X) \operatorname{sen}\delta$$

Donde:
P = Capacidad de manejo de potencia
V = Magnitud de voltaje de operación de la línea.
X = Reactancia serie de la línea.
δ = Diferencia angular entre los voltajes de los extremos que conecta la línea

Para cada elemento la caída de tensión esta dada por la ley de ohm como.

$$Vx = jXI$$

Donde: X = Reactancia del elemento.
I = La corriente a través de elemento.

La reactancia del elemento aumenta conforme aumenta su longitud y en consecuencia aumenta la caída de tensión. Para una potencia de transmisión dada, el aumentar la caída de tensión de operación reduce la corriente de transmisión.

1.3.1 Niveles de transporte de energia.

1.3.1.1 Transmisión.

En este nivel se realizan los movimientos de energía desde los centros de generación a las subestaciones importantes ubicadas en los centros de consumo. Se realizan estos movimientos a niveles altos de tensión para minimizar las pérdidas de potencia activa.

1.3.1.2 Subtransmisión.

En este nivel de transporte se manejan niveles de voltaje regulares desde las subestaciones de extra y alta tensión a las estaciones de distribución.

1.3.1.3 Distribución.

En este nivel se realiza la distribución de la energía eléctrica desde las subestaciones de distribución y hacia los consumidores finales.

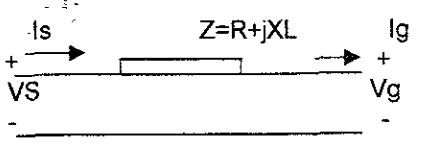
1.3.2 Modelos para líneas de transmisión.

Una línea de transmisión tiene su resistencia, inductancia, capacitancia y conductancia distribuidas uniformemente a lo largo de su longitud, y puede representarse para diversos estudios mediante su equivalente por fase y por unidad de longitud.

1.3.2.1 Líneas Cortas (hasta 80 km.).

Este tipo de líneas se representan como un circuito en el cual se desprecia el arreglo de capacitancia y se considera solo la impedancia.

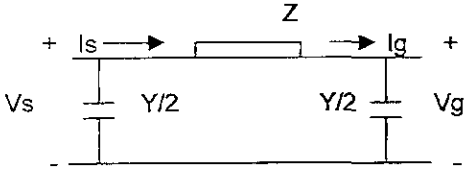
De esta forma el voltaje suministrado será igual al voltaje al neutro en el extremo receptor mas la caída de voltaje debida a la circulación de corriente en la impedancia: $Vs = Vg + I(R+jXL)$



Circuito Equivalente para líneas cortas

1.3.2.2 Líneas medias (hasta 240 Km).

Debido al incremento de la longitud, se deriva una capacitancia que se *incluye en una red de trabajo denominada Π ó T. La resistencia del aislamiento* puede en general considerarse como infinita, especialmente en líneas aéreas.



Circuito equivalente Pi para líneas medias

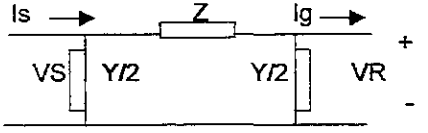
donde: $Vs=Vg+IZ$, $Y\pi=Y/2$, $Z\pi=Z$

1.3.2.3 Líneas largas (más de 240 Km).

En este tipo de líneas se debe de considerar el efecto de los diferentes parámetros distribuidos en la línea.

De esta forma la corriente y el voltaje suministrado están relacionados por las expresiones:

$$VS = VR+IR\,Z\pi + VR\,Y\pi 2$$
$$IS=IR+VR+VRY\,\pi 2+VSY\,\pi 1$$



Circuito Pi de una línea de transmisión larga

1.4 Distribución de energía eléctrica.

Un sistema de distribución eléctrico es el conjunto de elementos encargados de suministrar la energía a partir de una subestación de potencia y hacia el usuario. La función del sistema de distribución es tomar la energía eléctrica y distribuirla a los usuarios en niveles de tensión y seguridad adecuados.

El sistema de distribución debe de proyectarse de modo que pueda ser *ampliado paulatinamente con el fin de asegurar un servicio adecuado y continuo* para la carga existente, buscando el mínimo costo de operación.

Los sistemas de distribución pueden adoptar diversas disposiciones, ya sea en líneas aéreas ó subterráneas y con diversas topologías de red, todo dependiendo de la densidad de carga en un área determinada y del tipo de carga. De acuerdo a su construcción se pueden clasificar en: Sistemas aéreos, sistemas subterráneos y sistemas mixtos.

1.4.1 Sistemas aéreos.

1.4.1.1 Líneas primarias.

Son las encargadas de llevar a la energía desde la subestación hasta los transformadores de distribución.

1.4.1.1.1 Troncal.

Es el tramo de mayor capacidad en el alimentador, aquí se transmite la energía eléctrica desde la subestación a los ramales. Generalmente se utilizan calibres gruesos, dependiendo de la densidad de carga

1.4.1.1.2 Ramal.

Es la parte del alimentador primario energizado a través de un troncal, aquí se conectan los transformadores de distribución y servicios particulares a media tensión.

1.4.1.2 Transformadores de distribución.

Son equipos encargados de cambiar la tensión primaria a un valor menor, de tal manera que el usuario pueda utilizarla sin necesidad de equipos e instalaciones costosas y peligrosas.

La magnitud del transformador se selecciona en función de la magnitud de la carga, teniendo cuidado en considerar los factores que influyen en ella como por ejemplo el factor de manda.

El número de fases del transformador es función del número de fases de alimentación primaria y del número de fases de los elementos que componen la carga.

1.4.1.3 Líneas Secundarias.

Las líneas secundarias distribuyen la energía desde los transformadores de distribución hasta las acometidas en los usuarios.

1.4.1.4 Sistemas secundarios de distribución.

Los sistemas secundarios de distribución se clasifican de acuerdo al número de hilos.

1.4.1.4.1 Monofásico de dos hilos.

Este sistema se alimenta de un transformador monofásico con un secundario de solo dos hilos.

La corriente de carga esta dada por:

$$I=\frac{P}{VCos\phi}$$

1.4.1.4.2 Monofásico de tres hilos.

Este sistema se alimenta de un transformador monofásico con un secundario del que salen tres hilos, con el neutro derivándose del centro del devanado. Para este tipo de circuito la potencia de la carga se equilibra entre los dos hilos de fase y neutro. La tensión en el extremo de la carga es "V" y la resistencia es "R".

La corriente de carga esta dada por:

$$I=\frac{P}{2VCos\phi}$$

1.4.1.4.3 Trifásico de cuatro hilos.

Este sistema se alimenta de un transformador trifásico con un devanado secundario del que salen cuatro hilos, con el hilo neutro derivándose del punto de conexión en los devanados. Es el sistema que permite distribuir la energía con mayor eficiencia.

La corriente de carga esta dada por:

$$I=\frac{P}{3VCos\phi}$$

1.4.1.5 Acometidas y Sistemas de medición.

Esta parte del sistema realiza la unión entre el sistema eléctrico de la empresa suministradora con las instalaciones del usuario.

Las acometidas proporcionan la tensión primaria ó secundaria; esto depende de la magnitud de la carga del cliente.

El sistema de medición se realiza a baja ó alta tensión dependiendo del tipo de acometida.

1.4.1.6 Equipo de Protección contra sobre corrientes y sobretensiones.

Estos elementos garantizan la integridad del sistema protegiéndolo contra corrientes de fallas empleando fusibles e interruptores y contra sobretensiones originadas por descargas atmosféricas, utilizando apartarrayos e hilos de guarda

1.4.2 Sistemas subterráneos.

Los sistemas subterráneos son aquellos que se localizan en el subsuelo de la tierra. En cuanto al tipo de operación existen dos tipos fundamentales de redes de distribución:

1.4.2.1 Radial.

Un sistema radial es aquel que en el flujo de energía tiene una sola trayectoria de la fuente de generación a la carga, lo que origina que una falla en determinado componente de la red interrumpe todos los servicios.

1.4.2.2 Paralelo.

Los sistemas en paralelo cuentan con más de una trayectoria del flujo de energía para alimentar a los consumidores, este tipo de operación es utilizado principalmente en redes subterráneas de baja tensión debido a que es un sistema complejo y costoso

1.5 Utilización de la energía eléctrica:

La carga global de un sistema esta constituida por un gran número de cargas individuales del tipo industrial, comercial y residencial.

La carga absorbe generalmente potencia real y potencia reactiva, por ejemplo en un motor de inducción, la carga es puramente inductiva, en lamparas incandescentes y calentadores eléctricos, etc., las cargas son puramente resistivas y absorben únicamente potencia real.

La potencia suministrada en cada instante por un sistema es la suma de la potencia absorbida por las cargas más las perdidas en el sistema. La conexión y desconexión de las cargas individuales es un fenómeno aleatorio, donde la potencia total varía en función del tiempo siguiendo una curva que se puede predeterminar con bastante aproximación.

La variación de potencia tomada por las cargas que tienen diferentes voltajes es de importancia, si se considera la manera en la cuál estas se representan en estudios de estabilidad.

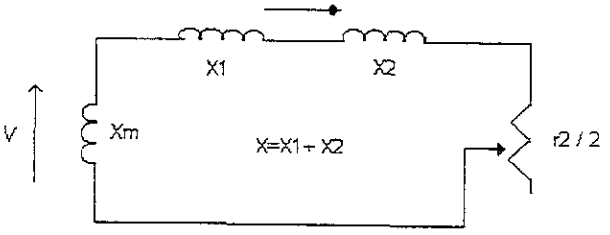
1.5.1 Cargas.

La composición típica de cargas en los sistemas es:

- motores de inducción	50 - 70%
- Lamparas y calentadores	20 - 25%
- motores síncronos	10 %
- perdidas en transmisión	10 – 12 %

1.5.2 Motores de inducción.

Las características van a ser determinadas por el uso de un circuito simplificado. En este circuito se asume que la carga mecánica en el eje es constante.



Circuito equivalente de un motor de inducción

Donde: X1= Reactancia del estator.
X2=Reactancia del rotor referida al estator.
Xm = Reactancia de magnetización.
r2= resistencia del rotor.

La potencia eléctrica esta dada por:

$$Pelec=Pmec=P$$

$$P= 3V^2 r2S / r2^2 + (sX)^2$$

1.5.3 **Lámparas**.

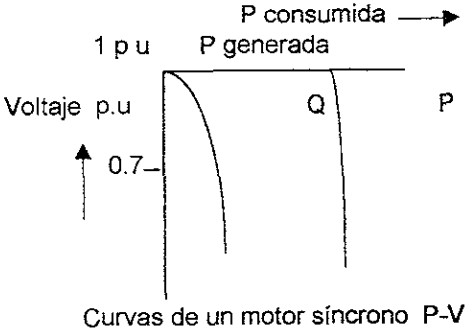
Las lámparas son elementos independientes de la frecuencia y no consumen potencia reactiva. Las lamparas fluorescentes así como las de sodio-mercurio distorsionan la corriente cuando esta fluye por ellas.

1.5.4 Calentadores.

Los calentadores son elementos que mantienen una resistencia constante durante los cambios de voltaje, produciendo variaciones en la potencia que se consume.

1.5.5 Motores Síncronos.

En los motores síncronos la potencia consumida es aproximadamente constante. Para una excitación dada los diversos cambios en la dirección del motor se obtienen con la reducción del voltaje aplicado.



Curvas de un motor síncrono P-V y Q –V.

1.5.6 Características de cargas de frecuencia.

Las cargas de frecuencia se producen debido a que los cambios de voltaje se acompañan siempre por cambios de frecuencia. Los cambios pequeños de frecuencia producen efectos en la potencia y en la potencia reactiva consumida, que usualmente se desprecian en los cálculos.

CAPÍTULO II

Líneas de Transmisión

2.1 Introducción y conceptos básicos.

Una línea de transmisión tiene diferentes propósitos, sirve para transportar energía desde los centros de producción a los centros de consumo, además de proporcionar interconexión para la transferencia de energía en casos de emergencia.

Los sistemas de transmisión presentan valores característicos determinados por su configuración, por su material y por el tamaño de los conductores.

En los sistemas de transmisión se tienen ciertas limitantes para la transmisión de potencia, por eso se busca siempre un balance tecnico-economico que permita una capacidad real en la línea, buscando aumentar en lo posible la transferencia de potencia.

Se tienen 2 parámetros que limitan la transferencia de potencia, la reactancia y la impedancia cuya combinación de ambas forma: la impedancia característica. Estos parámetros se pueden modificar de una forma importante al cambiar el calibre de los conductores ó en otros casos compensar la reactancia

La reactancia de una línea que se encuentra entre la generación y la carga produce limitaciones en la transferencia de energía, en la estabilidad y la caída de tensión. Para disminuir la reactancia se ha recurrido a diferentes técnicas, una de ellas es la de agrupar a los conductores por fase. Para una estructura de doble circuito, al cambiar las fases de un circuito con respecto a otro, la reactancia y la capacidad de transferencia varían, ó también si se adoptan arreglos triangulares en lugar de arreglos verticales la capacidad de transferencia también varia.

Otra opción para disminuir las pérdidas consiste en incrementar la tensión nominal de transmisión sin cambiar el calibre de los conductores.

Una línea de transmisión tiene tres parámetros principales que influyen en su comportamiento de una manera directa, dandole su función como componente de una red eléctrica.

Las tres constantes eléctricas fundamentales que se tienen presente son :

Resistencia eléctrica..... .. ... R, en ohm.
Inductancia.. .... L, en henrys.
Capacitancia..... .....C, en faradios.

2.1.1 Resistencia eléctrica.

La resistencia eléctrica consiste en la oposición al paso de la corriente eléctrica, esta oposición origina la conversión de una parte de la energía eléctrica que circula por las líneas en calor, ocasionando perdida de potencia.

El termino resistencia ó "resistencia efectiva" de un conductor esta dado por la expresión:

$$R = \frac{\text{Perdida de potencia}}{I^2} \quad \Omega$$

donde la perdida de potencia esta dada en watt y la corriente rms que circula por el conductor en amperes.

La resistencia eléctrica de un conductor es directamente proporcional a la resistividad del material con el que está construido (ρ), a la longitud del conductor e inversamente proporcional a su sección la cuál se expresa de la siguiente forma:

$$R = \frac{\rho L}{A} \text{ [ohm]}$$

donde:
- ρ es la resistividad volumétrica del conductor.
- L es la longitud del conductor.
- A es la sección del conductor, en el caso de un cable es la suma de las secciones rectas de los hilos componentes.

La resistencia de un conductor varía en función de la temperatura, lo que origina que en los cálculos industriales se opere habitualmente con el valor de la resistencia que se da en las tablas de conductores.

2.1.2 Inductancia.

La inductancia es la propiedad de un circuito que relaciona la FEM inducida por la variación del flujo con la velocidad de variación de la corriente. Estas fuerzas electromotrices se denominan también de autoinducción.

Se da el nombre de inductancia a la relación entre el flujo creado por la corriente en el circuito y a la intensidad de corriente que circula por el mismo.

Por definición: $\Phi(\text{phi}) = L\, i, \quad L = \Phi / i;$

La inductancia depende de la forma del circuito y de la naturaleza del medio en que esté situado.

La fuerza electromotriz de autoinducción ea viene dada por la expresión:

$$e_a = \frac{-d\Phi}{dt} = -d\,\frac{Li}{dt}$$

en donde si L es constante, tenemos:

$$e_a = -L\, di/dt$$

La expresión anterior sirve para dar la siguiente definición de inductancia: "La inductancia es la relación, con signo cambiado, entre la FEM de autoinducción y la velocidad de variación de la intensidad de corriente".

2.1.3 Capacitancia.

La capacitancia entre los conductores se define como la carga por unidad de diferencia de potencial. Al aplicar una diferencia de potencial entre los extremos de dos conductores separados por un dieléctrico, estos adquieren una carga eléctrica "q" que es proporcional al voltaje "V" aplicado y a una constante de proporcionalidad "C" denominada capacitancia, la cuál depende de la naturaleza del dieléctrico, de las dimensiones de los conductores y de la distancia de separación entre ellos.

$$Q = C\,V$$

El efecto de la capacitancia en líneas puede ser pequeño y muchas veces de desprecia en líneas de potencia que tienen menos de 80 km. de largo, en líneas largas de alta tensión la capacitancia llega a tener gran importancia, especialmente en el factor de potencia, estabilidad y rendimiento de la línea.

2.2 Impedancias de secuencia positiva, negativa y cero

Las impedancias de secuencia positiva y negativa se representan por circuitos trifásicos equilibrados, estas se obtienen de la acción combinada de la resistencia e inductancia existentes en el sistema. La impedancia de secuencia cero se representa por tres fasores de igual módulo y fase

2.2.1 Impedancia inductiva de secuencia positiva, negativa y cero.

Dentro de los conductores se tienen líneas de flujo cambiantes, las cuales proporcionan voltaje inducido dentro del circuito y por lo tanto inductancia. Para obtener un valor aproximado de la inductancia en una línea de transmisión se debe de considerar el flujo dentro y fuera del conductor.

En general es más deseable utilizar la reactancia inductiva que la inductancia. La reactancia inductiva de un conductor en una línea monofásica de dos conductores se representa como:

$$XL = 2\pi fL = 2\pi f \times 2 \times 10^{-7} \ln \frac{Dm}{Ds}$$

Donde:

Dm =distancia entre conductores.

Ds= Radio medio geométrico equivalente.

2.2.2 Radio medio geométrico y distancia media geométrica.

El radio medio geométrico se define como el radio exterior de un conductor tubular de espesor infinitesimal que para una misma corriente, produce el mismo flujo total que el conductor real al cuál sustituye.

La distancia media geométrica desde un punto a un grupo de otros puntos, es la *media geométrica* de las distancias desde un punto a cada uno de los otros puntos. Si hay "n" elementos, la media geométrica de las distancias es la raíz n-esima del producto de las "n" distancias.

2.2.3 Impedancia de secuencia positiva y negativa.

Si se tiene un circuito trifásico con neutro en donde la corriente resultante del desequilibrio regresa por este, el retorno por tierra se hará por una serie de caminos irregulares y de sección variable.

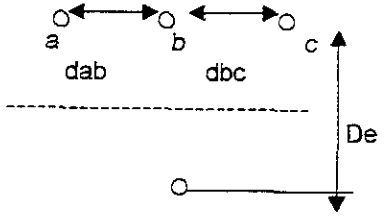
Si se supone que la tierra tiene resistividad uniforme, el circuito real puede sustituirse por un conductor ficticio, colocado bajo tierra a una distancia (*De*) de los conductores de la línea, el cuál es función de la resistividad del terreno y de la *frecuencia de operación*

$$De = 658\sqrt{\frac{\rho}{f}} \ (m)$$

donde:

ρ = resistividad del terreno en Ω/m³

f = frecuencia en ciclos por segundo (Hz).



Para este circuito las impedancias propias de los conductores son expresadas por las siguientes expresiones:

$$Zaa\_n = (Raa+0.000988f+\Delta Raa\_n)+j[0.002892f\log\left(\frac{658\sqrt{\rho/f}}{raa}\right)+\Delta Xaa\_n]\ (\Omega/km)$$

$$Zbb\_n=(Rbb+0.000988f+\Delta Rbb\_n)+j[0.002892f\log\left(\frac{658\sqrt{\rho/f}}{rbb}\right)+\Delta Xbb\_n]\ (\Omega/km)$$

$$Zcc\_n=(Rcc+0.000988f+\Delta Rcc\_n)+j[0.002892f\log\left(\frac{658\sqrt{\rho/f}}{rcc}\right)+\Delta Xcc\_n]\ (\Omega/km)$$

donde:

Raa = Rbb = Rcc = Resistencia en ohms de los conductores.
raa = rbb = rcc = radio medio geométrico de los conductores en m.
f = frecuencia en ciclos por segundo (Hz).
ρ = resistividad del terreno en ohms/m^3.

Los incrementos quedan definidos de la siguiente forma:

$$-\Delta\ Raa\_n = \Delta Xaa\_n = 1.015\ f\left(\frac{ha}{\sqrt{\rho/f}}\right) x\ 10^{-6}\quad \Omega/km$$

$$-\Delta\ Rbb\_n = \Delta Xbb\_n = 1.015\ f\left(\frac{hb}{\sqrt{\rho/f}}\right) x\ 10^{-6}\quad \Omega/km$$

$$-\Delta\ Rcc\_n = \Delta Xcc\_n = 1.015\ f\left(\frac{hc}{\sqrt{\rho/f}}\right) x\ 10^{-6}\quad \Omega/km$$

donde : ha = altura del conductor a sobre el suelo en m.
hb = altura del conductor b sobre el suelo en m.
hc = altura del conductor c sobre el suelo en m.

Las impedancias mutuas entre los conductores incluyendo el efecto del circuito de tierra se definen como:

$$Zab\_n = 0.000988f+\Delta\ Rab\_n+j\ [0.002892f\log\left(\frac{658\sqrt{\rho/f}}{dab}\right)+\Delta Xab\_n]\ \Omega/km$$

$$Zac\_n = 0.000988f + \Delta\, Rac\_n + j\,[0.002892f \log\left(\frac{658 \cdot \sqrt{\rho/f}}{dac}\right) + \Delta Xac\_n]\ \Omega/km$$

$$Zbc\_n = 0.000988f + \Delta\, Rbc\_n + j\,[0.002892f \log\left(\frac{658 \cdot \sqrt{\rho/f}}{dbc}\right) + \Delta Xbc\_n]\ \Omega/km$$

donde:

dab = distancia entre los conductores a y b en m.
dac = distancia entre los conductores a y c en m.
dbc = distancia entre los conductores b y c en m

En las ecuaciones de impedancias propias, la resistencia está constituida por la resistencia del conductor más la resistencia del circuito por tierra, la cuál corresponde al término 0.000988 f + Δ Raa_n.

La resistividad del terreno depende del tipo de terreno y del grado de humedad que este contenga.

El factor De es mucho mayor que la altura de los conductores sobre el piso, de esta forma las reactancias propias y mutuas de cada conductor, incluyendo el efecto del circuito de tierra son independientes de la altura de los conductores.

Por lo tanto, las impedancias de secuencia positiva, negativa y cero están dadas por las siguientes expresiones:

$$Z11=Z22=\left(\frac{1}{3}\right)(Zaa\_n+Zbb\_n+Zcc\_n)-\left(\frac{1}{3}\right)(Zab\text{-}n+Zac\_n+Zbc\_n)$$

$$Z00=\left(\frac{1}{3}\right)(Zaa\_n+Zbb\_n+Zcc\_n)+\left(\frac{2}{3}\right)(Zab\_n+Zac\_n+Zbc\_n$$

$$Z12=\left(\frac{1}{3}\right)(Zaa\_n+Zbb\_n\ a^2+Zcc\_n\ a)+\left(\frac{2}{3}\right)(Zab\_n\ a+\ a^2\ Zac\_n+Zbc\_n)$$

$$Z21=\left(\frac{1}{3}\right)(Zaa\_n+Zbb\_n\ a+Zcc\_n\ a^2)+\left(\frac{2}{3}\right)(Zab\_n\ a^2+a\ Zac\_n+Zbc\_n)$$

$$Z10=Z02=\left(\frac{1}{3}\right)(Zaa\_n+Zbb\_na+Zcc\_n\ a^2)-\left(\frac{1}{3}\right)(Zab\_n\ a+\ a^2\ Zac\_n+Zbc\_n)$$

$$Z20=Z01=\left(\frac{1}{3}\right)(Zaa\_n+Zbb\_n\ a^2+Zcc\_n\ a)-\left(\frac{1}{3}\right)(Zab\_n\ a+Zac\_n\ a^2+Zbc\_n)$$

Sustituyendo las expresiones de las impedancias propias y mutuas en las expresiones anteriores tenemos que la expresión general para las impedancias de secuencia positiva y negativa es:

$$Z11=Z22= R+j0.002892f \log\left(\frac{\sqrt[3]{dabxdacxdbc}}{rg}\right) \ (\Omega/\text{Km.})$$

donde:

R =Raa=Rbb=Rcc = resistencia efectiva de cada conductor.
rg =raa=rbb=rcc = radio medio geométrico de cada conductor.
$\sqrt{dabxdacxdbc}$ =distancia media geométrica entre conductores.

2.2.4 Impedancia de secuencia cero.

Para la impedancia de secuencia cero se tiene la siguiente expresión:

$$Zoo=R+0.002964f+ \Delta Roo+ j\left[0.008676f\log 10\left(\frac{658\sqrt{\frac{\rho}{f}}}{\sqrt[3]{rg(dabxdacxdbc)^{2}}}\right)+\Delta Xoo\right]$$

Cuyas unidades están dadas en (Ω /Km.).

donde los factores de corrección debidos a las alturas de los conductores se definen como:

$$-\Delta R00= \Delta X00\ \ 1.015\, f\left(\frac{(ha+hb+hc)}{\sqrt{\rho/f}}\right)x\,10^{-6} \quad (\Omega/\text{Km})$$

Si se tienen transposiciones entre los tres circuitos de la línea, es decir si los conductores se intercambian de posición a intervalos regulares, de tal forma que cada conductor ocupe la posición de cada uno de los conductores sobre una distancia igual, las impedancias mutuas entre secuencias son iguales a cero. En líneas de transmisión en las que no se tienen transposiciones, las impedancias mutuas de conductores son de valor muy pequeño lo cuál permite despreciarlas.

2.3 Impedancia capacitiva de secuencia positiva, negativa y cero.

La capacitancia de una línea de transmisión es el resultado de la diferencia de potencial entre los conductores y origina que ellos se carguen de la misma forma que las placas de un capacitor cuando hay una diferencia de potencial entre ellas.

La capacitancia afecta tanto a la caída de voltaje a lo largo de la línea, como a la eficiencia, el factor de potencia de la línea y la estabilidad del sistema del cuál la línea forma parte.

2.3.1 Reactancia Capacitiva de secuencia positiva y negativa.

Las reactancias están dadas por las expresiones siguientes:

$$X11=X22=\left(\frac{6.596}{f}\right)\log\left[\frac{\sqrt[3]{dabxdacxdbc}}{r}\times\frac{\sqrt[3]{daa'xdbb'xdcc'}}{\sqrt[3]{dab'xdca'xdbc'}}\right] \quad (M\Omega xKm.)$$

Si la distancia entre los conductores es pequeña comparada con la altura de los conductores sobre el piso, se tiene que la reactancia de secuencia positiva y negativa se represente mediante:

$$X11=X22=\left[\frac{6.596}{f}\right]\log\frac{\sqrt[3]{dabxdacxdbc}}{r} \quad (M\Omega xKm)$$

2.3.2 Reactancia Capacitiva de secuencia cero.

De la misma forma que la reactancia de secuencia positiva y negativa, la expresión para la secuencia cero esta dada por:

$$Xoo=\left(\frac{6.596}{f}\right)\log 10\left(\frac{\sqrt[3]{daa'\,x\,dbb'\,x\,dcc'\,x\,d^2ab'\,x\,d^2ca'\,x\,d^2bc'}}{r\ \sqrt[3]{(dab\,x\,dac\,x\,dbc)^2}}\right) \quad (M\Omega xKm)$$

La expresión anterior puede escribirse también de la siguiente forma:

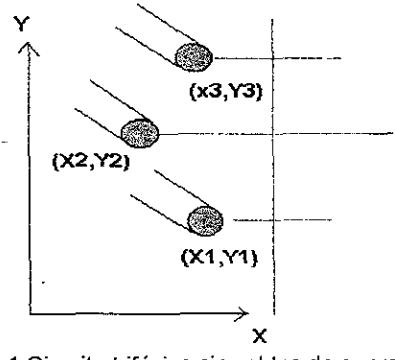
$$Xoo=\left(\frac{6.596}{f}\right)x3\log 10\left(\frac{\sqrt[9]{daa'\,x\,dbb'\,x\,dcc'\,x\,d^2ab'\,x\,d^2ca'\,x\,d^2bc'}}{\sqrt[3]{r\ \sqrt[3]{(dab\,x\,dac\,x\,dbc)^2}}}\right) \quad (M\Omega xKm)$$

El numerador del quebrado del logaritmo indica la distancia media geométrica entre los conductores y sus imágenes, el denominador representa el radio medio geométrico del conductor ficticio, el cuál es equivalente a los tres conductores en paralelo.

2.4 Análisis de los diferentes circuitos.

2.4.1 Un Circuito trifásico sin cables de guarda.

Para obtener la impedancia de secuencia cero de un circuito trifásico sin cables de guarda, esta se obtiene mediante un circuito el cuál puede reducirse a un circuito monofásico equivalente constituido por un solo conductor, el cuál resulta de combinar los conductores de fase.



1 Circuito trifásico sin cables de guarda.

La impedancia inductiva de secuencia cero del circuito está dada por la expresión:

$$Zoop=R+0.002964\,f+\Delta Roop+j\,0.008676\,f\log 10\left(\frac{658\sqrt{\rho/f}}{\sqrt[3]{r\,(DMGabc)^2}}\right)+\Delta Xoop \quad \Omega/Km.$$

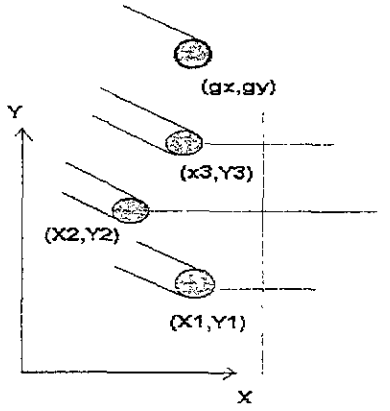
donde:

ΔXoop= - ΔRoop =Terminos de corrección.
R=Resistencia efectiva del conductor.
DMGabc= Distancia media geométrica entre los 3 conductores.
r= Radio medio geométrico de cada conductor.

2.4.2 Un circuito trifásico con 1 cable de guarda.

Para obtener la impedancia de secuencia cero en líneas de transmisión trifásicas con un cable de guarda, tenemos un circuito en el que el conductor puede ser representado por medio de un conductor equivalente a uno ó más circuitos trifásicos en paralelo. Por cada fase circulan corrientes de secuencia

circuitos trifásicos en paralelo. Por cada fase circulan corrientes de secuencia cero iguales y el hilo de guarda se puede representar como un conductor equivalente a uno o más hilos de guarda en paralelo, por donde también circulan corrientes de secuencia cero iguales.



1 circuito trifásico con 1 cable de guarda

En este caso se tendrá un radio medio geométrico equivalente para los tres conductores de:

$$RMG=\sqrt[9]{r^3 \times d^2ab \times d^2ac \times d^2bc} = \sqrt[3]{r(DMGabc)^2}$$

Donde:
r= es el radio medio geométrico de cada conductor.
DMG abc = distancia media geométrica de los 3 conductores.

La resistencia del conductor equivalente será igual a la tercera parte de la resistencia de una fase:

$$Rl= R/3.$$

Y la resistencia equivalente de secuencia cero equivalente será:

$$Rol=3R1=R.$$

En este circuito, la impedancia equivalente de secuencia cero del sistema estará dada por la expresión:

$$Zoo=ZooL-\left(\frac{Zoolg^2}{Zoog}\right)$$

donde:

ZooL = Impedancia propia del conductor.
Zoog = Impedancia propia del cable de guarda.
Zoolg =Impedancia mutua entre el conductor y el cable de guarda.

Despreciando los factores de corrección y basándose en las fórmulas de Carson tenemos que

$$Zool = R+0.002964\ f+j0.008676\ f \log\left[\frac{De}{RMGL}\right]\ (\Omega/Km)$$

$$Zoog = 3\ Rg+0\ 002964\ f+j0.008676\ f \log\left[\frac{De}{RMGg}\right]\ (\Omega/Km)$$

$$Zoolg = 0.002964\ f+j0\ 008676\ f \log\left[\frac{De}{DMGLg}\right]\ (\Omega/Km)$$

donde

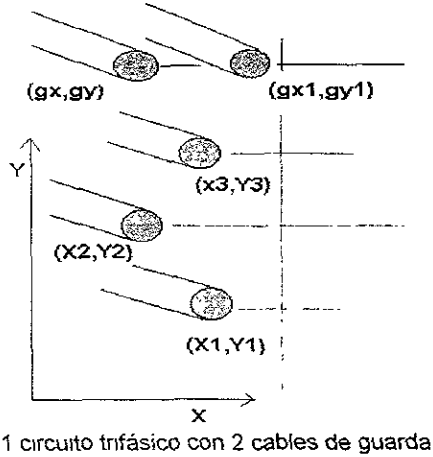
$De = 658\sqrt{\frac{\rho}{f}}$ metros

RMGg = radio medio geométrico del hilo de guarda.
DMGlg = Distancia media geométrica entre los conductores y el hilo de guarda.

$DMGlg = \sqrt[3]{dagxdbgxdcg}$

2.4.3 Un circuito trifásico con 2 cables de guarda.

Para obtener la impedancia de secuencia cero de 1 circuito trifásico con 2 cables de guarda, este sistema se puede representar por un circuito trifásico con dos cables de guarda, el cuál se representa por medio de un conductor equivalente y un solo hilo de guarda.



1 circuito trifásico con 2 cables de guarda

Para éste sistema tenemos que el radio geométrico del conductor equivalente a los tres conductores está dado por:

$$RMGl = \sqrt[3]{r(DMGabc)^2}$$

donde:

r=radio medio geométrico de cada conductor.

DMGabc= distancia media geométrica entre los tres conductores.

$$\sqrt[3]{dabxdacxdbc}$$

Se tiene un radio medio geométrico para el cable equivalente a los hilos de guarda en paralelo, el cuál queda definido como:

$$RMGg = \sqrt{rg\,dgg'}$$

donde:

rg = radio medio geométrico de cada cable de guarda.

dgg´ = distancia entre los cables de guarda.

La impedancia de secuencia cero estará dada por la expresión:

$$Zoo = Zool - \left(\frac{Zoolg^2}{Zoog}\right)$$

En donde las expresiones que forman la impedancia de secuencia cero son:

$$Zool = Rc + 0.002964\ f + j0.008676\ f \log\left[\frac{De}{RMGL}\right]\ (\Omega/Km.)$$

$$Zoog = \left(\frac{3rg}{2}\right) + 0.002964\ f + j0.008676\ f \log\left[\frac{De}{RMGg}\right]\ (\Omega/Km.)$$

$$Zoolg = 0.002964\ f + j0.008676\ f \log\left[\frac{De}{DMGLg}\right]\ (\Omega/Km.)$$

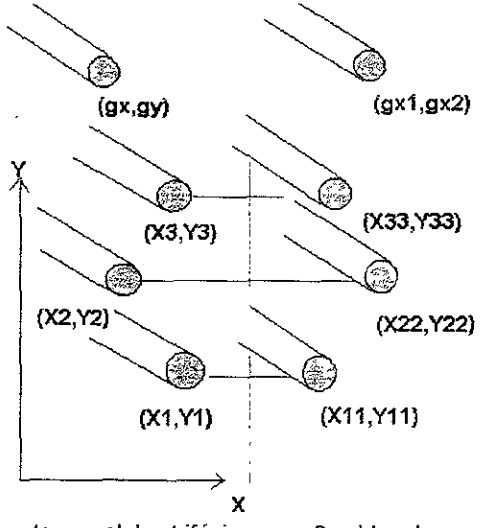
Donde:

DMGlg = distancia media geométrica entre los conductores y los hilos de guarda.

$$DMGlg = \sqrt[6]{dag \times dag' \times dbg \times dbg' \times dcg \times dcg'}$$

2.4.4 Dos circuitos paralelos trifásicos con 2 cables de guarda.

Para obtener la impedancia de secuencia cero en una línea de transmisión con dos circuitos trifásicos con 2 cables de guarda, tenemos que para este tipo de circuito, los seis conductores del sistema se van a sustituir por un conductor ficticio, y un solo cable de guarda.



2 circuitos paralelos trifásicos con 2 cables de guarda

El radio medio geométrico de los conductores es:

$$RMGL = \sqrt[36]{r^6 \times d^2ab \times d^2ac \times d^2aa' \times d^2ab' \times d^2ac' \times d^2bc \times d^2ba' \times d^2bb'}$$

$$\overline{\times d^2 bc' \times d^2 ca' \times d^2 cb' \times d^2 cc' \times d^2 a'b' \times d^2 a'c' \times d^2 b'c'.}$$

Los dos conductores de guarda se sustituyen por un cable ficticio cuyo radio medio geométrico es:

$$RMGg = \sqrt{rg\, dgg'}$$

La distancia media geométrica entre los conductores y el cable de guarda se obtiene mediante:

$$DMG\,lg = \sqrt[12]{dag \times dag' \times dbg \times dbg' \times dcg \times dcg' \times da'g \times da'g' \times db'g \times db'g' \times dc'g \times dc'g'.}$$

De igual manera que en los casos anteriores, la impedancia de secuencia cero estará dada por la expresión:

$$Zoo=Zool-\left(\frac{Zoolg^2}{Zoog}\right)$$

donde:

$$Zool=Rc+0.002964\ f+j0.008676\ f\log\left[\frac{De}{RMGL}\right]\ (\Omega/Km.)$$

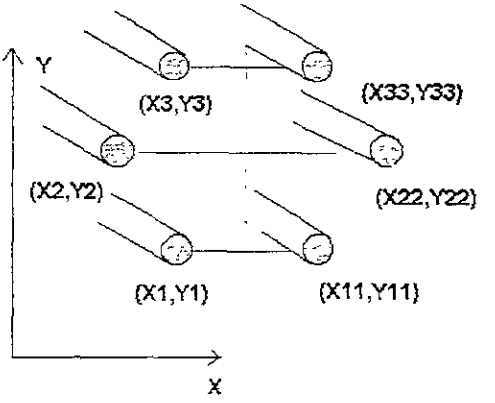
$$Zoog=\left[\frac{3Rg}{2}\right]+0.002964\ f+j0.008676\ f\log\left[\frac{De}{RMGg}\right]\ (\Omega/Km)$$

$$Zoolg=0.002964\ f+j0.008676\ f\log\left[\frac{De}{DMGLg}\right]\ (\Omega/Km)$$

En este caso Rc y Rg son las resistencias del conductor y del cable de guarda respectivamente.

2.4.5 *Dos circuitos paralelos trifásicos sin cables de guarda.*

La impedancia de secuencia cero de dos circuitos trifásicos en paralelo sin cables de guarda, se obtiene considerando que los dos circuitos paralelos se sustituyen por dos sistemas monofásicos de un solo hilo cada uno con regreso común por tierra.



2 circuitos paralelos trifásicos sin cables de guarda

La impedancia de secuencia cero se obtiene tomando en cuenta la *inducción mutua entre circuitos.*

La impedancia mutua de secuencia cero entre los dos circuitos se calcula como:

$$Zoom = 3Zm\text{-}n.$$

donde:
Zoom= Impedancia mutua de secuencia cero entre los dos circuitos.

de esta manera:

$$Zm\text{-}n = 0.00098\ f + \Delta Rm\text{-}n + j[0\ 002892\ f \log \frac{\left(658\sqrt{\frac{\rho}{f}}\right)}{(DMG)_{I-II}} + \Delta Xm\text{-}n] \quad (\Omega/Km)$$

La distancia media geométrica entre los dos circuitos trifásicos I y II esta dada por:

$$d\ I\text{-}II = \sqrt[18]{d^2aa' + d^2ab' + d^2ac' + d^2ba' + d^2bb' + d^2bc' + d^2ca' + d^2cb' + d^2cc'}$$

El factor de corrección para la resistencia y reactancia mutua es el siguiente:

$$-\Delta Rm\text{-}n = \Delta Xm\text{-}n = 1.015\ f \left(\frac{h_I + h_{II}}{2\sqrt{\frac{\rho}{f}}}\right) x10^{-6} \quad (\Omega/Km).$$

donde:
Las distancias hI y hII entre los conductores I y II son .

$$hI = \frac{ha + hb + hc}{3} \quad Y \quad hII = \frac{ha' + hb' + hc'}{3}$$

Basándose en los resultados anteriores tenemos que la impedancia mutua está dada por la siguiente expresión.

$$Zoom = 0.002964\ f + \Delta Room + j\ [0.008676\ f \log \frac{\left(658\sqrt{\frac{\rho}{f}}\right)}{d_{I-II}} + \Delta\ Xoom] \quad (\Omega/Km)$$

Donde los factores Δ Room y Δ Xoom, están relacionados por la expresión:

$$-\Delta Room = \Delta Xoom = 1.015\ f \left(\frac{3(h_I + h_{II})}{2\sqrt{\frac{\rho}{f}}}\right) x10^{-6}$$

Por lo tanto la impedancia de secuencia cero de los dos circuitos en paralelo esta dada por

CAPÍTULO III

Análisis de Corto Circuito

3.1 Conceptos básicos.

3.1.1 Potencia de corto circuito:

La potencia de corto circuito se define como el producto de la magnitud de la corriente de corto circuito (Icc) por la magnitud del voltaje entre líneas nominal del sistema (V) y por raíz de tres.

Para un corto circuito trifásico, donde la corriente tiene una magnitud de Icc3ϕ, la potencia de corto circuito trifásico es por definición:

$$Scc3\phi = \sqrt{3}\ V\ Icc3\phi$$

Para un corto circuito monofásico, con una corriente de magnitud Iccϕ, la potencia de corto circuito monofásico es:

$$Scc\ \phi = \sqrt{3}\ V\ Icc\phi$$

3.2 Descripción de fallas.

Una falla consiste en la conexión intencionada o no de dos ó más conductores que ordinariamente operan con una diferencia de potencial.

Esta conexión se produce por contacto físico entre elementos mecánicos o por medio de un arco. En el caso de contacto físico, la tensión se reduce a cero en el punto de contacto, o a un valor muy bajo en el segundo.

Las corrientes de corto circuito suelen ser muy superiores a las admisibles en conductores y máquinas debido a consideraciones de capacidad térmica. Las temperaturas elevadas que resultan pueden ocasionar daños en los conductores, recociendo el metal y carbonizando el aislamiento. Durante el tiempo en que dura la falla, la tensión en el sistema y en su cercanía es tan pequeña que hace imposible la operación de los equipos.

3.2.1 Fallas temporales.

Las fallas temporales son aquellas que son eliminadas en un periodo corto de tiempo al desenergizar el equipo. Durante el tiempo de contacto se puede formar un arco que permanece mientras la línea ésta energizada, pero si ésta se desenergiza rápido por la acción de dispositivos automáticos los daños físicos pueden no ser de consideración, permitiendo poner en servicio la línea en un breve espacio de tiempo.

3.2.2 Fallas Permanentes.

Las fallas permanentes son aquellas en donde una avería en el aislamiento ó en la estructura causa daños importantes, de tal manera que hace imposible la operación del equipo en forma rápida y obliga a efectuar reparaciones costosas.

3.2.3 Fallas en Líneas.

Las fallas en líneas de transmisión se producen generalmente al unirse estas por la acción del viento, de un árbol, de una grúa, ó por defecto o fallo en las estructuras que las soportan. Las fallas también se presentan por sobretensiones ocasionadas por rayos y por la interrupción de una corriente muy fuerte; estás pueden causar descargas en arco sobre los aisladores, entre el conductor y el soporte, ó entre conductores. La suciedad depositada sobre los aisladores puede *también ocasionar descargas sobre ellos, aún a tensiones normales.*

3.2.4 Fallas en transformadores.

Las fallas en los transformadores son el resultado del deterioro del aislamiento, combinado con las sobretensiones debidas a fenómenos transitorios como los rayos u otras causas.

Las sobretensiones aplicadas pueden ocasionar cortocircuitos entre espiras de una bobina si tienen estas un aislamiento defectuoso. El aislamiento principal también puede fallar, permitiendo que se produzcan arcos entre el primario y el secundario, o entre los embobinados y partes conectadas a tierra tales como los núcleos o la carcaza.

3.2.5 Fallas en generadores.

Los generadores pueden fallar por ruptura del aislamiento entre espiras adyacentes dentro de una ranura, lo que conduce a un cortocircuito en una sola espira. También puede ocurrir una falla si se produce la rotura del aislamiento entre bobinas y la estructura de acero en que están situadas. La rotura del aislamiento entre bobinas diferentes alojadas en la misma ranura ocasionan la puesta en cortocircuito de varias secciones de la maquina.

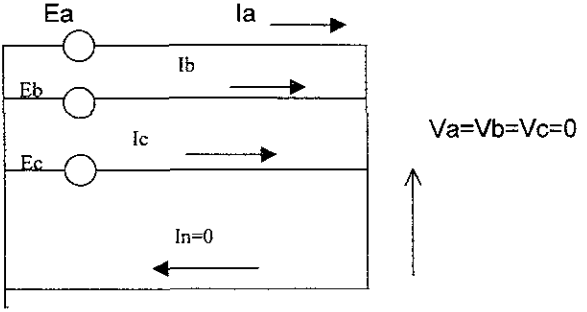
3.3 Tipos de fallas.

3.3.1 Fallas Simétricas.

En el caso de un corto circuito trifásico las corrientes de falla de cada fase son de la misma magnitud y constituyen un sistema de corrientes trifásico equilibrado. Por lo tanto el análisis se realiza considerando únicamente el circuito de secuencia positiva.

3.3.1.1 Falla Trifásica.

En este tipo de falla no se tiene ningún desequilibrio, ya que el efecto se produce en las líneas de una forma simultánea.



Falla Trifásica

Los voltajes en las tres fases son iguales a cero, y en el punto de falla tenemos que:

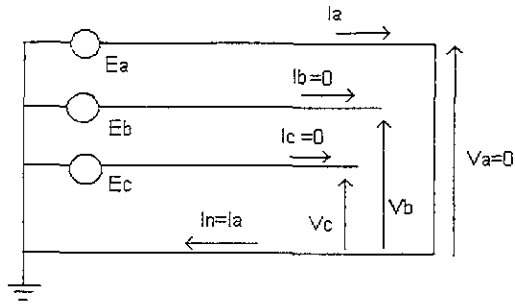
$$Va=Vb=Vc=0, \quad Ia+Ib+Ic=0.$$

3.3.2 Fallas Asimétricas.

Las fallas asimétricas son las que más se presentan en los sistemas de potencia, estas ocurren a través de una impedancia ó por causa de conductores abiertos.

3.3.2.1 Falla monofásica de línea a tierra.

Para una falla monofásica a tierra desde la línea "a", los segmentos de las 3 líneas se conectan como se muestra:



Falla monofásica de línea a tierra

Debido a la falla, no fluyen corrientes en la fase b y en la fase c, por lo tanto son iguales a cero

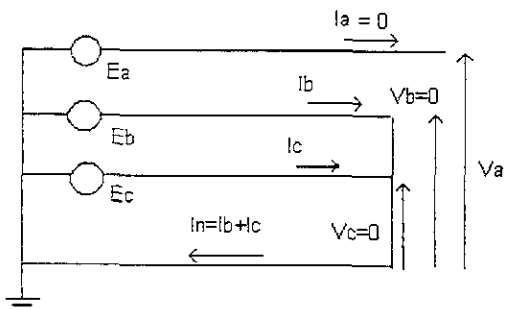
La corriente se concentra en la fase a, en donde tiende a infinito y ocasiona que el voltaje se reduzca a cero.

Las expresiones en las líneas son las siguientes;

Fase a: $Va = Zala = Za\ (\infty) = 0$
Fase b: $Ib = 0$, $Vb \neq 0$.
Fase c: $Ic = 0$, $Vc \neq 0$.

3.3.2.2 Falla doble de línea a tierra (Bifásica a tierra).

Para este tipo de falla se tiene la siguiente conexión de las líneas:



Falla doble de línea a tierra

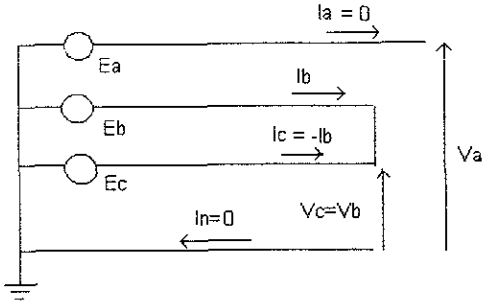
En este caso no fluye ninguna corriente en la fase a por lo que es igual a cero. La falla origina que en la línea b y c se concentre la totalidad de la corriente, ocasionando que ambas corrientes crezcan a infinito y disminuyan a cero los voltajes.

Las expresiones en las líneas son las siguientes:

Línea b: $Ib \neq 0$, $Vb = 0$.
Línea c: $Ic \neq 0$, $Vc = 0$.
Línea a: $Ia = 0$, $Va \neq 0$.

3.3.2.3 Falla entre dos fases (Bifásica).

Para representar esta falla se conectan las líneas de acuerdo al diagrama siguiente:



Falla entre dos fases

En esta falla no se tiene flujo de corriente en la fase a, la corriente que se presenta está entre las fases b y c, siendo esta de igual magnitud, misma dirección y de sentido contrario.

Las expresiones para las líneas son:

Línea b y c: $Ib = -Ic$, $Vb = Vc \neq 0$.
Línea a: $Ia=0$, $Va \neq 0$.

3.4 Impedancia de secuencia positiva, negativa y cero.

Un circuito trifásico desbalanceado puede reducirse para su estudio en un circuito equivalente, en donde las caídas de voltaje en cada fase se expresan mediante un sistema de ecuaciones.

Las caídas de voltaje se obtienen sobre la base de las corrientes de fase, así como de las impedancias mutuas e impedancia propias de cada línea.

$$\begin{bmatrix} Ea \\ Eb \\ Ec \end{bmatrix} - \begin{bmatrix} Va \\ Vb \\ Vc \end{bmatrix} = \begin{bmatrix} Zaa & Zab & Zac \\ Zba & Zbb & Zbc \\ Zca & Zcb & Zcc \end{bmatrix} \begin{bmatrix} Ia \\ Ib \\ Ic \end{bmatrix}$$

donde :

Ea,Eb y Ec : Son las fuerzas electromotrices aplicadas a las fases.
Va,Vb y Vc: Son las caídas de voltaje en cada fase.
Ia,Ib e Ic: Corrientes en cada fase.
Zaa, Zbb, Zcc: Impedancias propias de cada fase.
Zba,Zca,Zcb, etc.: Impedancias mutuas entre fases.

Si expresamos los voltajes y corrientes a partir de sus componentes simétricas, se obtiene la matriz de impedancias de secuencia positiva, negativa y cero, definida como:

$$\begin{bmatrix} Ea1 \\ Ea2 \\ Ea0 \end{bmatrix} - \begin{bmatrix} Va1 \\ Va2 \\ Va0 \end{bmatrix} = \begin{bmatrix} Z11 & Z12 & Z10 \\ Z21 & Z22 & Z20 \\ Z01 & Z02 & Z00 \end{bmatrix} \begin{bmatrix} Ia1 \\ Ia2 \\ Ia0 \end{bmatrix}$$

Donde:

Z11, Z22 y Z00, son los elementos en la diagonal principal y se denominan respectivamente impedancias propias de secuencia positiva, negativa y cero

Los elementos fuera de la diagonal se denominan impedancias mutuas.

Por lo tanto las caídas de voltaje de cada secuencia son función de las corrientes de secuencia positiva, negativa y cero.

3.4.1 Obtención de las impedancias propias y mutuas de secuencia positiva, negativa y cero.

A partir de la matriz de impedancias y considerando a la matriz de componentes simétricas, tenemos que:

$$\begin{bmatrix} Z11 & Z12 & Z10 \\ Z21 & Z22 & Z20 \\ Z01 & Z02 & Z00 \end{bmatrix} = \frac{1}{3} \begin{bmatrix} 1 & a & a^2 \\ 1 & a^2 & a \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} Zaa & Zab & Zac \\ Zba & Zbb & Zbc \\ Zca & Zcb & Zcc \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ a^2 & a & 1 \\ a & a^2 & 1 \end{bmatrix}$$

Resolviendo la matriz de impedancias tenemos que las impedancias propias se representan mediante las expresiones:

Z11=(1/3)(Zaa+Zbb+Zcc)-(1/3)(Zab+Zac+Zbc)
Z22=(1/3)(Zaa+Zbb+Zcc)-(1/3)(Zab+Zac+Zcb)
Z00=(1/3)(Zaa+Zbb+Zcc)+(2/3)(Zab+Zac+Zbc)

Las impedancias mutuas se representan mediante las expresiones:

Z12=(1/3)(Zaa+Zbba²+Zcca)+(2/3)(Zaba+Zaca²+Zbc)
Z10=(1/3)(Zaa+Zbba+zcc a²)-(1/3)(Zaba²+Zaca+Zbc)
Z21=(1/3)(Zaa+Zbba+zcc a²)+(2/3)(Zaba²+Zaca+Zbc)
Z20=(1/3)(Zaa+Zbba²+Zcca)-(1/3)(Zaba+Zaca²+Zbc)
Z01=(1/3)(Zaa+Zbba²+Zcca)-(1/3)(Zaba+Zaca²+Zbc)
Z02=(1/3)(Zaa+Zbba+Zcca²)-(1/3)(Zaba²+Zaca+Zbc)

De las expresiones anteriores se determina que las impedancias mutuas entre secuencias positiva, negativa y cero no son iguales.

En el caso de sistemas trifásicos simétricos solo se tienen impedancias propias, debido a que los conductores se encuentran espaciados simétricamente, lo que origina que las impedancias mutuas se reduzcan a cero. De esta forma las corrientes de cada secuencia producen únicamente caídas de voltaje de la misma secuencia

3.5 Componentes simétricas.

El método de las componentes simétricas es útil para determinar las corrientes y voltajes en todas las partes del sistema después de que ha ocurrido una falla. Cualquier falla asimétrica dará origen al flujo de corrientes desbalanceadas en el sistema.

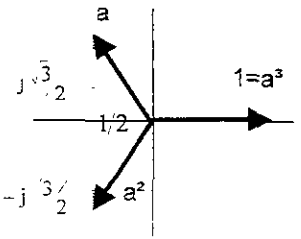
Un sistema desequilibrado de n vectores relacionados entre sí, se puede descomponer en n sistemas de vectores equilibrados, denominados "Componentes Simétricos" de los vectores originales; un sistema directo de secuencia positiva, un sistema inverso de secuencia negativa, y un sistema de secuencia cero.

3.5.1 Operador "a".

El operador a se define como un número complejo de módulo unidad y de argumento $2\pi/3$.

$$a = 1\angle\frac{2\pi}{3} = 1\angle 120°$$

Al multiplicar un fasor por el operador a, se obtiene un nuevo fasor de igual módulo que el primero y girado 120° en sentido positivo.



A partir del diagrama se obtienen las siguientes relaciones:

$a = 1 = \cos 120° + j\,\text{sen}\,120° = -\frac{1}{2} + j\frac{\sqrt{3}}{2} = e^{j120}$

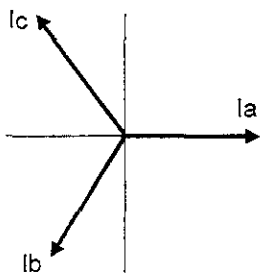
$a^2 = 1 = \cos 240° + j\,\text{sen}\,240° = -\frac{1}{2} - j\frac{\sqrt{3}}{2} = e^{j240°}$

$a^3 = 1 = \cos 360° + j\,\text{sen}\,360° = 1 + j0 = e^{j360°}$

El factor "a" nos sirve para representar a los sistemas trifásicos senoidales desbalanceados en sistemas balanceados, además de que nos ayuda a definir las componentes simétricas.

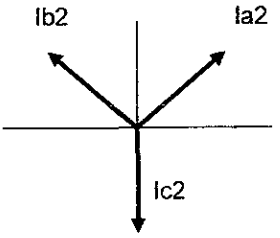
3.5.2 Componentes de secuencia positiva.

Es un sistema trifásico cuyos componentes están formados por tres vectores de igual módulo, con diferencias de fase entre dos vectores consecutivos de 120°, y con una secuencia de fases a, b y c.



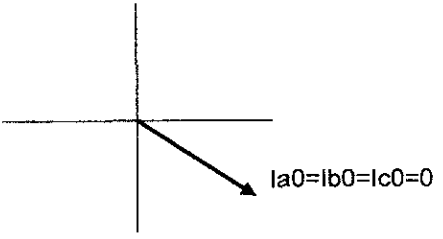
3.5.3 Componentes de secuencia negativa.

Es un sistema trifásico cuyos componentes están formados por tres vectores de igual módulo, con diferencias de fase entre dos vectores consecutivos de 120°, y con la secuencia de fases a. c y b.



3.5.4 Componentes de secuencia cero.

En este sistema sus componentes están formados por tres vectores de igual módulo y con diferencia de fase entre ellos nula.



La suma de los tres sistemas: de secuencia positiva, negativa y cero nos representan un sistema de tres vectores desequilibrados, que se puede representar por el sistema de ecuaciones siguiente.

$$Ia=Ia1+Ia2+Ia0$$
$$Ib=Ib1+Ib2+Ib0$$
$$Ic=Ic1+Ic2+Ic0$$

Si sustituimos en las expresiones anteriores el operador "a", las ecuaciones quedaran de la siguiente forma:

$$Ia = Ia1+Ia2+Ia0.$$
$$Ib = a^2Ia2+aIa2+Ia0.$$
$$Ic = aIa1+ a^2Ia2+Ia0.$$

Donde expresando en forma matricial tenemos:

$$\begin{bmatrix} Ia \\ Ib \\ Ic \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ a^2 & a & 1 \\ a & a^2 & 1 \end{bmatrix} \begin{bmatrix} Ia1 \\ Ia2 \\ Ia0 \end{bmatrix}$$

Despejando las corrientes de secuencia positiva Ia1, la de secuencia negativa Ia2, y la de secuencia cero Ia0, tenemos que:

$$Ia0 = \frac{Ia + Ib + Ic}{3}$$

$$Ia1 = \frac{Ia + aIb + a^2Ic}{3}$$

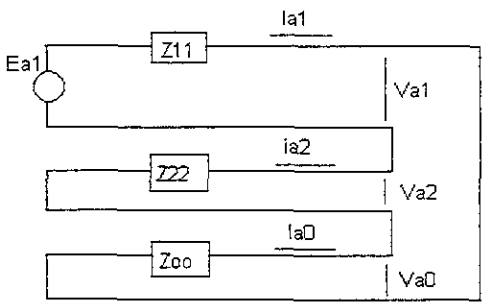
$$Ia2 = \frac{Ia + a^2Ib + aIc}{3}$$

Las ecuaciones obtenidas nos sirven para descomponer un sistema desequilibrado en 3 sistemas de fasores independientes, para resolver así un sistema asimétrico ó desbalanceado.

3.6 Obtención de las corrientes de fase y de los voltajes al neutro de los diferentes tipos de falla.

3.6.1 Falla monofásica de línea a tierra.

El circuito equivalente se obtiene al conectar en serie los circuitos equivalentes de secuencia positiva, negativa y cero



Falla monofásica de línea a tierra.

En el punto donde se tiene la falla se tienen las siguientes expresiones para las corrientes de fase:

$$Ia=Ia1+Ia2+Ia0= \frac{3(Ea1)}{Z11 + Z22 + Z00}$$

$$Ib=Ic=0$$

Los voltajes al neutro en el punto de falla son.

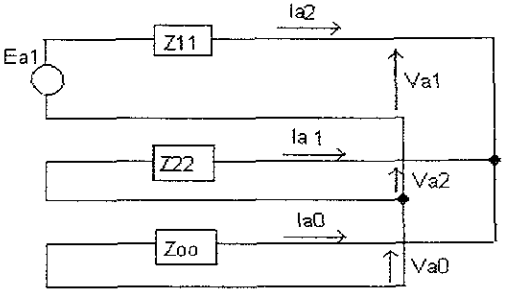
$$Va = Ea1-(Z11+Z22+Z00)\frac{Ea1}{Z11+Z22+Z00}=0$$

$$Vb=\left[\frac{(a^2-a)Z22+(a^2-1)Z00}{Z11+Z22+Z00}\right]Ea1$$

$$Vc=\left[\frac{(a-a^2)Z22+(a-1)Z00}{Z11+Z22+Z00}\right]Ea1$$

3.6.2 Falla doble de línea a tierra.

Utilizando la conexión de circuitos de secuencia positiva, negativa y cero, las tres redes de secuencia deben ser conectadas en paralelo en el punto de falla para simular la falla doble de línea a tierra



Falla doble de línea a tierra.

Utilizando las componentes simétricas en el circuito, se obtienen las corrientes de secuencia positiva, negativa y cero:

$Ia=0$

$$Ib=\left(\frac{(a^2-1)Z22+(a^2-a)Z00}{Z11Z22+Z11Z00+Z22Z00}\right)Ea1$$

$$Ic=\left(\frac{(a-1)Z22+(a-a^2)Z00}{Z11Z22+Z11Z00+Z22Z00}\right)Ea1$$

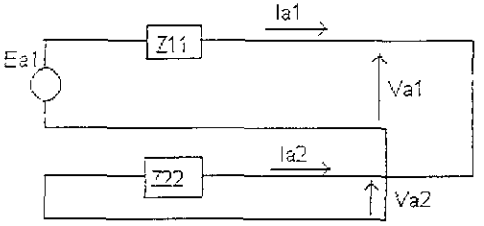
También obtenemos los voltajes al neutro Va, Vb y Vc por medio de las expresiones siguientes:

$$Va=\left(\frac{3(Z22Z00)}{Z11Z22+Z11Z00+Z22Z00}\right)Ea1$$

$$Vb=Vc=0$$

3.6.3 Falla entre dos fases.

En esta falla las redes de secuencia positiva y negativa deben ser conectadas en paralelo por el punto de falla para simular un fallo línea a línea.



Falla entre dos fases.

Puesto que no hay conexión a tierra en el punto de falla, la corriente de secuencia cero es igual a cero.

Las corrientes de las 3 fases son:

Ia=0

$$Ib=\frac{a^2-a}{Z11+Z22}Ea1$$

$$Ic=\frac{a-a^2}{Z11-Z22}Ea1$$

Así como los voltajes Va, Vb y Vc al neutro son:

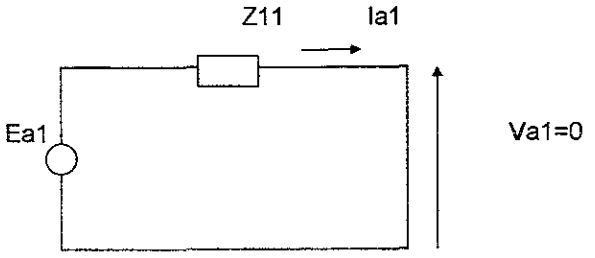
$$Va=\frac{2Z22}{Z11+Z22}Ea1$$

$$Vb=-\frac{Z22}{Z11+Z22}Ea1$$

$$Vc=-\frac{Z22}{Z11+Z22}Ea1$$

3.6.4 Falla trifásica.

Al no existir corrientes ni voltajes de secuencia negativa y cero, todas las cantidades que intervienen son de secuencia positiva.



Falla trifásica

En el punto de falla se tienen las siguientes cantidades de fase utilizando las componentes simétricas:

$$Ia=Ia1$$
$$Ib=a^2Ia1$$
$$Ic=aIa1$$

Los voltajes al neutro se representan como:

$$Va=Va1$$
$$Vb=a^2Va1$$
$$Vc=aVa1$$

3.7 Calculo de fallas por el método de nodos.

El método de nodos se relaciona ampliamente con el de las componentes simétricas y consiste en obtener las componentes simétricas de la corriente de falla. Los valores de corriente y voltaje se encontraran en los diferentes nodos del sistema por medio de la matriz de impedancias.

Los elementos de la matriz de impedancias se obtienen en primera instancia por inspección de los elementos de la red eléctrica, basándose en las admitancias del sistema.

La formulación sistemática de las ecuaciones determinadas en los nodos de un circuito, tiene su principio en la aplicación de la ley de corrientes de Kirkchoff, la cuál constituye la base para la solución de los sistemas de potencia por medio de las computadoras digitales.

La corriente que circula hacia la red eléctrica procedente de las fuentes de corriente conectadas a un nodo, es igual a la suma de varios productos.

Tenemos que la forma general de representar una red eléctrica con n+1 nodos es la siguiente:

$$Y11V1+Y12V2+.....+Y1nVn=If1$$
$$Y21V1+Y22V2+.....+Y2nVn=If2$$
$$Y31V1+Y32V2+.....+Y3nVn=If3$$
$$---$$
$$Yn1V1+Yn2V2+.....+YnnVn=Ifn$$

Donde :

If1,If2 ,....., Ifn indican las corrientes de las fuentes de corriente
V1,V2 ,....., Vn indican los voltajes de los nodos con respecto al neutro.
Y11,Y22 ,..., Ynn son las admitancias propias.
Y12,Y13 ,..., Ypq son las admitancias mutuas.

Las admitancias propias en los nodos se forman por la suma de todas las admitancias conectadas directamente hacia ellos. Las admitancias mutuas son aquellas que se encuentran conectadas entre los nodos de la red eléctrica.

En un nodo cualquiera se tiene que cada producto es igual a la tensión en dicho nodo, multiplicado por la suma de todas las admitancias que terminan en él.

Cada producto corresponde a la corriente que fluye en el nodo si la tensión es nula en los restantes.

El sistema de ecuaciones anterior se puede representar como un sistema matricial, mediante la matriz de admitancias de nodos, y se identifica como Y bus.

$$\begin{bmatrix} Y11 & Y12 & .. & Y1n \\ Y21 & Y22 & ... & Y2n \\ . & ... & .. & .. \\ Yn1 & Yn2 & ... & Ynn \end{bmatrix} x \begin{bmatrix} V1 \\ V2 \\ . \\ Vn \end{bmatrix} = \begin{bmatrix} If1 \\ If2 \\ ... \\ Ifn \end{bmatrix}$$

La inversa de la matriz de admitancias de nodos se conoce como la matriz de impedancias de bus y se identifica como Z bus, la cuál involucra todas las impedancias del sistema, y por definición:

$$Y^{-1} \text{ Bus} = Z \text{ bus}$$

$$\left[Y^{-1}\right] = [Zbus] = \begin{bmatrix} Z11 & Z12 & ...Z1n \\ Z21 & Z22 & ..Z2n \\ Zn1 & Zn2 & ..Znn \end{bmatrix}$$

Como Y bus es simétrica con respecto a la diagonal principal, de igual manera lo es Z bus.

Para calcular el voltaje al neutro en cualquier nodo afectado por la falla trifásica, solo se tiene que multiplicar la impedancia de falla, por la corriente de falla.

$$Vf = Zf\ If$$

Para calcular las distintas corrientes de la red durante el cortocircuito se tiene la siguiente expresión:

$$Ijk = Yjk(Vfj - Vfk)$$

Donde:

Ijk= corriente que circula por la rama comprendida entre el nodo j y el nodo k.
Yjk= admitancia de la rama comprendida entre el nodo j y el nodo k.
Vfj=Voltaje al neutro en el nodo j.
Vfk=Voltaje al neutro del nodo k.

3.7.1 Pasos a seguir para el cálculo de fallas por el método de nodos.

1.-Dibujar el circuito equivalente de secuencia positiva para las condiciones que se tengan, indicando las impedancias de cada rama en valores en por unidad.

2.-Calculo de la matriz de admitancias de bus (Ybus).

3.-Inversión de la matriz de admitancias y obtención de la matríz de impedancias de bus (Zbus).

4.-Cálculo de las corrientes de corto circuito trifásico en el punto de falla en valores en por unidad.

5.-Cálculo de los voltajes durante la falla en valores en por unidad.

6.-Cálculo de las corrientes que circulan por las líneas de transmisión y por los generadores debidas a fallas en valores en por unidad.

7.- Cálculo de las aportaciones en los generadores a las corrientes de cortocircuito.

Se concluye que la matriz de impedancias de bus es importante y muy útil para efectuar cálculos de fallas basándose en una computadora digital, aunque se presentan algunas dificultades en su desarrollo, una de ellas es que se requiere de mucha memoria, lo que origina mucho tiempo de proceso presentando esto como un limitante principal en los cálculos.

3.8 Simplificaciones en el cálculo de fallas.

1.-Se ignoran las cargas conectadas al sistema, así como la capacitancia de las líneas y la excitación de los transformadores.

2.-Si no se conocen los voltajes en los diferentes puntos, se considera su valor en por unidad igual a 1 y en fase.

3.- Se ignora la resistencia de los elementos de la red, ya que es mucho menor que la reactancia inductiva de los mismos.

4.-La impedancia de falla es igual a cero, debido a que la corriente de cortocircuito es muy alta.

CAPÍTULO IV

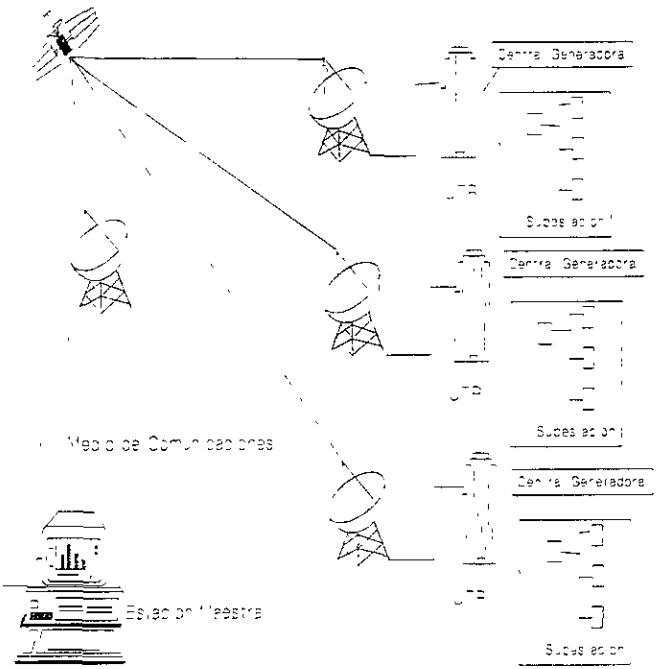
La Computadora Digital

4.1 FUNCIONES BÁSICAS DE UN SISTEMA DE CONTROL SUPERVISORIO

Para poder controlar un proceso se requiere tener acceso a sus parámetros de operación y a los elementos que permitan controlarlo. En el manejo de redes eléctricas, los sitios donde se puede tener acceso a los parámetros de operación son las centrales generadoras y las subestaciones eléctricas.

En cada uno de estos sitios con datos a captar y dispositivos a controlar se instalan terminales remotas (UTR), que centralizan su información en una estación maestra a través de un sistema de comunicaciones. Cada una está encargada de sus funciones de control y supervisión, siendo éstas las siguientes:

- Adquisición de datos.
- Control de dispositivos.
- Almacenaje y manejo de información.
- Proporcionar información al operador.
- Transmisión de información.



Elementos de un Sistema de Control Supervisorio

4.2 FUNCIONES BÁSICAS DE LA ESTACIÓN MAESTRA

4.2.1 Adquisición de Datos

El sistema debe ser capaz de acceder todos los datos de los puntos de interés, en una escala de tiempo que esté de acuerdo con las características dinámicas del sistema a ser supervisado o controlado.

En algunos sistemas de energía, se requieren ciclos de muestreo de la información de uno a 30 segundos cada ciclo. Estos datos pueden ser binarios o valores analógicos continuos. La adquisición típica de datos utiliza ambos tipos de entradas.

Ejemplo de adquisición binaria.

Entradas de estado	Abierto / Cerrado Alarmado / Normal
Contadores de pulsos	1 pulso = 1 cuenta = unidad / hora.

4.2.2 Control de Dispositivos

El sistema deberá proporcionar al operador la capacidad de control de elementos remotos integrantes de la red eléctrica o del proceso. Debe estar disponible una variedad de modos de control según el tipo de dispositivo a controlar, incluyendo contactos de cierre temporizado, momentáneos, continuos o de sello (LATCH).

4.2.3 Almacenamiento de Información

El sistema debe tener la capacidad de almacenar la información adquirida durante los muestreos y desplegarla. Existen dos modos: período corto, en el orden de algunos segundos, donde se proporciona una indicación inmediata del estado del sistema; y el período largo, para preparación de reportes y uso de planeación como referencia histórica.

4.2.4 Presentación de la Información al Operador

Cualquier sistema de control supervisorio y adquisición de datos (SCADA) consta de una interfaz Hombre/Máquina, que proporciona acceso al operador a los datos colectados de manera conveniente y lo habilita para tomar acción del sistema de control.

Se generan imágenes visuales de la red eléctrica incluyendo todos los equipos a ser controlados, mostrando su estado actual, así como las tensiones, megawatts, corrientes, etc. y se generan reportes, listados de mediciones y alarmas.

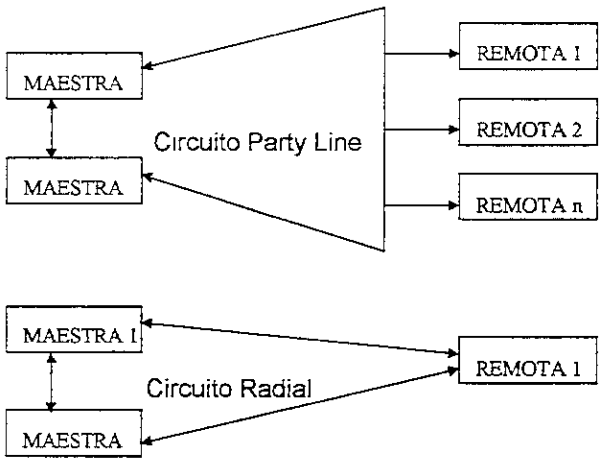
4.3 TIPOS DE CONTROL SUPERVISORIO

Existen tres configuraciones de acuerdo con su aplicación funcional.

1. *Sistema Maestra - Unidades Terminales Remotas*
2. *Sistema Maestra - Subsistemas Remotos Distribuidos.*
3. *Sistemas Abiertos - Estaciones Maestras Conectadas en Red.*

4.3.1 El Sistema Maestra - Unidades Terminales Remotas.

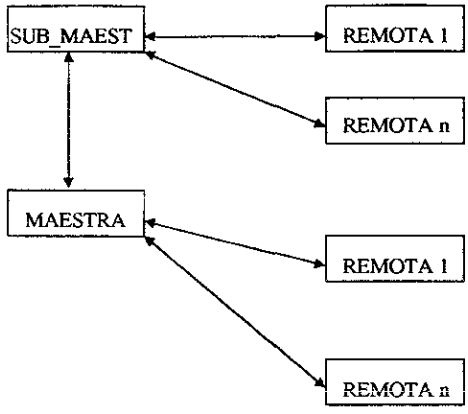
Se utiliza para obtener el control y la supervisión de pequeños procesos como son: sistemas de bombeo de agua (pozos), sistemas de distribución, etc.



Sistema Maestra - Unidades Terminales Remotas

4.3.2 El Sistema Maestra - Subsistemas Remotos Distribuidos.

Se utiliza para la supervisión y control de sistemas más complejos como centrales generadoras automatizadas y redes de distribución donde la supervisión se hace hasta nivel de poste.

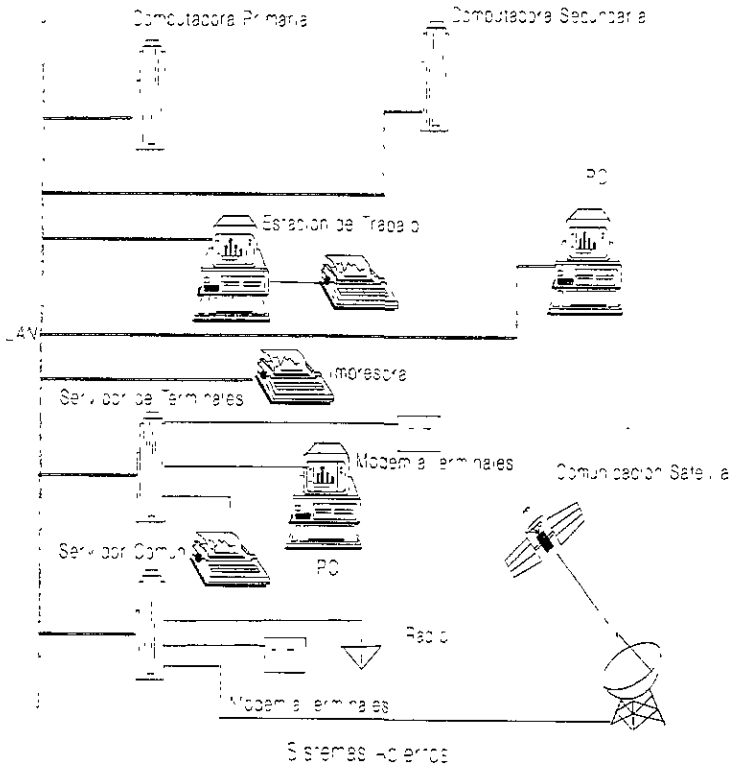


Sistema Maestra - Subsistema Remotos Distribuidos

4.3.3 Sistemas Abiertos - Estaciones Maestra Conectadas en una Red de Cómputo.

En ellas participan varias estaciones maestras, entre las cuales existe el intercambio de información mediante el uso de protocolos universales de comunicaciones, que no dependen del equipo utilizado (hardware).

Por lo general las estaciones maestras son duales, es decir, todo el equipo o dispositivo que al fallar, deja fuera todo el sistema es redundante y con transferencia automática para no perder la información ni la disponibilidad del equipo.



Sistemas Abiertos

4.4 Crisis del Software.

La velocidad a la que avanza la tecnología de hardware de computadoras es sorprendente; año con año se logran avances que conducen a la construcción de computadoras más veloces, más compactas y más baratas. Sin embargo, en el aspecto de software no parece haber un desarrollo similar. Mientras los costos de hardware han disminuido continuamente, los de software han hecho lo contrario. La construcción de software no es una tarea fácil y en muchas ocasiones los proyectos de programación sobregiran los presupuestos de tiempo y dinero.

4.5 El problema de mantenimiento de software.

La construcción de software ha recibido la atención de los expertos desde hace mucho tiempo; en la década de los 70's se consiguieron avances significativos hacia el desarrollo de metodologías de forma sistemática y a bajo costo. Como resultado de esos esfuerzos surgieron técnicas que, como el diseño estructurado y el desarrollo descendente (top-down), han sido las herramientas utilizadas por los programadores para construir software. Aunque dichas técnicas han sido empleadas durante mucho tiempo en proyectos realmente complejos, la mayoría de los ingenieros de software coinciden en afirmar que sufren de tres grandes deficiencias:

1. Los productos que resultan al emplear éstas técnicas son poco flexibles.

2. Los programadores que las usan tienden a concentrarse en el diseño y la implementación del sistema, sin tomar en cuenta su vida posterior.

3. No alientan al programador a aprovechar el trabajo de proyectos anteriores.

La desventaja de utilizar una metodología que se concentra en el diseño inicial del sistema se hace evidente si se toma en cuenta que la vida útil de un producto de software puede ser cinco o seis veces más grande que el lapso en que se desarrolla; por ejemplo, un sistema que se desarrolla en uno o dos años puede mantenerse trabajando durante un período que va de cinco a quince años. Los gastos que se hacen durante éste último período (gastos de mantenimiento) representan alrededor del 70% del costo total del sistema.

4.6 Reutilización de código.

La *reutilización de código* es otro de los factores que no se toma en cuenta en los métodos de diseño tradicionales. Cualquier programador que desarrolla un sistema nuevo debe escribir una buena cantidad de código que ha escrito anteriormente una y otra vez. Las rutinas de búsqueda, ordenamiento, manejo de menús y despliegue de ventanas, entre otras, se repiten continuamente en proyectos diferentes. Los métodos de desarrollo de software deben alentar al

programador a utilizar el código escrito previamente por él mismo o por otros programadores.

Tradicionalmente se han empleado tres tipos de reutilización: de personal, de diseño y de código fuente. En la primera, a un proyecto nuevo se le asignan programadores que tienen experiencia en el desarrollo de proyectos semejantes. La segunda consiste en emplear el diseño de un sistema para desarrollar otro sistema similar. La tercera y última forma de reutilización se da cuando un programador utiliza parte de un programa escrito con anterioridad para crear un nuevo programa.

Sin embargo, éstas tres formas de reutilización se han empleado en forma muy limitada, por lo que es necesario buscar técnicas más generales.

4.7 Modularidad.

La modularidad es una de las herramientas de diseño más poderosas para facilitar el desarrollo y mantenimiento de sistemas de software. La modularidad permite definir un sistema complejo en términos de unidades más pequeñas y manejables; cada una de esas unidades (o *módulos*) se encarga de manejar un aspecto local de todo el sistema, interactuando con otros módulos para cumplir con el objetivo global.

La mayoría de los lenguajes de programación actuales alientan el uso de la modularidad: en lenguajes estructurados como C o Pascal, la modularidad se basa en el concepto de función (también llamada procedimiento o subrutina); en lenguajes más evolucionados, como Ada o Modula-2, los módulos corresponden a conjuntos de funciones relacionados con las estructuras de datos que manipulan esas funciones (paquetes, en la terminología de Ada).

Sin embargo, para ser realmente útil, el concepto de módulo debe ser aún más sofisticado. Según Meyer, las propiedades de un módulo deberían ser las siguientes:

a) *Decomponibilidad*: Un método de diseño modular debe permitir descomponer un problema de diseño en subproblemas más pequeños que pueden resolverse en forma independiente.

b) *Componibilidad*: Una vez que se cuente con un conjunto de módulos que realizan una función específica, se debe alentar al programador a usar esos módulos para construir nuevos programas.

c) *Comprensibilidad*: El lector de un programa o librería debe ser capaz de entender el funcionamiento de cada módulo sin necesidad de consultar el texto de otros módulos.

d) *Continuidad*: Un cambio pequeño en las especificaciones de un programa debe causar cambios en un sólo módulo o en un conjunto pequeño de ellos.

e) *Protección*: Un error de ejecución en el funcionamiento de un módulo no debe expandirse hacia los demás módulos.

Estas cinco propiedades pueden alentarse con las siguientes estrategias:

a) Un módulo debe corresponder con una unidad sintáctica del lenguaje (una subrutina, un paquete, una *clase*); esta unidad debe poder ser compilada por separado, tal vez para ser almacenada en una librería (con esto se mejoran la componibilidad y comprensibilidad del sistema).

b) Los módulos deben tener pocas interfaces (medios de comunicación con otros módulos), y éstas deben ser pequeñas. Un número pequeño de interfaces aumenta la independencia de los módulos, lo cual hace más fácil el proceso de componer nuevos sistemas a partir de módulos prefabricados, ayuda a evitar que los errores en un módulo se propaguen por todo el sistema y hace que cada módulo de un sistema sea más comprensible.

c) Cada módulo debe ocultar su implementación y algoritmos internos al resto del sistema (principio de ocultamiento de información). No se debe permitir que un módulo modifique los elementos internos de otros módulos; sino que la comunicación entre ellos debe realizarse mediante interfaces explícitas y bien definidas (esto favorece la comprensibilidad y la protección modular).

4.8 La programación orientada a objetos.

La programación orientada a los objetos (POO) es un método de diseño que tiene como objetivo establecer técnicas de desarrollo de software para producir sistemas modulares. Un sistema orientado a objetos se puede entender fácilmente, por lo que su desarrollo, depuración y mantenimiento se facilitan en gran medida.

En primera instancia, la POO se basa en una idea relativamente sencilla: un programa de computadora es un modelo que representa un subconjunto del mundo real; la estructura de ese programa simplifica en gran medida, si cada una de las entidades (u objetos) del problema que se esta modelando corresponde directamente con un objeto que se pueda manipular internamente en el programa.

Un ejemplo sencillo puede ser un programa de nómina: cualquier empleado de cierta empresa constituye una entidad real que tal vez represente dentro de un programa con un conjunto de variables o con una estructura (o registro).

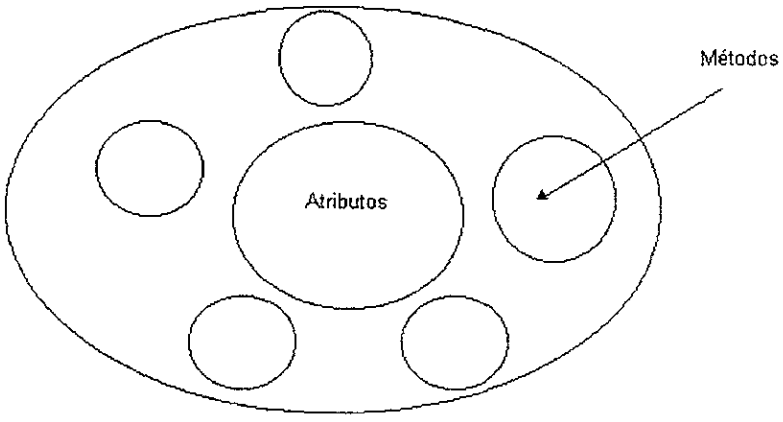
El proceso de representar entidades reales con elementos internos a un programa recibe el nombre de *abstracción de datos*. La abstracción es una descripción simplificada que resalta solamente las características esenciales de un objeto de la realidad, despreciando las características no esenciales. La abstracción de datos no se concentra en la representación interna de un objeto (un entero, una cadena de bits, un arreglo), sino en sus atributos (con el nombre, sueldo y edad de un empleado) y en las operaciones que se pueda realizar sobre ese objeto (como calcular el sueldo de un empleado o los impuestos que debe pagar). De esta forma, un tipo de dato abstracto se puede describir concentrándose en las operaciones que manipulan a los objetos de ese tipo, sin caer en los detalles de representación y manipulación de los datos.

La abstracción de datos es un concepto común en los lenguajes de programación moderna. Muchos lenguajes proporcionan un tipo de datos para números de punto flotante; cuando un programador utiliza datos de ese tipo, puede hacer operaciones como suma, multiplicación y exponenciación con esos datos, pero no es necesario que se preocupe por la representación de los números (p.e. cuatro palabras de máquina en las que se almacenan la mantisa y el exponente, junto con sus signos) ni por la forma en que el procesador realiza las operaciones (suma de enteros, comparación, desplazamiento lógico, etc.).

4.9 Mecanismos Básicos de la Programación Orientada a Objetos.

4.9.1 Objetos

Un programa tradicional se compone de procedimientos y datos, un programa orientado a objetos se compone de objetos. Un objeto es una encapsulación genérica de datos y los procedimientos, también llamados métodos, para modificar dichos datos, también llamados atributos. Por lo tanto un objeto contiene operaciones que definen su comportamiento y por otra variables que definen su estado. Un ejemplo sería el objeto generador, con sus atributos: voltaje, potencia, impedancias de secuencia; y por métodos: cambio de base, impresión de resultados.

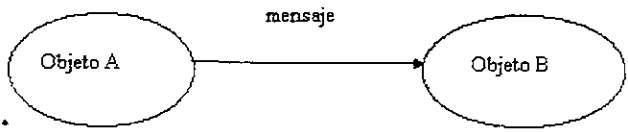


Un objeto podría ser real o abstracto, como en los siguientes ejemplos:

- una factura
- una organización
- una figura en un programa de dibujo
- una pantalla con la que interactúa un usuario
- un campo o nodo de la pantalla de una herramienta CASE
- un mecanismo en un dispositivo de robótica
- todo un plano de ingeniería
- un texto y fotografías utilizadas en la plana de un periódico
- un avión
- el vuelo de un avión
- una reservación aérea
- un icono en la pantalla al que un usuario puede apuntar y "abrir"
- un proceso para llenar un pedido
- el proceso par escribir esta línea.

4.9.2 Mensaje

El mensaje es la información que se mandan entre dos objetos, para que el objeto receptor ejecute uno de sus métodos previa petición del objeto transmisor. Los objetos pueden ser muy complejos, puesto que contener muchos subobjetos, éstos a su vez pueden contener subobjetos, etc. La persona que utilice el objeto no tiene que conocer la complejidad interna, sino la forma de comunicarse con él y la forma en que responde. El conjunto de mensajes a los que un objeto puede responder se le llama protocolo.



Cuando el objeto A envía una solicitud al objeto B, una operación del objeto A llama a una operación del objeto B, lo que en cierto modo provoca el manejo de los datos del objeto B. El objeto B devuelve, por lo general, un valor al objeto A.

Por ejemplo, consideremos una instancia del tipo de objeto *cilindro*. Podemos enviar una solicitud para obtener la altura del objeto cilindro. La operación altura regresa el valor *altura* almacenado. Además, podemos enviar una solicitud para obtener el volumen del objeto. La operación volumen utiliza el siguiente método para calcular y devolver el valor *volumen* del objeto:

PI*RADIO*2*ALTURA

En código en C++ quedaría:

```
class cilindro{
        private:
            double altura, radio;
         public:
             //constructor
             cilindro(double alt, double rad): altura(alt), radio(rad){}
             // método volumen
              double volumen(){
                    return 3.141592*radio*2*altura;
              }
};
```

4.9.3 Métodos

Los métodos, también llamados funciones miembro, se implementan en una clase y determina cómo tiene que actuar un objeto cuando recibe un mensaje Los métodos especifican la forma en que se controlan los datos de un objeto. Los métodos en un tipo de objeto sólo hacen referencia a las estructuras de datos de este tipo de objetos; las variables asociadas (estructuras de datos) permitirán almacenar información para dicho objeto. Un método puede enviar mensajes a otros objetos solicitando una acción de ellos o quizá un intercambio de información.

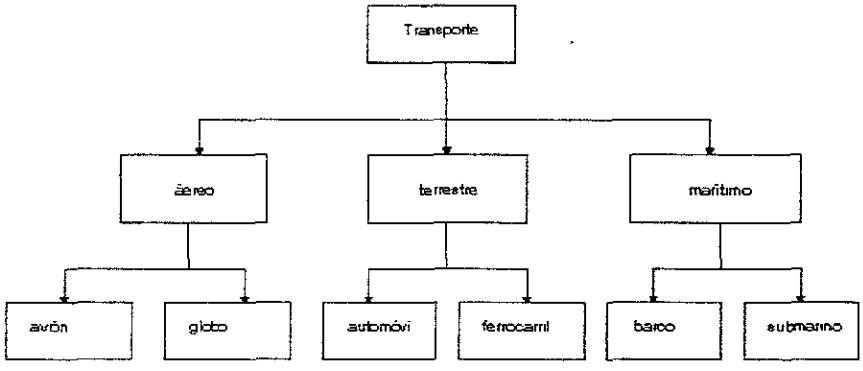
4.9.4 Clases

Una clase es una implantación de un tipo de objetos. Especifica una estructura de datos y los métodos operativos permisibles que se aplican a cada uno de sus objetos. La clase se puede considerar como una plantilla para generar otros objetos. Una clase describe los métodos y atributos que definen las características comunes de los objetos de esa clase.

En el mundo real no existen clases, nosotros hacemos dichas clasificaciones para poder conocer más nuestro entorno y poder generalizar nuestro conocimiento a otras situaciones comunes. Un ejemplo sería modelar un automóvil con todas las características que todo automóvil tiene en común (número de llantas, número de ocupantes, color, etc.).

4.9.5 Subclases

Una idea principal de la Orientación a Objetos es el hecho de reutilizar código, y el hecho de no redundar en almacenar el mismo atributo



Ejemplo de una estructura jerárquica de clases

Del ejemplo anterior en el cual se muestra una estructura jerárquica de clases de tres niveles, en teoría la estructura jerarquía podría ser de cualquier número de niveles pero en la práctica se ha visto que es recomendable no poder una jerarquización mayor a 5 niveles, ya que si exageramos en lugar de ser un apoyo las subclase se convertirán en una carga

4.10 Características de la Programación Orientada a Objetos.

Las características fundamentales de la POO son: abstracción, encapsulamiento, herencia y polimorfismo. Hay autores que aceptan otras características como son: herencia múltiple, identidad, clasificación.

4.10.1 Abstracción:

Consiste en la representación de las características esenciales de algo sin incluir antecedentes o detalles irrelevantes. Esto es, por medio de la abstracción conseguimos no detenernos en los detalles concretos de las cosas que no interesen en cada momento, sino generalizar y centrarse en los aspectos que permitan tener una visión global del tema.

Una abstracción que describa un conjunto de objetos en términos de una estructura de datos encapsulada u oculta y las operaciones sobre esa estructura, se le denominará como un *tipo abstracto de datos*.

4.10.2 Encapsulamiento:

El encapsulamiento u ocultamiento de la información se refiere a la práctica de incluir dentro de un objeto todo lo que necesita, de tal forma que ningún otro objeto necesite conocer su estructura interna. Esta característica nos permite ver al objeto como una caja negra, en la que se ha metido de alguna manera toda la información relacionada con dicho objeto.

En C++, el ocultamiento de la información esta directamente relacionado por la clase. La clase es una abstracción, porque en ella se definen las propiedades y los atributos genéricos de un determinado conjunto de individuos con características comunes, y es una abstracción porque constituye la caja negra que encierra los datos que constan los objetos como los métodos que permiten manipularlos.

4.10.3 Herencia:

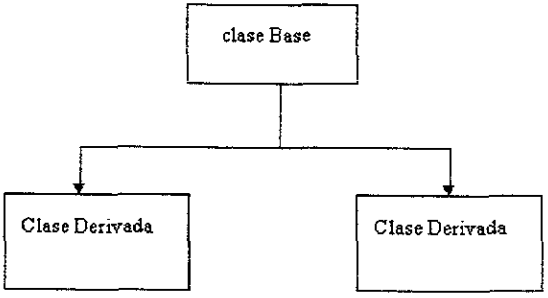
La herencia es el mecanismo para compartir automáticamente métodos y atributos entre las clases y sus subclases. Una clase implanta el tipo de objeto. Una subclase hereda propiedades de su clase padre; una sub-subclase hereda propiedades de las subclases, etc. Una subclase puede heredar la estructura de datos y los métodos, o algunos de los métodos, de su superclase. También tiene sus métodos e incluso tipos de datos propios.

Esta característica de la POO está fuertemente ligada a la reutilización del código. Esto es, el código de cualquiera de las clases existentes pueden ser utilizado sin más que crear una clase derivada de ella.

La herencia puede ser de 2 tipos: Simple y Múltiple.

4.10.3.1 Herencia Simple.

Es cuando una clase base hereda a una o varias clases hijas (derivadas).



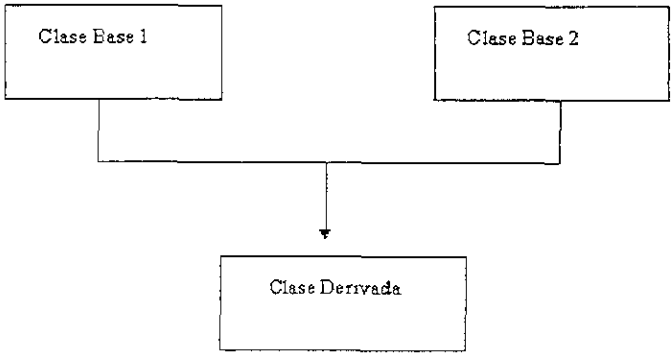
Ejemplo de Herencia Simple

Un ejemplo en C++ sería:

```
class Padre{
      private:
          char nombre[30];
          long telefono;
      public:
          ...
};

class Hijo : public Padre{
        private:
           char nombre[30];
           ...
         public:
          void Datos(){
                cout<< "El nombre :"<<nombre;
                cout<< " Nombre del padre:"<<padre.:nombre;
        }
},
```

4.10.3.2 Herencia Múltiple.

Es cuando varias clases base generan un clase derivada. El mejor ejemplo es la de la familia la clase Padre y Madre generaran a la Hijo.



Herencia Múltiple

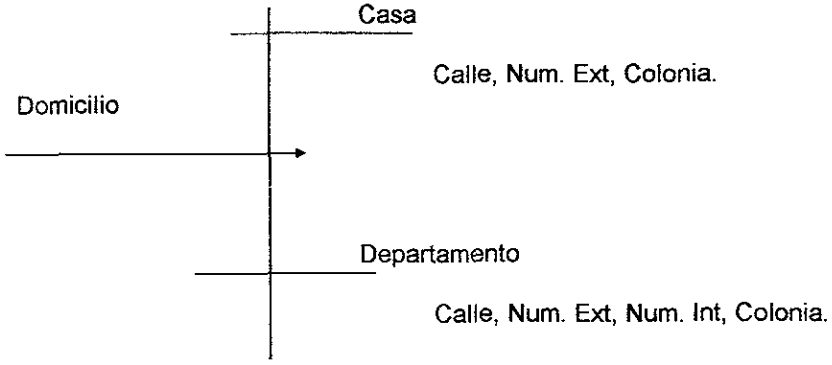
Un ejemplo en C++ sería:

```
class Padre{
      private:
          char nombre[30];
           long telefono;
      public:
          .
};

class Madre{
      private:
          char nombre[30];
           long telefono,
      public:
          ...
};


class Hijo : public Padre, public Madre{
       private:
          char nombre[30];
         ...
        public:
         void Datos(){
               cout<< "El nombre :"<<nombre;
               cout<< " Nombre del padre:"<<padre :nombre;
               cout<<" Nombre de la madre:"<<madre::nombre;
       }
};
```

4.10.4 Polimorfismo:

Polis muchos, *morfos* formas. Esta característica nos permite implementar múltiples formas de un mismo método, dependiendo cada una de ellas de la clase sobre la que se realice la implementación. En otra palabra nosotros tendremos una sola interfaz pero varios métodos, asociados.

Un ejemplo sería cuando preguntamos por el método Domicilio de las clases Casa y Departamento. Cuando se ejecutará el método para un objeto del tipo Casa sería suficiente con indicar calle, número exterior, colonia; pero si se ejecuta para un objeto del tipo Departamento será necesario indicar la calle, el número exterior, la colonia pero también el número interior. Así que la POO, le da la ventaja al programador de no requerir recordar detalles como este y enfocarse en la solución del problema, como en el ejemplo nosotros llamaremos al método Domicilio y el compilador sabrá que método ejecutar dependiendo si está aplicado a una Casa o a un Departamento.



En C++ sería:

```
class Casa{
      private:
         int numero;
         char *calle, *colonia;
      public:
         void Domicilio(){
            cout<<"domicilio:"<<calle<<"#"<<numero<<"col."<<colonia;
         }
};

class Departamento{
      private:
         int numero_ext, numero_int;
         char *calle, *colonia;
      public:
         void Domicilio(){
```

```
            cout<<"domicilio:"<<calle<<"#"<<numero_ext
                <<"int."<<numero_int<<"col."<<colonia;
        }
};
```

Así cuando creemos un objeto de la clase casa y otro objeto de la clase Departamento y llamemos a el método Domicilio() se ejecutará el correspondiente, dependiendo la clase.

Se puede considerar tres tipos de polimorfismo: funciones sobrecargadas, operadores sobrecargados, funciones virtuales.

4.10.4.1 Funciones sobrecargadas:

La sobrecarga de funciones es una característica de C++ que hace los programas más legibles. Consiste en volver a declarar una función anteriormente declarada, con distinto número y/o tipo de parámetros.

Normalmente, cada función tiene su propio nombre que la distingue de las demás. No obstante, se puede presentar casos en los que varias funciones ejecuten la misma tarea sobre objetos de diferentes tipos, y puede resultar conveniente que dichas funciones tengan el mismo nombre. Pues bien C++ permite definir funciones con el mismo nombre si el número y/o tipo de los parámetros son diferentes. Cabe señalar que ésta característica no era posible en el tan usado lenguaje C, ya que si hubiéramos creado una función int suma(int, int) y otra float suma(float, float), nos hubiera marcado un error el compilador por una redeclaración de funciones, apesar que realmente son distingibles de ahí que en C++ se pueda distinguir.

Ejemplo en C++:

```
int suma(int a, int b){
        return a+b;
}

float suma(int f1,int f2){
        return f1+f2;
}

char * suma(char *s1,char *s2){
        char *temp;     /* creamos una cadena temporal del tamaño de la suma de
                        los parámetros. */
        temp = new char[strlen(s1)+strlen(s2)+1];
        strcpy(temp,s1); // copia el contenido de s1 a temp
        strcat(temp,s2); // concatena a temp el contenido de s2
        return temp;    // devuelve temp
}

void main(){
    cout<<suma(2,3);  // se ejecuta int suma(int,int);
```

```
    cout<<suma(2.2,3.3);// se ejecuta float suma(float,float);
    cout<<suma("HOLA"," MUNDO"); // se ejecuta char* suma(char*,char*);
}
```

4.10.4.2 Operadores Sobrecargados:

El término de operador sobrecargado se refiere a un operador que es capaz de desarrollar su función en varios contextos diferentes sin necesidad de otras operaciones adicionales. Por ejemplo, la suma a + b, en la práctica para nosotros supondrá operaciones diferentes dependiendo de que estemos trabajando en el campo de los números reales o en el campo de los número complejos. Si dotamos al operador + para que además de sumar reales, permita también sumar complejos, dependiendo esto del tipo de los operandos, entonces diremos que el operador + está sobrecargado.

La sobrecarga de operadores resulta especialmente útil cuando se trata de trabajar con tipos abstractos de datos que definen objetos pertenecientes al campo de las matemáticas; por ejemplo, operaciones con número complejos o con el espacio vectorial (x,y,z), como en el ejemplo se muestra.

Ejemplo en C++:

```
class tres_D{
    int x,y,z;
  public:
    void asigna(int x1,int y1,int z1){
       x=x1; y=y1; z=z1;
    }
   tres_D operator+(tres_D t){
       tres_D temp;
       temp.x=x+t.x;
       temp.y=y+t.y;
       temp.z=z+t.z;
       return temp;
    }
 tres_D operator-(tres_D t){
       tres_D temp;
       temp.x=x-t.x;
       temp.y=y-t.y;
       temp.z=z-t.z;
       return temp;
    }
  tres_D &operator=(const tres_D &t){
      x=t.x; y=t.y; z=t.z;
      return *this;
 }


};
```

```
void main(){
      tres_D c,a,b;
      a.asigna(1,1,1),
      b.asigna(2,2,2);
      c=a+b;  // c valdría (3,3,3)
      c=a-c;   // c valdría (-2,-2,-2)
}
```

Veasé que los operadores + y - están sobrecargados ya que dichos operadores no están previamente definidos para la clase tres_D hasta que nosotros explícitamente le decimos al compilador lo que queremos que interpreté al usarlos para ésta clase.

4.10.4.3 Funciones virtuales:

Una función virtual es una función miembro de una clase base que puede ser redefinida en cada una de las clases derivadas de ésta, y una vez redefinida puede ser accedida mediante un apuntador o referencia a la clase base, resolviéndose la llamada en función del tipo del objeto apuntado.

Una función virtual se declara colocando la palabra clave virtual antes de la declaración de la función miembro perteneciente a la clase base. Las redefiniciones que realicemos de esta función en las clases derivadas no necesitan incorporar en su declaración la palabra clave virtual; son declaradas implícitamente funciones virtuales. No obstante, se puede utilizar el especificador virtual en la función de la clase derivada que redefine a la virtual, pero su uso es redundante

Ejemplo en C++:

```
class A{
      protected:
          int a;
      public:
          A(int a1):a(a1){} //constructor
          virtual int quien(){   // función virtual
              return a;
          }
};

class B:public A{
      protected:
          int b;
      public:
          B(int a1,int b1):b(b1),A(a1){} //constructor
},

class C:public A{
      public:
          C(int a1):A(a1){} //constructor
```

```
        int quien(){
            return a*a;
        }
};

void main(void){
    A *p, objetoA(2);
    B objetoB(3,4);
    C objetoC(5),
     p=&objetoA;
     cout<<p->quien(), //ejecutaría el método quien() de la clase A
     p=&objetoB;  /* obsérvese que con el mismo apuntador del tipo de la base
            podemos acceder a las clases derivadas.*/
      cout<<p->quien(); /* ejecutaría el método quien() de la clase A, ya que no
      está redeclarado en la clase B*/
      p=&objetoC;
      cout<<&objetoC;
      cout<<p->quien(); // ejecutaría el método quien() de la clase C
}
```

4.11 Ventajas de la Programación Orientada a Objetos

Algunas de las ventajas que tiene la Programación Orientada a Objetos son:

- Los objetos bien diseñados son la base para sistemas que se construyen a partir de módulos reutilizables, dando lugar a una mayor productividad.
- La reutilización de las clases que han sido probadas en circunstancias reales, en proyectos anteriores, dan lugar a sistemas de mayor calidad y con menos errores.
- La herencia hace posible definir sistemas más fiables, más fáciles de ampliar y menos costos de mantener.
- La encapsulación ayuda a construir sistemas más seguros.
- Los modelos son más realistas, debido a la misma concepción del objetos.
- Existe mayor nivel de automatización con las bases de datos.
- Mejores herramientas *CASES*.
- Se explota más la computación paralela (procesamiento paralelo).
- Mejor comunicación entre los profesionales de los sistemas de información y los empresarios.
- Estabilidad. Al tener todo probado se vuelve el sistema más estable.
- El diseñador piensa en términos del comportamiento de objetos y no de detalles de bajo nivel.

4.12 Desventajas de la Programación Orientada a Objetos

Como en toda nueva metodología siempre hay algunas desventajas y la Orientación a Objetos no es la excepción entre ellas se encuentran:

- *Renuencia al cambio.* Muchas personas de las que tienen el poder de la *decisión en la creación de sistemas orientados a objetos no lo hacen debido a* que no confían en una nueva metodología o por no cambiar sus sistemas estructurales siguen programando en esos viejos lenguajes

- *Falta de Asesores* Cuando se tienen problemas grandes en cuanto al análisis y el diseño de un sistema grande, muchas veces no tenemos quien nos ayude, así que esto se vuelve un problema.

- *Curva de aprendizaje.* Hay costes inevitables, asociados al entrenamiento y el aprendizaje tiene a ser más lento que en los lenguajes estructurales.

- *Falta de herramientas.* Debido a la renuencia al cambio, se siguen construyendo herramientas para el desarrollo de sistemas estructurales y pocas para la orientación a objetos; acabe mencionar que las orientadas a objetos tienen una calidad excelente como lo es *Rational Rose* , *Delphi*, *C++ Builder*, entre otros .

- *Bibliotecas de clases.* Ya que es preciso desarrollar dichas bibliotecas de clases, lo que obliga al usuario a tomar conocimientos de las mismas.

- *Lentitud.* La ejecución de un programa orientado a objetos es más lenta

Pero a pesar de estas desventajas la Orientación a Objetos es y será una excelente opción para la creación de sistemas computacionales de calidad y excelente presentación. Para ahondar un mayormente en el Análisis y Diseño se recomienda leer el Apéndice A o remitirse a las referencias bibliográficas.

CAPÍTULO V

Sistema Computalizado para el Cálculo de Parámetros de Líneas de Transmisión

El software para esta aplicación se denomina CPLT: Cálculo de Parámetros en Líneas de Transmisión, este software obtiene los parámetros de líneas de transmisión en circuitos trifásicos simples ó dobles con ó sin cables de guarda.

Este software permite en primera instancia a los estudiantes y profesores de nivel superior utilizar la ventaja de la computadora digital para el análisis de líneas de transmisión reduciendo tiempo en procesos de cálculo grandes.

5.I Especificaciones técnicas.

Se tienen especificaciones técnicas para que el programa cumpla con el funcionamiento optimo para el cuál fue diseñado, se necesitan consideraciones propias tanto de la computadora que se vaya a utilizar, como en el software de aplicación.

5.1.1 Sobre el equipo de cómputo.

Se recomienda para el equipo de cómputo:

- Procesador 386 ó superior.
- 4 Mb de memoria RAM.
- 4 Mb de espacio en disco duro.
- Windows 3.1 ó superior.
- Impresora compatible.

5.1.2 Sobre los datos a introducir.

Se tienen consideraciones solamente en el tipo de dato a introducir y específicamente en las unidades métricas que se utilizan.

Los datos requeridos para que el programa realice los cálculos son.

- Frecuencia Hz.
- Resistencia del conductor R_conductor en ohms/km.
- Resistencia del cable de guarda Rg_guarda en ohms/km.
- Resistividad de la tierra ρ.
- Radio medio geométrico del conductor RMG en m.
- Radio medio geométrico del cable de guarda RMG_guarda en cm.
- Posición X y posición Y del conductor dada en m.
- Posición X y posición Y de los cables de guarda dada en m (en caso de ser necesarios).

Los resultados que se obtienen son

- Impedancia de secuencia cero.
- Impedancia de secuencia positiva y negativa.

Los datos de salida se muestran tanto en forma rectangular como en coordenadas cartesianas, así también como en ohm/km. u ohm /milla.

5.2 Manual de usuario.

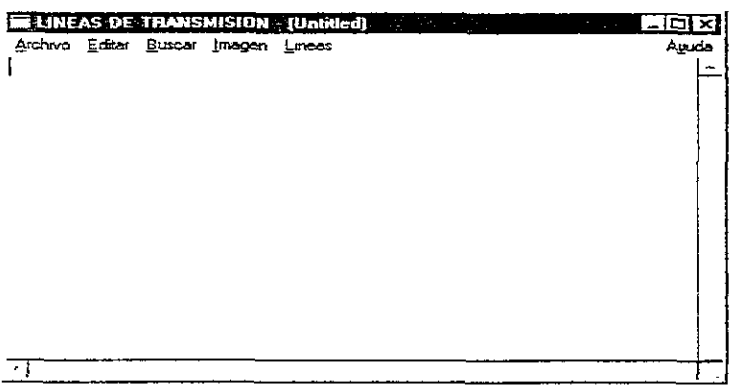
El programa calcula las impedancias de secuencia positiva, negativa y cero para los diferentes circuitos que se tienen actualmente en estudio. El programa ofrece características importantes en pantalla que hacen de este una herramienta muy amigable y versátil, al proporcionar un manejo de archivos con los datos de entrada y de salida de una manera ágil y sencilla.

5.3 Impedancias de secuencia positiva, negativa y cero.

La impedancia representa una oposición al paso de la corriente eléctrica. Las impedancias de secuencia positiva negativa y cero permiten representar la impedancia total de un sistema trifásico senoidal desequilibrado, como la suma de tres sistemas monofasicos. Un sistema de secuencia positiva, un sistema de secuencia negativa, y un sistema de secuencia cero.

5.4 Pantalla.

La pantalla principal ó de ejecución del programa ofrece características interesantes basadas en menús de cortina.



5.4.1 Archivo.

La opción de manejo de archivos, ofrece varias opciones para el usuario: nuevo archivo, abrir archivo, salvar, salvar como... y salir, al igual como aparecen en la pantalla de ejecución de la mayoría de las aplicaciones actuales, esto sirve para la recuperación ó manipulación del archivo de cualquier unidad de lectura y para salvar un archivo existente.

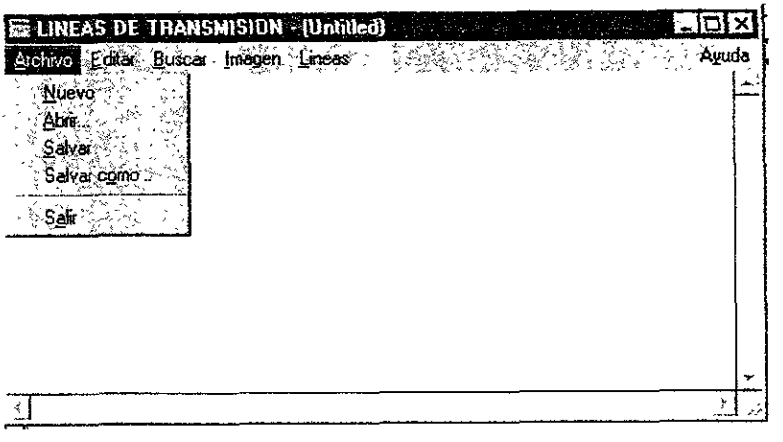
Nuevo archivo: Permite crear un archivo nuevo, con un nombre nuevo para un sistema nuevo.

Abrir archivo: Permite abrir un archivo existente ya sea de entrada (archivo.dat), de salida (archivo. clt) ó de texto, para recuperarlo y modificarlo si se requiere.

Salvar: Guarda un archivo al que ya se le ha dado un nombre previamente, ó pide el nombre si no se le ha asignado alguno.

Salvar como: Guarda un archivo nuevo asignando el nombre deseado.

Salir: Sale de la pantalla de aplicación y pregunta antes si el usuario quiere terminar con la sesión.



5.4.2 Editar.

La opción de editar Deshacer, Cortar, Copiar, Pegar, Borrar y limpiar todo; Permite la modificación del archivo existente de salida ó para editar algún archivo de texto que se requiera crear, la pantalla de ejecución permite esta utilidad independientemente de la aplicación principal.

Deshacer: Destruye las ultimas modificaciones que se hayan realizado en la pantalla de aplicación, sea un archivo de salida, de entrada, una palabra ó un texto.

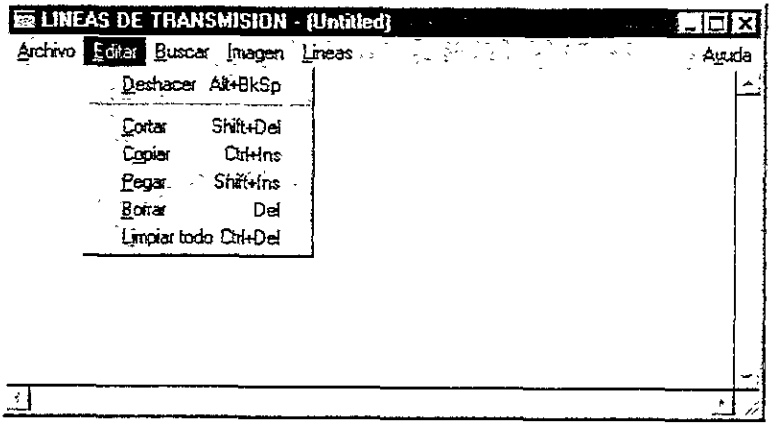
Cortar: Elimina lo que se haya seleccionado previamente mediante el mouse.

Copiar: Copia algún texto ó palabra que se haya seleccionado previamente con el mouse, sin eliminarlo de su lugar original.

Pegar: Pega en cualquier parte, algún texto ó palabra que se haya seleccionado previamente mediante el mouse.

Borrar: Elimina totalmente algún texto ó palabra que se haya escrito en la pantalla de aplicación

Limpiar todo: Limpia toda la pantalla de aplicación de una forma general.



5.4.3 Buscar.

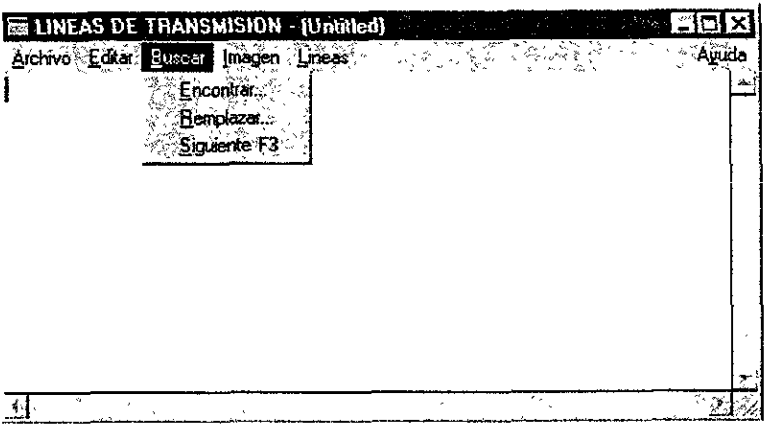
La opción buscar: Encontrar, remplazar y siguiente (F3), permite encontrar palabras en específico y remplazar alguna palabra por otra que se requiera, de una forma automática. Lo anterior será en cualquier archivo de salida o cualquier archivo de texto.

Encontrar: Encuentra una palabra que se requiera, basta con escribir la palabra y oprimir aceptar para que el sistema la busque y la encuentre de forma automática.

Remplazar: Reemplaza una palabra por otra, solo se debe escribir la palabra a cambiar y la palabra por la cuál se va a sustituir. Se tiene algunas opciones para

casos específicos como: caso sensitivo para una búsqueda minuciosa, todos los cambios en donde se cambian todas las palabras existentes automáticamente y preguntar al remplazar en donde se pregunta si se quiere realizar el cambio de palabra en cada ocasión que se encuentre una de ellas.

Siguiente (F3): Esta opción permite al usuario avanzar en cada momento palabra por palabra e ir realizando un cambio a la vez.



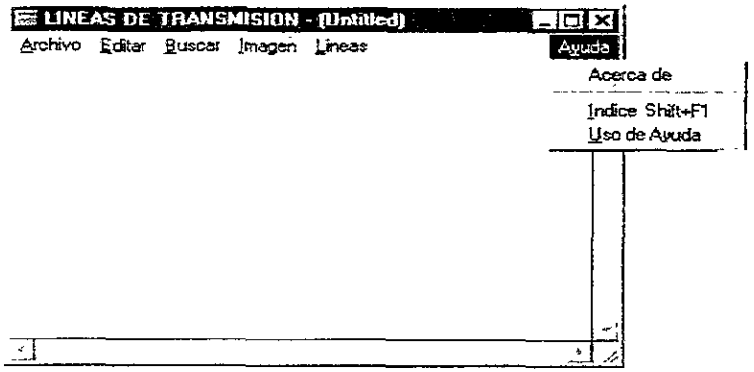
5.4.4 Ayuda.

La opción de ayuda: acerca de, Index, using help, presenta información propia de la elaboración del programa y da el acceso a la ayuda general de windows.

Acerca de: Proporciona la identificación del programa mediante el nombre del mismo, nombre de los creadores del programa y fecha de su elaboración.

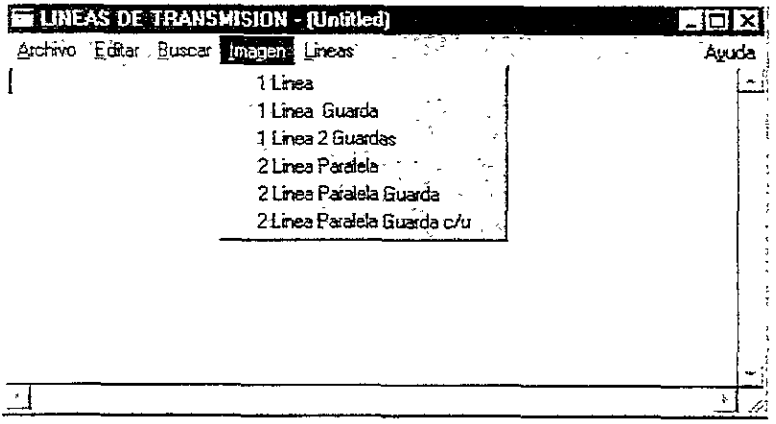
Index: Presenta el índice para la ayuda propia del programa.

Using Help: Llama al índice de la ayuda de windows.



5.4.5 Imagen.

La opción de imagen proporciona la imagen de los diferentes circuitos que se analizan así como la indicación de sus coordenadas.

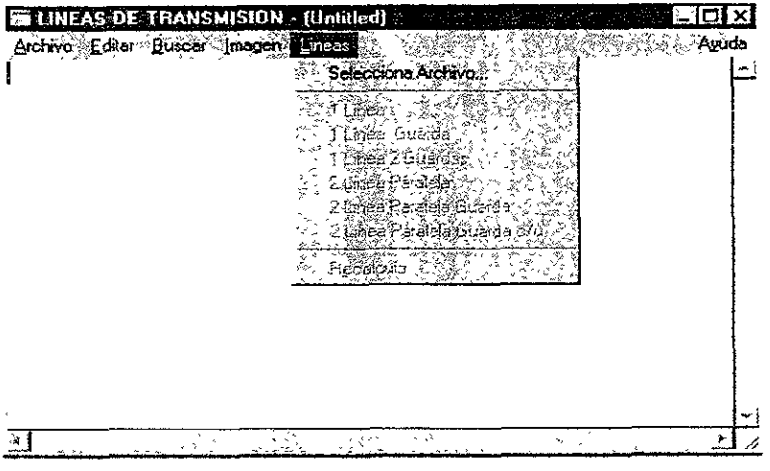


5.4.6 Líneas.

La opción más importante para el programa es la denominada Líneas, esta opción además de involucrar la opción seleccionar archivo en donde se manejaran los datos de entrada y salida del sistema, permite tener acceso a las diferentes topologías de sistemas que se pueden analizar.

Seleccionar archivo: Esta opción permite elegir un archivo de datos existente ó crear un archivo de datos nuevo. En el archivo seleccionado se introducirán los parámetros del sistema.

Recalcúla: La opción final es la llamada Recalcúla, es de las importantes, ya que permite introducir nuevos valores y analizar las impedancias de secuencia positiva, negativa y cero de los circuitos las veces que sea necesario, dando la posibilidad de nuevos análisis con características de parámetros distintas.



5.4.6.1 Sistemas a analizar.

1 Circuito Trifasico Sin Cables de Guarda.
1 Circuito Trifasico con un cable de guarda.
1 Circuito Trifasico con dos cables de guarda.
2 Circuitos Trifásicos en paralelo sin cables de guarda.
2 Circuitos Trifásicos en paralelo con dos cables de guarda.
2 Circuitos Trifásicos en paralelo con dos cables de guarda por circuito.

5.4.6.1.1 Un Circuito Trifasico Sin Cables de Guarda.

Este circuito permite la obtención de las impedancias de secuencia positiva, negativa y cero para un circuito trifásico sin cables de guarda, los datos de entrada que serán introducidos son: Resistencia del conductor, frecuencia de operación, resistividad de la tierra, radio medio geométrico del conductor (RMG_C), y las coordenadas de ubicación de los conductores.

5.4.6.1.2 Un Circuito Trifasico con un cable de guarda.

Este circuito permite la obtención de las impedancias de secuencia positiva, negativa y cero para un circuito trifásico con un cable de guarda, los datos de entrada que serán introducidos son: Resistencia del conductor, resistencia del cable de guarda, frecuencia de operación, resistividad de la tierra, radio medio geométrico del conductor (RMG_C), radio medio geométrico del cable de guarda coordenadas de ubicación de los conductores y del cable de guarda.

5.4.6.1.3 Un Circuito Trifasico con dos cables de guarda.

Este circuito permite la obtención de las impedancias de secuencia positiva, negativa y cero para un circuito trifásico con dos cables de guarda, los datos de entrada que serán introducidos son: Resistencia del conductor, resistencia de los cables de guarda, frecuencia de operación, resistividad de la tierra, radio medio geométrico del conductor (RMG_C), radio medio geométrico de los cables de guarda, coordenadas de ubicación de los conductores y de los dos cables de guarda (gx ,gy).

5.4.6.1.4 Dos Circuitos Trifásicos en paralelo sin cables de guarda.

Este circuito permite la obtención de las impedancias de secuencia positiva, negativa y cero para dos circuitos trifásicos en paralelo sin cables de guarda, los datos de entrada que serán introducidos son: Resistencia de los conductores, frecuencia de operación, resistividad de la tierra, radio medio geométrico de los conductores (RMG_C), y coordenadas de ubicación de los conductores (x1,y1) y (x11,x22) respectivamente.

5.4.6.1.5 Dos Circuitos Trifásicos en paralelo con dos cables de guarda.

Este circuito permite la obtención de las impedancias de secuencia positiva, negativa y cero para dos circuitos trifásicos en paralelo con dos cables de guarda, los datos de entrada que serán introducidos son: Resistencia de los conductores, resistencia de los cables de guarda, frecuencia de operación, resistividad de la tierra, radio medio geométrico de los conductores (RMG_C), radio medio geométrico de los cables de guarda, así como las coordenadas de ubicación de los conductores de ambos circuitos (x1 ,y1) y (x11,x22) y de los cables de guarda de ambos circuitos respectivamente (gx,gy), (gx1,gy1).

5.4.6.1.6 Dos Circuitos Trifásicos en paralelo con dos cables de guarda cada uno.

Este circuito permite la obtención de las impedancias de secuencia positiva, negativa y cero para dos circuitos trifásicos en paralelo con dos cables de guarda por circuito, los datos de entrada que serán introducidos son: Resistencia de los conductores, resistencia de los cables de guarda, frecuencia de operación, resistividad de la tierra, radio medio geométrico de los conductores (RMG_C), radio medio geométrico de los cables de guarda, además de las coordenadas de ubicación de los conductores de los dos circuitos (x1 ,y1) y (x11,x22) y de los cables de guarda (gx,gy), (gx1,gy1), (gx2,gy2) y (gx3,gy3) respectivamente.

5.5 Coordenadas de ubicación

Las coordenadas de ubicación de los conductores indican la distancia que tienen estos hacia dos ejes ficticios, dibujados uno de forma horizontal y el otro de forma vertical respectivamente, estos dos ejes permiten agilizar el proceso de captura de forma más eficiente, dando lugar a un marco de referencia de distancias, que facilite y generalice para esta aplicación la obtención de los valores del radio medio geométrico RMG y de la distancia media geométrica DMG, de los distintos conductores ó cables de guarda involucrados en el sistema en estudio.

5.5.1 Eje horizontal.

El eje horizontal indica la referencia de coordenadas para los conductores referidos a tierra. Las coordenadas horizontales de sistemas con un circuito se definen con subíndice sencillo (x1,x2,x3...), y los sistemas con 2 circuitos tendrán al segundo circuito identificado con subíndice doble (x11,x22,x33...).

5.5.2 Eje vertical.

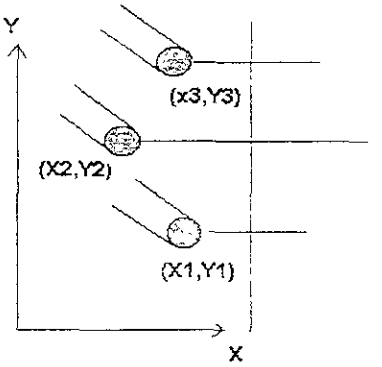
El eje vertical indica la referencia de las coordenadas de los conductores referida a un eje especifico definido por el usuario. Al igual que en el eje horizontal, los sistemas con un circuito se definen con subíndice sencillo (y1,y2,y3...), y los sistemas con 2 circuitos tendrán al segundo circuito identificado con subíndice doble (y11,y22,y33...).

En el caso de las coordenadas de referencia para los cables de guarda, estos quedaran definidos para el eje horizontal con: (gx, gx1, gx2...), y para el eje vertical con (gy, gy1, gy2, .), creando así parejas de coordenadas de la forma (gx,gy) para el primer cable de guarda, y (gxn, gyn) para el enésimo cable de guarda.

En términos generales la ubicación de cada conductor quedara definida por un conjunto de dos coordenadas, horizontal y vertical (Xn,Yn).

Este gráfico muestra la forma en que se piden los datos de las coordenadas en la ventana de captura de los distintos sistemas.

Eje vertical



Eje horizontal

5.6 Datos de entrada.

Se elige la opción seleccionar archivo del menú Líneas, al aparecer la ventana creamos un archivo con el nombre que deseamos (archivo.dat) en cualquier directorio elegido por el usuario (a, b, c, etc.) y oprimimos la tecla aceptar. Este archivo nuevo es el que almacenara los parámetros del sistema, y el cuál será llamado posteriormente.

El almacenar los datos de cualquier sistema eléctrico, da la flexibilidad de manejar posteriormente este archivo en distintas ocasiones, sin necesidad de volverlo a crear, reduciendo tiempo y minimizando proceso de computo.

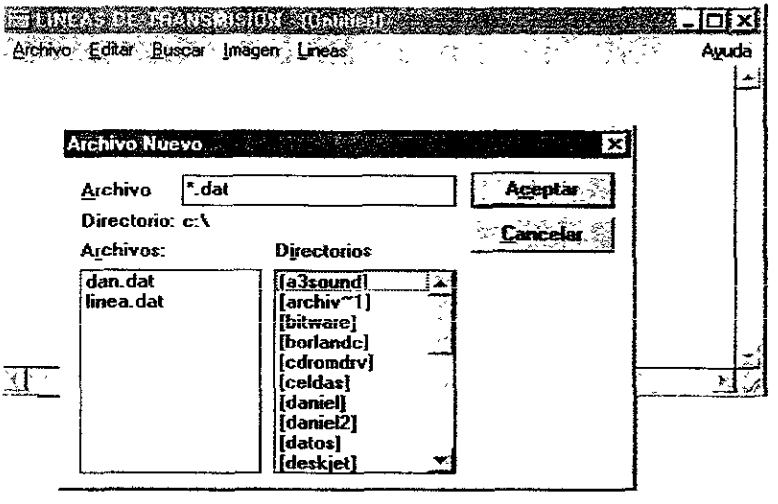
Al tener el archivo creado, seleccionamos el menú Líneas de la pantalla de aplicación y seleccionamos el modelo del sistema eléctrico que se desee analizar.

En los diferentes campos de captura que aparecen en la caja de dialogo, se especifica el tipo de dato a introducir, los valores serán de resistencia, radio medio geométrico y coordenadas de ubicación de los conductores e hilos de guarda, así como la frecuencia de operación del sistema, además de que se tiene la opción de suspender el proceso en cualquier momento al oprimir el botón cancelar.

Al terminar de insertar los valores de la manera correcta y oprimir aceptar, la pantalla de captura desaparece y en ese momento se regresa a la pantalla principal que se encuentra en blanco si es que no se ha ejecutado algún ejercicio previamente.

En este paso los datos del sistema ya se encuentran almacenados en el archivo que se creo al principio, el cuál tiene un nombre asignado y una

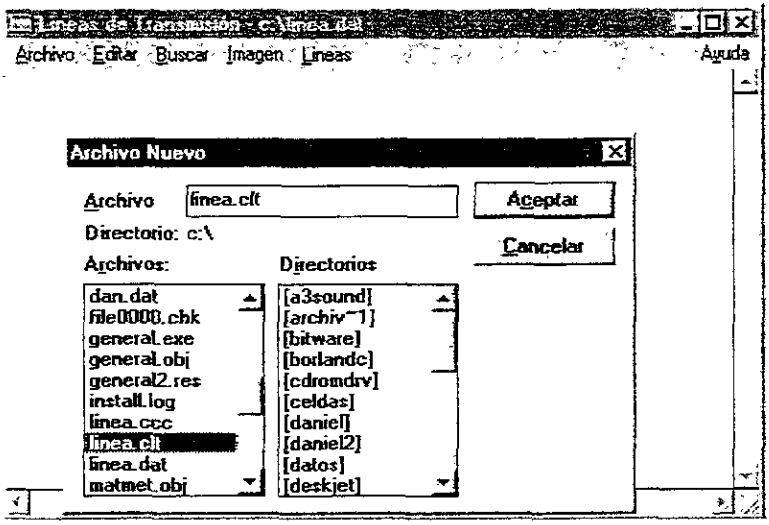
terminación (.dat). Desde este momento el programa inicia los cálculos correspondientes para obtener los valores de las impedancias del sistema.



5.7 Datos de salida.

Cuando el programa termina de realizar los cálculos, los datos de salida con los valores de las impedancias, se encuentran en un archivo de salida localizado en la dirección C ó la que el usuario especifique. Para observar el archivo de salida, seleccionamos en la pantalla principal la opción archivo, elegimos abrir archivo y aparecerá la caja de dialogo archivo nuevo, buscamos en la ruta adecuada y localizamos el archivo de salida, el cuál tiene el mismo nombre del archivo de datos de entrada pero terminación (archivo.clt).

El archivo de salida muestra los valores de las impedancias del sistema, tanto en forma compleja a+jb, como en forma fasorial, módulo y ángulo.

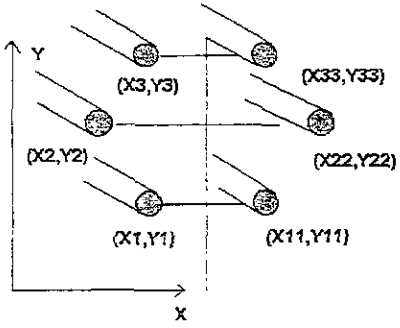


5.8 Captura.

En este apartado ejemplificaremos la forma mediante la cuál se capturan los datos en la pantalla de aplicación del programa CPLT, para esto utilizaremos un sistema de líneas de transmisión formado por dos circuitos Trifasico paralelos sin cables de guarda, de esta forma los diferentes circuitos eléctricos que se estudiaran en esta aplicación, seguirán la misma secuencia en la captura de los datos que el ejemplo.

5.8.1 Ejemplo.

Dos circuitos trifásicos en paralelo sin cables de guarda.



Parámetros de los conductores:

Diámetro de cada conductor = 2.048 cm.
Resistencia de cada conductor = 0.146 ohms/km.
Frecuencia del sistema = 60 Hz.
Resistividad del terreno (p) = 100 ohms/m/m².

Este circuito tiene 6 conductores, 3 por circuito y cada uno tiene un par de coordenadas que le dan posición con respecto a los ejes de referencia (xn,yn). La referencia con el eje horizontal es **x** y se considera la tierra física ó suelo, y la referencia con el eje vertical es **y**, considerando un eje ficticio vertical.

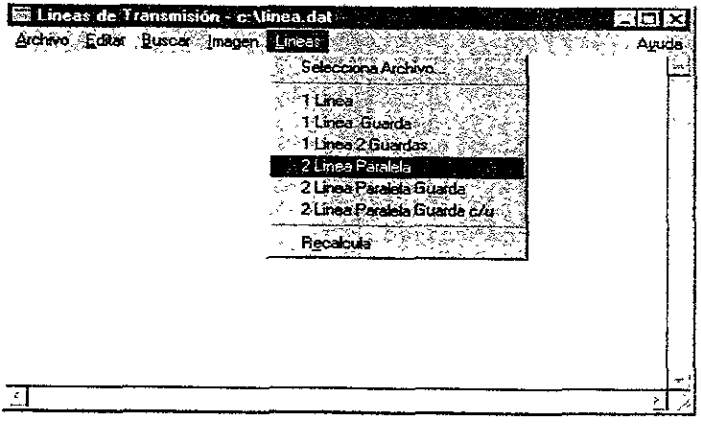
5.8.2 Pasos a seguir para introducir los datos en la pantalla de aplicación:

5.8.2.1 Obtener las coordenadas de ubicación de los diferentes conductores.

El circuito 1 esta formado por los conductores a,b y c.
Coordenadas de a = (x1,y1).
Coordenadas de b = (x2,y2).
Coordenadas de c = (x3,y3).

El circuito 2 esta formado por los conductores a1, b1 y b2.
Coordenadas de a1 = (x11,y11).
Coordenadas de b1 = (x22,y22).
Coordenadas de c1 = (x33,y33).

1) Estando en la pantalla de aplicación, elegimos el menú Líneas, seleccionamos archivo y elegimos la opción **2 Línea Paralela,** en este momento aparecerá la ventana de captura con los diferentes campos.



2) Al aparecer la pantalla de captura para este sistema con 2 Circuitos Trifásicos en Paralelo, verificamos que la pantalla sea la correcta, tenemos 16 campos en los cuales se vaciaran los datos que se soliciten.

5.8.2.2 Introducir los parámetros del sistema.

Se sugiere introducir en primera instancia los parámetros propios del sistema y de los conductores.

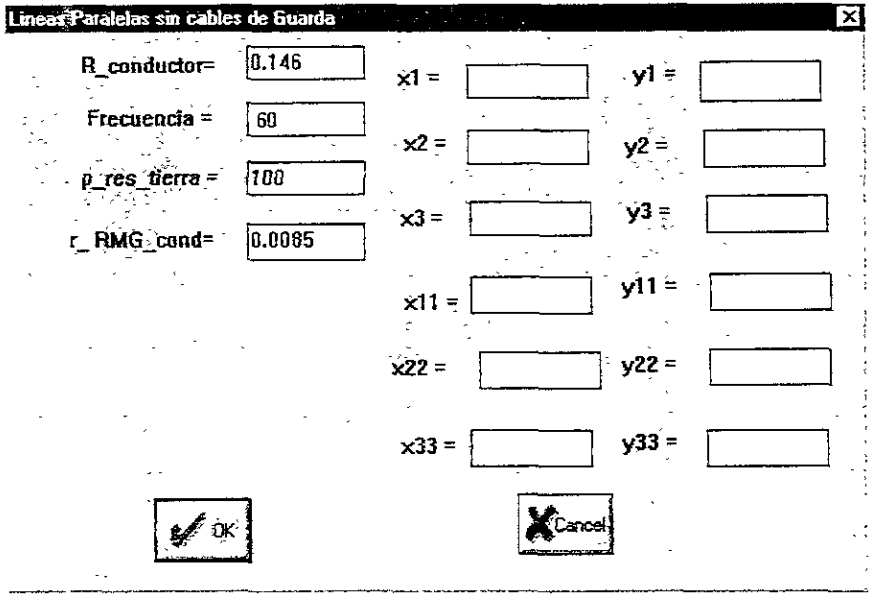
1.- Diámetro de cada conductor = 2 048 cm.

En este caso se toma el valor del radio medio geométrico que es 0.85 cm, lo pasamos a metros y se introduce en el campo denominado **r_RMG_Cond.**

2.- Resistencia de cada conductor = 0.146 ohms/km.
Se introduce en el campo denominado **R_Conductor.**

3.- Frecuencia del sistema = 60 Hz:
Se introduce en el campo denominado **Frecuencia.**

4.- Resistividad del terreno (p) = 100 ohms/m/m²:
Se introduce en el campo denominado **p_res_tierra.**



5.8.2.3 Introducir datos de coordenadas.

Al terminar de introducir los parámetros propios del sistema y de los conductores procedemos a introducir los valores propios de las coordenadas de los conductores y si el sistema lo requiere también de los cables de guarda.

Considerando para este caso el conductor b en el origen del sistema de referencia y las distancias en metros(m), tenemos que.

1.- Coordenadas de a = (x1,y1)=(.915, 13.716):
Introducimos los valores respectivamente en los campos denominados x1 y y1.

2.- Coordenadas de b = (x2,y2)=(0, 16 154)
Introducimos los valores respectivamente en los campos denominados x2 y y2.

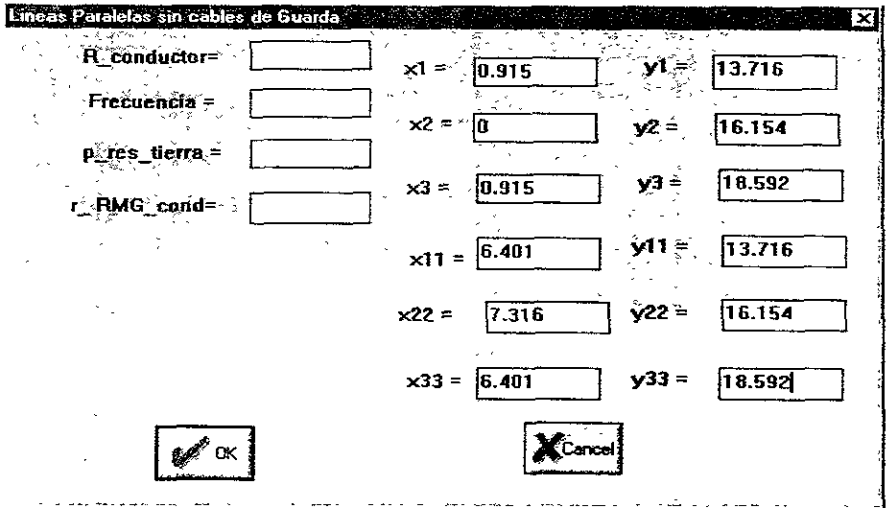
3.- Coordenadas de c = (x3,y3) = (.915, 18.592)
Introducimos los valores respectivamente en los campos denominados x3 y y3.

El circuito 2 esta formado por los conductores a1, b1 y b2.

4 - Coordenadas de a1 = (x11,y11) = (6.401,13.716)
Introducimos los valores respectivamente en los campos denominados x11 y y11.

5.- Coordenadas de b1 = (x22,y22) = (7.316,16.154).
Introducimos los valores respectivamente en los campos denominados x22 y y22.

6.- Coordenadas de c1 = (x33,y33) = (6.401, 18.592).
Introducimos los valores respectivamente en los campos denominados x33 y y33.



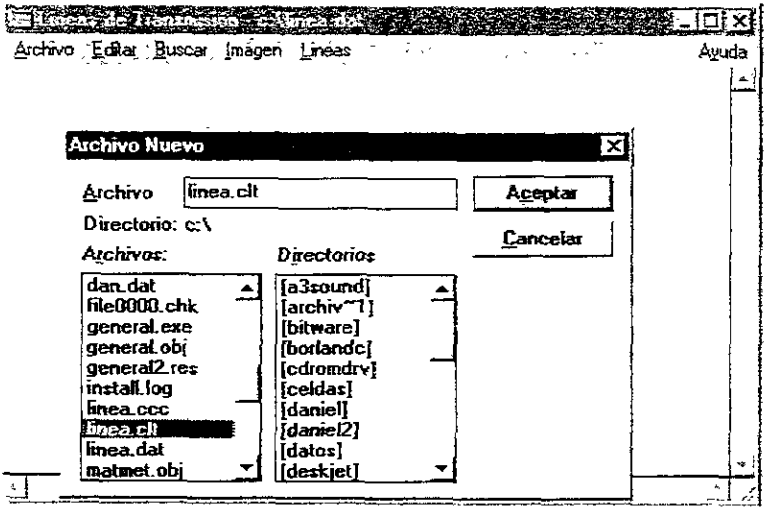
5.8.2.4 Aceptación de resultados.

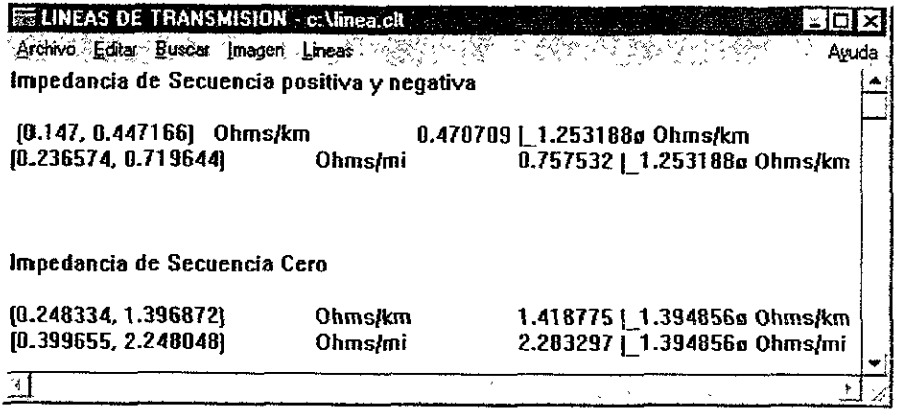
Cuando se termine de introducir los valores que se soliciten en los diferentes campos de captura de la forma correcta y no se tenga ninguna duda al respecto, se oprime la opción **OK** de la pantalla de captura, esta opción esta caracterizada con una paloma verde.

Inmediatamente después el sistema procede a realizar los cálculos para la obtención de la impedancia de la línea de transmisión. En caso de requerir alguna corrección o se tiene alguna duda en el proceso al introducir algún valor, se oprime la opción **CANCEL** de la pantalla de captura, caracterizada por una cruz roja.

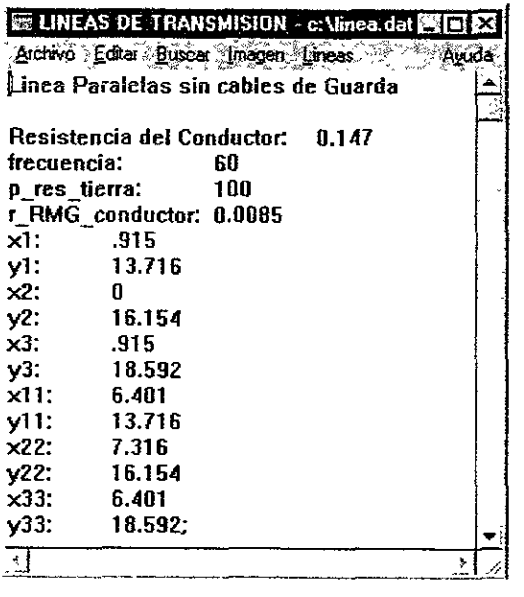
5.8.2.5 Obtención y observación de resultados.

Para poder observar los valores de la impedancia, elegimos el menú archivo en la pantalla de aplicación, seleccionamos abrir, buscamos en la ruta que el usuario eligió para guardar el archivo ó en la unidad C y abrimos el archivo que tiene el mismo nombre que el archivo de entrada pero terminación (archivo.clt) mediante la tecla OK. Inmediatamente después los valores de la impedancia de secuencia positiva, negativa y cero aparecerán en la pantalla de aplicación.





Salida de Resultados



Archivo de datos

CAPÍTULO VI

Sistema Computalizado para el Cálculo de Corto Circuito en S.E.P.

El software para esta aplicación se denomina CCC (Cálculo de Corto circuito), este software calcula las corrientes de corto circuito en los diferentes buses que conforman los diferentes sistemas eléctricos de potencia.

Este software permitirá en primera instancia a los estudiantes y profesores de nivel superior utilizar las ventajas de la computadora digital para el análisis del corto circuito en sistemas eléctricos de potencia, reduciendo tiempo en procesos que requieren de cálculos extensos.

6.1 Especificaciones técnicas.

Se tienen especificaciones técnicas para que el programa cumpla con el funcionamiento optimo para el cuál fue diseñado, se necesitan consideraciones propias tanto de la computadora que se vaya a utilizar, como en el software de aplicación.

En el caso del análisis de corto circuito, se propone para el software, un numero limite de elementos en el sistema eléctrico para un rendimiento total, aunque puede funcionar para un numero "N" de elementos conectados al sistema.

Las especificaciones técnicas también involucran la forma en que los valores de los parámetros se deben de introducir en la pantalla de captura, así como las unidades métricas que deben de tener estos, de igual manera se menciona la forma en que los resultados finales son mostrados.

6.1.1 Sobre el equipo de cómputo.

Se requiere para el equipo de cómputo:

- Procesador 386 ó superior.
- 4 Mb de memoria RAM.
- 4 Mb de espacio en disco duro.
- Windows 3.1 ó superior.
- Impresora compatible.

6.1.2 Sobre los datos a introducir

En el caso del estudio de corto circuito, los datos que se introducirán serán los referentes a los diferentes elementos que conforman el sistema eléctrico que se este analizando.

Los datos que se deben de suministrar además del voltaje base y de la potencia base, son los siguientes para cada elemento.

Para los generadores

- Potencia en MVA.
- Voltaje en KV.
- Número de bus al que esta conectado.
- Impedancias de secuencia positiva, negativa y cero.

Para los motores.
- Tipo de potencia Real en W ó Aparente en MVA.
- Factor de potencia si se requiere.
- Voltaje en KV.
- Número de bus al que esta conectado.
- Impedancias de secuencia positiva, negativa y cero.

Para las líneas.
- Número de los dos buses al que esta conectado.
- Impedancias de secuencia positiva, negativa y cero, en forma compleja (a,b).
  a: Parte real.
  b: Parte imaginaria.

Para los transformadores.
- Potencia en MVA.
- Voltaje en el bus de alta tensión en KV.
- Voltaje en el bus de baja tensión en KV.
- Número de bus del lado de alta tensión .
- Número de bus del lado de baja tensión.
- Impedancias de secuencia positiva, negativa y cero.

Los resultados que se obtienen son las corrientes de falla en cada bus.

- Falla línea a tierra.
- Falla línea a línea.
- Falla bifásica a tierra.
- Falla trifasica.

6.2 Manual de usuario.

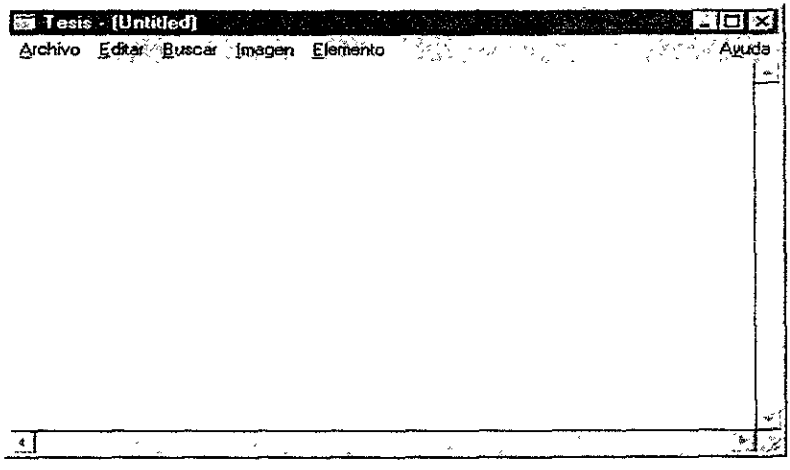
Un corto circuito consiste en la conexión intencionada o no de dos o más conductores que ordinariamente operan con una diferencia de potencial especifica. Esta conexión se produce por contacto fisico entre elementos mecánicos o por medio de un arco, ocasionando corrientes altas que dañan la operación de los elementos del sistema.

La determinación de los valores de corriente y voltaje ocasionados por un corto circuito es de suma importancia para seleccionar los interruptores, así como la aplicación de los relevadores de protección que actuarán durante la falla.

Se analizaran sistemas que posean generadores, transformadores, líneas y motores, introduciendo los parámetros de cada uno, así como el número del bus al que están conectados.

6.2.1 Pantalla.

La pantalla principal ó de ejecución del programa ofrece características interesantes basadas en menús de cortina.



6.2.1.1 Archivo.

La opción de manejo de archivos, ofrece varias opciones para el usuario: nuevo archivo, abrir archivo, salvar, salvar como y salir, al igual que como aparecen en la pantalla de ejecución de la mayoría de las aplicaciones actuales, esto sirve para la recuperación ó manipulación del archivo de cualquier unidad de lectura y para salvar un archivo existente.

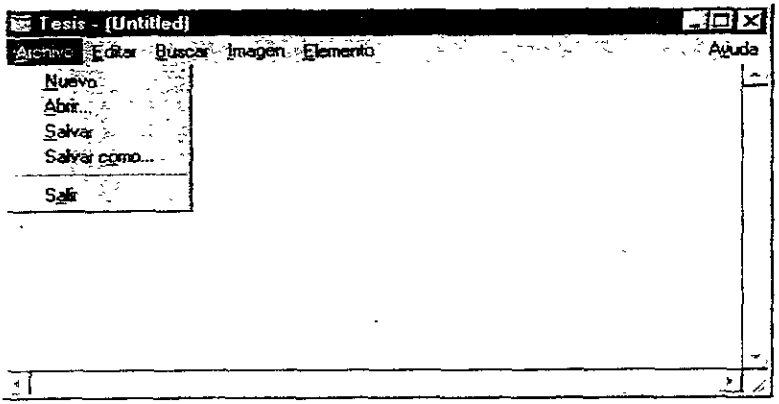
Nuevo archivo: Permite crear un archivo nuevo, con un nombre nuevo para un sistema nuevo.

Abrir archivo: Permite abrir un archivo existente ya sea de entrada (archivo.dat), de salida (archivo.ccc) ó de texto, para recuperarlo o modificarlo.

Salvar: Guarda un archivo al que ya se le ha dado un nombre previamente, o pide el nombre si no se le ha asignado alguno.

Salvar como: Guarda un archivo nuevo asignando el nombre deseado.

Salir: Sale de la pantalla de aplicación y pregunta antes si el usuario quiere terminar con la sesión.



6.2.1.2 Editar.

La opción de edición o de editar: Deshacer, Cortar, Copiar, Pegar, Borrar y limpiar todo; Permite la modificación del archivo existente de salida o para editar algún archivo de texto que se requiera utilizar, la pantalla de ejecución permite esta utilidad independientemente de la aplicación principal.

Deshacer: Destruye las ultimas modificaciones que se hayan realizado en la pantalla de aplicación, sea un archivo de salida ó de entrada, una palabra ó un texto.

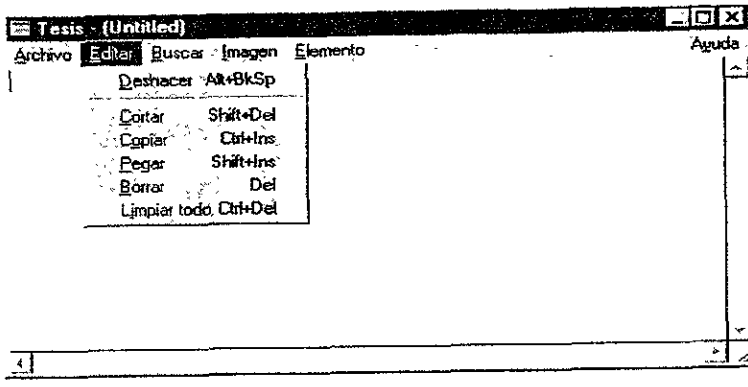
Cortar Elimina lo que se haya seleccionado previamente mediante el mouse.

Copiar: Copia algún texto o palabra que se haya seleccionado previamente con el mouse, sin eliminarlo de su lugar original.

Pegar: Pega en cualquier parte, algún texto ó palabra que se haya seleccionado previamente mediante el mouse.

Borrar: Elimina totalmente algún texto ó palabra que se haya escrito en la pantalla de aplicación.

Limpiar todo: Limpia toda la pantalla de aplicación de una forma general.



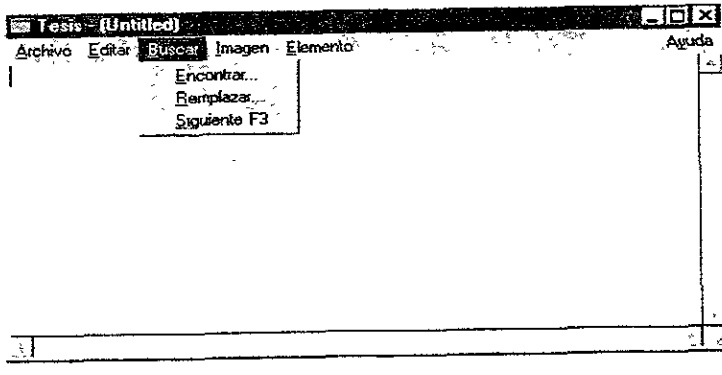
6.2.1.3 Buscar.

La opción Buscar: Encontrar, remplazar y siguiente (F3), permiten encontrar palabras en específico, y así remplazar alguna palabra por otra que se requiera de una forma automática. Lo anterior será en cualquier archivo de salida o cualquier archivo de texto.

Encontrar: Encuentra una palabra que se requiera, basta con escribir la palabra y oprimir aceptar para que el sistema la busque y la encuentre de forma automática.

Remplazar: Reemplaza una palabra por otra, solo se debe escribir la palabra a cambiar y la palabra por la cuál se va a sustituir. Se tiene algunas opciones para casos específicos como: caso sensitivo para una búsqueda minuciosa, todos los cambios en donde se cambian todas las palabras existentes automáticamente y preguntar al remplazar en donde se pregunta si se quiere realizar el cambio de palabra en cada ocasión que se encuentre una de ellas

Siguiente (F3): Esta opción permite al usuario avanzar en cada momento palabra por palabra e ir realizando un cambio a la vez.



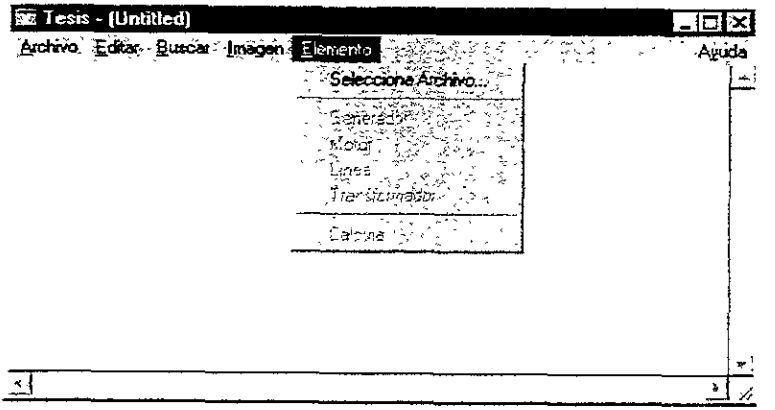
6.2.1.4 Imagen.

La opción de imagen proporciona la imagen de identificación del programa, esta imagen dura unos cuantos segundos.

6.2.1.5 Elemento.

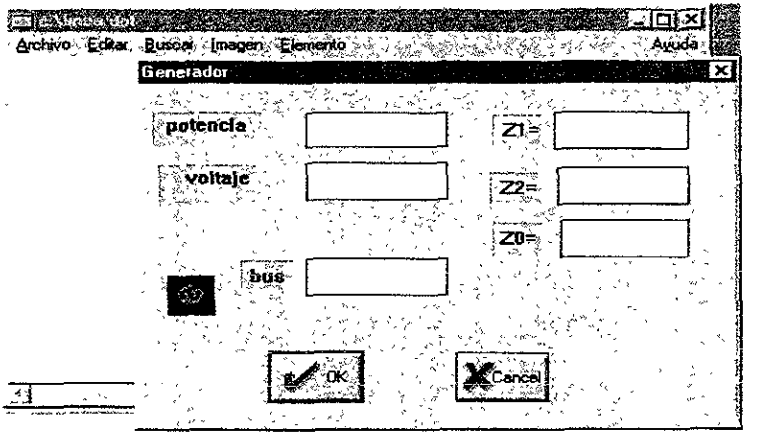
La opción Elemento: Seleccionar archivo, generador, motor, línea y transformador, permite el vaciado de la información de los distintos elementos que conforman el sistema eléctrico.

Seleccionar archivo: Esta opción permite elegir un archivo de datos existente ó crear un archivo de datos nuevo.



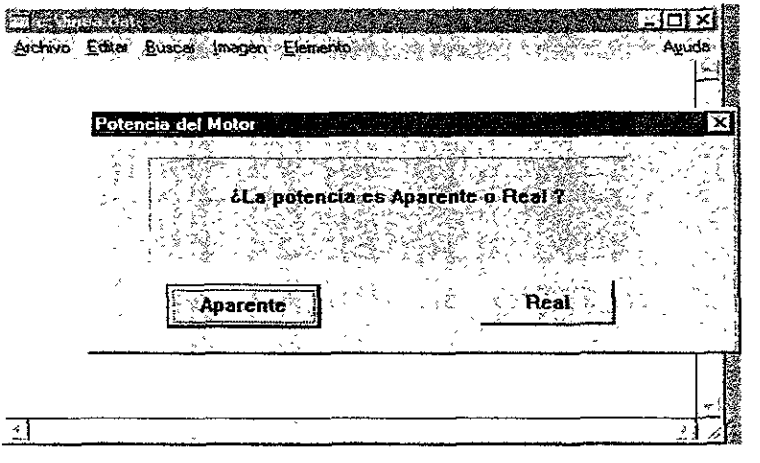
6.2.1.5.1 Generador.

En esta opción se deben de introducir los valores de potencia aparente, voltaje, impedancias de secuencia positiva, negativa y cero, así como el numero del bus al que esta conectado.

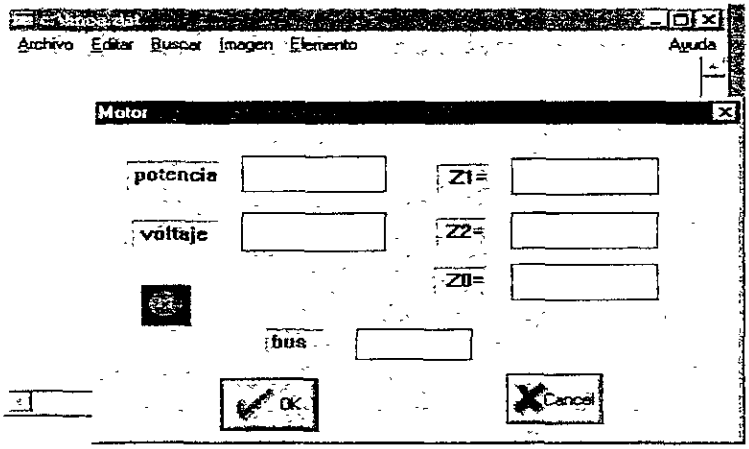


6.2.1.5.2 Motor.

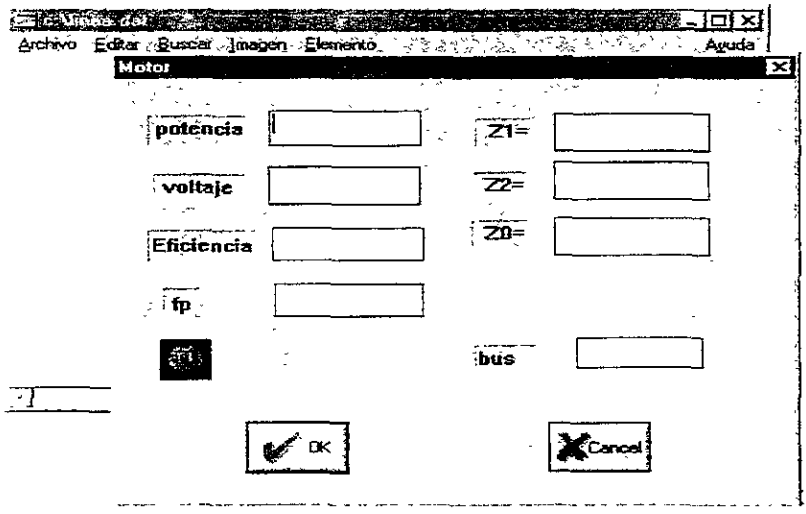
En esta opción se pueden introducir los valores de dos formas distintas.



1) Si se elige la opción en que el motor consume potencia aparente, los datos a introducir son: potencia, voltaje, impedancias de secuencia positiva, negativa y cero, así como el numero de bus al que esta conectado.

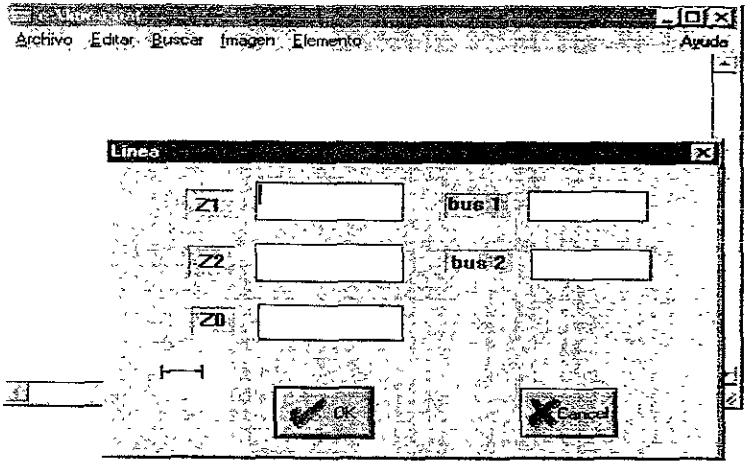


2) Si se elige la opción en que el motor consume potencia real los valores a introducir son: potencia, voltaje, eficiencia, factor de potencia, impedancias de secuencia positiva, negativa y cero, así como el numero del bus al que esta conectado.



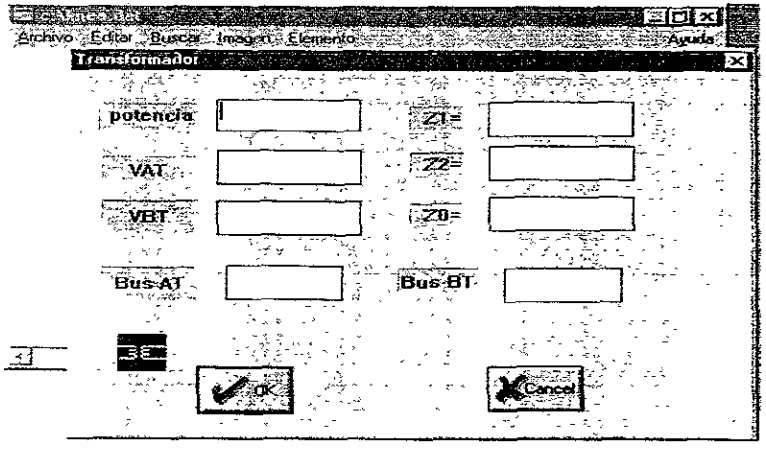
6.2.1.5.3 Línea.

En esta opción introducen los valores de las impedancias de secuencia positiva, negativa y cero entre paréntesis: (parte real, parte imaginaria), así como el numero de los dos buses entre los cuales se encuentra conectada.



6.2.1.5.4 Transformador.

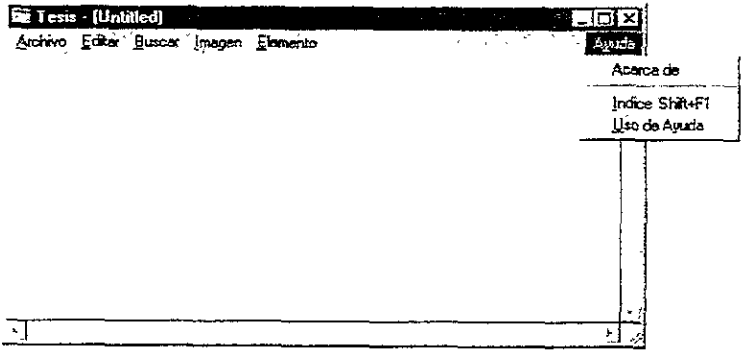
En esta opción se deben de introducir la potencia, el voltaje en el lado de alta tensión, el voltaje en el lado de baja tensión, los valores de las impedancias de secuencia positiva, negativa y cero, así como el numero de los dos buses, el del lado de alta tensión y el del lado de baja tensión.



6.2.1.6 Ayuda.

La opción de ayuda: acerca de, Index, using help, presenta información propia de la elaboración del programa, y da el acceso a la ayuda de windows.

Acerca de: Proporciona la identificación del programa mediante el nombre del mismo, nombre de los creadores del programa, así como la fecha de su elaboración.
Index: Presenta el índice para la ayuda propia del programa.
Using Help: Llama al índice de la ayuda general de windows.



6.3 Datos de entrada.

Se elige la opción seleccionar archivo del menú Elemento, al aparecer la ventana creamos un archivo con el nombre que deseemos (archivo.dat) en cualquier directorio elegido por el usuario (a, b, c, etc.) y oprimimos la tecla aceptar. Este archivo nuevo es el que almacenara los parámetros del sistema, y el cuál será llamado posteriormente.

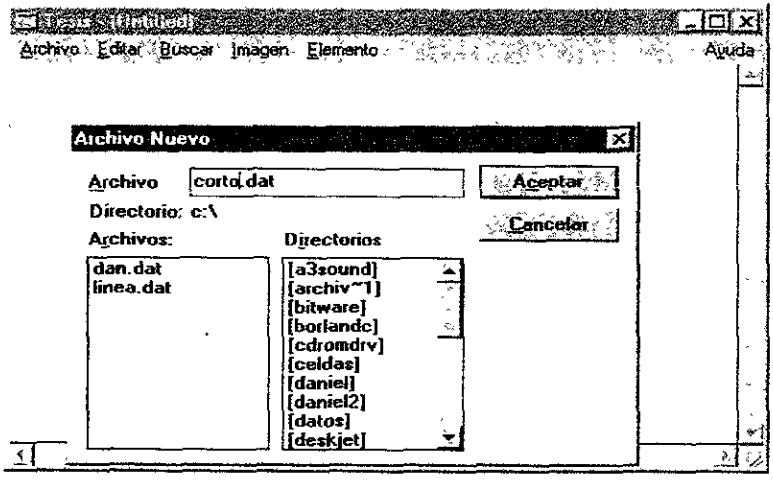
El almacenar los datos de cualquier sistema eléctrico, da la flexibilidad de manejar posteriormente este archivo en distintas ocasiones, sin necesidad de volverlo a crear, reduciendo tiempo y minimizando proceso de computo.

Al tener el archivo creado, elegimos el menú **elemento** de la pantalla de aplicación y seleccionamos alguno de los elementos del sistema, para introducir sus valores propios de operación en los diferentes campos que aparecen, cada campo especifica el tipo de dato a suministrar, además de que se tiene la opción de suspender el proceso al oprimir el botón cancelar.

Al terminar de insertar los valores de la manera correcta y oprimir aceptar, la pantalla de captura desaparece y en ese momento se regresa a la pantalla principal que se encuentra en blanco si es que no se ha ejecutado algún ejercicio previamente.

En este paso los datos del sistema ya se encuentran almacenados en el archivo que se creo al principio, el cuál tiene un nombre asignado y una terminación, ej. corto.dat.

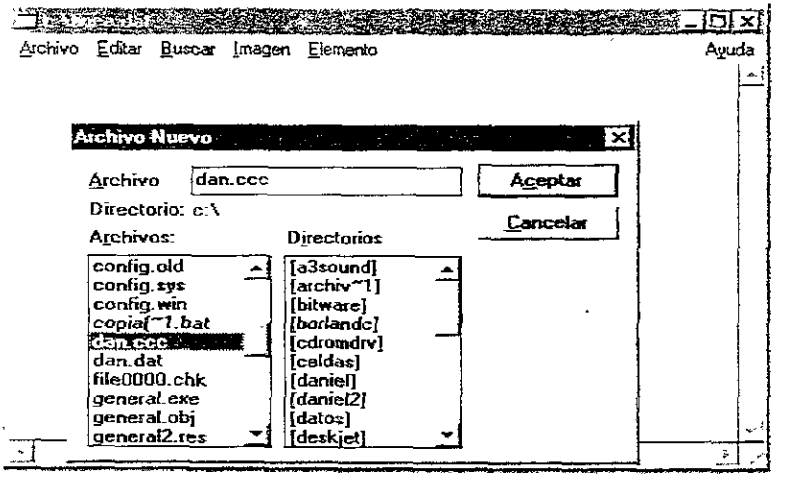
A partir de este momento el programa inicia los cálculos correspondientes para obtener los valores de la corriente y voltaje de corto circuito en los diferentes buses que conforman el sistema eléctrico propuesto



6.4 Datos de salida.

Al terminar el programa de realizar los cálculos, los datos del corto circuito, se encuentran en un archivo de salida localizado en la dirección que el usuario especifique. Para observar el archivo de salida, seleccionamos en la pantalla principal la opción archivo, elegimos abrir archivo y aparecerá la caja de dialogo archivo nuevo, buscamos en la ruta adecuada y localizamos el archivo de salida, el cuál tiene el mismo nombre del archivo de datos de entrada pero terminación (.ccc), eje. corto.ccc

El archivo de salida muestra los valores de los voltajes de corto circuito en los diferentes buses que conforman el sistema eléctrico.



6.5 Captura.

En este apartado ejemplificaremos la forma mediante la cuál se capturan los datos en la pantalla de aplicación del programa ZBUS, para esto utilizaremos un sistema eléctrico constituido por: 1 generador, 1 línea de transmisión, 2 transformadores y 2 motores.
Los sistemas que posean mas elementos de interconexión, seguirán la misma secuencia en la captura de los datos

6.5.1 Ejemplo.

Parámetros del sistema eléctrico:

Generador 1:
13.8 KV.
116.16 MVA.
X1=x2=x0= 0.0086.
BUS 1.

Línea 1:
1000 m.
X1=X2=X3=(0, 0 138)
BUS 1-2.

Transformador 1.
220/127 V
75 KVA.
X1=x2=x3= 0.021
BUS 2-3.

Transformador 2
440/250 V.
500 KVA
X1=x2=x3= 0.038
BUS 2-4

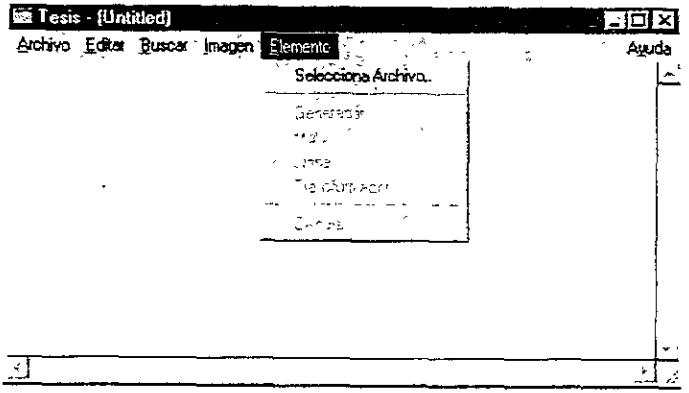
Motor 1:
127 V.
50 KVA
X1=x2=x3= 0.25
BUS 3.

Motor 2.
250 V.
450 KVA
X1=x2=x3= 0.19
BUS 4.

6.5.2 Pasos a seguir para introducir los datos en la pantalla de aplicación.

6.5.2.1 Seleccionar elemento.

Estando en la pantalla de aplicación elegimos el menú **Elemento**, seleccionamos un archivo ya existente ó creamos uno nuevo, posteriormente elegimos otra vez la opción **Elemento** para la selección de los diferentes elementos que forman nuestro sistema: Generador, motor, línea y transformador.



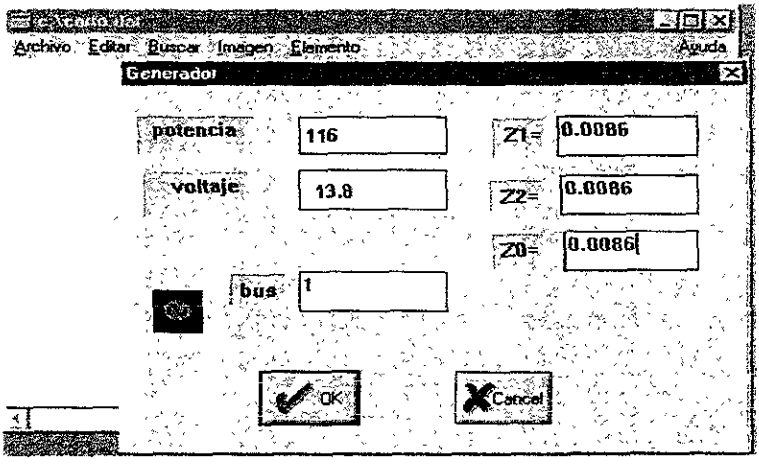
6.5.2.2 Introducir parámetros de Generadores.

Para los datos de los Generadores se tienen 6 campos de captura.

1.- Potencia.
2.- Voltaje.
3.- Impedancia de secuencia positiva, Z1.
4 - Impedancia de secuencia negativa, Z2.
5.- Impedancia de secuencia cero,Z0.
6.- Bus de conexión, BUS.

En este caso tenemos Generador 1:

1.- Potencia = 116.16 MVA.
2.- Voltaje = 13.8 KV.
3.- Impedancia de secuencia positiva, Z1 =0.0086 pu.
4.- Impedancia de secuencia negativa, Z2=0.0086 pu.
5.- Impedancia de secuencia cero,Z0 = 0.0086 pu.
6.- Bus de conexión, BUS = 1.



6.5.2.3 Introducir parámetros de transformadores.

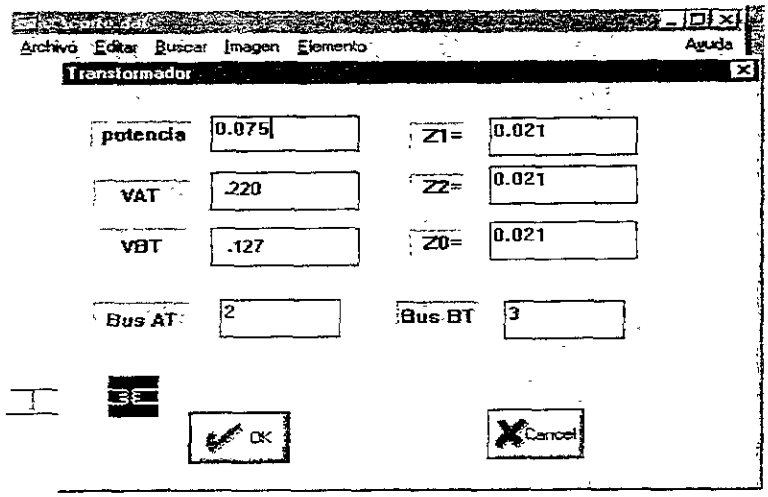
Para los datos de los transformadores se tienen 8 campos de captura.

1.- Potencia.
2.- Voltaje en el lado de alta tensión.
3.- Voltaje en el lado de baja tensión.
4.- Impedancia de secuencia positiva, Z1.
5.- Impedancia de secuencia negativa, Z2.
6.- Impedancia de secuencia cero,Z0.
7.- Bus de conexión en el lado de alta tensión, BUS AT.
8.- Bus de conexión en el lado de baja tensión, BUS BT.

En este caso tenemos.

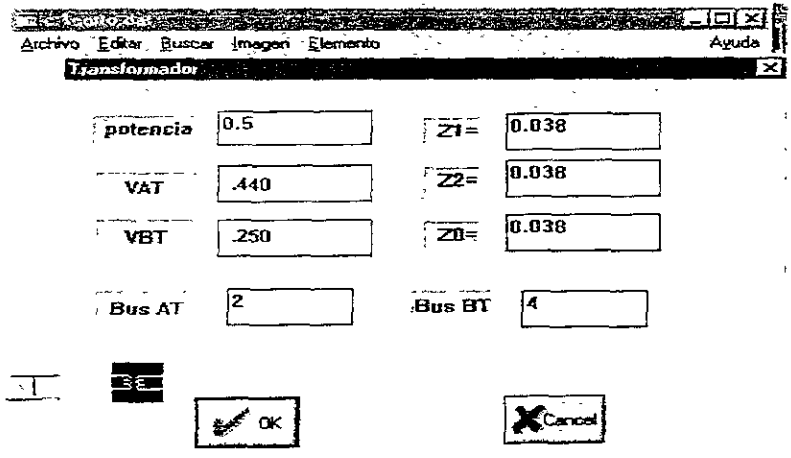
Transformador 1

1.- Potencia = 75 KVA.
2.- Voltaje en el lado de alta tensión = 220 V.
3.- Voltaje en el lado de baja tensión =127 V.
4.- Impedancia de secuencia positiva, Z1=0.021 pu.
5.- Impedancia de secuencia negativa, Z2=0.021 pu.
6.- Impedancia de secuencia cero,Z0=0.021 pu.
7.- Bus de conexión en el lado de alta tensión, BUS AT=2.
8.- Bus de conexión en el lado de baja tensión, BUS BT=3.



Transformador 2:

1.- Potencia = 500KVA.
2.- Voltaje en el lado de alta tensión = 440 V.
3.- Voltaje en el lado de baja tensión = 250 V.
4.- Impedancia de secuencia positiva, Z1 = 0.038 pu.
5.- Impedancia de secuencia negativa, Z2 =0.038 pu.
6.- Impedancia de secuencia cero,Z0 = 0.038 pu.
7.- Bus de conexión en el lado de alta tensión, BUS AT =2.
8.- Bus de conexión en el lado de baja tensión, BUS BT = 4.



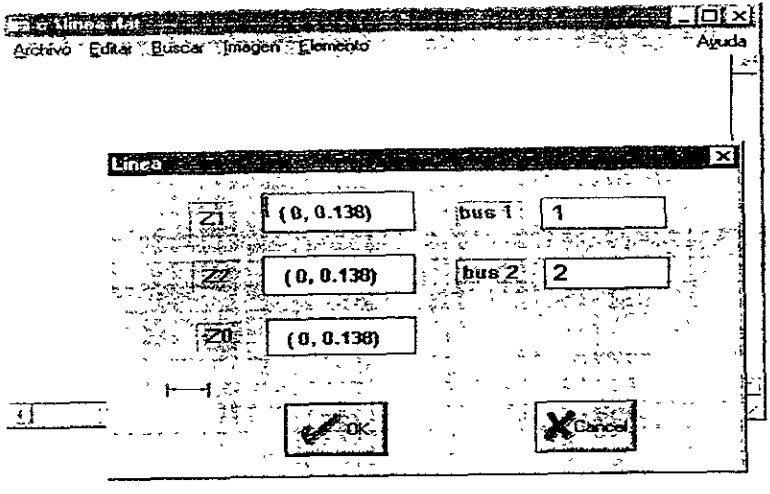
6.5.2.4 Introducir parámetros de las líneas de transmisión.

Para los datos de las líneas se tienen 5 campos de captura.

1.- Impedancia de secuencia positiva, Z1.
2.- Impedancia de secuencia negativa, Z2.
3.- Impedancia de secuencia cero,Z0.
4.- Bus de conexión 1, Bus1.
5.- Bus de conexión 2, Bus 2.

Línea 1:

1.- Impedancia de secuencia positiva, Z1 = 0.138.
2.- Impedancia de secuencia negativa, Z2 =0.138.
3.- Impedancia de secuencia cero,Z0 =0.138
4.- Bus de conexión 1, Bus1=1.
5.- Bus de conexión 2, Bus 2.=2



6.5.2.5 introducir parámetros de motores.

Para los datos de los Motores se tienen 8 campos de captura, si el dato que se pide para la potencia corresponde a la Potencia Real

1.- Potencia
2.- Voltaje.
3.- Eficiencia
4 - Factor de potencia.
3.- Impedancia de secuencia positiva, Z1.

4.- *Impedancia de secuencia negativa*, Z2.
5.- Impedancia de secuencia cero,Z0.
6.- Bus de conexión, BUS.

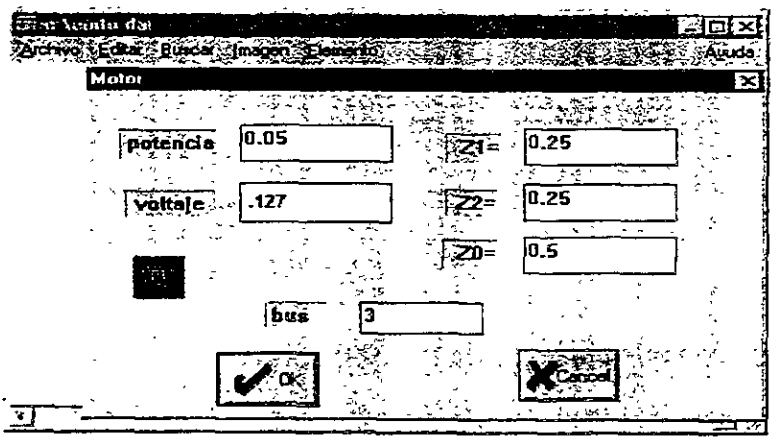
Y se tienen 6 campos de captura, si el dato que se pide para la potencia corresponde a la Potencia Aparente. .

1.- Potencia.
2.- Voltaje.
3.- Impedancia de secuencia positiva, Z1.
4.- Impedancia de secuencia negativa, Z2.
5.- *Impedancia de secuencia cero,Z0.*
6.- Bus de conexión, BUS.

En este caso tenemos que los 2 motores consumen potencia aparente:

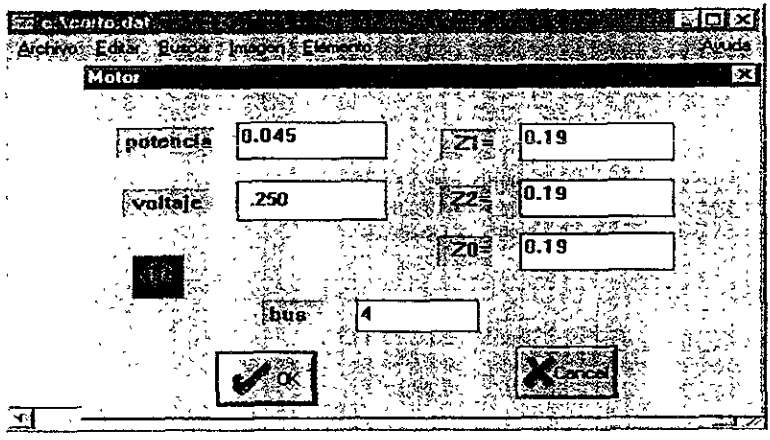
Motor 1:

1.- Potencia = 50 KVA.
2.- Voltaje =127 v.
3.- Impedancia de secuencia positiva, Z1=0.25 pu.
4.- Impedancia de secuencia negativa, Z2 =0.25 pu.
5.- *Impedancia de secuencia cero,Z0 =0.25 pu.*
6.- Bus de conexión, BUS.=3



Motor 2:

1.- Potencia =450 KVA.
2.- Voltaje = 250 V.
3.- Impedancia de secuencia positiva, Z1=0.19 pu.
4.- Impedancia de secuencia negativa, Z2=0.19 pu.
5.- Impedancia de secuencia cero,Z0=0.19 pu.
6.- Bus de conexión, BUS=4.



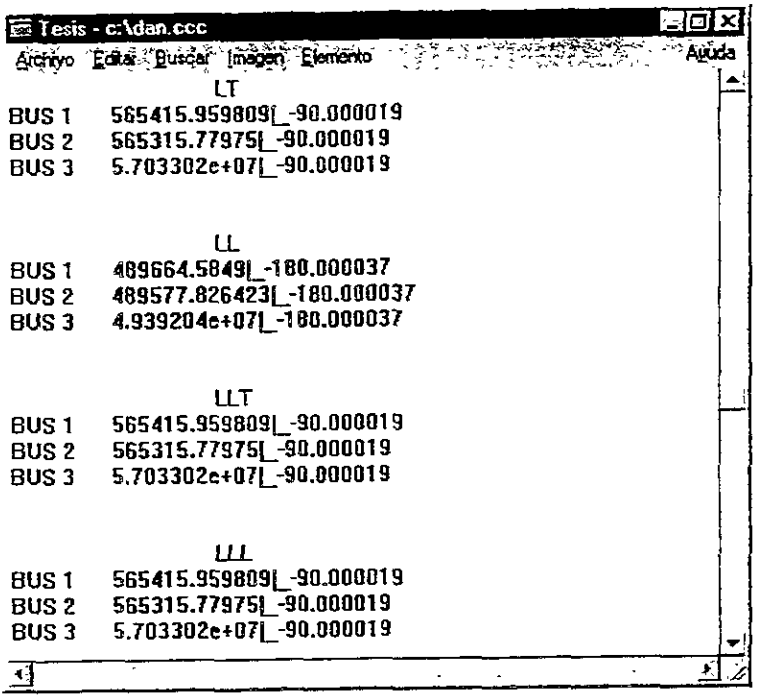
6.5.2.6 Aceptación de resultados.

Al terminar de introducir los parámetros de los diferentes elementos que conforman el sistema en los campos de captura de la forma correcta y no se tiene ninguna duda al respecto, se oprime la opcion **OK** en la pantalla de captura, esta opción esta caracterizada con una paloma verde.

Inmediatamente después el sistema procede a realizar los calculos para la obtención de la valores de corto circuito en todos los buses que tiene el sistema electrico analizado. En caso de requerir alguna corrección o se tiene alguna duda en el proceso al introducir algún valor, se oprime la opción **CANCEL** de la pantalla de captura, caracterizada esta por una cruz roja.

6.5.2.7 Obtención y observación de resultados.

Para poder observar los valores de corto circuito, elegimos el menú archivo en la pantalla de aplicación, seleccionamos abrir, buscamos en la ruta que el usuario eligió para guardar el archivo y abrimos el archivo mediante la tecla OK. Inmediatamente después los valores de corto circuito aparecerán en la pantalla de aplicación.


Tesis - c:\dan.ccc
Archivo Edita Buscar Imagen Elemento Ayuda
LT
BUS 1 565415.959809|_-90.000019
BUS 2 565315.77975|_-90.000019
BUS 3 5.703302e+07|_-90.000019
LL
BUS 1 489664.5849|_-180.000037
BUS 2 489577.826423|_-180.000037
BUS 3 4.939204e+07|_-180.000037
LLT
BUS 1 565415.959809|_-90.000019
BUS 2 565315.77975|_-90.000019
BUS 3 5.703302e+07|_-90.000019
LLL
BUS 1 565415.959809|_-90.000019
BUS 2 565315.77975|_-90.000019
BUS 3 5.703302e+07|_-90.000019

Conclusiones

La industria eléctrica, como otras industrias de energéticos, vive momentos importantes, una muy posible privatización con una entrada de capitales extranjeros y/o nacionales, la preocupación por el uso eficiente de los recursos energéticos, la concientización de los impactos ambientales debido a la generación, transformación, transportación y uso de los energéticos, el cambio al paradigma de un Desarrollo Sustentable, la globalización y la competencia que de ella emana; comprometen al Ingeniero a estar más consciente que nunca de su realidad y de su compromiso con el país y con todo el mundo que lo rodea, teniendo que aprender y dominar la técnica y la ciencia además de conocer la manera "moderna" de como se hacen las cosas.

La computadora nos da un amplia gama de posibilidades en el control de todos los sistemas eléctricos de potencia, algunos autores consideran que el control de la energía eléctrica solo es comparable con la conquista del espacio, debido a las inversiones que se han hecho, la tecnología que se llega a utilizar, la importancia que representa para un mayor avance, etc. Además del control, la computadora nos da la posibilidad de estudiar los sistemas para una mejor compresión de todos los fenómenos que en ellos están involucrados.

La topología de los sistemas eléctricos cada vez es más compleja, los sistemas tienden a ser mucho más interconectados entre sí lo cual hace su análisis muy complejo, lo que trae la necesidad de otros métodos y/o herramientas para el análisis y el diseño como lo es el software antes propuesto. Está interconexión de los sistemas es debida al desarrollo industrial, al compromiso de las empresas suministradoras de dar un servicio confiable, sin interrupciones, un servicio de calidad, en cuánto a su frecuencia y voltaje nominales del sistema. Europa es un claro ejemplo de ésta interconexión, ya que todos los países tienen enlace en una gran red internacional; México es también parte de esta tendencia ya que está unido con los países vecinos tanto en el norte como en el sur.

Las líneas de Transmisión tienen una importancia relevante en los sistemas eléctricos de potencia, ya que no tenemos las fuentes de generación cercanas a las cargas Por lo que al conocer más las propiedades eléctricas de las líneas de transmisión podremos modelar mejor todo el sistema eléctrico. El conocimiento de la impedancia de una línea es un trabajo arduo por lo cual, hay que darle a los diseñadores y/o analistas otras herramientas para un tiempo de respuesta menor y con una menor probabilidad de error.

Los softwares que se generan actualmente sea el área que fuere deben de basarse en técnicas modernas de programación como es la Orientación a Objetos, ya que tiene muchas ventajas sobre las técnicas estructurales, entre las cuales encontramos que son programas fácilmente modificables, adaptables, perfeccionables y flexibles. Llegará un día en que nosotros podamos comprar partes de un software y solo lo tengamos que armar tal y como lo hacemos en la

Ingeniería Eléctrica Electrónica, dando como consecuencia una mayor velocidad de construcción, mejores y más precisos diseños ya que las partes estarán del todo probadas y perfeccionadas logrando que, con un buen diseño general, el todo lo esté.

A continuación enumeramos los conceptos que se pueden ahondar para perfeccionar el software propuesto

Para el software de Cálculo de Corto Circuito:

1. Agregar otros elementos como lo son: Reactores, Capacitores, etc, que muy en particular tienen una importancia imprescindible en el control de reactivos y como consecuencia en el factor de potencia.

2. Agregar nuevas conexiones para los transformadores (delta-estrella, estrella-estrella, estrella-delta, delta-estrella aterrizada, etc.) y para los generadores y motores (estrella aterrizada, estrella aterrizada con impedancia, etc.), lo que traerá una secuencia negativa más cercana a todas las posibilidades existentes en la industria, generación, transmisión y distribución.

3. Mejorar el algoritmo de Inversión de las matrices por uno que soporte una mayor cantidad de elementos en un procesamiento menor y/o mas confiable, con lo que será necesario crear un depurador de la memoria ya que el uso de los recursos del sistema llega a ser un punto determinante en el cálculo del corto circuito por este método.

Para el software de el Cálculo de Parámetros de Líneas de Transmisión:

1. Considerar otros casos de topologías, para el cálculo de la impedancia de la línea de transmisión.

2. Considerar el caso de fases constituidas por varios conductores, ya que es una forma de construcción de las líneas muy usada.

La Universidad Nacional Autónoma de México muy en especial la Facultad de Ingeniería debe generar software para temas específicos como lo es el de los sistemas eléctricos de potencia, ya que dichas aplicaciones son muy caros y en ocasiones inexistentes y sería un desperdicio tener la capacidad de crearlos y no hacerlo, ya que podríamos aprovecharlos tanto como una herramienta en la docencia y en un futuro comercializarlos. Dicha generación de software no es exclusiva de las ramas de ingeniería sino de otros tópicos como lo son la Economía, las Finanzas, las Humanidades, etc.

Tenemos la esperanza que éste proyecto trascienda y haya más personas interesadas en desarrollar proyectos en apoyo a nuestra muy olvidada rama de la Ingeniería Eléctrica, ya que sin ella no existiría aquello llamado la "era tecnológica"...

. dixi

Apéndices

APÉNDICE A

Análisis y Diseño Orientado a Objetos

A.1 Metodología OMT (Metodología Técnica de Orientación a Objetos)

Una parte muy importante en el mundo de la orientación a objetos es la metodologías, ya que dan pauta a la construcción de software siguiendo unos pasos predefinidos que cualquier persona con el conocimiento mínimo sobre la metodología pueda entender el análisis y el diseño propuesto para la realización del proyecto. Con las metodologías lo que se busca es eliminar las barreras en la interacción de las partes, hablar todos un mismo lenguaje. Además de que nosotros ya no dependeremos de una persona en particular, si nuestro proyecto queda truncado por la pérdida de personal, solo será necesario contratar a otra persona que sepa sobre la metodología con la que trabajamos.

La OMT (Metodología Técnica de la Orientación a Objetos) es una de las técnicas más usadas en el mundo de la Orientación a Objetos. Su importancia también radica en que es uno de los tres pilares de la mayor técnica usada en la Orientación a Objetos que es la UML (Metodología del Lenguaje Unificado).

Se caracteriza por tres modelos:

Modelos

1. *Objetos (Estático).* Este modelo se caracteriza por la creación de un diagrama donde se muestren las clases y sus relaciones; podríamos hacer una excelente analogía con el modelo de entidad-relación, tan usado en las bases de datos relacionales En está parte del proceso es muy importante identificar las clases y que relación guardan con las demás que puede ser una simple relación o una herencia múltiple.

2. *Dinámico.* Se denota la interacción de las clases a lo largo del tiempo. Aquí es muy importante que pasa primero y que subsecuente a dicha acción. Muchas veces nos basamos en escenarios para caracterizar todas las acciones en un entorno lo más real posible. Esto es como que nosotros, los diseñadores, nos convirtiéramos en manejadores de títeres y realizáramos una obra con los títeres que construyamos. En está analogía el títere sería nuestra clase y la obra el sistema que estamos creando. Un elemento fundamental en este modelo es el tan afamado diagrama de estados, donde por excelencia se usa en la electrónica digital, para entender mejor la complejidad de nuestro problema.

3. *Funcional.* La importancia en este modelo es la transformación de los datos. Por primera vez nos paramos para estudiar el o los valores los cuales representan un elemento esencial en la solución de nuestro problema Por ejemplo al modelar un sistema de cajeros automáticos uno de los valores que

tiene una relevancia extrema es el saldo de la cuenta, ya que cuanto nos pide el dinero hay que cotejar con el saldo, luego hay que disminuir dicho saldo en igual proporción a la solicitada por el usuario. En este modelo las principales herramientas es el diagrama de flujo de datos y el diagrama de objetos. En el diagrama de objetos nosotros estudiamos cual es el producto, cual es el proceso y cual es el consumo.

A.2 Ciclo de Vida de un Sistema Orientado a Objetos según OMT

A.2.1 ANÁLISIS

La meta es desarrollar un modelo de lo que se quiere que el sistema haga. El modelo es expresado en términos de objetos y relaciones, control del comportamiento dinámico y de las transformaciones funcionales

El proceso para capturar los requerimientos y de consulta con el cliente deberá continuar durante todo el análisis.

1. Escribir u obtener una descripción inicial del problema. (definición del problema).
2. Construir un modelo de objetos.

   - Identificar clases de objetos
   - Iniciar un diccionario de datos que contenga la descripción de las clases, atributos y relaciones.
   - Agregar las asociaciones entre clases.
   - Agregar los atributos a los objetos.
   - Organizar y simplificar las clases de objetos con la herencia.
   - Pruebe las interfaces utilizando escenarios, y realice varias iteraciones de los pasos anteriores antes de continuar.
   - Agrupe las clases en módulos, basados en lo cercano de sus relaciones y funcionalidad.

MODELO DE OBJETOS=diagrama del MO+diccionario de datos

3. Construir un modelo dinámico.

   - Preparar escenarios de situaciones.
   - Identificar eventos entre objetos y preparar un rastreo de eventos por escenario.
   - Preparar un diagrama de flujo de eventos para el sistema.
   - Desarrollar un diagrama de estado para cada clase que tenga un comportamiento dinámico interesante.
   - Verifique la consistencia de los eventos distribuidos en el diagrama de estado.

MODELO DINÁMICO=diagrama de estados+diagrama de flujo de eventos global.

4. Construya un modelo funcional.

   - Identifique los datos de entrada y salida.
   - Utilice un diagrama de flujo de datos tanto como sea necesario para mostrar las dependencias funcionales.
   - Describa qué hace cada función.
   - Identifique restricciones.
   - Especificar criterios de optimización.

MODELO FUNCIONAL=diagrama de flujo de datos+restricciones.

5. Verifique, repase y ajuste los tres modelos.

- Agregue operaciones importantes que fueron descubiertas durante la preparación del modelo funcional al modelo de objetos. No muestre todas las operaciones durante el análisis, sólo las más importantes deben aparecer.
- Verifique que las clases, relaciones, atributos y operaciones sean consistentes y completos de acuerdo al nivel de abstracción seleccionado. Compare los tres módulos con la definición del problema y los puntos relevantes que se tengan sobre el conocimiento del dominio de la aplicación, y pruebe los modelos utilizando escenarios.
- Repase los pasos anteriores tanto como sea necesario para completar el análisis.

Documento de análisis=definición del problema+modelo de objetos+modelo dinámico+modelo funcional.

A.2.2 DISEÑO

Durante el diseño del sistema, estructuras de alto nivel deben ser seleccionadas.

1. Organizar el sistema en subsistemas.
2. Identificar problemas de concurrencia.
3. Definir tareas específicas a los subsistemas.
4. Seleccionar estrategias básicas para la implementación en términos de almacenamiento de datos, estructuras de datos, archivos o bases de datos.
5. Identificar los recursos globales y determinar los mecanismos de control de accesos a ellos.
6. Escoger un enfoque apropiado para el control de la implementación
7. Considere condiciones de frontera.
8. Establezca prioridades.

Documento de diseño=estructura básica de la arquitectura del sistema, como también decisiones estratégicas de alto nivel.

A.2.3 DISEÑO DE OBJETOS

Durante el diseño de objetos elaboramos un modelo de análisis y proveemos las bases detalladas para la implementación. Hacemos las decisiones que sean necesarias para realizar el sistema sin caer en detalles particulares debido a un lenguaje o DBMS. El diseño de objetos inicia apoyándose en las situaciones del mundo real del modelo del análisis dirigiéndose al enfoque computacional que se requiere para llevar a cabo la implementación práctica

1. Obtener las operaciones para el modelo de objetos a partir de los otros modelos:

   - Encontrar una operación por cada proceso del modelo funcional.
   - Definir una operación por cada evento del modelo dinámico, dependiendo de los mecanismos de control de la implementación.

2. Diseñar algoritmos para implementar las operaciones:

   - Escoger algoritmos que minimicen el costo de implementación.
   - Seleccionar las estructuras de datos apropiadas para los algoritmos
   - Definir nuevas clases internas y operaciones tanto como sea necesario.
   - Asignar responsabilidades a las operaciones que no estén claramente asociadas a una clase.

3. Optimizar las interfaces a los datos:

   - Añade relaciones redundantes para minimizar los accesos costosos y que maximizan la conveniencia.
   - Reacomodar los cálculos para obtener mayor eficiencia.
   - Guardar valores derivados para evitar calcularlos nuevamente, sobre todo en expresiones complejas.

4. Ajustar las clases para aumentar la herencia

   - Reacomodar y ajustar las clases y operaciones para incrementar la herencia
   - Extraer comportamientos comunes agrupando clases.

5. determine la representación exacta de los atributos.
6. Empacar clases y relaciones en módulos.

Documento de diseño de objetos=modelo de objetos detallado+modelo dinámico detallado+modelo funcional detallado

APÉNDICE B

Comparación con otros Sistemas

La comparación de este software se llevara acabo contra el software Power System Analysis and Design Software de Duncan Glover, y contra el software denominado HAWK-40, la comparación se llevara acabo en lo que se refiere a la descripción general del mismo, entrada de datos y salida de resultados, alcance limitaciones, método de solución y equipo de computo requerido.

1.1 Diseño de un sistema digital para el estudio de líneas de transmisión y corto circuito.

1.1.1 Descripción.

El software para esta aplicación se denomina CPLT (Cálculo de Parámetros de Líneas de Transmisión), este software calcula los parámetros de líneas de transmisión en circuitos trifásicos simples ó dobles con o sin cables de guarda.

Este software permitirá en primera instancia a los estudiantes y profesores de nivel superior utilizar la ventaja de la computadora digital para el análisis de líneas de transmisión reduciendo tiempo en procesos de cálculo grandes.

1.1.2 Entrada de datos.

Se tienen especificaciones técnicas para que el programa cumpla con el funcionamiento optimo para el cuál fue diseñado, se necesitan consideraciones propias tanto de la computadora que se vaya a utilizar, como en el software de aplicación.

En el caso del cálculo de líneas de transmisión y de corto circuito, se tienen consideraciones solamente en el tipo de dato a introducir y específicamente en las unidas métricas que se utilizan.

Los datos requeridos para que el software realice los cálculos en el estudio de líneas de transmisión son.

- Frecuencia Hz.
- Resistencia del conductor R_conductor ohms/km.
- Resistencia del cable de guarda Rg_guarda ohms/km.
- Resistividad de la tierra ρ.
- Radio medio geométrico del conductor RMG en m
- Radio medio geométrico del cable de guarda RMG_guarda en m.
- Posición X y posición Y del conductor dada en m.
- Posición X y posición Y de los cables de guarda dada en m (en caso de ser necesarios)

Los datos requeridos para que el software realice los cálculos en el estudio de corto circuito además del voltaje base y de la potencia base para el sistema, son los siguientes.

Para los generadores.
- Potencia en MVA
- Voltaje en KV.
- Número de bus al que esta conectado.
- Impedancias de secuencia positiva, negativa y cero.

Para los motores
- Tipo de potencia Real en W ó Aparente en MVA
- Factor de potencia si se requiere.
- Voltaje en KV
- Número de bus al que esta conectado.
- Impedancias de secuencia positiva, negativa y cero.

Para las líneas.
- Número de los dos buses al que esta conectado
- Impedancias de secuencia positiva, negativa y cero, en forma compleja (a, b).
  a: Parte real.
  b: Parte compleja.

Para los transformadores.
- Potencia en MVA.
- Voltaje en el bus de alta tensión en KV
- Voltaje en el bus de baja tensión en KV.
- Número de bus del lado de alta tensión
- Número de bus del lado de baja tensión
- Impedancias de secuencia positiva, negativa y cero

1.1.3 Salida de resultados.

En ambas aplicaciones los resultados serán mostrados mediante archivos que se crean después de realizados los cálculos la forma de diferenciarlos con los de entrada es mediante la terminación (CLT para líneas y .CCC para corto circuito).

Los datos de salida para líneas se muestran en coordenadas cartesianas y coordenadas fasoriales, así como en ohms /km u ohms /milla.
Los resultados que se obtienen son.

- Impedancia de secuencia cero.
- Impedancia de secuencia positiva y negativa

Los resultados que se obtienen para el corto circuito son las corrientes de falla en cada bus.

- Falla línea a tierra.
- Falla línea a línea.
- Falla bifásica a tierra
- Falla trifasica

1.1.4 Alcance y limitaciones.

Para el software de líneas de transmisión se tienen modelos de circuitos ya establecidos que se deben de seguir, no siendo esta una limitante para llevar acabo un análisis exhaustivo y profesional.

En el caso del análisis de corto circuito, se propone para el software, un numero límite de elementos en el sistema eléctrico para un rendimiento total, aunque puede funcionar teóricamente para un numero "N" de elementos conectados al sistema.

1.1.5 Métodos de solución.

El software se base principalmente para el estudio de líneas de transmisión se basa en el uso de las corrientes de secuencia cero, positiva y negativa.

En el caso del estudio de corto circuito se basa en la utilización de matrices para la realización de los cálculos

1.1.6 Equipo de cómputo recomendado.

Se requiere para el equipo de cómputo:

- Procesador 386 ó superior.
- 4 Mb de memoria RAM.
- 3 Mb de espacio en disco duro.
- Windows 3.1 ó superior.
- Impresora compatible.

2.2 Power system analysis and design software by Duncan Glover.

2.2.1 Descripción.

El software Power system analysis and design calcula los parámetros para un análisis general de los sistemas eléctricos de potencia. Se tiene aplicación propia del software solo para algunos capitulos en especifico estos son: Componentes simétricas (Capitulo 3), Constantes de línea (capitulo 5), Lineas de transmisión (Capitulo 6), Corto circuitos simétricos (Capitulo 8), Transitorios en líneas de transmisión (Capitulo 12) y estabilidad transitoria (Capitulo 13).

Este software permite a los estudiantes e ingenieros analizar sistemas de potencia eléctricos de una forma extensa y completa así como el diseño de estudios mas detallados, permitiendo trabajar con problemas reales de gran complejidad.

2.2.2 Entrada de datos.

Los datos son introducidos mediante un archivo con nombre y terminación .dat. En el se almacenaran los parámetros que en cada caso de análisis se vayan requiriendo, siguiendo un orden especifico dado por el tipo de análisis.

Los datos de entrada de los sistemas de potencia deberán de ser convertidos a valores en por unidad o a un sistema base común.

Cada programa que utilice un archivo de datos tiene la opción "display the data file", para mostrar los datos del archivo que se este utilizando, o también el usuario del programa puede acceder a la opción de mostrar los datos en el monitor.

Para el análisis de líneas de transmisión, el software obtiene la impedancia serie y la matriz de admitancia de derivación en líneas de transmisión aéreas de circuitos simples ó circuitos dobles trifásicos

Los datos requeridos son.

- Número de circuitos NC.
- Resistividad de la tierra RHO.
- Voltaje de línea y de línea a línea en KV.
- Frecuencia Hz.
- Número de cables neutros para los diferentes circuitos.

Posteriormente se deben de introducir otros datos que también se requieren para el análisis.

- Resistencia del conductor R en Ohms/Km.
- Radio medio geométrico del conductor RMG en cm
- Diámetro exterior del conductor D en cm.
- Posición X y posición Y del conductor dada en m.
- Número de conductores por cable Nb

Para el análisis de corto circuito, el software calcula la corriente de falla simétrica para un corto circuito trifásico en un sistema de N buses. También se analizan las contribuciones de la corriente de falla en maquinas síncronas, líneas de transmisión y transformadores conectados al bus de falla.

Los datos que se requieren para este análisis son:
Para las maquinas síncronas

- Número de la máquina.
- Número de bus.
- Reactancia de secuencia positiva de cada máquina en por unidad.

Para las líneas de transmisión.
Número de línea.
Los dos buses a los cuales se conecta la línea.
Reactancia de secuencia positiva de cada línea en por unidad.

Para los transformadores los datos son similares a los de la línea, solo que en el caso de la reactancia se refiere a la reactancia del tipo leakage también en por unidad.

2.2.3 Salida de resultados.

La salida de resultados se presenta mediante un archivo de datos de salida, el cuál permite al usuario examinar el contenido del archivo de datos para poderlo procesar.

Por cada programa que utilice un archivo de datos, el usuario del programa puede almacenar los datos de 3 o 5 casos diferentes. Cada vez que se revisen los archivos de datos de entrada de un programa, el programa escribirá sobre el archivo de datos, esto es reemplazara el archivo de datos en el directorio con los valores nuevos, de esta forma no se incrementara el espacio ocupado en el disco y los valores buscados no cambiaran.

Los resultados que se obtienen son.

- Corrientes de falla.
- Voltajes en el bus durante la falla.
- Matriz de impedancia de bus.

Se tiene la opción de mostrar los resultados en formato exponencial o en formato decimal.

2.2.4 Alcance y limitaciones.

Este software es en primera instancia dedicado para fines educativos, no para resolver problemas que involucren sistemas eléctricos de potencia de dimensiones muy grandes, pero en estudios de corto circuito, flujos de potencia y estabilidad transitoria, el software es capaz de soportar hasta arriba de 100 buses.

2.2.5 Método de solución.

Este software se basa en un proceso matricial.

2.2.6 Equipo de computo utilizado.

- PC compatible IBM
- 2 MB de espacio en disco duro.
- Sistema operativo Microsoft Windows 3.1 o superior.
- Impresora compatible.

3.3 Software HAWK-40.

3.3.1 Descripción.

Este software calcula los parámetros de líneas de transmisión que son utilizados en diversos análisis de sistemas eléctricos de potencia, estudios de corto circuito, flujos de potencia y estudios transitorios.

3.3.2 Entradas de datos.

Los datos de las líneas son introducidos mediante un archivo, que deberá de tener un orden especifico indicado en el instructivo.

- Grupo de líneas.
- Voltaje de operación.
- Identificación de la línea.
- Subestaciones externas que conecta la línea.
- Numero de conductores e hilos de guarda.
- Longitud de la línea (Km.).
- Resistencia base.
- Frecuencia base.
- MVA Base.
- Resistencia del conductor.
- Reactancia del conductor.
- Diámetro exterior del conductor.
- Coordenadas cartesianas de localización de cada conductor incluyendo hilos de guarda.

3.3.3 Salida de resultados.

La salida de resultados se presenta mediante un archivo de datos de salida, se debe de mencionar la ruta y nombre del archivo.

Los resultados que se muestran son:

- *Datos alimentados de entrada.*

- Valores en por unidad de las componentes simétricas de secuencia positiva y cero, de los valores de impedancia y admitancia de shunt.
- Valores de impedancia serie incluyendo en caso de ser mas de un circuito, los valores mutuos y constantes ABCD.
- Impedancia Zpi equivalente, cuya base de potencia fue dada como dato de entrada y Tensión base igual a la de la línea.

3.3.4 Alcance y limitaciones.

Este software calcula parámetros en estructuras con hasta 4 circuitos , ya sean todos ellos sencillos, o bien 2 circuitos con doble conductor por fase incluyendo 4 hilos de guarda

Las limitaciones se presentan en cuanto al numero de conductores:

- Número de circuitos.
  4 sencillos un conductor por fase o 2 con doble conductor por fase.

- Número de hilos de hilos de guarda: 4

- 3 fases x número de conductores por fase x número de hilos de guarda ≤ 16.

3.3.5 Método de solución.

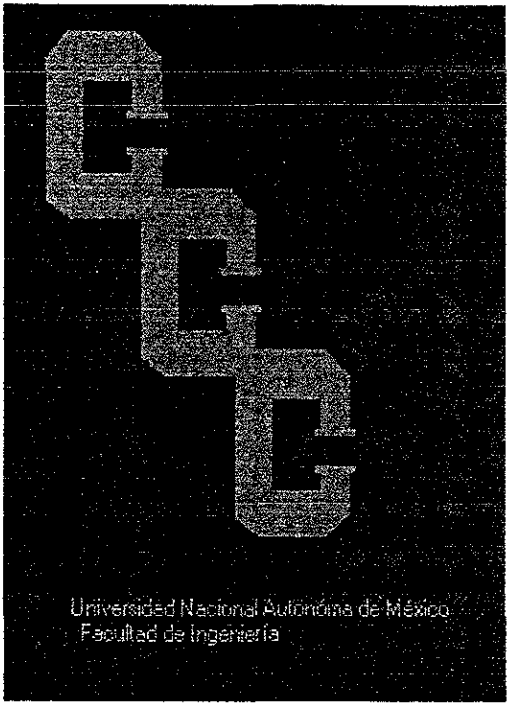
El software HAWK-40 procesa líneas con conductores agrupados, el procedimiento utilizado se basa en un cálculo matricial.

3.3.6 Equipo de computo utilizado.

- PC compatible IBM XT.
- 550 kb de memoria.
- 3 Mb en disco duro
- Sistema operativo MS-DOS.
- Software HAWK-40.
- Impresora

APÉNDICE C

Código de CCC


Universidad Nacional Autónoma de México
Facultad de Ingeniería

Este archivo contiene todos las declaraciones de las clases más importantes en la solución del problema de análisis de corto circuito. Contiene un amplio conjunto de definiciones, necesarias para el buen funcionamiento de las cajas de dialogo principalmente.

```
/*
                    ZBUS.H

*/

// Modificado el 27 de abril de 1999
// siendo las 23.30 hrs.
#define ID_C91 5555
#define ID_C92 6666

#define CM_U_HELPINDEX          0x200
#define CM_U_HELPHELP           0x201

#define CM_HELP  201
#define CM_BORRAR 202
#define CM_SALIR  203
#define CM_CAJA  1110
#define CM_SONIDO 3333
#define FunBotIzq Ventana::WMLButtonDown
#define FunBotDer Ventana::WMRButtonDown

#define NULO hInstance, hPrevInstance,lpCmdLine, nCmdShow
#define CreaVentana TMyApp MyApp
#define AbreVentana(): MyApp Run(); return MyApp.Status;

#define main() int PASCAL WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,LPSTR
lpCmdLine, int nCmdShow)

#define f1 Ventana::fun1
#define f2 Ventana::fun2
#define f3 Ventana::fun3
#define f4 Ventana::fun4
#define f5 Ventana::fun5
#define f6 Ventana::fun6
#define f7 Ventana::fun7
#define f8 Ventana ::fun8
#define f9 Ventana  fun9
#define f10 Ventana::fun10


#define CM_ALUM1 801
#define CM_ALUM2 802
#define CM_ALUM3 803

#define CM_GEN 804
#define CM_MOT 805
#define CM_TRAN 806
```

```
#define ID_M1  901
#define ID_M2  902
#define ID_M3  903
#define ID_M4  904
#define ID_M5  905
#define ID_M6  906
#define ID_M7  907
#define ID_M8  908

#define ID_C41 401
#define ID_C42 402
#define ID_C43 403
#define ID_C44 404
#define ID_C45 333
#define ID_C46 334

#define ID_C31 405
#define ID_C32 406
#define ID_C33 407
#define ID_C34 408
#define ID_C35 409

#define ID_C21 408
#define ID_C22 409

#define ID_C11 410

#define CM_NUMINPUT 301
#define ID_NUM1 101
#define ID_NUM2 105
#define ID_OP 107
#define CM_CIR 401
#define CM_ELI 402
#define CM_LINEA 403
#define CM_ARCO 404
#define CM_IMAGEN 666

#define MAXNUM 15
#define MAXOP 2
#define MAX 50
#define MAXC 60

#define CM_F1 501
#define CM_F2 502
#define CM_F3 503
#define CM_F4 504
#define CM_F5 505
#define CM_F6 506
#define CM_F7 507
#define CM_F8 508
#define CM_F9 509
#define CM_F10 510
#define CM_ABOUT 511
#define CM_ESCOGE 777
```

```
#define MAXPATH 80
#include <groupbox.h>
#include "resource.h"
#include <stdio.h>
#include <math.h>
#include <COMPLEX.h>
#include <bwcc.h>
#include <bradio.h>
#include <bgrpbox.h>
#include <stdlib.h>
#include <array.h>
#include <abstarry.h>
#include <owl.h>
#include <inputdia.h>
#include <string.h>
#include <control.h>
#include <edit.h>
#include <dialog.h>
#include <ctype.h>
#include <iostream.h>
#include <clstypes.h>
#include <filedial.h>
#include <BSTATBMP H>
#include <dos.h>
#include <time h>
#include "filewnd.h"

#include "c:\tsep\abril\zarch\zarch.cpp"
//#include "c:\cabal!~1\tesis\abril\zarch\zarch.cpp"

#define cam31 "Z1="
#define cam32 "Z2="
#define cam33 "Z0="
#define cam34 "bus AT"
#define cam35 "bus BT"


#define ID_P1 10001
#define ID_P2 10002

#define CM_ARCHSAL  2103
#define CM_POT 2104

float Resultado;
int sonido=5;
int valor;
char variable[MAXC];



struct TTransferStruct {
        char Num1[MAXNUM];
   char Num2[MAXNUM];
```

```
  char Oper[MAXOP];
};

struct TTransfiere2 {
  char C1[MAXNUM];
  char C2[MAXNUM];
};

struct TTransfiere4{
        char C1[MAXC];
        char C2[MAXC];
        char C3[MAXC];
        char C4[MAXC];
        char C5[MAXC];
     char C6[MAXC];
};

struct TTransfiere5{
        char C1[MAXC],
        char C2[MAXC],
        char C3[MAXC],
        char C4[MAXC],
        char C5[MAXC],
     char C6[MAXC];
};

struct TTransfiere6{
        char C1[MAXC];
        char C2[MAXC],
        char C3[MAXC];
        char C4[MAXC];
     char C5[MAXC];
        char C6[MAXC],
     char C7[MAXC];
     char C8[MAXC],
};

struct TTransfiere3{
        char C1[MAXC];
        char C2[MAXC],
        char C3[MAXC],
        char C4[MAXC];
     char C5[MAXC],
};


_CLASSDEF(TPoint)
class TPoint: public Object
{
public:
 int X, Y;
 TPoint(int AX, int AY) {X = AX, Y = AY;}
 virtual classType isA() const { return __firstUserClass; }
 virtual Pchar nameOf() const { return "TPoint"; }
```

```
  virtual hashValueType hashValue() const { return 0; }
  virtual int isEqual(RCObject APoint) const
    { return X == ((RTPoint)APoint) X && Y == ((RTPoint)APoint).Y. }
  virtual void printOn(Rostream outputStream) const
    { outputStream << "(" << X << "," << Y << ")", }
};

_CLASSDEF(TPointArray)
class TPointArray   public Array
{
public.
  TPoint Array(int upper, int lower = 0, sizeType aDelta = 0)
    : Array(upper, lower, aDelta){};
  virtual classType isA() const { return __firstUserClass + 1; }
  virtual Pchar nameOf() const { return "TPointArray", }
},



//_CLASSDEF(Ventana)
class _EXPORT Ventana : public TFileWindow
{
public:
  HMENU hmenu;
  HINSTANCE hinst,
  HWND hwnd,
  HDC DragDC,
  BOOL LineaLibre,Ele,Cir,Arc,Lin1,IsNewFile,
  HPEN ThePen;
  HBITMAP alicia,
  HDC hdc;
    int PenSize,contador,contadorx,contadory,xi,yi,
  TTransferStruct TransferStruct;
  TTransfiere4 Transfiere4;
  TTransfiere3 Transfiere3,
  TTransfiere6 Transfiere6;
  TTransfiere5 Transfiere5,
  TTransfiere2 Transfiere2;
  PTPointArray Points;
   char FileName[MAXPATH];
  Ventana(PTWindowsObject AParent, LPSTR ATitle, LPSTR AFileName);
  ~Ventana(),
  char * dotostr(double,int);
  void Duerme(int),
  void Imagen()
     =[CM_FIRST+CM_IMAGEN];
  void DrawBMP( HDC DC, int X, int Y, HBITMAP BitMap ),
  virtual BOOL CanClose().
  void Escoge()=[CM_FIRST+CM_ESCOGE];
  void SetPenSize(int NewSize);
  virtual void WMLButtonDown(RTMessage Msg)
  = [WM_FIRST + WM_LBUTTONDOWN];
  virtual void WMLButtonUp(RTMessage Msg)
    = [WM_FIRST + WM_LBUTTONUP]
```

```
virtual void WMRButtonUp(RTMessage Msg)
  = [WM_FIRST + WM_RBUTTONUP];
virtual void WMMouseMove(RTMessage Msg)
  = [WM_FIRST + WM_MOUSEMOVE];
virtual void WMRButtonDown(RTMessage Msg)
= [WM_FIRST + WM_RBUTTONDOWN];
virtual void CMHelp(RTMessage Msg)
  = [CM_FIRST + CM_HELP],
void About(void)
    =[CM_FIRST + CM_ABOUT ],
void Dibuja_Linea(RTMessage Msg),
void Despliega(float cadena),
void Despliega(int cadena),
void Despliega(char *cadena);
void Despliega(char cadena),
void Despliega(complex cadena),
void Salta(void);
void Activa(),
void Mensaje(char *titulo,char *mensaje);
int& Pide(char *titulo, char *mensaje, int &);
float& Pide(char *titulo, char *mensaje, float &);
char& Pide(char *titulo, char *mensaje, char *);
char& Pide(char *titulo,char *mensaje,char&);
void Borrar(void)
        =[CM_FIRST + CM_BORRAR],
void Salir(void)
  =[CM_FIRST + CM_SALIR];
void Circulo(int x=10,int y=10,int radio=20)=[CM_FIRST+CM_CIR];
void X();
void Elipse(int a=10,int b=10,int h=10,int k=10)=[CM_FIRST+CM_ELI];
void Arco(int x1,int y1,int x2,int y2,int x3,int y3,int x4,int y4)
  =[CM_FIRST+CM_ARCO];
void Rectangulo(int x1=10,int x2=10,int x3=100,int x4=100);
void RectanguloBR(int x1=50,int y1=50, int x2=150, int y2=150, int r=20,int e=20);
virtual void Gen(char *, char *, char *, char *,char *,char *) = [CM_FIRST + CM_GEN];
int Guarda(char *,char *,char *,char  *,char *,char *);
int Guardam(char *,char *,char *,char  *,char *,char *);
int Guardat(char *,char *,char *,char  *,char *,char *,char *,char *);
int Guarda(char *,char *,char *,char *,char *);
virtual void Mot(char *,char*,char*,char*,char*,char*,char*,char*)=[CM_FIRST +CM_MOT],
virtual void Trans(char *,char*,char*,char*,char*,char*,char*,char*,char *)=[CM_FIRST +CM_TRAN];
virtual void Lin(char *, char *, char *,char*,char *) = [CM_FIRST + CM_ALUM3];
friend void Enlace(float);
virtual void Potencia(char *, char *)=[CM_FIRST + CM_POT];
void fun3()=[CM_F3+CM_FIRST];
void fun4()=[CM_F4+CM_FIRST];
void fun5()=[CM_F5+CM_FIRST];
void fun6()=[CM_F6+CM_FIRST],
void fun7()=[CM_F7+CM_FIRST];
 virtual void CMUHelpIndex( RTMessage ) = [CM_FIRST + CM_U_HELPINDEX];
        virtual void CMUHelpHelp( RTMessage ) = [CM_FIRST + CM_U_HELPHELP];

};
```

```
class Generador : public TDialog{
public:
   char Camp1[MAXC] ;
   char Camp2[MAXC] ;
   char Camp3[MAXC] ;
   char Camp4[MAXC] ;
         char Camp5[MAXC] ;
   char Camp6[MAXC] ,
   Generador(PTWindowsObject AParent, LPSTR name),
   virtual BOOL CanClose();

private
   void FillBuffers();
};

class Motor · public TDialog{
public.
   char Camp1[MAXC] ;
   char Camp2[MAXC] ;
   char Camp3[MAXC] ;
   char Camp4[MAXC] ;
   char Camp5[MAXC] ,
   char Camp6[MAXC] ;
   char Camp7[MAXC] ;
   char Camp8[MAXC] ;
   Motor(PTWindowsObject AParent, LPSTR name);
   virtual BOOL CanClose();

private:
   void FillBuffers();
};

class Pregunta : public TDialog{
public:
   Pregunta(PTWindowsObject AParent, LPSTR name);
   virtual BOOL CanClose();

private:
   void FillBuffers();
};

class Motor2 : public TDialog{
public:
   char Camp1[MAXC] ;
   char Camp2[MAXC] ;
   char Camp3[MAXC] ;
   char Camp4[MAXC] ;
   char Camp5[MAXC] ;
   char Camp6[MAXC] ;
   Motor2(PTWindowsObject AParent, LPSTR name);
   virtual BOOL CanClose();

private.
   void FillBuffers().
```

```
};

class Transfor   public TDialog{
public.
   char Camp1[MAXC] ;
   char Camp2[MAXC] ;
   char Camp3[MAXC] ,
   char Camp4[MAXC] ,
   char Camp5[MAXC] ;
   char Camp6[MAXC] ;
   char Camp7[MAXC] ;
   char Camp8[MAXC] ;
   Transfor(PTWindowsObject AParent. LPSTR name);
   virtual BOOL CanClose();

private:
   void FillBuffers();
};

class Linea : public TDialog{
public:
   char Camp1[MAXC] ;
   char Camp2[MAXC] ;
   char Camp3[MAXC] ;
   char Camp4[MAXC] ;
   char Camp5[MAXC] ;
   Linea(PTWindowsObject AParent. LPSTR name);
   virtual BOOL CanClose(),

private:
   void FillBuffers();

};

class Pot · public TDialog{
public:
   char Camp1[MAXC] ;
   char Camp2[MAXC] ;
   Pot(PTWindowsObject AParent, LPSTR name).
   virtual BOOL CanClose().

private:
   void FillBuffers().

};


const unsigned int SW_SIZE = 40.
enum SQUARESTATUS { SQS_UNOCCUPIED. SQS_X. SQS_O };
SQUARESTATUS UserSide:

class TMyApp . public TApplication
{
public·
```

```
 TMyApp(LPSTR AName, HINSTANCE hInstance, HINSTANCE hPrevInstance,
   LPSTR lpCmdLine, int nCmdShow)
        : TApplication(AName, hInstance, hPrevInstance, lpCmdLine, nCmdShow) {};
 virtual void InitMainWindow();
 virtual void InitInstance();
};
void TMyApp: InitInstance()
{
 TApplication::InitInstance();
 if ( Status == 0 )
   HAccTable = LoadAccelerators(hInstance, "FileCommands");
}
```

Este es el archivo principal, ya que se tiene las definiciones de la mayoría de las funciones miembro de las clases

```
/*

            ZBUS CPP


*/


#define cam41 "potencia="
#define cam42 "voltaje="
#define cam43 "Z1="
#define cam44 "Z2="
#define cam45 "Z0="
#define cam46 "eficiencia="
#define cam47 "fp"
#define cam48 "bus="

#define cam51 "potencia="
#define cam52 "voltaje="
#define cam53 "Z1="
#define cam54 "Z2="
#define cam55 "Z0="
#define cam56 "bus="

#define cam31 "Z1="
#define cam32 "Z2="
#define cam33 "Z0="
#define cam34  "bus="

#include "zbus.h"

//******************** Para la ayuda ******************************

void Ventana::CMUHelpIndex( RTMessage )
{
        WinHelp( HWindow, "CC1.HLP", HELP_INDEX, 0L );
```

```
}

void Ventana::CMUHelpHelp( RTMessage )
{
        WinHelp( HWindow, "CC1.HLP", HELP_HELPONHELP, 0L );
}



//********************* Para escoger achivo de entrada****
void Ventana::Escoge()
{
 InvalidateRect(HWindow, NULL, TRUE);
 IsNewFile=1;

 if ( GetApplication()->ExecDialog(new TFileDialog(this, SD_FILEOPEN,
  strcpy(FileName, "*.dat"))) == IDOK ){
        fstream salida(FileName, ios::in|ios::out|ios::noreplace);
    if(!salida){
          if(MessageBox(HWindow, "¿ Quieres que lo reemplace ?", "El archivo ya existe ",
           MB_YESNO | MB_ICONEXCLAMATION)==IDYES){
              fstream salida(FileName,ios::out|ios::trunc);
                 salida.close();
                 Activa();
                 SetWindowText(HWindow, (LPSTR) FileName);
          }
          else{
                Activa();
             SetWindowText(HWindow, (LPSTR) FileName);
       }

    }
    else{

          Activa();
          SetWindowText(HWindow, (LPSTR) FileName);
    }
 }

}

//********************Control en el menu ************

void Ventana ::Activa(){
   HMENU hMenu = GetMenu (HWindow);
   EnableMenuItem(hMenu, ID_P1, MF_ENABLED);
   EnableMenuItem(hMenu, CM_F3, MF_ENABLED);
   EnableMenuItem(hMenu, CM_F4, MF_ENABLED);
   EnableMenuItem(hMenu, CM_F5, MF_ENABLED);
   EnableMenuItem(hMenu, CM_F6, MF_ENABLED);
   EnableMenuItem(hMenu, CM_F7, MF_ENABLED);
}

void Ventana ::Duerme(int incremento){
```

```
  time_t to,tf;
  to =tf=time(NULL);
  while(tf<to+incremento)
        tf=time(NULL);

}

void Enlace(float  origen){ Resultado=origen; }

//***************Constructor de Generador *************

Generador::Generador(PTWindowsObject AParent, LPSTR name)
          :TDialog(AParent, name)
{
 new TEdit(this, ID_C41,
    sizeof(((Ventana *)Parent)->Transfiere4.C1));
 new TEdit(this, ID_C42,
          sizeof(((Ventana *)Parent)->Transfiere4.C2));
 new TEdit(this, ID_C43,
          sizeof(((Ventana *)Parent)->Transfiere4.C3));
 new TEdit(this, ID_C44,
          sizeof(((Ventana *)Parent)->Transfiere4 C4));
 new TEdit(this, ID_C45,
          sizeof(((Ventana *)Parent)->Transfiere4.C5));
 new TEdit(this, ID_C46,
          sizeof(((Ventana *)Parent)->Transfiere4.C6));
 TransferBuffer = (void far*)
             &(((Ventana *)Parent)->Transfiere4),
}

BOOL Generador::CanClose()
{
 FillBuffers();
 return TRUE;
}


void Generador::FillBuffers()
{
  GetDlgItemText(HWindow, ID_C41, Camp1, MAXC);
  GetDlgItemText(HWindow, ID_C42, Camp2, MAXC);
  GetDlgItemText(HWindow, ID_C43, Camp3, MAXC);
  GetDlgItemText(HWindow, ID_C44, Camp4, MAXC);
  GetDlgItemText(HWindow, ID_C45, Camp5, MAXC);
  GetDlgItemText(HWindow, ID_C46, Camp6, MAXC);
}

//******************* Constructor de Motor ********************

Motor::Motor(PTWindowsObject AParent, LPSTR name)
          :TDialog(AParent, name)
{
 new TEdit(this, ID_M1,
    sizeof(((Ventana *)Parent)->Transfiere6 C1));
```

```
 new TEdit(this, ID_M2,
          sizeof(((Ventana *)Parent)->Transfiere6.C2)),
 new TEdit(this, ID_M3,
          sizeof(((Ventana *)Parent)->Transfiere6.C3)),
 new TEdit(this, ID_M4,
          sizeof(((Ventana *)Parent)->Transfiere6 C4)),
 new TEdit(this, ID_M5,
          sizeof(((Ventana *)Parent)->Transfiere6.C5)),
 new TEdit(this, ID_M6,
          sizeof(((Ventana *)Parent)->Transfiere6 C6));
 new TEdit(this, ID_M7,
          sizeof(((Ventana *)Parent)->Transfiere6.C7));
 new TEdit(this, ID_M8,
          sizeof(((Ventana *)Parent)->Transfiere6.C8));
 TransferBuffer = (void far*)
            &(((Ventana *)Parent)->Transfiere6);
}

BOOL Motor::CanClose()
{
 FillBuffers();
 return TRUE;
}


void Motor::FillBuffers()
{
  GetDlgItemText(HWindow, ID_M1, Camp1, MAXC);
  GetDlgItemText(HWindow, ID_M2, Camp2, MAXC);
  GetDlgItemText(HWindow, ID_M3, Camp3, MAXC);
  GetDlgItemText(HWindow, ID_M4, Camp4, MAXC);
  GetDlgItemText(HWindow, ID_M5, Camp5, MAXC);
  GetDlgItemText(HWindow, ID_M6, Camp6, MAXC);
  GetDlgItemText(HWindow, ID_M7, Camp7, MAXC);
  GetDlgItemText(HWindow, ID_M8, Camp8, MAXC);

}

Motor2::Motor2(PTWindowsObject AParent, LPSTR name)
          :TDialog(AParent, name)
{
 new TEdit(this, ID_M1,
   sizeof(((Ventana *)Parent)->Transfiere5.C1));
 new TEdit(this, ID_M2,
          sizeof(((Ventana *)Parent)->Transfiere5 C2)),
 new TEdit(this, ID_M3,
          sizeof(((Ventana *)Parent)->Transfiere5.C3)),
 new TEdit(this, ID_M4,
          sizeof(((Ventana *)Parent)->Transfiere5.C4)),
 new TEdit(this, ID_M5,
          sizeof(((Ventana *)Parent)->Transfiere5.C5));
 new TEdit(this, ID_M6,
          sizeof(((Ventana *)Parent)->Transfiere5 C6));
 TransferBuffer = (void far*)
```

```
            &(((Ventana *)Parent)->Transfiere5);
}

BOOL Motor2 .CanClose()
{
 FillBuffers();
 return TRUE;
}


void Motor2::FillBuffers()
{
  GetDlgItemText(HWindow, ID_M1, Camp1, MAXC);
  GetDlgItemText(HWindow, ID_M2, Camp2, MAXC);
  GetDlgItemText(HWindow, ID_M3, Camp3, MAXC);
  GetDlgItemText(HWindow, ID_M4, Camp4, MAXC);
  GetDlgItemText(HWindow, ID_M5, Camp5, MAXC);
  GetDlgItemText(HWindow, ID_M6, Camp6, MAXC);
}
//**********************Pregunta para el motor**************

Pregunta::Pregunta(PTWindowsObject AParent, LPSTR name)
          :TDialog(AParent, name)
{

}

BOOL Pregunta::CanClose()
{
 FillBuffers();
 return TRUE;
}


void Pregunta::FillBuffers()
{

}

//********************** Constructor del Transformador ************

Transfor::Transfor(PTWindowsObject AParent, LPSTR name)
           :TDialog(AParent, name)
{
 new TEdit(this, ID_M1,
    sizeof(((Ventana *)Parent)->Transfiere6.C1));
 new TEdit(this, ID_M2,
          sizeof(((Ventana *)Parent)->Transfiere6.C2));
 new TEdit(this, ID_M3,
          sizeof(((Ventana *)Parent)->Transfiere6.C3));
 new TEdit(this, ID_M4,
          sizeof(((Ventana *)Parent)->Transfiere6.C4));
 new TEdit(this, ID_M5,
          sizeof(((Ventana *)Parent)->Transfiere6.C5));
```

```
new TEdit(this, ID_M6,
        sizeof(((Ventana *)Parent)->Transfiere6.C6));
new TEdit(this, ID_M7,
        sizeof(((Ventana *)Parent)->Transfiere6.C7));
new TEdit(this, ID_M8,
        sizeof(((Ventana *)Parent)->Transfiere6.C8));
TransferBuffer = (void far*)
            &(((Ventana *)Parent)->Transfiere6);
}

BOOL Transfor::CanClose()
{
 FillBuffers();
 return TRUE;
}


void Transfor::FillBuffers()
{
  GetDlgItemText(HWindow, ID_M1, Camp1, MAXC);
  GetDlgItemText(HWindow, ID_M2, Camp2, MAXC);
  GetDlgItemText(HWindow, ID_M3, Camp3, MAXC);
  GetDlgItemText(HWindow, ID_M4, Camp4, MAXC);
  GetDlgItemText(HWindow, ID_M5, Camp5, MAXC);
  GetDlgItemText(HWindow, ID_M6, Camp6, MAXC);
  GetDlgItemText(HWindow, ID_M7, Camp7, MAXC);
  GetDlgItemText(HWindow, ID_M8, Camp8, MAXC);
 }

 //************** Constructor de Linea ******************************

 Linea::Linea(PTWindowsObject AParent, LPSTR name)
          :TDialog(AParent, name)
 {
  new TEdit(this, ID_C31,
    sizeof(((Ventana *)Parent)->Transfiere3.C1));
  new TEdit(this, ID_C32,
          sizeof(((Ventana *)Parent)->Transfiere3.C2));
  new TEdit(this, ID_C33,
          sizeof(((Ventana *)Parent)->Transfiere3.C3));
  new TEdit(this, ID_C34,
          sizeof(((Ventana *)Parent)->Transfiere3.C4));
  new TEdit(this, ID_C35,
          sizeof(((Ventana *)Parent)->Transfiere3.C5));
  TransferBuffer = (void far*)
              &(((Ventana *)Parent)->Transfiere3);
  }

  BOOL Linea::CanClose()
  {
   FillBuffers();
   return TRUE;
  }
```

```
void Linea::FillBuffers()
{
  GetDlgItemText(HWindow, ID_C31, Camp1, MAXC);
  GetDlgItemText(HWindow, ID_C32, Camp2, MAXC);
  GetDlgItemText(HWindow, ID_C33, Camp3, MAXC);
  GetDlgItemText(HWindow, ID_C34, Camp4, MAXC);
  GetDlgItemText(HWindow, ID_C35, Camp5, MAXC);
}

void Ventana::Lin(char *L1,char *L2, char *L3,char *L4,char *L5)
{
 char NumInfo[4*MAXC+50] ;

 strcpy(L1,""); strcpy(L2,""); strcpy(L3,""); strcpy(L4,""); strcpy(L5,"");
 if ( GetModule()->ExecDialog(
            new Linea(this,"LI")) == IDOK )
 {      strcpy(L1,Transfiere3.C1);
    strcpy(L2,Transfiere3.C2);
        strcpy(L3,Transfiere3.C3);
        strcpy(L4,Transfiere3.C4);
        strcpy(L5,Transfiere3.C5);
        Guarda(L1,L2,L3,L4,L5);

 }

}

//************** Constructor de Potencia **********************

Pot::Pot(PTWindowsObject AParent, LPSTR name)
        :TDialog(AParent, name)
{
 new TEdit(this, ID_C91,
   sizeof(((Ventana *)Parent)->Transfiere2.C1));
 new TEdit(this, ID_C92,
        sizeof(((Ventana *)Parent)->Transfiere2.C2));
 TransferBuffer = (void far*)
            &(((Ventana *)Parent)->Transfiere2);
}

BOOL Pot::CanClose()
{
 FillBuffers();
 return TRUE;
}


void Pot::FillBuffers()
{
  GetDlgItemText(HWindow, ID_C91, Camp1, MAXC);
  GetDlgItemText(HWindow, ID_C92, Camp2, MAXC);
}
```

```
void Ventana::Potencia(char *L1,char *L2)
{
 char NumInfo[2*MAXC+55] ;
 strcpy(L1,""); strcpy(L2,"");
 if ( GetModule()->ExecDialog(
            new Pot(this,"BASE")) == IDOK )
 {      strcpy(L1,Transfiere2 C1),
    strcpy(L2,Transfiere2.C2);
        if(Calcula(FileName,strtod(L1,NULL),strtod(L2,NULL))==0){
          MessageBox(HWindow, "El archivo de Salida se creo con éxito", "Éxito ",
           MB_ICONEXCLAMATION);
    }
        else{
          MessageBox(HWindow, "Hubó problemas en la salida", "Error ",
           MB_ICONEXCLAMATION);
    }

 }

 }


 //****************** Guarda para  el Generador **********************

 int Ventana::Guarda(char *L1,char *L2,char *L3,char *L4,char *L5,char *L6){
  char *p=new char[strlen(L1)+strlen(L2)+strlen(L3)+strlen(L4)+strlen(L5)+strlen(L6)+20];
  strcpy(p,"g,");
  strcat(p,L1);
  strcat(p,";"),
  strcat(p,L2);
  strcat(p,";");
  strcat(p,L3);
  strcat(p,";"),
  strcat(p,L4);
  strcat(p,";");
  strcat(p,L5);
  strcat(p,";");
  strcat(p,L6);
  strcat(p,";\n");
        fstream salida(FileName, ios::in|ios::out|ios: nocreate);
        if(!salida){
                MessageBox(HWindow,"Debe especificar un archivo de datos","Error",
 MB_OK|MB_ICONEXCLAMATION);
                return 0;
         }
       else{
         salida.seekp(0,ios::end);
            while(*p)
            salida put(*p++);
            salida close();
          return 1;
```

```
    }
}
//********************** Guarda para el Motor **********************

int Ventana::Guardam(char *L1,char *L2,char *L3,char *L4,char *L5,char *L6){

  char *p=new char[strlen(L1)+strlen(L2)+strlen(L3)+strlen(L4)+strlen(L5)+strlen(L6)+20];
  strcpy(p,"m;");
  strcat(p,L1);
  strcat(p,";");
  strcat(p,L2);
  strcat(p,";");
  strcat(p,L3);
  strcat(p,";");
  strcat(p,L4);
  strcat(p,",");
  strcat(p,L5);
  strcat(p,",");
  strcat(p,L6);
  strcat(p,".\n");

        fstream salida(FileName, ios::in|ios::out|ios::nocreate);
        if(!salida){
                MessageBox(HWindow,"Debe especificar un archivo de datos","Error",
MB_OK|MB_ICONEXCLAMATION);
                return 0;
        }
     else{
        salida.seekp(0,ios::end);
            while(*p)
            salida.put(*p++);
            salida.close();
        return 1;
     }
}

//********************** Guarda para el Transformador **********************

int Ventana::Guardat(char *L1,char *L2,char *L3,char *L4,char *L5,char *L6,char *L7,char *L8){
  char *p=new char[strlen(L1)+strlen(L2)+strlen(L3)+strlen(L4)+strlen(L5)+strlen(L6)+strlen(L7)
                +strlen(L8)+20];

  strcpy(p,"t;");
  strcat(p,L1);
  strcat(p,";");
  strcat(p,L2);
  strcat(p,";");
  strcat(p,L3);
  strcat(p,",");
  strcat(p,L4);
  strcat(p,",");
  strcat(p,L5);
  strcat(p,",");
  strcat(p,L6);
```

```
  strcat(p,";");
  strcat(p,L7);
  strcat(p,";");
  strcat(p,L8);
  strcat(p,",\n");

        fstream salida(FileName, ios::in|ios::out|ios::nocreate);
        if(!salida){

                MessageBox(HWindow,"Debe especificar un archivo de datos","Error",
MB_OK|MB_ICONEXCLAMATION);
                return 0;
        }
    else{
       salida.seekp(0,ios :end);
          while(*p)
          salida put(*p++);
          salida close();
       return 1;
    }

}

//********************** Guarda para la Linea ***********************

int Ventana::Guarda(char *L1,char *L2,char *L3,char *L4,char *L5){
  char *p=new char[strlen(L1)+strlen(L2)+strlen(L3)+strlen(L4)+strlen(L5)+8];
  strcpy(p,"l;");
  strcat(p,L1);
  strcat(p,";");
  strcat(p,L2);
  strcat(p,",");
  strcat(p,L3);
  strcat(p,",");
  strcat(p,L4);
  strcat(p,";");
  strcat(p,L5);
  strcat(p,";\n");
        fstream salida(FileName, ios :in|ios::out|ios :nocreate);
        if(!salida){

                MessageBox(HWindow,"Debe especificar un archivo de datos","Error",
MB_OK|MB_ICONEXCLAMATION);
                return 0;
        }
    else{
       salida.seekp(0,ios::end);
          while(*p)
          salida.put(*p++);
          salida close();
       return 1;
    }
}
```

```
//****************** Caja de Dialogo del Generador ******************

void Ventana::Gen(char *L1,char *L2,char *L3, char *L4, char *L5,char *L6)
{
 char NumInfo[3*MAXC+100] ;
 strcpy(L1,""); strcpy(L2,""); strcpy(L3,""); strcpy(L4,""); strcpy(L5,""); strcpy(L6,"");
 if ( GetModule()->ExecDialog(
            new Generador(this,"GEN")) == IDOK)
 {
  strcpy(L1,Transfiere4.C1);
  strcpy(L2,Transfiere4.C2);
  strcpy(L3,Transfiere4.C3);
  strcpy(L4,Transfiere4.C4);
  strcpy(L5,Transfiere4.C5);
  strcpy(L6,Transfiere4.C6);
  Guarda(L1,L2,L3,L4,L5,L6);



 }
            }
}
//********************** Caja de Dialogo del Motor en HP o MVA***************

void Ventana::Mot(char *L1,char *L2,char *L3, char *L4, char *L5,char *L6="",char *L7="",char *L8="")
{
 if(GetModule()->ExecDialog(new Pregunta(this,"PREG"))==IDCANCEL){
 char NumInfo[8*MAXC+100] ;
 strcpy(L1,""); strcpy(L2,""); strcpy(L3,""); strcpy(L4,""); strcpy(L5,"");
 strcpy(L6,""); strcpy(L7,""); strcpy(L8,"");
 if ( GetModule()->ExecDialog(
            new Motor(this,"MOT")) == IDOK )
 {

  strcpy(L1,Transfiere6.C1);
  strcpy(L2,Transfiere6.C2);
  strcpy(L3,Transfiere6.C3);
  strcpy(L4,Transfiere6.C4);
  strcpy(L5,Transfiere6.C5);
  strcpy(L6,Transfiere6.C6);
  strcpy(L7,Transfiere6.C7);
  strcpy(L8,Transfiere6.C8);
  double temp=0.746*strtod(L1,NULL)/(strtod(L4,NULL)*strtod(L3,NULL));
  char fg[20];
  Guardam(dotostr(temp,4),L2,L5,L6,L7,L8);

 }

}
else{
 char NumInfo[6*MAXC+50] ;
 strcpy(L1,""); strcpy(L2,""); strcpy(L3,""); strcpy(L4,""); strcpy(L5,"");
 if ( GetModule()->ExecDialog(
            new Motor2(this,"MOT2")) == IDOK )
```

```
 {


  strcpy(L1,Transfiere5 C1);
  strcpy(L2,Transfiere5 C2);
  strcpy(L3,Transfiere5.C3);
  strcpy(L4,Transfiere5 C4);
  strcpy(L5,Transfiere5.C5);
  strcpy(L6,Transfiere5 C6);
  Guardam(L1,L2,L3,L4,L5,L6);

 }

}
}

/*****************************************/
char* Ventana::dotostr(double d,int i){
    double temp;
    char g[20],h[21];
    temp=d*pow(10,i),
    itoa(temp,g,10),
    for(int j=strlen(g),l=strlen(g)+1;j>=0,j--,l--){
            if(strlen(g)-i-1==j) h[l--]='.';
          h[l]=g[j],
    }
    return h,
}

/*****************************************
                              */
void Ventana::Trans(char *L1,char *L2,char *L3, char *L4, char *L5,char *L6,char *L7,char *L8)
{
 char NumInfo[3*MAXC+100] ;
 strcpy(L1,""); strcpy(L2,""); strcpy(L3,""), strcpy(L4,""); strcpy(L5,"");
 strcpy(L6,""); strcpy(L7,""), strcpy(L8,"");
 if ( GetModule()->ExecDialog(
             new Transfor(this,"TRAN")) == IDOK )
 {
  strcpy(L1,Transfiere6.C1);
  strcpy(L2,Transfiere6.C2),
  strcpy(L3,Transfiere6 C3),
  strcpy(L4,Transfiere6.C4);
  strcpy(L5,Transfiere6 C5);
  strcpy(L6,Transfiere6.C6);
  strcpy(L7,Transfiere6 C7);
  strcpy(L8,Transfiere6.C8);
  Guardat(L1,L2,L3,L4,L5,L6,L7,L8);
 }

}


Ventana .Ventana(PTWindowsObject AParent, LPSTR ATitle, LPSTR AFileName)
```

```
:TFileWindow(AParent, ATitle, AFileName) {

AssignMenu("COMMANDS");
alicia = LoadBitmap( GetApplication()->hInstance, "ccc" );
LineaLibre = FALSE;
Cir=FALSE;
// Lin=FALSE;
Eli=FALSE;
Arc=FALSE;
strcpy(TransferStruct.Num1,"");
strcpy(TransferStruct.Num2,"");
strcpy(TransferStruct.Oper,"");

strcpy(Transfiere2.C1,"");
strcpy(Transfiere2.C2,"");

strcpy(Transfiere4.C1,"");
strcpy(Transfiere4.C2,"");
strcpy(Transfiere4.C3,"");
strcpy(Transfiere4.C4,"");
strcpy(Transfiere4.C5,"");
strcpy(Transfiere4.C6,"");

strcpy(Transfiere5.C1,"");
strcpy(Transfiere5.C2,"");
strcpy(Transfiere5.C3,"");
strcpy(Transfiere5.C4,"");
strcpy(Transfiere5.C5,"");
strcpy(Transfiere5.C6,"");

strcpy(Transfiere3.C1,"");
strcpy(Transfiere3.C2,"");
strcpy(Transfiere3.C3,"");
strcpy(Transfiere3.C4,"");
strcpy(Transfiere3.C5,"");

strcpy(Transfiere6.C1,"");
strcpy(Transfiere6.C2,"");
strcpy(Transfiere6.C3,"");
strcpy(Transfiere6.C4,"");
strcpy(Transfiere6.C5,"");
strcpy(Transfiere6.C6,"");
strcpy(Transfiere6.C7,"");
strcpy(Transfiere6.C8,"");
PenSize = 1;
ThePen = CreatePen(PS_SOLID, PenSize, 0);
Points = new TPointArray(50, 0, 50);
yi=xi=0;
Imagen();

}

void Ventana:: Imagen(){
 hdc=GetDC(HWindow);
```

```
 DrawBMP( hdc, 211, 90,alicia);
 Duerme(3);
 Borrar();
 ReleaseDC( HWindow, hdc );
}

Ventana::~Ventana()
{
 delete Points;
 DeleteObject(ThePen);
}


void Ventana: DrawBMP( HDC DC, int X, int Y, HBITMAP BitMap )
{
        HDC MemDC;
        BITMAP bm;
        BOOL MadeDC=FALSE;

        if ( DC == 0 )
        {
                DC = GetDC( HWindow );
                MadeDC = TRUE;
        }
        else
                MadeDC = FALSE;
        MemDC = CreateCompatibleDC( DC );
        SelectObject( MemDC, BitMap );
        GetObject( alicia, sizeof( bm ), ( LPSTR ) &bm );
        BitBlt( DC, X, Y, bm.bmWidth, bm.bmHeight, MemDC, 0, 0, SRCCOPY );
        DeleteDC( MemDC );
        if ( MadeDC )
                ReleaseDC( HWindow, DC );
}


void Ventana::About()
{
 GetApplication()->ExecDialog(new TDialog(this, "ABOUT"));
};

void Ventana  SetPenSize(int NewSize)
{
 DeleteObject(ThePen);
 ThePen = CreatePen(PS_SOLID, NewSize, 0);
 PenSize = NewSize;
}

BOOL Ventana  CanClose()
{
 return MessageBox(HWindow, "Esto terminará con su sesión ",
 "Salir de Corto Circuito" , MB_YESNO | MB_ICONEXCLAMATION) == IDYES;
 }
```

```
void Ventana::Mensaje(char *titulo,char *mensaje){
   MessageBox(HWindow,titulo,mensaje, MB_OK);
}

void Ventana::Borrar(void){
            InvalidateRect(HWindow,NULL,TRUE);
        contador=contadorx=contadory=0;
}

void Ventana::Salir(void){ exit(0); }


void Ventana::WMMouseMove(RTMessage Msg)
{


}

void Ventana::WMLButtonUp(RTMessage)
{

}

void Ventana::WMRButtonUp(RTMessage)
{

}


int& Ventana::Pide(char *titulo, char *mensaje, int &a){
 char InputText[6];
  sprintf(InputText, "%d");
  if ( GetApplication()->ExecDialog(new TInputDialog(this, titulo,
          mensaje, InputText, sizeof InputText)) == IDOK )
  {
     a = atoi(InputText);
     return a;
  }

}

float& Ventana: Pide(char *titulo, char *mensaje, float &flotante){
 char InputText[6];
  sprintf(InputText, "%f");
  if ( GetApplication()->ExecDialog(new TInputDialog(this, titulo,
    mensaje, InputText, sizeof InputText)) == IDOK )
  {
     flotante = atof(InputText);
     return flotante;


  }

}
```

```
char& Ventana::Pide(char *titulo, char *mensaje, char *cad1){
 char cad[100];
 sprintf(cad,"%c"),
 if ( GetApplication()->ExecDialog(new TInputDialog(this, titulo,
  mensaje,cad,sizeof cad)) == IDOK )
 {
   strcpy(cad1,cad),
    return *cad1;
 }

}



void Ventana..CMHelp(RTMessage)
{
 MessageBox(HWindow, "No implementado", "Ayuda", MB_OK);
}

void TMyApp::InitMainWindow()
{
 MainWindow = new Ventana(NULL, Name,"");
}

void Ventana::Despliega(char *cadena){
 HDC DC;
 char S[100];
  if(contador>=80)
                  {
                   contadory++;
                   contadorx=0,
 }
 sprintf(S, "%s",cadena),

 DC = GetDC(HWindow);
 TextOut(DC, contadorx, contadory, S, strlen(cadena));
 ReleaseDC(HWindow, DC);
 Salta();
}

 void Ventana.::Despliega(char cadena){
 HDC DC,
 char S[100];
  if(contador>=80)
                  {
                   contadory++;
                   contadorx=0,
 }
 sprintf(S, "%c",cadena);
 DC = GetDC(HWindow).
 TextOut(DC, contadorx, contadory, S, strlen(S));
 ReleaseDC(HWindow, DC),
 Salta(),
```

```
}

void Ventana::Despliega(int cadena){
  HDC DC;
 char S[16];
  if(contador>=80)
                    {
                    contadory++;
                    contadorx=0;
  }
 sprintf(S, "%i",cadena);
 DC = GetDC(HWindow);
 TextOut(DC, contadorx, contadory, S, strlen(S));
 ReleaseDC(HWindow, DC);
 Salta();
}

void Ventana::Despliega(complex cadena){
  HDC DC;
 char S[50];
 sprintf(S, "%lf+i %lf",real(cadena),imag(cadena));
 DC = GetDC(HWindow);
 TextOut(DC, contadorx, contadory, S, strlen(S));
 ReleaseDC(HWindow, DC);
 Salta();
}

void Ventana::Despliega(float cadena){
   char *lcr,*numero; int a,b;
          lcr=new char [100];
          numero=new char [100];
          numero=fcvt(cadena,10,&a,&b);
          int i=0;
          if(b){ lcr[i]='-'; i++; }
          for(;i<a;i++)
                    lcr[i]=numero[i];
          lcr[i]='.';
          for(;i<=(strlen(numero)-a);i++)
                    lcr[i+1]=numero[i];

          if(contador>=80)
                    {
                    contadory++;
                    contadorx=0;
                    }
  HDC DC;
  if(contador>=80)
                    {
                    contadory++;
                    contadorx=0;
  }
  sprintf(lcr, "%f",cadena);
  DC = GetDC(HWindow);
  TextOut(DC, contadorx, contadory, lcr, strlen(lcr));
```

```
 ReleaseDC(HWindow, DC);
 Salta();
}

void Ventana :Salta(){ contadorx=0; contadory+=20;}

void Ventana::Dibuja_Linea(RTMessage Msg){
Points->flush();

 if ( !LineaLibre )
 {
  LineaLibre = TRUE;
  SetCapture(HWindow);
        DragDC = GetDC(HWindow);
  SelectObject(DragDC, ThePen);
  MoveTo(DragDC, Msg.LP.Lo, Msg.LP.Hi);
        Points->add(* (new TPoint(Msg.LP.Lo, Msg.LP Hi)));
 }
}

void Ventana :Circulo(int x,int y,int radio)
{
  if ( !Cir )
  {

    Cir = TRUE;
          SetCapture(HWindow);
          hdc = GetDC(HWindow);
          Ellipse(hdc,x,y,radio,radio);
  }
}



void Ventana::X(){
    HDC DC;
    DC = GetDC(HWindow);
    MoveTo(DC,0,0);
    LineTo(DC,SW_SIZE,SW_SIZE);
    MoveTo(DC,SW_SIZE,0);
    LineTo(DC,0,SW_SIZE);
}


void Ventana::Elipse(int a,int b,int h,int k){
    HDC DC;
    DC = GetDC(HWindow);
    Ellipse(DC,-a+h,b+k,a+h,-b+k);
}

void Ventana :Rectangulo(int x1,int x2,int x3,int x4){
          HDC DC;
          DC=GetDC(HWindow);
          Rectangle(DC,x1,x2,x3,x4);
```

```
}

void Ventana::RectanguloBR(int x1,int y1, int x2, int y2, int r,int e){
    HDC DC;
        DC=GetDC(HWindow);
        RoundRect(DC,x1,y1,x2,y2,r,e);
}


void Ventana::Arco(int xa,int ya,int xb,int yb,int xc,int yc,int xd,int yd){

    HDC DC;
        DC = GetDC(HWindow);
        Chord(DC,xa,ya,xb,yb,xc,yc,xd,yd);
}


void
FunBotIzq(RTMessage Msg) /*Función Boton Izquierdo*/
{



}

void
FunBotDer(RTMessage Msg) /*Función Boton Derecho*/
{
 Despliega(dotostr(3.1415,4));

}


void
f3(){
 char Y[20],O[20],V[20],I[20],N[20];
 Lin(Y,O,V,I,N);

}
void
f4(){
 char Y[20],O[20],V[20],I[20],N[20],A[20];
 Gen(Y,O,V,I,N,A);
}
void
f5(){
 char Y[20],O[20],V[20],I[20],N[20],A[20],M[20];
 Mot(Y,O,V,I,N,A,M);
}
void
f6(){
 char Y[20],O[20],V[20],I[20],N[20],A[20],M[20],E[20];
 Trans(Y,O,V,I,N,A,M,E);
}
```

```
void
f7(){
 char Y[20],O[20];
 Potencia(Y,O);
}


main()
{

 CreaVentana("Tesis", NULO),
 AbreVentana();
}
```

Continene todas los elementos gráficos para la creación de los menús, la creación de las cajas de diálogo, iconos, BITMAP etc Es muy importante para la programación de aplicaciones en WINDOWS.

```
/*

            ZBUS RC


*/
#define CM_U_HELPINDEX          0x200
#define CM_U_HELPHELP           0x201

#define CM_HELP  201
#define CM_BORRAR 202
#define CM_SALIR  203
#define CM_CAJA  1110
#define CM_SONIDO 3333

//#define FunBotIzq Ventana.:WMLButtonDown
//#define FunBotDer Ventana::WMRButtonDown

//#define NULO hInstance, hPrevInstance,lpCmdLine. nCmdShow
//#define CreaVentana TMyApp MyApp
//#define AbreVentana();  MyApp.Run(); return MyApp.Status,

//#define main() int PASCAL WinMain(HINSTANCE hInstance. HINSTANCE hPrevInstance,LPSTR
lpCmdLine, int nCmdShow)
#define CM_ALUM1 801
#define CM_ALUM2 802
#define CM_ALUM3 803

#define CM_GEN 804
#define CM_MOT 805
#define CM_TRAN 806

#define ID_M1  901
#define ID_M2  902
#define ID_M3  903
```

```
#define ID_M4  904
#define ID_M5  905
#define ID_M6  906
#define ID_M7  907
#define ID_M8  908

#define ID_C41 401
#define ID_C42 402
#define ID_C43 403
#define ID_C44 404
#define ID_C45 333
#define ID_C46 334

#define ID_C31 405
#define ID_C32 406
#define ID_C33 407
#define ID_C34 408
#define ID_C35 409

#define ID_C21 408
#define ID_C22 409

#define ID_C11 410

#define CM_NUMINPUT 301
#define ID_NUM1 101
#define ID_NUM2 105
#define ID_OP 107
#define CM_CIR 401
#define CM_ELI 402
#define CM_LINEA 403
#define CM_ARCO 404

#define MAXNUM 15
#define MAXOP 2
#define MAX 50
#define MAXC 60

#define CM_F1 501
#define CM_F2 502
#define CM_F3 503
#define CM_F4 504
#define CM_F5 505
#define CM_F6 506
#define CM_F7 507
#define CM_F8 508
#define CM_F9 509
#define CM_F10 510

#define MAXPATH 80
#define ID_C01  1002
#define ID_C141 1001
#define ID_C142 1002
#define CM_GEN 804
```

```
#define CM_TRAN 806

#define CM_CAJA 1110

#define ID_C1 401
#define ID_C2 402
#define ID_C3 403
#define ID_C4 404
#define ID_C5 405
#define CM_NUMINPUT 301
#define ID_NUM1 101
#define ID_NUM2 105
#define ID_OP 107
#define CM_HELP  201
#define CM_BORRAR 202
#define CM_SALIR  203
#define CM_IMAGEN 666
#define CM_ESCOGE 777

#define CM_F1 501
#define CM_F2 502
#define CM_F3 503
#define CM_F4 504
#define CM_F5 505
#define CM_F6 506
#define CM_F7 507
#define CM_F8 508
#define CM_F9 509
#define CM_F10 510

#define CM_ABOUT 511

#include <windows h>
#include <owlrc.h>

#include "c \caball~1\tesis\febrero\menus\inputdia.dlg"
#include "c \caball~1\tesis\febrero\menus\filedial.dlg"
#include "c \caball~1\tesis\febrero\menus\stdwnds.dlg"
//#include <inputdia dlg>
//#include <filedial dlg>
//#include "stdwnds.dlg"

#include "editacc.rc"
#include "fileacc.rc"
#include "editmenu rc"

#define ID_C41 401
#define ID_C42 402
#define ID_C43 403
#define ID_C44 404
#define ID_C45 333
#define ID_C46 334

#define ID_C31 405
```

```
#define ID_C32 406
#define ID_C33 407
#define ID_C34 408


//#define CM_ABOUT 908

#define ID_M1  901
#define ID_M2  902
#define ID_M3  903
#define ID_M4  904
#define ID_M5  905
#define ID_M6  906
#define ID_M7  907

#define cam31 "Z1="
#define cam32 "Z2="
#define cam33 "Z0="
#define cam34 "bus"
/*

COMMANDS  MENU LOADONCALL MOVEABLE PURE DISCARDABLE
BEGIN
 POPUP "&Archivo"
 BEGIN
  MenuItem  "&Nuevo", CM_FILENEW
  MenuItem  "&Abrir...", CM_FILEOPEN
  MenuItem  "&Salvar", CM_FILESAVE
  MenuItem  "Salvar c&omo...", CM_FILESAVEAS
  MenuItem  SEPARATOR
  MenuItem  "S&alir", CM_EXIT
 END
 POPUP "&Editar"
 BEGIN
  MenuItem  "&Deshacer\aAlt+BkSp", CM_EDITUNDO
  MenuItem  SEPARATOR
  MenuItem  "&Cortar\aShift+Del", CM_EDITCUT
  MenuItem  "C&opiar\aCtrl+Ins", CM_EDITCOPY
  MenuItem  "&Pegar\aShift+Ins", CM_EDITPASTE
  MenuItem  "&Borrar\aDel", CM_EDITDELETE
  MenuItem  "L&impiar todo\aCtrl+Del", CM_EDITCLEAR
 END
 POPUP "&Buscar"
 BEGIN
  MenuItem  "&Encontrar...", CM_EDITFIND
  MenuItem  "&Remplazar ..", CM_EDITREPLACE
  MenuItem  "&Siguiente\aF3", CM_EDITFINDNEXT
 END
 POPUP "A&yuda"
        BEGIN
                MENUITEM "&Index\aShift+F1", CM_U_HELPINDEX
                MENUITEM "&Using help", CM_U_HELPHELP
                MENUITEM SEPARATOR
                MENUITEM "Acerca de ", CM_ABOUT
```

```
    END
      MENUITEM "&Imagen", CM_IMAGEN
      POPUP "&Elemento"
      BEGIN
       MENUITEM "Selecciona Archivo...",CM_ESCOGE
            MenuItem  SEPARATOR
            MENUITEM "Generador", CM_F4, GRAYED
            MENUITEM "Motor", CM_F5, GRAYED
            MENUITEM "Linea", CM_F3, GRAYED
            MENUITEM "Transformador", CM_F6, GRAYED
      END
END


*/

COMMANDS MENU
BEGIN
      POPUP "&Archivo"
      BEGIN
            MENUITEM "&Nuevo", CM_FILENEW
            MENUITEM "&Abrir ..", CM_FILEOPEN
            MENUITEM "&Salvar", CM_FILESAVE
            MENUITEM "Salvar c&omo.. ", CM_FILESAVEAS
            MENUITEM SEPARATOR
            MENUITEM "S&alir", CM_EXIT
      END

      POPUP "&Editar"
      BEGIN
            MENUITEM "&Deshacer\aAlt+BkSp", CM_EDITUNDO
            MENUITEM SEPARATOR
            MENUITEM "&Cortar\aShift+Del", CM_EDITCUT
            MENUITEM "C&opiar\aCtrl+Ins", CM_EDITCOPY
            MENUITEM "&Pegar\aShift+Ins", CM_EDITPASTE
            MENUITEM "&Borrar\aDel", CM_EDITDELETE
            MENUITEM "L&impiar todo\aCtrl+Del", CM_EDITCLEAR
      END

      POPUP "&Buscar"
      BEGIN
            MENUITEM "&Encontrar...", CM_EDITFIND
            MENUITEM "&Remplazar...", CM_EDITREPLACE
            MENUITEM "&Siguiente\aF3", CM_EDITFINDNEXT
      END

      MENUITEM "&Imagen", CM_IMAGEN
      POPUP "&Elemento"
      BEGIN
            MENUITEM "Selecciona Archivo. .", CM_ESCOGE
            MENUITEM SEPARATOR
            MENUITEM "Generador", CM_F4, GRAYED
            MENUITEM "Motor", CM_F5, GRAYED
            MENUITEM "Linea", CM_F3, GRAYED
```

```
            MENUITEM "Transformador", CM_F6, GRAYED
        END

        POPUP "A&yuda", HELP
        BEGIN
            MENUITEM "Acerca de ", CM_ABOUT
            MENUITEM SEPARATOR
            MENUITEM "&Indice\aShift+F1", CM_U_HELPINDEX
            MENUITEM "&Uso de Ayuda", CM_U_HELPHELP
        END

END


GEN DIALOG 49, 17, 202, 136
STYLE WS_CAPTION | WS_SYSMENU | DS_MODALFRAME | WS_POPUP
CAPTION "Generador"
BEGIN
        EDITTEXT ID_C41, 57, 11, 48, 14
        EDITTEXT ID_C42, 57, 31, 48, 15
        EDITTEXT ID_C43, 140, 9, 48, 17
        EDITTEXT ID_C44, 141, 31, 53, 15, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C45, 143, 54, 50, 15
        EDITTEXT ID_C46, 63, 80, 48, 15, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        CONTROL "potencia", 106, "static", SS_RIGHT | WS_CHILD, 7, 12, 30, 10
        CONTROL "voltaje", 106, "static", SS_RIGHT | WS_CHILD, 8, 33, 30, 13
        CONTROL "Z2=", 106, "static", SS_RIGHT | WS_CHILD, 122, 33, 13, 10
        CONTROL "Z1=", 106, "static", SS_RIGHT | WS_CHILD, 122, 12, 14, 11
        CONTROL "Z0=", 106, "static", SS_RIGHT | WS_CHILD | WS_VISIBLE, 122, 57, 13, 8
        CONTROL "bus", 106, "static", SS_RIGHT | WS_CHILD | WS_VISIBLE, 39, 86, 13, 8
        ICON "ICON_3", -1, 15, 60, 16, 16, WS_CHILD | WS_VISIBLE
        DEFPUSHBUTTON "&Aceptar", IDOK, 46, 113, 29, 14, WS_CHILD | WS_VISIBLE |
WS_TABSTOP
        PUSHBUTTON "&Cancelar", IDCANCEL, 128, 113, 33, 14, WS_CHILD | WS_VISIBLE |
WS_TABSTOP
END

MOT DIALOG 27, 29, 213, 166
STYLE WS_CAPTION | WS_SYSMENU | DS_MODALFRAME | WS_POPUP
CAPTION "Motor"
BEGIN
        EDITTEXT ID_M1, 48, 12, 48, 14, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_M2, 48, 34, 48, 15, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_M3, 49, 58, 48, 13, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_M4, 50, 80, 47, 13, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_M5, 138, 13, 48, 15, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
```

```
        EDITTEXT ID_M6, 138, 32, 48, 15, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_M7, 138, 54, 48, 15, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_M8, 145, 101, 39, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        CONTROL "potencia", 106, "static", SS_RIGHT | WS_CHILD, 12, 15, 28, 10
        CONTROL "voltaje", 106, "static", SS_RIGHT | WS_CHILD, 13, 38, 24, 9
        CONTROL "Z0=", 106, "static", SS_RIGHT | WS_CHILD | WS_VISIBLE, 115, 56, 14, 8
        CONTROL "Z1=", 106, "static", SS_RIGHT | WS_CHILD, 114, 16, 16, 9
        CONTROL "Z2=", 106, "static", SS_RIGHT | WS_CHILD | WS_VISIBLE, 114, 37, 15, 8
        CONTROL "Eficiencia", 106, "static", SS_RIGHT | WS_CHILD, 11, 61, 32, 8
        LTEXT "bus", -1, 114, 105, 16, 8, WS_CHILD | WS_VISIBLE | WS_GROUP
        ICON "ICON_2", -1, 14, 102, 16, 16, WS_CHILD | WS_VISIBLE
        LTEXT "fp", -1, 18, 84, 8, 8, WS_CHILD | WS_VISIBLE | WS_GROUP
        DEFPUSHBUTTON "&Aceptar", IDOK, 51, 140, 30, 13, WS_CHILD | WS_VISIBLE |
WS_TABSTOP
        PUSHBUTTON "&Cancelar", IDCANCEL, 126, 140, 33, 14, WS_CHILD | WS_VISIBLE |
WS_TABSTOP
END

MOT2 DIALOG 27, 29, 213, 122
STYLE WS_CAPTION | WS_SYSMENU | DS_MODALFRAME | WS_POPUP
CAPTION "Motor"
BEGIN
        EDITTEXT ID_M1, 48, 12, 48, 14, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_M2, 48, 34, 48, 15, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_M3, 138, 13, 48, 15, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_M4, 138, 32, 48, 15, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_M5, 140, 54, 48, 15, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_M6, 86, 79, 39, 12
        CONTROL "potencia", 106, "static", SS_RIGHT | WS_CHILD, 12, 15, 28, 10
        CONTROL "voltaje", 106, "static", SS_RIGHT | WS_CHILD, 13, 38, 24, 9
        CONTROL "Z0=", 106, "static", SS_RIGHT | WS_CHILD | WS_VISIBLE, 115, 56, 14, 8
        CONTROL "Z1=", 106, "static", SS_RIGHT | WS_CHILD, 114, 16, 16, 9
        CONTROL "Z2=", 106, "static", SS_RIGHT | WS_CHILD | WS_VISIBLE, 114, 37, 15, 8
        LTEXT "bus", -1, 58, 80, 16, 8, WS_CHILD | WS_VISIBLE | WS_GROUP
        ICON "ICON_2", -1, 15, 62, 16, 16, WS_CHILD | WS_VISIBLE
        DEFPUSHBUTTON "&Aceptar", IDOK, 40, 104, 29, 14, WS_CHILD | WS_VISIBLE |
WS_TABSTOP
        PUSHBUTTON "&Cancelar", IDCANCEL, 130, 105, 34, 14, WS_CHILD | WS_VISIBLE |
WS_TABSTOP
END

PREG DIALOG 27, 17, 213, 77
STYLE WS_CAPTION | WS_SYSMENU | DS_MODALFRAME | WS_POPUP
CAPTION "Potencia del Motor"
BEGIN
```

```
        CONTROL "\277La potencia es Aparente o Real ?", 106, "static", SS_RIGHT | WS_CHILD |
WS_VISIBLE, 42, 17, 115, 24
        DEFPUSHBUTTON "Aparente", IDOK, 41, 42, 33, 14, WS_CHILD | WS_VISIBLE |
WS_TABSTOP
        PUSHBUTTON "Real", IDCANCEL, 132, 40, 27, 14, WS_CHILD | WS_VISIBLE |
WS_TABSTOP
END

TRAN DIALOG 15, 17, 225, 158
STYLE WS_CAPTION | WS_SYSMENU | DS_MODALFRAME | WS_POPUP
CAPTION "Transformador"
BEGIN
        EDITTEXT ID_M1, 48, 12, 48, 14
        CONTROL "potencia", 106, "static", SS_RIGHT | WS_CHILD, 12, 15, 28, 10
        CONTROL "VAT", 106, "static", SS_RIGHT | WS_CHILD, 18, 39, 13, 7
        EDITTEXT ID_M2, 48, 34, 48, 15
        EDITTEXT ID_M3, 48, 56, 48, 15
        CONTROL "VBT", 106, "static", SS_RIGHT | WS_CHILD, 18, 59, 14, 10
        CONTROL "Z1=", 106, "static", SS_RIGHT | WS_CHILD | WS_VISIBLE, 113, 16, 16, 8
        EDITTEXT ID_M4, 138, 13, 48, 15
        CONTROL "Z2=", 106, "static", SS_RIGHT | WS_CHILD, 113, 36, 16, 8
        EDITTEXT ID_M5, 138, 32, 48, 15
        CONTROL "Z0=", 106, "static", SS_RIGHT | WS_CHILD | WS_VISIBLE, 116, 58, 13, 8
        EDITTEXT ID_M6, 138, 54, 48, 15
        EDITTEXT ID_M7, 51, 84, 39, 15
        LTEXT "Bus AT", -1, 14, 88, 24, 8, WS_CHILD | WS_VISIBLE | WS_GROUP
        EDITTEXT ID_M8, 142, 85, 39, 15
        LTEXT "Bus BT", -1, 109, 87, 24, 8, WS_CHILD | WS_VISIBLE | WS_GROUP
        ICON "ICON_5", -1, 15, 114, 16, 16, WS_CHILD | WS_VISIBLE
        DEFPUSHBUTTON "&Aceptar", IDOK, 60, 116, 30, 14, WS_CHILD ! WS_VISIBLE |
WS_TABSTOP
        PUSHBUTTON "&Cancelar", IDCANCEL, 134, 115, 34, 14, WS_CHILD | WS_VISIBLE |
WS_TABSTOP
END

LI DIALOG 31, 33, 198, 114
STYLE WS_CAPTION | WS_SYSMENU | DS_MODALFRAME | WS_POPUP
CAPTION "Linea"
BEGIN
        EDITTEXT ID_C31, 48, 7, 48, 15
        EDITTEXT ID_C32, 48, 31, 48, 15
        EDITTEXT ID_C33, 49, 55, 47, 14, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER '
WS_GROUP | WS_TABSTOP
        CONTROL "", ID_C34, "EDIT", ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_GROUP | WS_TABSTOP, 136, 10, 39, 12
        EDITTEXT ID_C35, 137, 33, 39, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        CONTROL "Z1", 106, "static", SS_RIGHT | WS_CHILD | WS_VISIBLE, 28, 12, 9, 10
        CONTROL "Z2", 106, "STATIC", SS_RIGHT | WS_CHILD | WS_VISIBLE, 29, 34, 9, 9
        LTEXT "bus 1", -1, 110, 11, 19, 8
        LTEXT "bus 2", -1, 111, 34, 20, 8
        ICON "ICON_4", -1, 16, 75, 16, 16, WS_CHILD | WS_VISIBLE
        LTEXT "Z0", -1, 30, 57, 10, 8, WS_CHILD | WS_VISIBLE | WS_GROUP
```

```
        DEFPUSHBUTTON "&Aceptar", IDOK, 55, 82, 29, 13, WS_CHILD | WS_VISIBLE |
WS_TABSTOP
        PUSHBUTTON "&Cancelar", IDCANCEL, 123, 82, 32, 14, WS_CHILD | WS_VISIBLE |
WS_TABSTOP
END
/*
BASE DIALOG 34, 29, 168, 114
STYLE WS_CAPTION | WS_SYSMENU | DS_MODALFRAME | WS_POPUP
CLASS "BorDlg"
CAPTION "Caracteristicas del Sistema"
BEGIN
        EDITTEXT ID_C91, 57, 11, 48, 14, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C92, 57, 43, 48, 15, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C93, 58, 79, 48, 17, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        CONTROL "", IDOK, BUTTON_CLASS, BS_DEFPUSHBUTTON | WS_CHILD | WS_VISIBLE |
WS_TABSTOP, 126, 10, 37, 24
        CONTROL "", IDCANCEL, BUTTON_CLASS, BS_PUSHBUTTON | WS_CHILD | WS_VISIBLE
| WS_TABSTOP, 127, 53, 36, 24
        CONTROL "potencia base", 106, "static", SS_RIGHT | WS_CHILD | WS_VISIBLE, 7, 12, 41, 17
        CONTROL "voltaje base (bus 1)", 106, "static", SS_RIGHT | WS_CHILD | WS_VISIBLE, 9, 44,
43, 19
        CONTROL "num. de buses", 106, "static", SS_RIGHT | WS_CHILD | WS_VISIBLE, 16, 79, 29, 16
        CONTROL "", 107, "BorShade", BSS_GROUP | WS_CHILD | WS_VISIBLE, 8, 11, 47, 24
        CONTROL "", 108, "BorShade", BSS_GROUP | WS_CHILD | WS_VISIBLE, 8, 41, 46, 27
        CONTROL "", 109, "BorShade", BSS_GROUP | WS_CHILD | WS_VISIBLE, 10, 76, 43, 24
END
*/
ABOUT DIALOG 18, 23, 142, 136
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Acerca de..."
BEGIN
        CONTROL "Lorenzo Caballero Rosas", 103, "STATIC", SS_CENTER | WS_CHILD |
WS_VISIBLE, 24, 53, 99, 10
        CONTROL "Copyright \251 1999", 104, "STATIC", SS_CENTER | WS_CHILD | WS_VISIBLE,
22, 71, 101, 10
        CONTROL "UNAM FI DIE", 108, "STATIC", SS_CENTER | WS_CHILD | WS_VISIBLE, 23, 81,
103, 9
        CTEXT "Corto Circuito", -1, 25, 7, 92, 8, WS_CHILD | WS_VISIBLE | WS_GROUP
        CTEXT "Carlos Daniel Ramirez Briseño", -1, 21, 63, 103, 8, WS_CHILD | WS_VISIBLE |
WS_GROUP
        CONTROL "ICON_1", -1, "STATIC", SS_ICON | WS_CHILD | WS_VISIBLE | WS_GROUP, 63,
25, 16, 16
        DEFPUSHBUTTON "&OK", IDOK, 63, 102, 27, 14
END

// codigo de los iconos...
```

Contiene todo lo concerniente a la generación del archivo de salida y el análisis del archivo de entrada, veasé que se usan archivos de forma plana, ya que tendrán la ventaja de poderse transmitidos y interpretados en cualquier computadoras que contenga el software de cálculo de corriente de corto circuito

```
/*
        ZARCH.CPP

*/

/*#include "c:\caball~1\tesis\abril\zarch\archi.cpp"
#include "c.\caball~1\tesis\abril\zarch\ybus cpp"*/
#include "c \tsep\abril\zarch\ybus.cpp"
#include "c:\tsep\abril\zarch\archi cpp"
#include "c \tsep\abril\zarch\corto.cpp"
#include <fstream.h>
#include <ctype h>
class GeneradorA;
class MotorA;
class TransformadorA;
class LineaA;

class zarchivo: public archivo{
     int g,m,t,l,buses;
     int verifica();
   public:
     void cuenta();
     zarchivo(char *L):archivo(L){
                    buses=g=m=t=l=0.
                    verifica();
     }
     friend char * pedazo(char *,int&);
     char * renglon(double &);
     friend void imprime(zarchivo F);
     int gen(){return g;}
     int mot(){return m.}
     int tra(){return t;}
     int lin(){return l;}
     int bus().
     GeneradorA ArmaG(char *).
};

void imprime(zarchivo F){
   clrscr();
   cout<<"generadores: "<<F g;
   cout<<"\nmotores:    "<<F.m.
   cout<<"\ntransformadores. "<<F t.
   cout<<"\nlineas:      "<<F.l:
   getche();
}

int zarchivo::verifica(){
   ifstream aut(arch);
```

```
  if(!aut){
          cerr<<"No puedo abrir el archivo";
    exit(0);
   }
   char chayo[100];
   int respuesta=0;
   while(aut){
        aut getline(chayo,sizeof(chayo));
        for(int i=1;i<strlen(chayo);i++){
            if(isalpha(chayo[i])) respuesta++;
            else
                if(isdigit(chayo[i])){
                }
                else{
                  if(chayo[i]=='('|| chayo[i]==')'||
                              chayo[i]==';'|| chayo[i]==','|| chayo[i]=='.'){}
                  else{
                       respuesta++;
                  }
                }
        }
   }
   aut close();
 // cout<<respuesta;
   return respuesta;
}

void zarchivo::cuenta(){
  ifstream aut(arch);
  if(!aut){
          cerr<<"No puedo abrir el archivo";
    exit(0);
   }
   char chayo;
   while(aut){
     char buffer[100];
     aut getline(buffer,sizeof(buffer));
     if(buffer[0]=='g') g++;
     if(buffer[0]=='m') m++;
     if(buffer[0]=='t') t++;
     if(buffer[0]=='l') l++;
   }
   aut.close();
}

int zarchivo::bus(){
  ifstream aut(arch);
  if(!aut){
     cerr<<"No puedo abrir el archivo";
     exit(0);
   }
   while(aut){
     char buffer[100],temp[20];
     int chayo;
```

```
    aut.getline(buffer,sizeof(buffer));
    chayo=strlen(buffer)-2;
          buffer[chayo];
    for(int i=chayo;buffer[i]!=';';i--){}
    for(;i<=chayo;i++)
          temp[i-chayo]=buffer[i];
    temp[i-chayo]='\0';
  //  cout<<temp;
  //  getche();
    if(atoi(temp)>buses) buses=atoi(temp);
   }
 // cout<<"\nEl mayor es "<<buses;
  aut.close();
  return buses;
}


char * zarchivo::renglon(double &j){
  fstream aut(arch,ios::out|ios::in);
  char buffer[100];
  if(!aut){
    cerr<<"No puedo abrir el archivo";
    getche();
    exit(0);
   }
  aut.seekg(j,ios::beg);
  aut.getline(buffer,sizeof(buffer));
  j=aut.tellg();
       aut.close();
  // getche();
  return buffer;
}

char* pedazo(char *l,int &inicio){
        char *temp=new char[20];
   strcpy(temp,"");
   for(int i=inicio;l[i]!=':';i++){
         temp[i-inicio]=l[i];
   }
   temp[i-inicio]='\0';
   inicio=i+1;
        //cout<<temp;
   //getche();
   return temp;

}

double atod(char *l){
   char  *endptr;
   return strtod(l,&endptr);
}

complex atocom(char *l){
   char im[20],re[20];
```

```
    strcpy(im,"");
          strcpy(re,"");
    if(l[0]!='(')
    {
          strcpy(re,"0");
          strcpy(im,l);
    }
    else{
    for(int i=1;l[i]!=',';i++)
          re[i-1]=l[i];
    re[i-1]='\0';
    int j=i;
    for(/*i=strlen(re)+1*/;l[i]!=')';i++)
          im[i-j]=l[i+1];
          im[i]='\0';
    }
    char *end1,*end2;
    return complex(strtod(re,&end1),strtod(im,&end2));
}

TransformadorA ArmaT(char *l){
   complex Z1, Z2,Z0;
   double VAT, VBT,S,busAT,busBT;
   int inicio=2;
   S=atod(pedazo(l,inicio));
   VAT=atod(pedazo(l,inicio));
   VBT=atod(pedazo(l,inicio));
          Z1=atocom(pedazo(l,inicio));
   Z2=atocom(pedazo(l,inicio));
   Z0=atocom(pedazo(l,inicio));
   busAT=atod(pedazo(l,inicio));
   busBT=atod(pedazo(l,inicio));
          return TransformadorA(Z1,Z2,Z0,VAT,VBT,S,busAT,busBT);

}

LineaA ArmaL(char *l){
          double bus1,bus2;
          int inicio=2;
          complex Z1,Z2,Z0;
          Z1=atocom(pedazo(l,inicio));
          Z2=atocom(pedazo(l,inicio));
          Z0=atocom(pedazo(l,inicio));
          bus1=atod(pedazo(l,inicio));
          bus2=atod(pedazo(l,inicio));
    //    cout<<Z1<<" "<<Z2<<" "<<Z0<<" "<<bus1<<" "
    //      <<bus2;
    //    getche();
          return LineaA(Z1,Z2,Z0,bus1,bus2);
}

GeneradorA ArmaG(char *l){
   complex Z1,Z2,Z0;
          double S,VN,bus;
```

```
    int inicio=2;
    S=atod(pedazo(l,inicio));
    VN=atod(pedazo(l,inicio));
    Z1=atocom(pedazo(l,inicio));
    Z2=atocom(pedazo(l,inicio));
    Z0=atocom(pedazo(l,inicio));
    bus=atod(pedazo(l,inicio));
    return GeneradorA(Z1,Z2,Z0,S,VN,bus);
}

MotorA ArmaM(char *l){
    complex Z1,Z2,Z0;
        double S,VN,bus;
    int inicio=2;
    S=atod(pedazo(l,inicio));
    VN=atod(pedazo(l,inicio));
    Z1=atocom(pedazo(l,inicio));
    Z2=atocom(pedazo(l,inicio));
    Z0=atocom(pedazo(l,inicio));
    bus=atod(pedazo(l,inicio));
        return MotorA(Z1,Z2,Z0,S,VN,bus);
}

int Calcula(char *FileName,double SB,double VB){
            char *s=new char[strlen(FileName)+1];
          // strcpy(s,"c:\\salida lin");
            strcpy(s,"");
            strncpy(s,FileName,strlen(FileName)-3);
            s[strlen(FileName)-3]='\0';
            strncat(s,"ccc",3);
    fstream salida(s,ios::out|ios::trunc);
    if(!salida)
                return -2;
    double b,g,t,l,m,cg=0,ct=0,cl=0,cm=0;
    char p[40];
        zarchivo Prueba(FileName);
        Prueba.cuenta();
        Bus *B;
        GeneradorA *G=new GeneradorA[g=Prueba.gen()];
        TransformadorA *T=new TransformadorA[t=Prueba.tra()];
        LineaA *L=new LineaA[l=Prueba.lin()];
        MotorA *M=new MotorA[m=Prueba.mot()];
        double j=0;
        for(int i=0;i<l+m+t+g;i++){
         strcpy(p,Prueba.renglon(j));
         if(p[0]=='g'){
                G[cg]=ArmaG(p);
                cg++;
         }

         if(p[0]=='m'){
                M[cm]=ArmaM(p);
                cm++;
         }
```

```
        if(p[0]=='t'){
                T[ct]=ArmaT(p);
                ct++;
        }
        if(p[0]=='l'){
                L[cl]=ArmaL(p);
                cl++;
        }
        }
        b=Prueba.bus();
        salida<<"Buses "<<b<<"\n";
        B=new Bus[b];
        for(i=0;i<b;i++)
           B[i].asigna_ID(i+1);
        for(i=0;i<g;i++)
          salida<<G[i];
        for(i=0;i<t;i++)
           salida<<T[i];
        for(i=0;i<l;i++)
           salida<<L[i];
        for(i=0;i<m;i++)
           salida<<M[i];
        comp_simetricas(G,T,L,M,B,g,t,l,m,b,SB,VB);
        matriz sal1(b),sal2(b),sal0(b);
        sal1=ybus1(G,T,L,M,B,g,t,l,m,b);
        sal2=ybus2(G,T,L,M,B,g,t,l,m,b);
        sal0=ybus0(G,T,L,M,B,g,t,l,m,b);
        corto_circuito cortazo(sal1,sal2,sal0);
        // cortazo saca(fstream salida);
        cortazo.Icck3(salida,B,b,SB);
        salida.close();
        return 0;
}
```

Este archivo contiene los modelos a objetos de los elementos más importantes constituyente del sistema eléctrico de potencia. Veáse la forma conveniente como se usa la herencia para evitar la repetición de código ya sea por datos o métodos en común.

```
/*
        YBUS CPP

*/

#include<iostream.h>
#include<math.h>
#include<complex.h>
#include<conio.h>
#include<string.h>

//#include"c:\caball~1\tesis\abril\zarch\matmet.cpp"
#include"c:\tsep\abril\zarch\matmet.cpp"

class GeneradorA.
```

```
class MotorA;
class LineaA;
class TransformadorA;
class Bus;

class EquipoElectrico{
        protected:
                complex Z1,Z2,Z0,z1,z2,z0,y1,y2,y0;
                double VB,IB;
                char nombre[20];
                double bus,ID;
        public:
                EquipoElectrico(){}
                double mod(complex Z){
                  return sqrt(norm(Z));
                }
           /*   void mete_bus(){
                  cout<<"en que bus esta conectado ";
                  cin>>bus;
                }*/
                void saca_pu(){
                  cout<<"Los valores por unidad son ";
                  cout<<"z1= "<<z1<<"\tz2= "<<z2<<"\tz0= "<<z0<<endl;
                }
                complex saca_y1(){ return y1;}
                complex saca_y2(){ return y2;}
                complex saca_y0(){ return y0;}
                int saca_bus(){return bus;}
                //int saca_ID(){return ID; }
};

class GeneradorA:public EquipoElectrico{
        protected:
                double S;
                char conex;
                float fp;
                double VN;
        public:
        GeneradorA(complex pZ1,complex pZ2,complex pZ0,double pS,double pVN,double pbus=0){
           Z1=pZ1; Z2=pZ2; Z0=pZ0; S=pS; VN=pVN; bus=pbus;
           strcpy(nombre,"generador");
        }
        GeneradorA(){strcpy(nombre,"generador"); }
        friend comp_simetricas(GeneradorA*,TransformadorA*,LineaA*,MotorA*,Bus*);
        friend ostream &operator<<(ostream &stream, GeneradorA obj);
        void cambio_base(double SB,double VB){
                  z1=Z1*(SB/S)*pow((VN/VB),2);
                  z2=Z2*(SB/S)*pow((VN/VB),2);
                  z0=Z0*(SB/S)*pow((VN/VB),2);
                  y1=1/z1; y2=1/z2; y0=1/z0;
        }

};
```

```
ostream &operator<<(ostream &stream, GeneradorA obj){
        stream<<obj nombre<<endl;
        stream <<"Z1="<<obj.Z1<<"\tZ2="<<obj.Z2<<"\tZ0="<<obj Z0<<endl;
        stream<<"Voltaje="<<obj.VN<<" V "<<endl;
        stream<<"Potencia="<<obj S<<" VA "<<endl,
        stream<<"bus="<<obj.bus<<endl<<endl;
        return stream.
}


class TransformadorA: public EquipoElectrico{
         protected:
        double S;
        double VAT,VBT;
        char conexA, conexB;
        int busAT,busBT.
         public:
                TransformadorA(){strcpy(nombre,"transformador");}
         double Voltaje(char a){if(a=='b') return VBT; else return VAT, }
         TransformadorA(complex pZ1,complex pZ2,complex pZ0,double pVAT,double pVBT,double
pS,double pbus1=0,double pbus2=0){
           Z1=pZ1; Z2=pZ2; Z0=pZ0; VAT=pVAT; VBT=pVBT; S=pS; strcpy(nombre,"transformador");
           busAT=pbus1; busBT=pbus2;
         }
         void cambio_base(double SB,double VB){
           z1=Z1*(SB/S)*pow((VAT/VB),2);
           z2=Z2*(SB/S)*pow((VAT/VB),2);
           z0=Z0*(SB/S)*pow((VAT/VB),2);
           y1=1/z1; y2=1/z2; y0=1/z0;
         }
         int saca_busAT(){ return busAT;}
         int saca_busBT(){ return busBT;}
         double RT(){ return VBT/VAT; }
         friend comp_simetricas(GeneradorA*,TransformadorA*,LineaA*,MotorA*,Bus*);
        // friend matriz ybus(GeneradorA*,TransformadorA*,LineaA*,MotorA*,Bus*);
         friend validar_conexion(TransformadorA *T,LineaA *L,Bus *B,int t,int l,int b),
         friend ostream &operator<<(ostream &stream, TransformadorA obj);
};

ostream &operator<<(ostream &stream, TransformadorA obj){
        stream<<obj.nombre<<endl;
        stream <<"Z1="<<obj.Z1<<"\tZ2="<<obj.Z2<<"\tZ0="<<obj.Z0<<endl;
        stream <<"VAT="<<obj.VAT<<" V "<<"\tVBT="<<obj.VBT<<" V "<<endl;
        stream<<"Potencia="<<obj.S<<" MVA "<<endl<<endl;
        stream<<"busAT="<<obj.busAT<<endl;
        stream<<"busBT="<<obj busBT<<endl<<endl.
        return stream.
}


class LineaA: public EquipoElectrico{
         protected:
        double S;
        int bus1,bus2.
```

```
        char conexA, conexB;
          public:
          LineaA(){strcpy(nombre,"LineaA de Transmisien");}
          LineaA(complex pZ1,complex pZ2,complex pZ0,double pbus1=0,double pbus2=0){
                    Z1=pZ1;
          //        cout<<Z1;
                    Z2=pZ2; Z0=pZ0;
             bus1=pbus1; bus2=pbus2;
            strcpy(nombre,"LineaA de Transmisien");
          }
          void mete_bus(){
          cout<<"entre que buses esta conectado " ;
          cin>>bus1>>bus2;
                  }
          int saca_bus1(){ return bus1;}
          int saca_bus2(){ return bus2;}
          void cambio_base(double SB,double VB){
                    double ZB;
                    ZB=VB*VB/SB;
                    z1=Z1/ZB; z2=Z2/ZB, z0=Z0/ZB;
                    y1=1/z1; y2=1/z2; y0=1/z0;
          }
          //friend matriz ybus(GeneradorA*,TransformadorA*,LineaA*,MotorA*,Bus*);
          friend comp_simetricas(GeneradorA*,TransformadorA*,LineaA*,MotorA*,Bus*);
          friend validar_conexion(TransformadorA *T,LineaA *L,Bus *B,int t,int l,int b);
          friend ostream &operator<<(ostream &stream, LineaA obj);
};

ostream &operator<<(ostream &stream, LineaA obj){
        stream<<obj.nombre<<endl;
        stream <<"Z1="<<obj.Z1<<"\tZ2="<<obj.Z2<<"\tZ0="<<obj.Z0<<endl;
        stream<<"bus1="<<obj.bus1<<endl;
        stream<<"bus2="<<obj.bus2<<endl<<endl;
        return stream;
}


class MotorA: public EquipoElectrico{
          protected:
        double S,VN;
        char conexA;
        float fp,efi;
          public:
                MotorA(complex pZ1,complex pZ2,complex pZ0,double pS,double pVN,double pbus){
                        Z1=pZ1; Z2=pZ2; Z0=pZ0; S=pS; VN=pVN; bus=pbus;
                        strcpy(nombre,"motor");
                }
                MotorA(){strcpy(nombre,"motor");}
                void cambio_base(double SB,double VB){
                  z1=Z1*(SB/S)*pow((VN/VB),2);
                  z2=Z2*(SB/S)*pow((VN/VB),2);
                  z0=Z0*(SB/S)*pow((VN/VB),2);
                  y1=1/z1; y2=1/z2; y0=1/z0;
                }
```

```
        //      friend matriz ybus(GeneradorA*,TransformadorA*,LineaA*,MotorA*,Bus*);
                friend comp_simetricas(GeneradorA*,TransformadorA*,LineaA*,MotorA*,Bus*);
                friend ostream &operator<<(ostream &stream, MotorA obj);
};

ostream &operator<<(ostream &stream, MotorA obj){
        stream<<obj.nombre<<endl;
        stream <<"Z1="<<obj.Z1<<"\tZ2="<<obj.Z2<<"\tZ0="<<obj.Z0<<endl;
        stream<<"Voltaje nominal="<<obj.VN<<" V "<<endl;
        stream<<"Potencia="<<obj.S<<" VA "<<endl;
        stream<<"bus="<<obj.bus<<endl<<endl;
        return stream;
}

class Bus{
    int ID;
    double VB;
  public
    void asigna_VB(double V1){ VB=V1; }
    double saca_VB(){return VB;}
    void asigna_ID(int a){ ID=a; }
    int saca_bus(){return ID; }
    friend validar_conexion(TransformadorA *T,LineaA *L,Bus *B,int t,int l,int b);
    friend comp_simetricas(GeneradorA*,TransformadorA*,LineaA*,MotorA*,Bus*);
    //friend matriz ybus(GeneradorA*,TransformadorA*,LineaA*,MotorA*,Bus*);
};

validar_conexion(TransformadorA *T,LineaA *L,Bus *B,int t,int l,int b){
        for(int j=1;j<=b;j++){
                for(int i=1;i<=t;i++)
                  if(T[i].saca_busAT()==B[j].saca_bus()){
                    int temp2;
                    temp2=T[i].saca_busBT();
                        }
                for(i=1;i<=l;i++)
                  if(L[i].saca_bus1()==B[j].saca_bus()){
                    int temp2;
                    temp2=L[i].saca_bus2();
                    B[temp2].asigna_VB(B[j].saca_VB());
                  }
                }
}
// Formacion de la matriz de Impedancia Directa

matriz ybus1(GeneradorA *G, TransformadorA *T, LineaA *L,MotorA *M,Bus *B,int g,int t,int l,int m, int
b){
          matriz Temp(b);
          complex it=0;
/*        for(int i=0;i<g;i++)
                  cout<<G[i];
          getche();  */
         for(int i=0;i<b;i++)
          for(int j=0;j<b;j++){
                  if(i==j){
```

```
                lt=0;
                for(int k=0;k<g;k++){
                double F=G[k].saca_bus();
                if(F==j+1)
                        lt+=G[k].saca_y1();
                }
                for(k=0;k<m;k++){
                if(M[k].saca_bus()==j+1)
                        lt+=M[k].saca_y1();
                }
                for(k=0;k<t;k++){
                if(T[k].saca_busAT()==j+1)
                        lt+=T[k].saca_y1();
                if(T[k].saca_busBT()==j+1)
                        lt+=T[k].saca_y1();
                }
                for(k=0;k<l;k++){
                if(L[k].saca_bus1()==j+1)
                        lt+=L[k].saca_y1();
                if(L[k].saca_bus2()==j+1)
                        lt+=L[k].saca_y1();
                }
                Temp.M[i][j]=lt;
                }
                else{
                lt=0;
                Temp.M[i][j]=0;
                for(int k=0;k<t;k++){
                if(T[k].saca_busAT()==i+1 && T[k].saca_busBT()==j+1 ||
                        T[k].saca_busAT()==j+1 && T[k].saca_busBT()==i+1 )
                        Temp.M[i][j]+=-(T[k].saca_y1()),
                }
                for(k=0;k<l;k++){
                if(L[k].saca_bus1()==i+1 && L[k].saca_bus2()==j+1 ||
                        L[k].saca_bus1()==j+1 && L[k].saca_bus2()==i+1 )
                        Temp.M[i][j]+=-(L[k].saca_y1());
                }

                }

        }

        for(i=0;i<b;i++){
                for(int j=0;j<b;j++){
                        if(i!=j)
                        Temp.M[j][i]=Temp.M[i][j];
        //              cout<<Temp.M[j][i]<<endl;
          //            getche();
                }
        }
        return Temp;
}
// Formacion de la matriz de impedancia inversa
```

```
matriz ybus2(GeneradorA *G, TransformadorA *T, LineaA *L,MotorA *M,Bus *B,int g,int t,int l,int m, int
b){
        matriz Temp(b);
        complex lt=0;
       /* for(int i=0;i<g;i++)
              cout<<G[i];
        getche(); */
       for(int i=0;i<b;i++)
        for(int j=0;j<b;j++){
              if(i==j){
               lt=0;
               for(int k=0;k<g;k++){
               double F=G[k].saca_bus();
              if(F==j+1)
                     lt+=G[k].saca_y2();
               }
               for(k=0;k<m;k++){
              if(M[k].saca_bus()==j+1)
                     lt+=M[k].saca_y2();
               }
               for(k=0;k<t;k++){
              if(T[k].saca_busAT()==j+1)
                     lt+=T[k].saca_y2();
              if(T[k].saca_busBT()==j+1)
                     lt+=T[k].saca_y2();
               }
               for(k=0;k<l;k++){
              if(L[k].saca_bus1()==j+1)
                     lt+=L[k].saca_y2();
              if(L[k].saca_bus2()==j+1)
                     lt+=L[k].saca_y2();
               }
               Temp.M[i][j]=lt;
              }
               else{
               lt=0;
              Temp.M[i][j]=0;
              for(int k=0;k<t;k++){
               if(T[k].saca_busAT()==i+1 && T[k].saca_busBT()==j+1 ||
                     T[k].saca_busAT()==j+1 && T[k].saca_busBT()==i+1 )
                      Temp M[i][j]+=-(T[k].saca_y2());
              }
              for(k=0;k<l;k++){
               if(L[k].saca_bus1()==i+1 && L[k].saca_bus2()==j+1 ||
                     L[k].saca_bus1()==j+1 && L[k].saca_bus2()==i+1  )
                      Temp.M[i][j]+=-(L[k].saca_y2());
              }

               }

       }

       for(i=0,i<b.i++){
              for(int j=0,j<b;j++){
```

```
                        if(i!=j)
                        Temp.M[j][i]=Temp.M[i][j];
        //              cout<<Temp.M[j][i]<<endl;
         //             getche();
                }
        }
        return Temp;
}

matriz ybus0(GeneradorA *G, TransformadorA *T, LineaA *L,MotorA *M,Bus *B,int g,int t,int l,int m, int
b){
         matriz Temp(b);
         complex It=0;
        /*  for(int i=0;i<g;i++)
                cout<<G[i];
         getche(); */
        for(int i=0;i<b;i++)
         for(int j=0;j<b;j++){
                if(i==j){
                 It=0;
                 for(int k=0;k<g;k++){
                 double F=G[k].saca_bus();
                if(F==j+1)
                        It+=G[k].saca_y0();
                 }
                 for(k=0;k<m;k++){
                if(M[k].saca_bus()==j+1)
                        It+=M[k].saca_y0();
                 }
                 for(k=0;k<t;k++){
                if(T[k].saca_busAT()==j+1)
                        It+=T[k].saca_y0();
                if(T[k].saca_busBT()==j+1)
                        It+=T[k].saca_y0();
                 }
                 for(k=0;k<l;k++){
                if(L[k].saca_bus1()==j+1)
                        It+=L[k].saca_y0();
                if(L[k].saca_bus2()==j+1)
                        It+=L[k].saca_y0();
                 }
                 Temp.M[i][j]=It;
                }
                 else{
                 It=0;
                Temp.M[i][j]=0;
                for(int k=0;k<t;k++){
                 if(T[k].saca_busAT()==i+1 && T[k].saca_busBT()==j+1 ||
                        T[k].saca_busAT()==j+1 && T[k].saca_busBT()==i+1 )
                         Temp.M[i][j]+=-(T[k].saca_y0());
                }
                for(k=0;k<l;k++){
                 if(L[k].saca_bus1()==i+1 && L[k].saca_bus2()==j+1 ||
                        L[k].saca_bus1()==j+1 && L[k].saca_bus2()==i+1 )
```

```
                        Temp.M[i][j]+=-(L[k].saca_y0());
                }

                }

        }

        for(i=0,i<b,i++){
                for(int j=0;j<b,j++){
                        if(i!=j)
                        Temp M[j][i]=Temp.M[i][j];
        //              cout<<Temp.M[j][i]<<endl;
          //            getche();
                }
        }
        return Temp,
}

saca_pu(GeneradorA *G,TransformadorA *T,LineaA *L,MotorA *M,Bus *B,int g,int t,int l,int m, int b){
        for(int i=0,i<g,i++){
                cout<<"Generador "<<i<<endl;
                G[i] saca_pu(),
        }
        for(i=0,i<t,i++){
                cout<<"Transformador "<<i<<endl,
                T[i] saca_pu();
        }
        for(i=0;i<l;i++){
                cout<<"Linea de Transmision "<<i<<endl;
                L[i].saca_pu();
        }
        for(i=0,i<m,i++){
                cout<<"Motor "<<i<<endl;
                M[i].saca_pu();
        }
}

double comp_simetricas(GeneradorA *G,TransformadorA *T,LineaA *L,MotorA *M,Bus *B,double
g,double t,double l,double m,double b,double SB,double VB){
        int i;
        //double VB,SB;
        /* cout<<"Dame la potencia Base del sistema " ;
        cin>>SB,
        cout<<"Dame el voltaje Base en bus 1 ";
        cin>>VB.  */

        B[0].asigna_VB(VB);
                for(int j=0,j<b,j++){
                 for(i=0,i<t,i++){
                        if(T[i] saca_busAT()==B[j] saca_bus()){
                          int temp2;
                          temp2=T[i].saca_busBT();
                          double RTC;
                          RTC=T[i].RT();
```

```
                    B[temp2-1] asigna_VB(RTC*B[j].saca_VB()).
                    }

                  if(T[i].saca_busBT()==B[j].saca_bus()){
                    int temp2;
                    temp2=T[i].saca_busAT();
                    double RTC;
                    RTC=T[i].RT();
                    B[temp2-1].asigna_VB(B[j] saca_VB()/RTC);
                  }
                }
                for(i=0;i<l;i++){
                    if(L[i].saca_bus1()==B[j].saca_bus()){
                      int temp2;
                      temp2=L[i].saca_bus2();
                      B[temp2-1].asigna_VB(B[j].saca_VB());
                    }
                    if(L[i].saca_bus2()==B[j].saca_bus()){
                      int temp2;
                      temp2=L[i].saca_bus1();
                      B[temp2-1].asigna_VB(B[j].saca_VB());
                    }
                }
              }
        //    for(i=0;i<b;i++) cout<<"\nVB"<<i+1<<"= "<<B[i].saca_VB();
              for(i=0;i<g;i++){
                    for(j=0;j<b;j++){
                        if(G[i].saca_bus()==B[j].saca_bus())
                        G[i].cambio_base(SB,B[j].saca_VB());
                    }
              }
              for(i=0;i<t;i++){
                    for(j=0;j<b;j++){
                        if(T[i] saca_busAT()==B[j] saca_bus())
                        T[i].cambio_base(SB,B[j].saca_VB());
                    }
              }
              for(i=0;i<l;i++){
                    for(j=0;j<b;j++){
                        if(L[i].saca_bus1()==B[j].saca_bus())
                        L[i] cambio_base(SB,B[j].saca_VB());
                    }
              }
              for(i=0;i<m;i++){
                    for(j=0;j<b;j++){
                        if(M[i].saca_bus()==B[j].saca_bus())
                        M[i].cambio_base(SB,B[j].saca_VB());
                    }
              }
    return SB;
};
```

Este archivo tiene todo lo concerniente al manejo de las matrices, muy en especial en la metodología para su inversión. Este código se le dio un trato especializado ya que es un punto importante en el uso de ZBUS, y fue la forma más conveniente que encontramos para la solución práctica de dicha problematica.

```
/*
        MATMET.CPP
*/

#include<iostream.h>
#include<math.h>
#include<conio.h>
#include<complex.h>

#define ZNULO complex(0,0)

class GeneradorA;
class MotorA;
class LineaA;
class TransformadorA;
class Bus;

class matriz{
  public:
        double n;
        double static ni;
    complex **M,deter;
  public:

    void inst(){ cout<<ni; }
        matriz(const double n1);
    matriz(){n=0; ni++; }
        ~matriz();
        void chafa(double);
        void Pide( );
        friend matriz operator+(const matriz &A,const matriz &B);
        matriz& operator=(const matriz &B);
        friend matriz operator%(const matriz &A,int a);
        friend matriz operator*(const double l,const matriz &A);
        friend matriz operator*(const matriz &A,const double l);
        friend matriz operator*(const complex l,const matriz &A);
        friend matriz operator*(const matriz &A, const complex l);
        friend matriz operator*(const matriz &A,const matriz &B);
        friend matriz operator/(const matriz &A,double d);
        friend matriz operator/(double d,const matriz &A);
        void Imprime()const;
        friend matriz cofactor(const matriz &A,int i,int j);
        friend matriz transpuesta(const matriz &A);
        friend matriz adjunta(const matriz &A);
        friend complex det(const matriz &A);
        matriz inversa()const;
        friend matriz menores(const matriz &A);
        friend double modulo (const complex z);
```

```
        double dim(){ return n; }
        complex saca(double x,double y)const{ return M[x][y]; }
        friend matriz ybus(GeneradorA*,TransformadorA*,LineaA*,MotorA*,Bus*);
};

double matriz::n=1;

void matriz::chafa(double n1){n=n1;
                M=new complex*[n];
        for(int j=0;j<n;j++)
                M[j]=new complex[n];
        }

/*ostream &operator<<(ostream &stream, complex obj){

   stream<<obj.x<<" + j "<<obj.y<<"\n";
        return stream;
} */

/*istream &operator>>(istream &stream, complex &obj){
        stream>>obj.x>>","">>obj.y;
        return stream;
}

*/

double modulo(const complex z){
        return sqrt(norm(z));
}

matriz operator*(const matriz &A,const matriz &B){
        matriz temp(A.n);
        int i,j,k;
        for(i=0;i<A.n;i++)
                for(j=0;j<A.n;j++)
                        temp.M[i][j]=0;

        for(i=0;i<A.n;i++)
                for(j=0;j<A.n;j++){
                        for(k=0;k<A.n;k++)
                                temp.M[i][j]+=A.M[i][k]*B.M[k][j];
         // cout<<endl<<temp.M[i][j];
          }
        return temp;
}

matriz operator*(const double l, const matriz &A){
        matriz B(A.n);
        int i,j;
        for(i=0;i<A.n;i++)
          for(j=0;j<A.n;j++)
                B.M[i][j]=l*A.M[i][j];
        return B;
}
```

```
matriz operator*(const matriz &A,const double l){
        matriz B(A.n);
        int i,j;
        for(i=0,i<A.n;i++)
          for(j=0.i<A.n;j++)
                B.M[i][j]=l*A.M[i][j];
        return B;
}
matriz operator*(const complex l, const matriz &A){
        matriz B(A n);
        int i,j;
        for(i=0;i<A.n,i++)
          for(j=0;j<A.n;j++)
                B M[i][j]=l*A.M[i][j];
        return B;
}

matriz operator*(const matriz &A, const complex l){
        matriz B(A n);
        int i,j;
        for(i=0,i<A n;i++)
          for(j=0,i<A.n;j++)
                B.M[i][j]=l*A.M[i][j];
        return B;
}
matriz operator/(const complex l,const matriz &A){
        matriz B(A n);
        int i,j;
        for(i=0;i<A n;i++){
          for(j=0;j<A n,j++){
                B.M[i][j]=A.M[i][j]/l;
                //cout<<B.M[i][j];
          }//cout<<endl;
        }
        return B;
}

matriz operator/(const matriz &A,const complex l){
        matriz B(A n);
        int i,j;
        for(i=0;i<A.n;i++){
          for(j=0;j<A n;j++){
                B.M[i][j]= A.M[i][j]/l;
                //cout<<B.M[i][j];
          }
        //cout<<endl;
        }
        return B;
}

matriz  matriz(const double n1){
        n=n1;
  n1++;
```

```
        M=new complex*[n];
        for(int j=0;j<n;j++)
                M[j]=new complex[n];
}

matriz::~matriz(){
//  for(int j=0;j<n;j++)
 //         delete []M[j];
//  delete[]*M;
}



matriz operator%(const matriz &A,int a=-1){
        matriz temp(A.n);
        for(int i=0; i<temp.n;i++)
          for(int j=0;j<temp.n;j++){
                  temp.M[i][j]=A.M[j][i];
          }
    a++;
        return temp;
}

matriz operator+(const matriz &A,const matriz &B){
        matriz temp(A.n);
        for(int i=0; i<temp.n;i++)
          for(int j=0;j<temp.n;j++){
                  temp.M[i][j]=A.M[i][j]+B.M[i][j];
          }
        return temp;
}

matriz& matriz::operator=(const matriz &B){
        //if(n!=0) matriz::~matriz();
        //matriz::matriz(B.n);
        for(int i=0; i<n;i++)
          for(int j=0;j<n;j++)
                  M[i][j]=B.M[i][j];
        return *this;
}



void matriz::Pide(){
        int i,j;
   for(i=0;i<n;i++)
        for(j=0;j<n;j++){
                cout<<"M["<<i<<"]["<<j<<"]=";
                cin>>M[i][j];
        }
  // deter=det(*this);
}
void matriz::Imprime()const{
        for(int i=0,i<n;i++){ cout<<endl;
```

```
        for(int j=0;j<n;j++){
                cout<<M[i][j]<<" ";
        }
        }
        cout<<endl;
}

matriz transpuesta(const matriz &A){
        matriz B(A.n);
        for(int i=0,i<A.n;i++){
        for(int j=0;j<A.n;j++){
                B.M[j][i]=A.M[i][j];
        }
        }
        return B;
}
matriz adjunta(const matriz &A){
        matriz B(A.n);
        for(int i=0;i<A.n;i++){
        for(int j=0;j<A.n;j++){
                B M[i][j]=pow((-1),i+j)*det(cofactor(A,i,j));
        }
        }
        return transpuesta(B);
}

//matriz B1;

matriz cofactor(const matriz &A,int k,int p){
        matriz B1(A.n-1);
  //B1.chafa(A.n-1);
        int j=0,i=0;
        for(int l=0;i<A.n;i++){
          if(i==k){}
    else{
          for(int m=0;j<A.n;j++){
                if(j==p){}
                else{
                        B1.M[l][m]=A.M[i][j];
                        m++;
                }
          }
         l++;
         j=0;
         }
        }
        return B1;
}

complex det(const matriz &A){
        matriz B(A.n);
        complex temp,deter=0,menor;
        if(A.n>3 || A.n<2){
                for(int i=0,i<A n,i++){
```

```
                temp=pow(-1,i)*(A.M[i][0]);
                B=cofactor(A,i,0);
                menor=det(B);
                deter+=temp*menor;
              }
        return deter;
        }
        else{
        if(A.n==2) return (A.M[0][0]*A.M[1][1]-A.M[1][0]*A.M[0][1]);
        if(A.n==3) return
(A.M[0][0]*A.M[1][1]*A.M[2][2]+A.M[1][0]*A.M[2][1]*A.M[0][2]+A.M[2][0]*A.M[1][2]*A.M[0][1]-
A.M[2][0]*A.M[1][1]*A.M[0][2]-A.M[2][1]*A.M[1][2]*A.M[0][0]-A.M[2][2]*A.M[0][1]*A.M[1][0]);
        if(A.n==1) return 1;
        }
    return 0;
}

matriz menores(const matriz &A){
        matriz B(A.n-1),
        matriz C(A.n),
        complex menor,temp;
        for(int i=0;i<A.n;i++)
                for(int j=0;j<A.n;j++){
                        temp=pow(-1,i+j)*modulo(A.M[i][j]);
                        B=cofactor(A,i,j);
                        menor=det(B);
                        C.M[i][j]=menor*temp;
                }
        return transpuesta(C);
}


matriz matriz::inversa()const{
        /* cout<<"Deberas es la original \n";
        Imprime();
        getche();
                */
        complex chafisisima=M[0][0];
        matriz A1(n),temp(n);
        A1=*this;
        A1.M[0][0]=chafisisima;
        complex cha;
    /*  cout<<"Deberas es la original \n",
        Imprime(),
        getche();                       */
        // creando una matriz identidad
        for(int i=0;i<A1.n;i++)
          for(int j=0;j<A1.n;j++)
            if(i==j) temp.M[i][j]=1;
            else temp.M[i][j]=0;
        // cout<<"\nTemporal unitaria\n";
        // temp.Imprime();
         // haciendo los A.M[i][i] unitarios
    //  complex chaq=A1.M[0][0];
```

```
//    complex chaqu=M[0][0];
      for(i=0,i<A1.n;i++)
        for(int j=0,j<A1.n;j++)
          if(i==j){
                cha=A1.M[i][j];
                for(int k=0;k<A1.n;k++){
                  // if(cha!=0){
                        A1.M[i][k]/=cha;
                        temp.M[i][k]/=cha;
                  // }
                }
          }
      //cout<<"\nTemporal diagonal\n";
      //temp.Imprime();
      //A1 Imprime();
       // creando a una identidad y obtiendo finalmente la inversa
        for(i=0,i<A1 n;i++)
              for(int j=0,j<A1.n,j++){
                  if(i==j){
       //                 A1.Imprime();
       //                 temp.Imprime();
                        // para barrer los renglones
                        for(int k=0;k<A1.n;k++){
                                if(/*A1.M[k][i]!=0*/ k!=i){
                                 complex base=-A1.M[k][i];
                              // para barrer las columnas.
                              for(int l=0;l<A1.n;l++){
                              //    cout<<"\n\n"<<base;
                                    complex avalon=A1.M[i][l];
                              //    cout<<avalon;
                                    A1.M[k][l]+=(base*avalon);
                                    complex avalon2=temp.M[i][l];
                             //     cout<<avalon2;
                                    temp.M[k][l]+=(base*avalon2);
                              }
                                  complex chal=A1.M[k][k];
                                  if(chal!=0)
                              for(int h=0;h<A1 n;h++){
                                    A1.M[k][h]/=chal;
                                    temp.M[k][h]/=chal;
                              }
                       //       A1.Imprime();
                       //       temp.Imprime();
                              }
                            }
                        }

                }
      // cout<<"\nTemporal finalmente\n";
      // Imprime();
      // temp Imprime();
      // getche();
       return temp;
}
```

Este archivo contiene lo concerniente al cálculo del corto circuito, las datos necesarios, las funciones miembro indispensables para su determinación certera Observesé que el código podría haber sido puesto en pequeñas funciones, pero debido a que se trabaja una metodología orientada a objetos, se obtó por crear una nueva clase llamada corto.

```
/*
        CORTO.CPP

*/

#include<iostream.h>
#include<COMPLEX.h>
#include<math.h>

class corto{
  public:
    complex Icc_LT(complex Vk,complex Zp,complex Zn,complex Z0);
    complex Icc_LL(complex Vk,complex Zp,complex Zn);
    complex Icc_LLT(complex Vk,complex Zp,complex Zn,complex Z0);
    complex Icc_LLL(complex Vk,complex Zp);
    complex Vn(complex Vpn,complex Vk,complex Znk,complex Zkk);
    complex Iij(complex Vi,complex Vj,complex Zij);
};

complex corto::Icc_LT(complex Vk,complex Zp,complex Zn,complex Z0){
   return 3*Vk/(Zp+Zn+Z0);
}
complex corto::Icc_LL(complex Vk,complex Zp,complex Zn){
        return -complex(0,1)*pow(3,.5)*Vk/(Zp+Zn);
}
complex corto::Icc_LLT(complex Vk,complex Zp,complex Zn,complex Z0){
   return 3*Vk*Zn/(Zp*Zn+Zn*Z0+Z0*Zp);
}
complex corto::Icc_LLL(complex Vk,complex Zp){
        return Vk/Zp;
}
complex corto::Vn(complex Vpn,complex Vk,complex Znk,complex Zkk){
   return Vpn - Vk*Znk/Zkk;
}
complex corto::Iij(complex Vi,complex Vj,complex Zij){
        return (Vi-Vj)/Zij;
}

class corto_circuito:public corto{
   matriz *M1,*M2,*M0;
   public:
        corto_circuito(const matriz &Y1,const matriz &Y2,const matriz &Y0){
                M1=new matriz(Y1.dim());
                M2=new matriz(Y2.dim());
                M0=new matriz(Y0.dim());
                *M1=Y1.inversa();
                *M2=Y2.inversa();
```

```
            *M0=Y0.inversa();

    }
    void saca(){
      M1->Imprime();
      getche();
      M2->Imprime();
      getche();
      M0->Imprime();
      getche();
    }
    void Vnk(Bus *B,double b){
       for(int i=0;i<b;i++)
            for(int j=0;j<b;j++){
              cout<<"\nVoltajes Nodales\n ";
              cout<<Vn(B[i].saca_VB(),B[j] saca_VB(),M1->saca(i,j),M1->saca(j,j));
            }
    }

    double argu(complex Z){
       return arg(Z)*180/3 141592;
}

    void Icck3(fstream &salida,Bus *B,double b,double SB)const{
      double IB;
      complex ZLT,ZLL,ZLLT,ZLLL;
      salida<<"\n\n\t\tLT";
      for(int i=0;i<b;i++){
            salida<<"\nBUS "<<i+1;
            IB=SB*1000/(pow(3, 5)*B[i].saca_VB());
     ZLT=IB*Icc_LT(1,M1->saca(i,i),M2->saca(i,i),M0->saca(i,i));
     salida<<"\t"<<modulo(ZLT)<<"|_"<<argu(ZLT);
   }

      salida<<"\n\n\n\t\tLL";
      for(i=0;i<b;i++){
            salida<<"\nBUS "<<i+1;
            IB=SB*1000/(pow(3, 5)*B[i].saca_VB());
     ZLL=IB*Icc_LL(1,M1->saca(i,i),M2->saca(i,i));
     salida<<"\t"<<modulo(ZLL)<<"|_"<<argu(ZLL);
   }

      salida<<"\n\n\n\t\tLLT";
      for(i=0;i<b;i++){
            salida<<"\nBUS "<<i+1;
            IB=SB*1000/(pow(3, 5)*B[i].saca_VB());
            ZLLT=IB*Icc_LLT(1,M1->saca(i,i),M2->saca(i,i),M0->saca(i,i));
            salida<<"\t"<<modulo(ZLLT)<<"|_"<<argu(ZLLT);
      }


      salida<<"\n\n\n\t\tLLL";
      for(i=0;i<b;i++){
            salida<<"\nBUS "<<i+1;
```

```
                IB=SB*1000/(pow(3..5)*B[i].saca_VB());
                ZLLL=IB*Icc_LLL(1,M1->saca(i,i));
          salida<<"\t"<<modulo(ZLLL)<<"|_"<<argu(ZLLL);
           }

        }

};
```

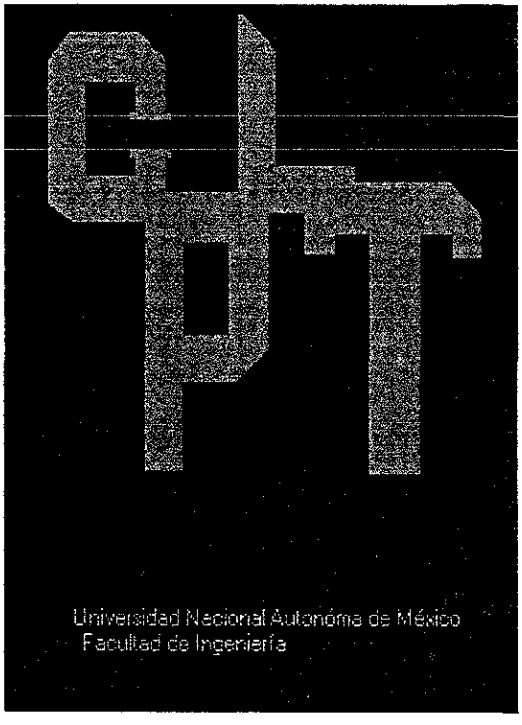
Este archivo sirve como base a ZARCH.CPP para el análisis del archivo de entrada y la síntesis en el archivo de salida Veasé que se creo una nueva clase llamada archivo con unos métodos necesarios tanto para ingresar información como para accederla

```
/*
          ARCHI.CPP

*/

#if !defined(__ARCHI_H)
#define __ARCHI_H

#include<iostream.h>
#include<fstream.h>
#include<stdlib.h>
#include<conio.h>
#include<string.h>

class archivo{
  protected:
        char *arch;
  public:
   archivo(char *l){
        arch=new char[strlen(l)+1];
        strcpy(arch,l);
   }
   ~archivo(){ delete []arch; }
    int entrada(char *);
    void salida();
};

int archivo::entrada(char *p){
    //  char *p="HJOSLADFASDL";
        fstream s(arch, ios::in|ios::out);
        if(!s){
                cout << "No puedo abrir el archivo";
                return 1;
        }
        s.seekp(0,ios::end);
        while(*p)
          s.put(*p++);
        s.close();
        return 0;
}
```

```
void archivo. salida(){
  ifstream aut(arch).
  if(!aut){
    cerr<<"No puedo abrir el archivo";
    exit(0).
   }
   while(aut){
    char buffer[100].
    aut.getline(buffer,sizeof(buffer),':');
    cout<<endl<<buffer;
    getche().
   }
  aut.close();
}
#endif
```

APÉNDICE D

Código de CPLT


Universidad Nacional Autónoma de México
Facultad de Ingeniería

Este archivo contiene las declaraciones de las clases necesarias para la solución del problema, además tiene la definición de las estructuras necesarias para las cajas de dialogo de captura También es importante porque va llamado a cada uno de los archivos en el orden adecuado para las necesidades de la compilación

```
/*

                        GRAL.CPP


*/

#if !defined(__GENERAL_H)
#define __GENERAL_H

#define CM_U_HELPINDEX          0x200
#define CM_U_HELPHELP           0x201

#include<stdio.h>
#include<string.h>
#include<owl.h>
#include<inputdia.h>
#include<dialog.h>
#include<fstream.h>
#include <filedial.h>
#include "filewnd.h"
#include <bwcc.h>
#include <edit.h>
#include "conio.h"
#include"stdio.h"
#include "imparchi.cpp"

// Errores
#define E_ARGUMENTO  MessageBox(HWindow,"Se ha cometido un error de argumento","Error",
MB_OK|MB_ICONEXCLAMATION);
#define E_ARCHIVOSALIDA MessageBox(HWindow,"Error en el archivo de Salida","Error",
MB_OK|MB_ICONEXCLAMATION);
#define E_ARCHIVOENTRADA MessageBox(HWindow,"Error en el archivo de Entrada","Error",
MB_OK|MB_ICONEXCLAMATION);
#define E_OPCION MessageBox(HWindow,"Error de opcion","Error",
MB_OK|MB_ICONEXCLAMATION);

#define EXITO MessageBox(HWindow,"Se ha creado el archivo de salida con Exito","EXITO", MB_OK);

#define ID_C1   101
#define ID_C2   102 //Es importante definir los campos y hay que asignar un numero
#define ID_C3   103
#define ID_C4   104
#define ID_C5   105
#define ID_C6   106

#define ID_C7   107
```

```
#define ID_C8   108 //Es importante definir los campos y hay que asignar un numero
#define ID_C9   109
#define ID_C10  110
#define ID_C11  111
#define ID_C12  112

#define ID_C13  113
#define ID_C14  114 //Es importante definir los campos y hay que asignar un numero
#define ID_C15  115
#define ID_C16  116
#define ID_C17  117
#define ID_C18  118

#define ID_C19  119
#define ID_C20  120 //Es importante definir los campos y hay que asignar un numero
#define ID_C21  121
#define ID_C22  122
#define ID_C23  123
#define ID_C24  124
#define ID_C25  125
#define ID_C26  1126

#define CM_DIALOGO6 126
#define CM_DIALOGO5 125
#define CM_DIALOGO1 131
#define CM_DIALOGO2 132
#define CM_DIALOGO3 133
#define CM_DIALOGO4 134

#define CM_LINEA 3000
#define CM_LINEAGUARDA 127
#define CM_LINEAGUARDA2 128
#define CM_LINEAPARALELA 129
#define CM_LINEAPARALELAGUARDA 130
#define CM_LINEAPARALELAGUARDA2 1131

#define DM_LINEA 13000
#define DM_LINEAGUARDA 1127
#define DM_LINEAGUARDA2 1128
#define DM_LINEAPARALELA 1129
#define DM_LINEAPARALELAGUARDA 1130
#define DM_LINEAPARALELAGUARDA2 11131
#define DM_RECALCULA 1999


#define CM_IMAGEN 666
#define CM_ABOUT 511
#define CM_HELP  201
#define CM_BORRAR 202
#define CM_SALIR  203
#define CM_ESCOGE 777
#define CM_RECALCULA 999

#include<control.h>
```

```
#include<edit.h>
#include<ctype.h>

#define MAXC 16

//**** 2 Ctos. TRIFASICOS CON 2 CABLES  DE GUARDA

struct TTransfiere12{
        char c1[MAXC];
        char c2[MAXC];
        char c3[MAXC];
        char c4[MAXC];
        char c5[MAXC];
        char c6[MAXC];
        char c7[MAXC];
        char c8[MAXC];
        char c9[MAXC];
        char c10[MAXC];
   //   char c11[MAXC];
   //   char c12[MAXC];
/*      char c13[MAXC];
        char c14[MAXC];
        char c15[MAXC];
 */
};

struct TTransfiere14{
        char c1[MAXC];
        char c2[MAXC];
        char c3[MAXC];
        char c4[MAXC];
        char c5[MAXC];
        char c6[MAXC];
        char c7[MAXC];
        char c8[MAXC];
        char c9[MAXC];
        char c10[MAXC];
        char c11[MAXC];
        char c12[MAXC];
        char c13[MAXC];
        char c14[MAXC];
 /*     char c15[MAXC];
        char c16[MAXC];
        char c17[MAXC];*/
};

struct TTransfiere16{
   char c1[MAXC];
        char c2[MAXC];
        char c3[MAXC];
        char c4[MAXC];
        char c5[MAXC];
        char c6[MAXC];
        char c7[MAXC];
```

```
        char c8[MAXC];
        char c9[MAXC],
        char c10[MAXC],
        char c11[MAXC];
        char c12[MAXC];
        char c13[MAXC];
        char c14[MAXC];
        char c15[MAXC];
        char c16[MAXC];
/*      char c17[MAXC],
        char c18[MAXC];
        char c19[MAXC];*/
},

struct TTransfiere18{
    char c1[MAXC];
    char c2[MAXC];
    char c3[MAXC],
    char c4[MAXC],
        char c5[MAXC];
        char c6[MAXC];
        char c7[MAXC];
        char c8[MAXC];
        char c9[MAXC];
        char c10[MAXC],
        char c11[MAXC],
        char c12[MAXC],
        char c13[MAXC],
        char c14[MAXC];
        char c15[MAXC];
        char c16[MAXC];
  //    char c17[MAXC];
  //    char c18[MAXC];
/*      char c19[MAXC];
        char c20[MAXC];
        char c21[MAXC].*/
};


struct TTransfiere22{
        char c1[MAXC];
        char c2[MAXC],
        char c3[MAXC];
        char c4[MAXC],
        char c5[MAXC];
        char c6[MAXC];
        char c7[MAXC];
        char c8[MAXC];
        char c9[MAXC];
        char c10[MAXC];
        char c11[MAXC],
        char c12[MAXC];
        char c13[MAXC],
        char c14[MAXC];
```

```
        char c15[MAXC];
        char c16[MAXC];
        char c17[MAXC];
        char c18[MAXC];
        char c19[MAXC];
        char c20[MAXC];
        char c21[MAXC];
        char c22[MAXC];
/*      char c23[MAXC];
        char c24[MAXC];
        char c25[MAXC];*/
};

struct TTransfiere24{
        char c1[MAXC];
        char c2[MAXC];
        char c3[MAXC];
        char c4[MAXC];
        char c5[MAXC];
        char c6[MAXC];
        char c7[MAXC];
        char c8[MAXC];
        char c9[MAXC];
        char c10[MAXC];
        char c11[MAXC];
        char c12[MAXC];
        char c13[MAXC];
        char c14[MAXC];
        char c15[MAXC];
        char c16[MAXC];
        char c17[MAXC];
        char c18[MAXC];
        char c19[MAXC];
        char c20[MAXC];
        char c21[MAXC];
        char c22[MAXC];
        char c23[MAXC];
        char c24[MAXC];
        char c25[MAXC];
   char c26[MAXC];
};

//********* CLASE 1 (CAJA 5) ***********

class Caja12:public TDialog{
        public:
        char Camp1[MAXC];
        char Camp2[MAXC];
        char Camp3[MAXC];
        char Camp4[MAXC];
        char Camp5[MAXC];
        char Camp6[MAXC];
        char Camp7[MAXC];
        char Camp8[MAXC];
```

```
        char Camp9[MAXC],
     char Camp10[MAXC];
   // char Camp11[MAXC];
  //      char Camp12[MAXC];
 /*       char Camp13[MAXC],
         char Camp14[MAXC];
         char Camp15[MAXC];*/

Caja12(PTWindowsObject Aparent,LPSTR name),
         virtual BOOL CanClose(), //Cerramos la caja
   private:
         void FillBuffers();
};

class Caja14:public TDialog{
         public:
         char Camp1[MAXC],
         char Camp2[MAXC];
         char Camp3[MAXC],
      char Camp4[MAXC],
         char Camp5[MAXC],
         char Camp6[MAXC],
         char Camp7[MAXC];
         char Camp8[MAXC];
         char Camp9[MAXC],
      char Camp10[MAXC];
         char Camp11[MAXC];
         char Camp12[MAXC];
      char Camp13[MAXC],
         char Camp14[MAXC],
/*       char Camp15[MAXC];
         char Camp16[MAXC];
         char Camp17[MAXC];*/

Caja14(PTWindowsObject Aparent,LPSTR name),
         virtual BOOL CanClose(), //Cerramos la caja
    private:
         void FillBuffers();
};

class Caja16:public TDialog{
         public:
         char Camp1[MAXC];
         char Camp2[MAXC];
         char Camp3[MAXC],
       char Camp4[MAXC];
         char Camp5[MAXC],
         char Camp6[MAXC];
         char Camp7[MAXC];
         char Camp8[MAXC];
         char Camp9[MAXC];
      char Camp10[MAXC];
         char Camp11[MAXC];
         char Camp12[MAXC],
```

```
      char Camp13[MAXC];
          char Camp14[MAXC];
          char Camp15[MAXC];
          char Camp16[MAXC];
/*        char Camp17[MAXC];
          char Camp18[MAXC];
          char Camp19[MAXC];*/

Caja16(PTWindowsObject Aparent,LPSTR name);
          virtual BOOL CanClose(); //Cerramos la caja
   private:
          void FillBuffers();
};

class Caja18:public TDialog{
          public:
          char Camp1[MAXC];
          char Camp2[MAXC];
          char Camp3[MAXC];
   char Camp4[MAXC];
          char Camp5[MAXC];
          char Camp6[MAXC];
          char Camp7[MAXC];
          char Camp8[MAXC];
          char Camp9[MAXC];
          char Camp10[MAXC];
          char Camp11[MAXC];
          char Camp12[MAXC];
      char Camp13[MAXC];
          char Camp14[MAXC];
          char Camp15[MAXC];
          char Camp16[MAXC];
     //   char Camp17[MAXC];
     // char Camp18[MAXC];
/*       char Camp19[MAXC];
          char Camp20[MAXC];
          char Camp21[MAXC];
 */

Caja18(PTWindowsObject Aparent,LPSTR name);
          virtual BOOL CanClose(); //Cerramos la caja
   private:
          void FillBuffers();
};

class Caja22:public TDialog{
          public:
          char Camp1[MAXC];
          char Camp2[MAXC];
          char Camp3[MAXC];
      char Camp4[MAXC];
          char Camp5[MAXC];
          char Camp6[MAXC];
          char Camp7[MAXC];
```

```
        char Camp8[MAXC];
        char Camp9[MAXC];
    char Camp10[MAXC];
        char Camp11[MAXC];
        char Camp12[MAXC];
    char Camp13[MAXC];
        char Camp14[MAXC],
        char Camp15[MAXC];
        char Camp16[MAXC];
        char Camp17[MAXC];
    char Camp18[MAXC];
        char Camp19[MAXC];
        char Camp20[MAXC];
    char Camp21[MAXC],
        char Camp22[MAXC];
/*      char Camp23[MAXC],
        char Camp24[MAXC];
        char Camp25[MAXC];*/

Caja22(PTWindowsObject Aparent,LPSTR name),
        virtual BOOL CanClose(); //Cerramos la caja
  private
        void FillBuffers(),
};

class Caja24:public TDialog{
        public
        char Camp1[MAXC],
        char Camp2[MAXC],
        char Camp3[MAXC],
    char Camp4[MAXC];
        char Camp5[MAXC];
        char Camp6[MAXC];
        char Camp7[MAXC];
        char Camp8[MAXC];
        char Camp9[MAXC];
    char Camp10[MAXC],
        char Camp11[MAXC];
        char Camp12[MAXC],
    char Camp13[MAXC],
        char Camp14[MAXC];
        char Camp15[MAXC];
        char Camp16[MAXC];
        char Camp17[MAXC];
    char Camp18[MAXC];
        char Camp19[MAXC],
        char Camp20[MAXC];
    char Camp21[MAXC],
        char Camp22[MAXC];
        char Camp23[MAXC];
        char Camp24[MAXC];
        char Camp25[MAXC],
        char Camp26[MAXC];
```

```
Caja24(PTWindowsObject Aparent,LPSTR name);
        virtual BOOL CanClose(), //Cerramos la caja
   private:
        void FillBuffers(),
};
// ********* FIN DE CLASE 1 (CAJA5) ********

//********* CLASE 2 (VENTANA) **********
class Ventana : public TApplication{
        public:
                Ventana(LPSTR AName, HINSTANCE hInstance, HINSTANCE hPrevInstance,
                LPSTR lpCmdLine, int nCmdShow)
                        : TApplication(AName, hInstance, hPrevInstance, lpCmdLine, nCmdShow){};
        virtual void InitMainWindow();
        virtual void InitInstance(); //****
};
//********* FIN DE CLASE 2 (VENTANA) *********

//********* CLASE 3 ( MI VENTANA) *********
class Miventana : public TFileWindow{
        int x,y,recalcula,
        HBITMAP escudo;
        HBITMAP BITMAP_1;
        HBITMAP BITMAP_2;
        HBITMAP BITMAP_3,
        HBITMAP BITMAP_4;
        HBITMAP BITMAP_5,
        HBITMAP BITMAP_6;
        HDC hdc,
        BOOL IsNewFile,
        char ArchNom[MAXPATH];
public:

  TTransfiere12 Transfiere12;
  TTransfiere14 Transfiere14;
  TTransfiere16 Transfiere16;
  TTransfiere18 Transfiere18,
  TTransfiere22 Transfiere22;
  TTransfiere24 Transfiere24;
  Miventana(PTWindowsObject AparentLPSTR ATitle, LPSTR AFileName);

  virtual void WMLButtonDown(RTMessage Msg)
    = [WM_FIRST + WM_LBUTTONDOWN];
  virtual void WMRButtonDown(RTMessage Msg)
    = [WM_FIRST + WM_RBUTTONDOWN];

  virtual BOOL CanClose();
  int Pide(char *titulo, char *mensaje),

  char * Dialogo12(char *,char *,char *,
                                        char *, char *, char *,
                                        char *, char *, char *,
                                        char *)=[CM_FIRST+CM_DIALOGO1];
```

```
char * Dialogo14(char *,char *,char *,char *,char *,char *,
                                char *,char *,char *,char *,char *,char *,
                                char *,char *)=[CM_FIRST+CM_DIALOGO2];

char * Dialogo16(char *,char *,char *,char *,char *,char *,
                                char *,char *,char *,char *,char *,char *,
                                char *,char *,char *,char *)=[CM_FIRST+CM_DIALOGO3];

char * Dialogo18(char *,char *,char *,char *,char *,char *,char *,
                                char *,char *,char *,char *,char *,char *,char *,
                                char *,char *,char *,char *)=[CM_FIRST+CM_DIALOGO4];

char * Dialogo22(char *,char *,char *,char *,char *,char *,char *,
                                char *,char *,char *,char *,char *,char *,char *,
                                char *,char *,char *,char *,char *,char *,char *,
                                char *,char *,char *,char *)=[CM_FIRST+CM_DIALOGO5];
char * Dialogo24(char *,char *,char *,char *,char *,char *,
                                char *,char *,char *,char *,char *,char *,char *,
                                char *,char *,char *,char *,char *,char *,char *,
                                char *,char *,char *,char *,char *,char *,char *,
                                char *,char *)=[CM_FIRST+CM_DIALOGO6];
void Linea(void);
void LC(void)=[CM_FIRST+CM_LINEA];

void LineaGuarda(void);
void LGC(void)=[CM_FIRST+CM_LINEAGUARDA];

void LineaGuarda2(void);
void LGC2(void)=[CM_FIRST+CM_LINEAGUARDA2];

void LineaParalela(void);
void LP(void)=[CM_FIRST+CM_LINEAPARALELA];



void LineaParalelaGuarda(void);
void LPG(void)=[CM_FIRST+CM_LINEAPARALELAGUARDA];


void LineaParalelaGuarda2(void);
void LP2G(void)=[CM_FIRST+CM_LINEAPARALELAGUARDA2];


void DLC(void)=[CM_FIRST+DM_LINEA];
void DLGC(void)=[CM_FIRST+DM_LINEAGUARDA];
void DLGC2(void)=[CM_FIRST+DM_LINEAGUARDA2];
void DLP(void)=[CM_FIRST+DM_LINEAPARALELA];
void DLPG(void)=[CM_FIRST+DM_LINEAPARALELAGUARDA];
void DLP2G(void)=[CM_FIRST+DM_LINEAPARALELAGUARDA2];



void Linea(int,int,int,int);
void Circulo(int x,int y,int radio);
```

```
 int Guarda(char *L1,char *L2,char *L3,char *L4,char *L5,char *L6,char *L7,char *L8,char *L9,char
*L10);

 int Guarda(char *L1,char *L2,char *L3,char *L4,char *L5,char *L6,char *L7,char *L8,char *L9,char
*L10,char *L11,char *L12,
                        char *L13,char *L14);

 int Guarda(char *L1,char *L2,char *L3,char *L4,char *L5,char *L6,char *L7,char *L8,char *L9,char
*L10,char *L11,char *L12,
                        char *L13,char *L14,char *L15,char *L16);

int Guardap(char *L1,char *L2,char *L3,char *L4,char *L5,char *L6,char *L7,char *L8,char *L9,char
*L10,char *L11,char *L12,
          char *L13,char *L14,char *L15,char *L16);

 int Guarda(char *L1,char *L2,char *L3,char *L4,char *L5,char *L6,char *L7,char *L8,char *L9,char
*L10,char *L11,char *L12,
                        char *L13,char *L14,char *L15,char *L16,char *L17,char *L18,char *L19,
                        char *L20,char *L21,char *L22);
 int Guarda(char *L1,char *L2,char *L3,char *L4,char *L5,char *L6,char *L7,char *L8,char *L9,char
*L10,char *L11,char *L12,
                        char *L13,char *L14,char *L15,char *L16,char *L17,char *L18,char *L19,
                        char *L20,char *L21,char *L22,char *L23,char *L24,char *L25,char *L26);
 void Activa();
 void Desactiva();
 void About()
    =[CM_FIRST+CM_ABOUT];
 void Duerme(int);
 void Imagen()
    =[CM_FIRST+CM_IMAGEN];
 void DrawBMP( HDC DC, int X, int Y, HBITMAP BitMap );
   void Borrar(void)
        =[CM_FIRST + CM_BORRAR];
   void Salir(void)
  =[CM_FIRST + CM_SALIR];
  virtual void Escoge()
    =[CM_FIRST+CM_ESCOGE];
  void Recalcula(void)
          =[CM_FIRST+CM_RECALCULA];
        virtual void CMUHelpIndex( RTMessage ) = [CM_FIRST + CM_U_HELPINDEX];
        virtual void CMUHelpHelp( RTMessage ) = [CM_FIRST + CM_U_HELPHELP];

};
void Ventana::InitInstance()
{
 TApplication::InitInstance();
 if ( Status == 0 )
   HAccTable = LoadAccelerators(hInstance, "FileCommands");
}

#include "generalc.cpp"
#include "caja24.cpp"
#include "caja22.cpp"
```

```
#include "caja12.cpp"
#include "caja14.cpp"
#include "caja16.cpp"
#include "caja18.cpp"
#include "general cpp"
#endif
```

Debido a que el código se extendió demasiado fue necesario crear este archivo el cual contiene las principales definiciones de las funciones miembro de las clases vistas en el archivo anterior.

```
/*
                        GENERALE.CPP

*/
//#include"general.h"

BOOL Miventana: CanClose()
{
 return MessageBox(HWindow, "Esto terminará con su sesión ",
   "Salir de Lineas de Transmusión", MB_YESNO | MB_ICONEXCLAMATION) == IDYES;
}

void Miventana::Borrar(void){
          InvalidateRect(HWindow,NULL,TRUE);
         // contador=contadorx=contadory=0;
}

void Miventana.:Salir(void){ exit(0). }

void Miventana.:Activa(){
   HMENU hMenu = GetMenu (HWindow);
   EnableMenuItem(hMenu, CM_LINEA, MF_ENABLED);
   EnableMenuItem(hMenu, CM_LINEAGUARDA, MF_ENABLED);
   EnableMenuItem(hMenu, CM_LINEAGUARDA2, MF_ENABLED);
   EnableMenuItem(hMenu, CM_LINEAPARALELA, MF_ENABLED);
        EnableMenuItem(hMenu, CM_LINEAPARALELAGUARDA, MF_ENABLED).
        EnableMenuItem(hMenu, CM_LINEAPARALELAGUARDA2, MF_ENABLED);
   EnableMenuItem(hMenu, CM_RECALCULA, MF_ENABLED);
}

void Miventana::Desactiva(){
   HMENU hMenu = GetMenu (HWindow);
   EnableMenuItem(hMenu, CM_LINEA, MF_GRAYED);
   EnableMenuItem(hMenu, CM_LINEAGUARDA, MF_GRAYED);
   EnableMenuItem(hMenu, CM_LINEAGUARDA2, MF_GRAYED);
   EnableMenuItem(hMenu, CM_LINEAPARALELA, MF_GRAYED);

        EnableMenuItem(hMenu, CM_LINEAPARALELAGUARDA, MF_GRAYED).
        EnableMenuItem(hMenu, CM_LINEAPARALELAGUARDA2, MF_GRAYED);
}

void Miventana.:DrawBMP( HDC DC, int X, int Y, HBITMAP BitMap )
```

```
{
        HDC MemDC;
        BITMAP bm;
        BOOL MadeDC=FALSE;

        if ( DC == 0 )
        {
                DC = GetDC( HWindow );
                MadeDC = TRUE;
        }
        else
                MadeDC = FALSE;
        MemDC = CreateCompatibleDC( DC );
        SelectObject( MemDC, BitMap );
        GetObject( BitMap ,sizeof( bm ), ( LPSTR ) &bm );
        BitBlt( DC, X, Y, bm.bmWidth, bm.bmHeight, MemDC, 0, 0, SRCCOPY );
        //delay(2000);
        DeleteDC( MemDC );
        if ( MadeDC )
                ReleaseDC( HWindow, DC );
}


void Miventana::About()
{
 GetApplication()->ExecDialog(new TDialog(this, "ABOUT"));
};

//********************** Saca la Imagen ************************
void Miventana::Imagen(){
 hdc=GetDC(HWindow);
 DrawBMP( hdc, 91, 90,escudo);
 Duerme(3);
 Borrar();
 ReleaseDC( HWindow, hdc );
}
//**********************Abre Archivo de entrada**********************
void Miventana::Escoge()
{
 InvalidateRect(HWindow, NULL, TRUE);
 IsNewFile=1;

 if ( GetApplication()->ExecDialog(new TFileDialog(this, SD_FILEOPEN,
  strcpy(ArchNom, "*.dat"))) == IDOK ){
        fstream salida(ArchNom, ios::in|ios::out|ios::noreplace);
    if(!salida){
        if(MessageBox(HWindow, "¿ Quieres que lo reemplace ?", "El archivo ya existe ",
         MB_YESNO | MB_ICONEXCLAMATION)==IDYES){
                fstream salida(ArchNom,ios::in|ios::out|ios::trunc);

                salida.close();
                Activa();
                char *chafu=new char[strlen(ArchNom)+strlen("Lineas de Transmisión - ")+1];
                strcpy(chafu,"Lineas de Transmisión - ");
```

```
                    strcat(chafu,ArchNom);
                SetWindowText(HWindow, (LPSTR) chafu);
            }

        }
        else{
            Activa();
                char *chafu=new char[strlen(ArchNom)+strlen("Lineas de Transmisión - ")+1];
            strcpy(chafu,"Lineas de Transmisión - ");
                    strcat(chafu,ArchNom);
            SetWindowText(HWindow, (LPSTR) chafu);

    }
    }
    else
    {

    }
}


//******************** Pierde Tiempo ***************************
void Miventana::Duerme(int incremento){
  DWORD l;
  l=GetTickCount();
  while(GetTickCount()<l+incremento*1000L);
}


//*************Guarda 12 ****************************************

int Miventana::Guarda(char *L1,char *L2,char *L3,char *L4,char *L5,char *L6,char *L7,char *L8,char
*L9,char *L10){
  char *p=new char[150+strlen(L1)+strlen(L2)+strlen(L3)+strlen(L4)+strlen(L5)+strlen(L6)+strlen(L7)+
            strlen(L8)+strlen(L9)+strlen(L10)];
  strcpy(p,"Linea12\n\n");
  strcat(p,"Radio del Conductor:\t");
  strcat(p,L1);
  strcat(p,"\nfrecuencia:\t");
  strcat(p,L2);
  strcat(p,"\np_res_tierra:\t");
  strcat(p,L3);
  strcat(p,"\nr_RMG_conductor:\t");
  strcat(p,L4);
  strcat(p,"\nx1:\t");
  strcat(p,L5);
  strcat(p,"\ny1:\t");
  strcat(p,L6);
  strcat(p,"\nx2:\t");
  strcat(p,L7);
  strcat(p,"\ny2:\t");
  strcat(p,L8);
  strcat(p,"\nx3:\t");
  strcat(p,L9);
```

```
  strcat(p,"\ny3 \t");
  strcat(p,L10);
  strcat(p,",\n");

        fstream salida(ArchNom,ios::out|ios::in);

        if(!salida){
                MessageBox(HWindow,"Debe especificar un archivo de datos","Error",
MB_OK|MB_ICONEXCLAMATION);
                return 0;
        }
     else{
        salida.seekp(0,ios::end);
           while(*p)
           salida.put(*p++);
           salida.close();
        return 1;
           }

}



//*************Guarda 14 ****************************************

int Miventana::Guarda(char *L1,char *L2,char *L3,char *L4,char *L5,
                                                char *L6,char *L7,char *L8,char *L9,char *L10,
                                                char *L11,char *L12,char*L13,char*L14){
  char *p=new char[150+strlen(L1)+strlen(L2)+strlen(L3)+strlen(L4)+strlen(L5)+strlen(L6)+strlen(L7)+
                strlen(L8)+strlen(L9)+strlen(L10)+strlen(L11)+strlen(L12)+strlen(L13)+strlen(L14)+8];
  strcpy(p,"Linea14\n\n");
  strcat(p,"Radio del Conductor:\t");
  strcat(p,L1);
  strcat(p,"\nfrecuencia:\t");
  strcat(p,L2);
  strcat(p,"\np_res_tierra:\t");
  strcat(p,L3);
  strcat(p,"\nr_RMG_conductor:\t");
  strcat(p,L4);
  strcat(p,"\nRg_guarda:\t");
  strcat(p,L5);
  strcat(p,"\nrg_RMG_guarda:\t");
  strcat(p,L6);
  strcat(p,"\nx1:\t");
  strcat(p,L7);
  strcat(p,"\ny1:\t");
  strcat(p,L8);
  strcat(p,"\nx2:\t");
  strcat(p,L9);
  strcat(p,"\ny2:\t");
  strcat(p,L10);
  strcat(p,"\nx3:\t");
  strcat(p,L11);
  strcat(p,"\ny3:\t");
```

```
  strcat(p,L12);
  strcat(p,"\ngx:\t"),
  strcat(p,L13);
  strcat(p,"\ngy:\t");
  strcat(p,L14),
  strcat(p,";\n");
        fstream salida(ArchNom, ios::in|ios..out);
        if(!salida){
                MessageBox(HWindow,"Debe especificar un archivo de datos","Error",
MB_OK|MB_ICONEXCLAMATION);
                return 0;
        }
     else{
        salida.seekp(0,ios: end);
           while(*p)
           salida.put(*p++);
           salida close();
        return 1,
     }
}

 //*************Guarda 16 ****************************************

int Miventana::Guarda(char *L1,char *L2,char *L3,char *L4,char *L5,char *L6,char *L7,char *L8,
                                        char *L9,char *L10,char *L11,char *L12,char *L13,char
*L14,
                                        char *L15,char *L16)
                                        {
 char *p=new char[150+strlen(L1)+strlen(L2)+strlen(L3)+strlen(L4)+strlen(L5)+strlen(L6)+strlen(L7)+
                strlen(L8)+strlen(L12)+strlen(L13)+strlen(L14)+strlen(L15)+strlen(L16)+8];
  strcpy(p,"Linea16\n\n");
  strcat(p,"Radio del Conductor:\t");
  strcat(p,L1);
  strcat(p,"\nfrecuencia:\t");
  strcat(p,L2),
  strcat(p,"\np_res_tierra:\t"),
  strcat(p,L3);
  strcat(p,"\nr_RMG_conductor:\t");
  strcat(p,L4);
  strcat(p,"\nRg_guarda:\t");
  strcat(p,L5);
  strcat(p,"\nrg_RMG_guarda:\t");
  strcat(p,L6);
  strcat(p,"\nx1:\t");
  strcat(p,L7);
  strcat(p,"\ny1:\t");
  strcat(p,L8);
  strcat(p,"\nx2:\t");
  strcat(p,L9);
  strcat(p,"\ny2:\t");
  strcat(p,L10);
  strcat(p,"\nx3:\t");
  strcat(p,L11);
  strcat(p,"\ny3:\t");
```

```
  strcat(p,L12),
  strcat(p,"\ngx:\t"),
  strcat(p,L13);
  strcat(p,"\ngy:\t");
  strcat(p,L14),
  strcat(p,"\ngx1:\t"),
  strcat(p,L15);
  strcat(p,"\ngy1:\t"),
  strcat(p,L16),
  strcat(p,",\n"),

        fstream salida(ArchNom, ios::in|ios::out),
        if(!salida){
                MessageBox(HWindow,"Debe especificar un archivo de datos","Error",
MB_OK|MB_ICONEXCLAMATION);
                return 0;
        }
     else{
        salida seekp(0,ios::end);
            while(*p)
                salida.put(*p++),
            salida.close(),
        return 1;
     }
}

//*************Guarda 18 ******************************************

int Miventana::Guardap(char *L1,char *L2,char *L3,char *L4,char *L5,char *L6,
                                        char *L7,char *L8,char *L9,char *L10,char *L11,char *L12,
                                        char *L13,char *L14,char *L15,char *L16){

  char *p=new char[150+strlen(L1)+strlen(L2)+strlen(L3)+strlen(L4)+strlen(L5)+strlen(L6)+strlen(L7)+
                   strlen(L8)+strlen(L9)+strlen(L10)+strlen(L11)+strlen(L12)+strlen(L13)+strlen(L14)+
                   strlen(L15)+strlen(L16)+8];

  strcpy(p,"Linea18\n\n"),
  strcat(p,"Radio del Conductor:\t"),
  strcat(p,L1);
  strcat(p,"\nfrecuencia:\t");
  strcat(p,L2),
  strcat(p,"\np_res_tierra:\t");
  strcat(p,L3);
  strcat(p,"\nr_RMG_conductor:\t"),
  strcat(p,L4);
   strcat(p,"\nx1:\t");
  strcat(p,L5);
   strcat(p,"\ny1:\t"),
  strcat(p,L6);
  strcat(p,"\nx2:\t"),
  strcat(p,L7);
  strcat(p,"\ny2:\t");
  strcat(p,L8);
    strcat(p,"\nx3:\t"),
```

```
  strcat(p,L9);
   strcat(p,"\ny3:\t");
  strcat(p,L10);
   strcat(p,"\nx11 \t");
  strcat(p,L11);
   strcat(p,"\ny11:\t");
  strcat(p,L12);
   strcat(p,"\nx22:\t");
  strcat(p,L13);
   strcat(p,"\ny22 \t");
  strcat(p,L14);
    strcat(p,"\nx33:\t");
  strcat(p,L15);
    strcat(p,"\ny33.\t");
  strcat(p,L16);
  strcat(p,";\n");

        fstream salida(ArchNom, ios::in|ios::out);
        if(!salida){
                MessageBox(HWindow,"Debe especificar un archivo de datos","Error",
MB_OK|MB_ICONEXCLAMATION);
                return 0;
          }
      else{
         salida.seekp(0,ios:.end);
            while(*p)
            salida put(*p++);
            salida.close();
         return 1;
      }
}


//*************Guarda 22 ****************************************

int Miventana::Guarda(char *L1,char *L2,char *L3,char *L4,char *L5,char *L6,char *L7,
                                        char *L8,char *L9,char *L10,char *L11,char *L12,char
*L13,char *L14,
                                        char *L15,char *L16, char *L17,char *L18, char *L19,char
*L20,
                                        char *L21, char *L22){

  char *p=new char[150+strlen(L1)+strlen(L2)+strlen(L3)+strlen(L4)+strlen(L5)+strlen(L6)+strlen(L7)+
                  strlen(L8)+strlen(L9)+strlen(L10)+strlen(L11)+strlen(L12)+strlen(L13)+strlen(L14)+
                  strlen(L15)+strlen(L16)+strlen(L17)+strlen(L18)+strlen(L19)+strlen(L20)+
                  strlen(L21)+strlen(L22)];
  strcpy(p,"Linea22\n\n");
  strcat(p,"Radio del Conductor:\t");
  strcat(p,L1);
  strcat(p,"\nfrecuencia.\t");
  strcat(p,L2);
  strcat(p,"\np_res_tierra:\t");
  strcat(p,L3);
  strcat(p,"\nr_RMG_conductor \t");
```

```
  strcat(p,L4);
  strcat(p,"\nRg_guarda:\t");
  strcat(p,L5);
  strcat(p,"\nrg_RMG_guarda:\t");
  strcat(p,L6);
  strcat(p,"\nx1:\t");
  strcat(p,L7);
  strcat(p,"\ny1:\t");
  strcat(p,L8);
  strcat(p,"\nx2:\t");
  strcat(p,L9);
  strcat(p,"\ny2:\t");
  strcat(p,L10);
  strcat(p,"\nx3:\t");
  strcat(p,L11);
  strcat(p,"\ny3:\t");
  strcat(p,L12);
  strcat(p,"\nx11:\t");
  strcat(p,L15);
  strcat(p,"\ny11:\t");
  strcat(p,L16);
  strcat(p,"\nx22:\t");
  strcat(p,L17);
  strcat(p,"\ny22:\t");
  strcat(p,L18);
  strcat(p,"\nx33:\t");
  strcat(p,L19);
  strcat(p,"\ny33:\t");
  strcat(p,L20);
  strcat(p,"\ngx1:\t");
  strcat(p,L21);
  strcat(p,"\ngy1:\t");
  strcat(p,L22);
  strcat(p,";\n");
                fstream salida(ArchNom, ios::in|ios::out);
        if(!salida){
                MessageBox(HWindow,"Debe especificar un archivo de datos","Error",
MB_OK|MB_ICONEXCLAMATION);
                return 0;
        }
    else{
       salida.seekp(0,ios::end);
          while(*p)
          salida.put(*p++);
          salida.close();
       return 1;
    }
}

//**************Guarda 24 ******************************************

int Miventana::Guarda(char *L1,char *L2,char *L3,char *L4,char *L5,char *L6,char *L7,
                                                char *L8,char *L9,char *L10,char *L11,char *L12,char
*L13,char *L14,
```

```
                                        char *L15,char *L16, char *L17,char *L18, char *L19,char
*L20,
                                        char *L21, char *L22,char *L23,char *L24,char *L25,char
*L26){

  char *p=new char[150+strlen(L1)+strlen(L2)+strlen(L3)+strlen(L4)+strlen(L5)+strlen(L6)+strlen(L7)+
                   strlen(L8)+strlen(L9)+strlen(L10)+strlen(L11)+strlen(L12)+strlen(L13)+strlen(L14)+
                   strlen(L15)+strlen(L16)+strlen(L17)+strlen(L18)+strlen(L19)+strlen(L20)+
                   strlen(L21)+strlen(L22)+strlen(L23)+strlen(L24)+strlen(L25)+strlen(L26)];

  strcpy(p,"Linea24\n\n");
  strcat(p,"Radio del Conductor:\t");
  strcat(p,L1);
  strcat(p,"\nfrecuencia:\t");
  strcat(p,L2);
  strcat(p,"\np_res_tierra:\t");
  strcat(p,L3);
  strcat(p,"\nr_RMG_conductor:\t");
  strcat(p,L4);
  strcat(p,"\nRg_guarda:\t");
  strcat(p,L5);
  strcat(p,"\nrg_RMG_guarda:\t");
  strcat(p,L6);
  strcat(p,"\nx1:\t");
  strcat(p,L7);
  strcat(p,"\ny1:\t");
  strcat(p,L8);
  strcat(p,"\nx2:\t");
  strcat(p,L9);
  strcat(p,"\ny2:\t");
  strcat(p,L10);
  strcat(p,"\nx3:\t");
  strcat(p,L11);
  strcat(p,"\ny3:\t");
  strcat(p,L12);
  strcat(p,"\nx11:\t");
  strcat(p,L13);
  strcat(p,"\ny11:\t");
  strcat(p,L14);
  strcat(p,"\nx22:\t");
  strcat(p,L15);
  strcat(p,"\ny22:\t");
  strcat(p,L16);
  strcat(p,"\nx33:\t");
  strcat(p,L17);
  strcat(p,"\ny33:\t");
  strcat(p,L18);
  strcat(p,"\ngx:\t");
  strcat(p,L19);
  strcat(p,"\ngy:\t");
  strcat(p,L20);
  strcat(p,"\ngx1:\t");
  strcat(p,L21);
  strcat(p,"\ngy1:\t");
```

```
  strcat(p,L22);
  strcat(p,"\ngx2:\t");
  strcat(p,L23);
  strcat(p,"\ngy2:\t");
  strcat(p,L24);
  strcat(p,"\ngx3:\t");
  strcat(p,L25);
  strcat(p,"\ngy3:\t");
  strcat(p,L26);
  strcat(p,";\n");
  fstream salida(ArchNom, ios::in|ios::out);
        if(!salida){
                MessageBox(HWindow,"Debe especificar un archivo de datos","Error",
MB_OK|MB_ICONEXCLAMATION);
                return 0;
        }
    else{
       salida.seekp(0,ios::end);
            while(*p)
            salida.put(*p++);
            salida.close();
       return 1;
    }
}

//********* FIN DE CLASE 3 (MI VENTANA) ********

void Miventana::Linea(int x1,int y1,int x2,int y2){
    HDC DC;
    DC=GetDC(HWindow);
    MoveTo(DC,x2,y2);
        LineTo(DC,x1,y1);

}
void Miventana::Circulo(int x,int y,int radio){

    HDC DC;
        DC = GetDC(HWindow);
        Ellipse(DC,x-radio,y+radio,x+radio,y-radio);
}
void Miventana::Linea(void){
  Linea(180,200,170,190);
  Linea(180,200,170,210);

  Linea(50,200,180,200);
  Linea(50,200,50,50);
  Linea(50,50,40,60);
  Linea(50,50,60,60);

  Circulo(100,80,10);
  Circulo(50,140,10);
  Circulo(140,200,10);
```

```
}

void Miventana::LineaGuarda(void){
 // Linea().
  Linea(180,200,170,190).
  Linea(180,200,170,210);
  Linea(50,200,180,200);
  Linea(50,200,50,50).
  Linea(50,50,40,60).
  Linea(50,50,60,60),

  Circulo(50,200,10);
  Circulo(115,140,10);
  Circulo(180,200,10);

  Circulo(115,70,5);


}
void Miventana::LineaGuarda2(void){
   LineaGuarda();
   Circulo(150,70,5),

}
void Miventana::LineaParalela(void){
  Linea().
  //Horizontal
  Linea(350,200,190,200),
  Linea(200,210,190,200);
  Linea(200,190,190,200),
  //Vertical
  Linea(350,200,350,50);
  Linea(350,50,340,60);
  Linea(350,50,360,60);
  Circulo(280,80,10);
  Circulo(350,140,10);
  Circulo(270,200,10);

}
void Miventana::LineaParalelaGuarda(void){
  LineaParalela();
  Circulo(240,30,5);
  Circulo(150,30,5).


}
void Miventana::LC(void){
 //       Linea().
         char L1[20],L2[20],L3[20],L4[20],L5[20],L6[20],L7[20],
         L8[20],L9[20],L10[20];

         Dialogo12(L1,L2,L3,L4,L5,L6,L7,L8,L9,L10);

}
```

```
void Miventana::LGC(void){
   //      LineaGuarda();
           char L1[20],L2[20],L3[20],L4[20],L5[20],L6[20],L7[20],
           L8[20],L9[20],L10[20],L11[20],L12[20],L13[20],L14[20];

           Dialogo14(L1,L2,L3,L4,L5,L6,L7,L8,L9,L10,L11,L12,L13,L14);

}
void Miventana::LGC2(void){
   //      LineaGuarda2();
           char L1[20],L2[20],L3[20],L4[20],L5[20],L6[20],L7[20],
           L8[20],L9[20],L10[20],L11[20],L12[20],L13[20],L14[20],L15[20],L16[20];

           Dialogo16(L1,L2,L3,L4,L5,L6,L7,L8,L9,L10,L11,L12,L13,L14,L15,L16);


}
void Miventana. LP(void){
    //      LineaParalela();
           char L1[20],L2[20],L3[20],L4[20],L5[20],L6[20],L7[20],
           L8[20],L9[20],L10[20],L11[20],L12[20],L13[20],L14[20],L15[20],L16[20],

           Dialogo18(L1,L2,L3,L4,L5,L6,L7,L8,L9,L10,L11,L12,L13,L14,L15,L16);

}
void Miventana  LPG(void){
   //      LineaParalelaGuarda();
           char L1[20],L2[20],L3[20],L4[20],L5[20],L6[20],L7[20],
           L8[20],L9[20],L10[20],L11[20],L12[20],L13[20],L14[20],L15[20],L16[20],L17[20],L18[20],
           L19[20],L20[20],L21[20],L22[20];


           Dialogo22(L1,L2,L3,L4,L5,L6,L7,L8,L9,L10,L11,L12,L13,L14,L15,L16,L17,L18,L19,L20,L21,L2
2);

}

void Miventana::LP2G(void){
    //      LineaParalelaGuarda();
           char L1[20],L2[20],L3[20],L4[20],L5[20],L6[20],L7[20],
           L8[20],L9[20],L10[20],L11[20],L12[20],L13[20],L14[20],L15[20],L16[20],L17[20],L18[20],
           L19[20],L20[20],L21[20],L22[20],L23[20],L24[20],L25[20],L26[20];

           Dialogo24(L1,L2,L3,L4,L5,L6,L7,L8,L9,L10,L11,
            L12,L13,L14,L15,L16,L17,L18,L19,L20,L21,L22,L23,L24,L25,L26);

}


void Miventana::DLC(void){
 hdc=GetDC(HWindow);
 DrawBMP( hdc, 91, 20,BITMAP_1);
 Duerme(5);
 Borrar();
```

```
  ReleaseDC( HWindow, hdc ),
}

void Miventana::DLGC(void){
   hdc=GetDC(HWindow);
  DrawBMP( hdc, 91, 20,BITMAP_2);
  Duerme(5);
  Borrar(),
  ReleaseDC( HWindow, hdc );
}
void Miventana: DLGC2(void){
   hdc=GetDC(HWindow),
  DrawBMP( hdc, 91, 20,BITMAP_3);
  Duerme(5);
  Borrar();
  ReleaseDC( HWindow, hdc );

}
void Miventana :DLP(void){
    hdc=GetDC(HWindow);
  DrawBMP( hdc, 91, 20,BITMAP_4);
  Duerme(5);
  Borrar();
  ReleaseDC( HWindow, hdc );
}
void Miventana::DLPG(void){
   hdc=GetDC(HWindow);
  DrawBMP( hdc, 91, 20.BITMAP_5);
  Duerme(5);
  Borrar();
  ReleaseDC( HWindow, hdc );
}

void Miventana::DLP2G(void){
  hdc=GetDC(HWindow);
  DrawBMP( hdc, 91, 20,BITMAP_6);
  Duerme(5);
  Borrar();
  ReleaseDC( HWindow, hdc );
}
```

Aquí empieza un conjunto de archivos que definen todo el código necesario para la ejecucción de la cajas de diálogo y su nombre define la cantidad de campos que están capturando es decir caja24 captura una cantidad de 24 campos distintos y de forma similar los demás archivos.

```
/*
        CAJA24.CPP

*/

//#include"general.h"
```

```
//*************** CONSTRUCTOR ***************************

Caja24::Caja24(PTWindowsObject Aparent,LPSTR name):TDialog(Aparent,name)
{
    new TEdit(this,ID_C1,
        sizeof(((Miventana *)Parent)->Transfiere24 c1)),//del tamaño de c1
    //hacemos un cast  del tipo ventana al tipo parent

    new TEdit(this,ID_C2,
        sizeof(((Miventana *)Parent)->Transfiere24 c2)),

    new TEdit(this,ID_C3,
        sizeof(((Miventana *)Parent)->Transfiere24.c3));

    new TEdit(this,ID_C4,
        sizeof(((Miventana *)Parent)->Transfiere24.c4)),
                //hacemos un cast  del tipo ventana al tipo parent
    new TEdit(this,ID_C5,
        sizeof(((Miventana *)Parent)->Transfiere24.c5)),

    new TEdit(this,ID_C6,
        sizeof(((Miventana *)Parent)->Transfiere24.c6)),

    new TEdit(this,ID_C7,
        sizeof(((Miventana *)Parent)->Transfiere24.c7));

    new TEdit(this,ID_C8,
        sizeof(((Miventana *)Parent)->Transfiere24.c8)),

    new TEdit(this,ID_C9,
        sizeof(((Miventana *)Parent)->Transfiere24.c9));

    new TEdit(this,ID_C10,
        sizeof(((Miventana *)Parent)->Transfiere24.c10));

    new TEdit(this,ID_C11,
        sizeof(((Miventana *)Parent)->Transfiere24.c11));

    new TEdit(this,ID_C12,
        sizeof(((Miventana *)Parent)->Transfiere24.c12));
                //hacemos un cast  del tipo ventana al tipo parent
    new TEdit(this,ID_C13,
        sizeof(((Miventana *)Parent)->Transfiere24 c13));

    new TEdit(this,ID_C14,
        sizeof(((Miventana *)Parent)->Transfiere24 c14));

    new TEdit(this,ID_C15,
        sizeof(((Miventana *)Parent)->Transfiere24.c15));

    new TEdit(this,ID_C16,
        sizeof(((Miventana *)Parent)->Transfiere24.c16));

    new TEdit(this,ID_C17,
```

```
            sizeof(((Miventana *)Parent)->Transfiere24.c17));

      new TEdit(this,ID_C18,
            sizeof(((Miventana *)Parent)->Transfiere24.c18)),

      new TEdit(this,ID_C19,
            sizeof(((Miventana *)Parent)->Transfiere24.c19));

      new TEdit(this,ID_C20,
            sizeof(((Miventana *)Parent)->Transfiere24.c20));

      new TEdit(this,ID_C21,
            sizeof(((Miventana *)Parent)->Transfiere24.c21)),

      new TEdit(this,ID_C22,
            sizeof(((Miventana *)Parent)->Transfiere24.c22));

      new TEdit(this,ID_C23,
            sizeof(((Miventana *)Parent)->Transfiere24.c23));

      new TEdit(this,ID_C24,
            sizeof(((Miventana *)Parent)->Transfiere24.c24));

      new TEdit(this,ID_C25,
            sizeof(((Miventana *)Parent)->Transfiere24.c25));

            new TEdit(this,ID_C26,
            sizeof(((Miventana *)Parent)->Transfiere24.c26));                    //hacemos un cast  del tipo
ventana al tipo parent
            TransferBuffer=(void far*)&(((Miventana*)Parent)->Transfiere24);


}  //************ FIN DE CONSTRUCTOR ************

BOOL Caja24::CanClose(){
        FillBuffers();
return TRUE;
}

void Caja24::FillBuffers(){
      //Para Dejar Los Valores en el Campo
          GetDlgItemText(HWindow,ID_C1,Camp1,MAXC);
          GetDlgItemText(HWindow,ID_C2,Camp2,MAXC);
          GetDlgItemText(HWindow,ID_C3,Camp3,MAXC);
          GetDlgItemText(HWindow,ID_C4,Camp4,MAXC);
          GetDlgItemText(HWindow,ID_C5,Camp5,MAXC);
          GetDlgItemText(HWindow,ID_C6,Camp6,MAXC);

          GetDlgItemText(HWindow,ID_C7,Camp7,MAXC);
          GetDlgItemText(HWindow,ID_C8,Camp8,MAXC);
          GetDlgItemText(HWindow,ID_C9,Camp9,MAXC);
          GetDlgItemText(HWindow,ID_C10,Camp10,MAXC);
          GetDlgItemText(HWindow,ID_C11,Camp11,MAXC);
```

```
        GetDlgItemText(HWindow,ID_C12,Camp12,MAXC);
        GetDlgItemText(HWindow,ID_C13,Camp13,MAXC);
        GetDlgItemText(HWindow,ID_C14,Camp14,MAXC);
        GetDlgItemText(HWindow,ID_C15,Camp15,MAXC);
        GetDlgItemText(HWindow,ID_C16,Camp16,MAXC);
        GetDlgItemText(HWindow,ID_C17,Camp17,MAXC);
        GetDlgItemText(HWindow,ID_C18,Camp18,MAXC);

        GetDlgItemText(HWindow,ID_C19,Camp19,MAXC);
        GetDlgItemText(HWindow,ID_C20,Camp20,MAXC);
        GetDlgItemText(HWindow,ID_C21,Camp21,MAXC);
        GetDlgItemText(HWindow,ID_C22,Camp22,MAXC);
    GetDlgItemText(HWindow,ID_C23,Camp23,MAXC);
        GetDlgItemText(HWindow,ID_C24,Camp24,MAXC);
        GetDlgItemText(HWindow,ID_C25,Camp25,MAXC);
        GetDlgItemText(HWindow,ID_C26,Camp26,MAXC);


}
//////////////////////

char* Miventana::Dialogo24( char *L1,char *L2,char *L3,char *L4,char *L5,
                                                    char *L6,char *L7,char *L8,char *L9,char
*L10,
                                                    char *L11,char *L12,char *L13,char *L14
                .char *L15,char *L16,char *L17, char *L18,
                                                    char *L19,char *L20,char *L21,char
*L22,char *L23,
                                                    char *L24,char *L25,char *L26){

        if(GetModule()->ExecDialog(new Caja24(this,"DIALOG_07"))==IDOK)
        {

    Desactiva();

  strcpy(L1,Transfiere24.c1);
  strcpy(L2,Transfiere24.c2);
  strcpy(L3,Transfiere24.c3);
  strcpy(L4,Transfiere24.c4);
  strcpy(L5,Transfiere24.c5);
  strcpy(L6,Transfiere24.c6);
  strcpy(L7,Transfiere24.c7);
  strcpy(L8,Transfiere24.c8);
  strcpy(L9,Transfiere24.c9);
  strcpy(L10,Transfiere24.c10);
  strcpy(L11,Transfiere24.c11);
  strcpy(L12,Transfiere24.c12);
  strcpy(L13,Transfiere24.c13);
  strcpy(L14,Transfiere24.c14);
  strcpy(L15,Transfiere24.c15);
  strcpy(L16,Transfiere24.c16);
  strcpy(L17,Transfiere24.c17);
  strcpy(L18,Transfiere24.c18);
  strcpy(L19,Transfiere24.c19);
```

```
  strcpy(L20,Transfiere24 c20);
  strcpy(L21,Transfiere24 c21);
  strcpy(L22,Transfiere24.c22);
  strcpy(L23,Transfiere24.c23);
  strcpy(L24,Transfiere24.c24);
  strcpy(L25,Transfiere24.c25);
  strcpy(L26,Transfiere24.c26);

  Guarda(L12,L2,L3,L4,L5,L6,L7,L8,L9,L10,L11,L12,L13,L14,
      L15,L16,L17,L18,L19,L20,L21,L22,L23,L24,L25,L26);

  recalcula=6;
  int q=Calcula(ArchNom,6);
  if(q==0) EXITO
  if(q==-1) E_ARGUMENTO
  if(q==-2) E_ARCHIVOSALIDA
  if(q==-3) E_OPCION
  if(q==-4) E_ARCHIVOENTRADA
return "";

  }
}

/*

        CAJA22.CPP

*/


//#include"general.h"
Caja22::Caja22(PTWindowsObject Aparent,LPSTR name):TDialog(Aparent,name)
{
    new TEdit(this,ID_C1,
        sizeof(((Miventana *)Parent)->Transfiere22.c1));//del tamaño de c1
    //hacemos un cast  del tipo ventana al tipo parent

    new TEdit(this,ID_C2,
        sizeof(((Miventana *)Parent)->Transfiere22 c2));

    new TEdit(this,ID_C3,
        sizeof(((Miventana *)Parent)->Transfiere22.c3));

    new TEdit(this,ID_C4,
        sizeof(((Miventana *)Parent)->Transfiere22.c4));
            //hacemos un cast  del tipo ventana al tipo parent
    new TEdit(this,ID_C5,
        sizeof(((Miventana *)Parent)->Transfiere22.c5));

    new TEdit(this,ID_C6,
        sizeof(((Miventana *)Parent)->Transfiere22.c6));

    new TEdit(this,ID_C7,
        sizeof(((Miventana *)Parent)->Transfiere22.c7));
```

```
    new TEdit(this,ID_C8,
        sizeof(((Miventana *)Parent)->Transfiere22.c8));

    new TEdit(this,ID_C9,
        sizeof(((Miventana *)Parent)->Transfiere22 c9));

    new TEdit(this,ID_C10,
        sizeof(((Miventana *)Parent)->Transfiere22 c10));

    new TEdit(this,ID_C11,
        sizeof(((Miventana *)Parent)->Transfiere22 c11));

    new TEdit(this,ID_C12,
        sizeof(((Miventana *)Parent)->Transfiere22 c12));
                //hacemos un cast  del tipo ventana al tipo parent
    new TEdit(this,ID_C13,
        sizeof(((Miventana *)Parent)->Transfiere22.c13)),

    new TEdit(this,ID_C14,
        sizeof(((Miventana *)Parent)->Transfiere22.c14)),

    new TEdit(this,ID_C15,
        sizeof(((Miventana *)Parent)->Transfiere22.c15));

    new TEdit(this,ID_C16,
        sizeof(((Miventana *)Parent)->Transfiere22.c16));

    new TEdit(this,ID_C17,
        sizeof(((Miventana *)Parent)->Transfiere22.c17));

    new TEdit(this,ID_C18,
        sizeof(((Miventana *)Parent)->Transfiere22.c18));

    new TEdit(this,ID_C19,
        sizeof(((Miventana *)Parent)->Transfiere22.c19)),

    new TEdit(this,ID_C20,
        sizeof(((Miventana *)Parent)->Transfiere22.c20));

    new TEdit(this,ID_C21,
        sizeof(((Miventana *)Parent)->Transfiere22.c21));

    new TEdit(this,ID_C22,
        sizeof(((Miventana *)Parent)->Transfiere22.c22)),

            //hacemos un cast  del tipo ventana al tipo parent
        TransferBuffer=(void far*)&(((Miventana*)Parent)->Transfiere22),
            //apunta a Transfiere22

} //************ FIN DE CONSTRUCTOR ************

BOOL Caja22: CanClose(){
    FillBuffers(),
```

```
return TRUE.
}

void Caja22  FillBuffers(){

        GetDlgItemText(HWindow,ID_C1,Camp1,MAXC);
        GetDlgItemText(HWindow,ID_C2,Camp2,MAXC);
        GetDlgItemText(HWindow,ID_C3,Camp3,MAXC);
        GetDlgItemText(HWindow,ID_C4,Camp4,MAXC);
        GetDlgItemText(HWindow,ID_C5,Camp5,MAXC);
        GetDlgItemText(HWindow,ID_C6,Camp6,MAXC);

        GetDlgItemText(HWindow,ID_C7,Camp7,MAXC);
        GetDlgItemText(HWindow,ID_C8,Camp8,MAXC);
        GetDlgItemText(HWindow,ID_C9,Camp9,MAXC);
        GetDlgItemText(HWindow,ID_C10,Camp10,MAXC);
        GetDlgItemText(HWindow,ID_C11,Camp11,MAXC);

        GetDlgItemText(HWindow,ID_C12,Camp12,MAXC);
        GetDlgItemText(HWindow,ID_C13,Camp13,MAXC);
        GetDlgItemText(HWindow,ID_C14,Camp14,MAXC);
        GetDlgItemText(HWindow,ID_C15,Camp15,MAXC);
        GetDlgItemText(HWindow,ID_C16,Camp16,MAXC);
        GetDlgItemText(HWindow,ID_C17,Camp17,MAXC);
        GetDlgItemText(HWindow,ID_C18,Camp18,MAXC);

        GetDlgItemText(HWindow,ID_C19,Camp19,MAXC);
        GetDlgItemText(HWindow,ID_C20,Camp20,MAXC);
        GetDlgItemText(HWindow,ID_C21,Camp21,MAXC);
        GetDlgItemText(HWindow,ID_C22,Camp22,MAXC);


}
//////////////////////

char* Miventana::Dialogo22( char *L1,char *L2,char *L3,char *L4,char *L5,
                                                        char *L6,char *L7,char *L8,char *L9,char
*L10,
                                                        char *L11,char *L12,char *L13,char *L14
        ,char *L15,char *L16,char *L17, char *L18,
                                                        char *L19,char *L20,char *L21,char *L22){

        if(GetModule()->ExecDialog(new Caja22(this,"DIALOG_06"))==IDOK)
        {
    Desactiva();

  strcpy(L1,Transfiere22 c1);
  strcpy(L2,Transfiere22 c2);
  strcpy(L3,Transfiere22 c3);
  strcpy(L4,Transfiere22.c4);
  strcpy(L5,Transfiere22.c5);
  strcpy(L6,Transfiere22.c6);
  strcpy(L7,Transfiere22.c7);
  strcpy(L8,Transfiere22 c8);
```

```
  strcpy(L9,Transfiere22.c9);
  strcpy(L10,Transfiere22.c10);
  strcpy(L11,Transfiere22.c11);
  strcpy(L12,Transfiere22.c12);
  strcpy(L13,Transfiere22.c13);
  strcpy(L14,Transfiere22.c14);
  strcpy(L15,Transfiere22.c15);
  strcpy(L16,Transfiere22.c16);
  strcpy(L17,Transfiere22.c17);
  strcpy(L18,Transfiere22.c18);
  strcpy(L19,Transfiere22.c19);
  strcpy(L20,Transfiere22.c20);
  strcpy(L21,Transfiere22.c21);
  strcpy(L22,Transfiere22.c22);

  Guarda(L12,L2,L3,L4,L5,L6,L7,L8,L9,L10,L11,L12,L13,L14,L15,L16,L17,L18,L19,L20,L21,L22);
  recalcula=5;
  int q=Calcula(ArchNom,5);
  if(q==0) EXITO
  if(q==-1) E_ARGUMENTO
  if(q==-2) E_ARCHIVOSALIDA
  if(q==-3) E_OPCION
  if(q==-4) E_ARCHIVOENTRADA
return "";

  }
}

/*

                        CAJA18.CPP


*/
Caja18  Caja18(PTWindowsObject Aparent,LPSTR name):TDialog(Aparent,name)
{
    new TEdit(this,ID_C1,
          sizeof(((Miventana *)Parent)->Transfiere18.c1));//del tamaño de c1
          //hacemos un cast  del tipo ventana al tipo parent

    new TEdit(this,ID_C2,
          sizeof(((Miventana *)Parent)->Transfiere18 c2));

    new TEdit(this,ID_C3,
          sizeof(((Miventana *)Parent)->Transfiere18 c3));

    new TEdit(this,ID_C4,
          sizeof(((Miventana *)Parent)->Transfiere18.c4)),
                //hacemos un cast  del tipo ventana al tipo parent
    new TEdit(this,ID_C5,
          sizeof(((Miventana *)Parent)->Transfiere18.c5));
```

```
    new TEdit(this,ID_C6,
          sizeof(((Miventana *)Parent)->Transfiere18.c6));

    new TEdit(this,ID_C7,
          sizeof(((Miventana *)Parent)->Transfiere18.c7));

    new TEdit(this,ID_C8,
          sizeof(((Miventana *)Parent)->Transfiere18.c8)),

    new TEdit(this,ID_C9,
          sizeof(((Miventana *)Parent)->Transfiere18 c9));

    new TEdit(this,ID_C10,
          sizeof(((Miventana *)Parent)->Transfiere18.c10));

    new TEdit(this,ID_C11,
          sizeof(((Miventana *)Parent)->Transfiere18.c11));

    new TEdit(this,ID_C12,
          sizeof(((Miventana *)Parent)->Transfiere18.c12));
                //hacemos un cast  del tipo ventana al tipo parent
    new TEdit(this,ID_C13,
          sizeof(((Miventana *)Parent)->Transfiere18.c13));

    new TEdit(this,ID_C14,
          sizeof(((Miventana *)Parent)->Transfiere18.c14));

    new TEdit(this,ID_C15,
          sizeof(((Miventana *)Parent)->Transfiere18.c15));

    new TEdit(this,ID_C16,
          sizeof(((Miventana *)Parent)->Transfiere18 c16));

                //hacemos un cast  del tipo ventana al tipo parent

          TransferBuffer=(void far*)&(((Miventana*)Parent)->Transfiere18);
                //apunta a Transfiere18

} //************ FIN DE CONSTRUCTOR ************

BOOL Caja18::CanClose(){
        FillBuffers();
return TRUE;
}

void Caja18::FillBuffers(){
    //Para Dejar Los Valores en el Campo
        GetDlgItemText(HWindow,ID_C1,Camp1,MAXC);
        GetDlgItemText(HWindow,ID_C2,Camp2,MAXC);
        GetDlgItemText(HWindow,ID_C3,Camp3,MAXC);
        GetDlgItemText(HWindow,ID_C4,Camp4,MAXC);
        GetDlgItemText(HWindow,ID_C5,Camp5,MAXC);
        GetDlgItemText(HWindow,ID_C6,Camp6,MAXC);
```

```
        GetDlgItemText(HWindow,ID_C7,Camp7,MAXC);
        GetDlgItemText(HWindow,ID_C8,Camp8,MAXC);
        GetDlgItemText(HWindow,ID_C9,Camp9,MAXC);
        GetDlgItemText(HWindow,ID_C10,Camp10,MAXC);
        GetDlgItemText(HWindow,ID_C11,Camp11,MAXC);
        GetDlgItemText(HWindow,ID_C12,Camp12,MAXC);
        GetDlgItemText(HWindow,ID_C13,Camp13,MAXC);
        GetDlgItemText(HWindow,ID_C14,Camp14,MAXC);
        GetDlgItemText(HWindow,ID_C15,Camp15,MAXC);
        GetDlgItemText(HWindow,ID_C16,Camp16,MAXC);

}
//////////////////////////

char* MiventanaDialogo18(char *L1,char *L2,char *L3,char *L4,char *L5,
                                                    char *L6,char *L7,char *L8,char *L9,char
*L10,
                                                    char *L11,char *L12,char *L13,char *L14
                                                    ,char *L15,char *L16){
         if(GetModule()->ExecDialog(new Caja18(this,"DIALOG_04"))==IDOK)
        {
         Desactiva();


  strcpy(L1,Transfiere18.c1);
  strcpy(L2,Transfiere18.c2);
  strcpy(L3,Transfiere18.c3);
  strcpy(L4,Transfiere18.c4);
  strcpy(L5,Transfiere18.c5);
  strcpy(L6,Transfiere18.c6);
  strcpy(L7,Transfiere18.c7);
  strcpy(L8,Transfiere18.c8);
  strcpy(L9,Transfiere18.c9);
  strcpy(L10,Transfiere18.c10);
  strcpy(L11,Transfiere18.c11);
  strcpy(L12,Transfiere18.c12);
  strcpy(L13,Transfiere18.c13);
  strcpy(L14,Transfiere18.c14);
  strcpy(L15,Transfiere18.c15);
  strcpy(L16,Transfiere18.c16);


  Guardap(L12,L2,L3,L4,L5,L6,L7,L8,L9,L10,L11,L12,L13,L14,L15,L16);
  recalcula=4;
  int q=Calcula(ArchNom,4);
  if(q==0) EXITO
  if(q==-1) E_ARGUMENTO
  if(q==-2) E_ARCHIVOSALIDA
  if(q==-3) E_OPCION
  if(q==-4) E_ARCHIVOENTRADA
return " ";

  }
}
```

```
/*

                        CAJA16.CPP

*/

Caja16::Caja16(PTWindowsObject Aparent,LPSTR name):TDialog(Aparent,name)
{
    new TEdit(this,ID_C1,
        sizeof(((Miventana *)Parent)->Transfiere16.c1));//del tamaño de c1
    //hacemos un cast  del tipo ventana al tipo parent

    new TEdit(this,ID_C2,
        sizeof(((Miventana *)Parent)->Transfiere16 c2));

    new TEdit(this,ID_C3,
        sizeof(((Miventana *)Parent)->Transfiere16.c3));

    new TEdit(this,ID_C4,
        sizeof(((Miventana *)Parent)->Transfiere16.c4));
            //hacemos un cast  del tipo ventana al tipo parent
    new TEdit(this,ID_C5,
        sizeof(((Miventana *)Parent)->Transfiere16.c5));

    new TEdit(this,ID_C6,
        sizeof(((Miventana *)Parent)->Transfiere16 c6));

    new TEdit(this,ID_C7,
        sizeof(((Miventana *)Parent)->Transfiere16.c7));

    new TEdit(this,ID_C8,
        sizeof(((Miventana *)Parent)->Transfiere16.c8));

    new TEdit(this,ID_C9,
        sizeof(((Miventana *)Parent)->Transfiere16.c9));

    new TEdit(this,ID_C10,
        sizeof(((Miventana *)Parent)->Transfiere16.c10));

    new TEdit(this,ID_C11,
        sizeof(((Miventana *)Parent)->Transfiere16 c11));

    new TEdit(this,ID_C12,
        sizeof(((Miventana *)Parent)->Transfiere16 c12));
            //hacemos un cast  del tipo ventana al tipo parent
    new TEdit(this,ID_C13,
        sizeof(((Miventana *)Parent)->Transfiere16.c13));
```

```
    new TEdit(this,ID_C14,
        sizeof(((Miventana *)Parent)->Transfiere16.c14)),

    new TEdit(this,ID_C15,
        sizeof(((Miventana *)Parent)->Transfiere16.c15)),

    new TEdit(this,ID_C16,
        sizeof(((Miventana *)Parent)->Transfiere16.c16)),

        TransferBuffer=(void far*)&(((Miventana*)Parent)->Transfiere16);


} //************ FIN DE CONSTRUCTOR ************

BOOL Caja16::CanClose(){
      FillBuffers();
return TRUE;
}

void Caja16::FillBuffers(){
    //Para Dejar Los Valores en el Campo
      GetDlgItemText(HWindow,ID_C1,Camp1,MAXC);
      GetDlgItemText(HWindow,ID_C2,Camp2,MAXC);
      GetDlgItemText(HWindow,ID_C3,Camp3,MAXC);
      GetDlgItemText(HWindow,ID_C4,Camp4,MAXC);
      GetDlgItemText(HWindow,ID_C5,Camp5,MAXC);
      GetDlgItemText(HWindow,ID_C6,Camp6,MAXC);

      GetDlgItemText(HWindow,ID_C7,Camp7,MAXC);
      GetDlgItemText(HWindow,ID_C8,Camp8,MAXC);
      GetDlgItemText(HWindow,ID_C9,Camp9,MAXC);
      GetDlgItemText(HWindow,ID_C10,Camp10,MAXC);
      GetDlgItemText(HWindow,ID_C11,Camp11,MAXC);

      GetDlgItemText(HWindow,ID_C12,Camp12,MAXC);
      GetDlgItemText(HWindow,ID_C13,Camp13,MAXC);
      GetDlgItemText(HWindow,ID_C14,Camp14,MAXC);
      GetDlgItemText(HWindow,ID_C15,Camp15,MAXC);
      GetDlgItemText(HWindow,ID_C16,Camp16,MAXC);

}
//////////////////////

char* Miventana::Dialogo16(char *L1,char *L2,char *L3,char *L4,char *L5,
            char *L6,char *L7,char *L8,char *L9,char *L10,
                  char *L11,char *L12,char *L13,char *L14
                        ,char* L15,char* L16){
    if(GetModule()->ExecDialog(new Caja16(this,"DIALOG_03"))==IDOK)
    {
     Desactiva();

  strcpy(L1,Transfiere16.c1);
  strcpy(L2,Transfiere16.c2);
  strcpy(L3,Transfiere16.c3);
```

```
  strcpy(L4,Transfiere16.c4);
  strcpy(L5,Transfiere16.c5);
  strcpy(L6,Transfiere16.c6);
  strcpy(L7,Transfiere16.c7);
  strcpy(L8,Transfiere16.c8);
  strcpy(L9,Transfiere16.c9);
  strcpy(L10,Transfiere16.c10);
  strcpy(L11,Transfiere16.c11);
  strcpy(L12,Transfiere16.c12);
  strcpy(L13,Transfiere16.c13);
  strcpy(L14,Transfiere16.c14);
  strcpy(L15,Transfiere16.c15);
  strcpy(L16,Transfiere16.c16);

  Guarda(L12,L2,L3,L4,L5,L6,L7,L8,L9,L10,L11,L12,L13,L14,L15,L16);
  recalcula=3;
  int q=Calcula(ArchNom,3);
  if(q==0) EXITO
  if(q==-1) E_ARGUMENTO
  if(q==-2) E_ARCHIVOSALIDA
  if(q==-3) E_OPCION
  if(q==-4) E_ARCHIVOENTRADA
return "";

  }
}

/*


                        CAJA14.CPP


*/

Caja14::Caja14(PTWindowsObject Aparent,LPSTR name):TDialog(Aparent,name)
{
    new TEdit(this,ID_C1,
    sizeof(((Miventana *)Parent)->Transfiere14.c1)),//del tamaño de c1
    //hacemos un cast  del tipo ventana al tipo parent

    new TEdit(this,ID_C2,
          sizeof(((Miventana *)Parent)->Transfiere14.c2));

    new TEdit(this,ID_C3,
          sizeof(((Miventana *)Parent)->Transfiere14.c3));

    new TEdit(this,ID_C4,
          sizeof(((Miventana *)Parent)->Transfiere14.c4));
                //hacemos un cast  del tipo ventana al tipo parent
    new TEdit(this,ID_C5,
          sizeof(((Miventana *)Parent)->Transfiere14.c5));

    new TEdit(this,ID_C6,
```

```
        sizeof(((Miventana *)Parent)->Transfiere14.c6));

    new TEdit(this,ID_C7,
        sizeof(((Miventana *)Parent)->Transfiere14 c7)),

    new TEdit(this,ID_C8,
        sizeof(((Miventana *)Parent)->Transfiere14 c8));

    new TEdit(this,ID_C9,
        sizeof(((Miventana *)Parent)->Transfiere14 c9));

    new TEdit(this,ID_C10,
        sizeof(((Miventana *)Parent)->Transfiere14 c10));

    new TEdit(this,ID_C11,
        sizeof(((Miventana *)Parent)->Transfiere14.c11));

    new TEdit(this,ID_C12,
        sizeof(((Miventana *)Parent)->Transfiere14 c12)),
            //hacemos un cast  del tipo ventana al tipo parent
    new TEdit(this,ID_C13,
        sizeof(((Miventana *)Parent)->Transfiere14.c13));

    new TEdit(this,ID_C14,
        sizeof(((Miventana *)Parent)->Transfiere14 c14)),

        TransferBuffer=(void far*)&(((Miventana*)Parent)->Transfiere14);
            //apunta a Transfiere22

} //************ FIN DE CONSTRUCTOR ************

BOOL Caja14::CanClose(){
        FillBuffers();
return TRUE;
}

void Caja14::FillBuffers(){
    //Para Dejar Los Valores en el Campo
        GetDlgItemText(HWindow,ID_C1,Camp1,MAXC);
        GetDlgItemText(HWindow,ID_C2,Camp2,MAXC);
        GetDlgItemText(HWindow,ID_C3,Camp3,MAXC);
        GetDlgItemText(HWindow,ID_C4,Camp4,MAXC);
        GetDlgItemText(HWindow,ID_C5,Camp5,MAXC);
        GetDlgItemText(HWindow,ID_C6,Camp6,MAXC);

        GetDlgItemText(HWindow,ID_C7,Camp7,MAXC);
        GetDlgItemText(HWindow,ID_C8,Camp8,MAXC);
        GetDlgItemText(HWindow,ID_C9,Camp9,MAXC);
        GetDlgItemText(HWindow,ID_C10,Camp10,MAXC);
        GetDlgItemText(HWindow,ID_C11,Camp11,MAXC);

        GetDlgItemText(HWindow,ID_C12,Camp12,MAXC);
        GetDlgItemText(HWindow,ID_C13,Camp13,MAXC);
        GetDlgItemText(HWindow,ID_C14,Camp14,MAXC);
```

```
}
//////////////////////

char* Miventana::Dialogo14(char *L1,char *L2,char *L3,char *L4,char *L5,
                    char *L6,char *L7,char *L8,char *L9,char *L10,
                                        char *L11,char *L12,char *L13,char *L14){

          if(GetModule()->ExecDialog(new Caja14(this,"DIALOG_02"))==IDOK)
          {
           Desactiva();

  strcpy(L1,Transfiere14.c1);
  strcpy(L2,Transfiere14.c2);
  strcpy(L3,Transfiere14.c3);
  strcpy(L4,Transfiere14.c4);
  strcpy(L5,Transfiere14.c5);
  strcpy(L6,Transfiere14.c6);
  strcpy(L7,Transfiere14 c7);
  strcpy(L8,Transfiere14.c8);
  strcpy(L9,Transfiere14.c9);
  strcpy(L10,Transfiere14.c10);
  strcpy(L11,Transfiere14.c11);
  strcpy(L12,Transfiere14.c12);
  strcpy(L13,Transfiere14.c13);
  strcpy(L14,Transfiere14.c14);

  Guarda(L12,L2,L3,L4,L5,L6,L7,L8,L9,L10,L11,L12,L13,L14);
  recalcula=2;
  int q=Calcula(ArchNom,2);
  if(q==0) EXITO
  if(q==-1) E_ARGUMENTO
  if(q==-2) E_ARCHIVOSALIDA
  if(q==-3) E_OPCION
  if(q==-4) E_ARCHIVOENTRADA

return " ";

  }

}

/*

                CAJA12.CPP


*/
Caja12::Caja12(PTWindowsObject Aparent,LPSTR name):TDialog(Aparent,name)
{

    new TEdit(this,ID_C1,
          sizeof(((Miventana *)Parent)->Transfiere12.c1));//del tamaño de c1
    //hacemos un cast  del tipo ventana al tipo parent
```

```
    new TEdit(this,ID_C2,
          sizeof(((Miventana *)Parent)->Transfiere12.c2));

    new TEdit(this,ID_C3,
          sizeof(((Miventana *)Parent)->Transfiere12.c3));

    new TEdit(this,ID_C4,
          sizeof(((Miventana *)Parent)->Transfiere12.c4));
                //hacemos un cast  del tipo ventana al tipo parent
    new TEdit(this,ID_C5,
          sizeof(((Miventana *)Parent)->Transfiere12.c5));

    new TEdit(this,ID_C6,
          sizeof(((Miventana *)Parent)->Transfiere12.c6));

    new TEdit(this,ID_C7,
          sizeof(((Miventana *)Parent)->Transfiere12.c7));

    new TEdit(this,ID_C8,
          sizeof(((Miventana *)Parent)->Transfiere12.c8));

    new TEdit(this,ID_C9,
          sizeof(((Miventana *)Parent)->Transfiere12.c9));

    new TEdit(this,ID_C10,
          sizeof(((Miventana *)Parent)->Transfiere12.c10));

          TransferBuffer=(void far*)&(((Miventana*)Parent)->Transfiere12);


} //************ FIN DE CONSTRUCTOR ************

BOOL Caja12::CanClose(){
        FillBuffers();
return TRUE;
}

void Caja12  FillBuffers(){
     //Para Dejar Los Valores en el Campo
        GetDlgItemText(HWindow,ID_C1,Camp1,MAXC);
        GetDlgItemText(HWindow,ID_C2,Camp2,MAXC);
        GetDlgItemText(HWindow,ID_C3,Camp3,MAXC);
        GetDlgItemText(HWindow,ID_C4,Camp4,MAXC);
        GetDlgItemText(HWindow,ID_C5,Camp5,MAXC);
        GetDlgItemText(HWindow,ID_C6,Camp6,MAXC);
        GetDlgItemText(HWindow,ID_C7,Camp7,MAXC);
        GetDlgItemText(HWindow,ID_C8,Camp8,MAXC);
        GetDlgItemText(HWindow,ID_C9,Camp9,MAXC);
        GetDlgItemText(HWindow,ID_C10,Camp10,MAXC);

}
//////////////////////
```

```
char* Miventana.:Dialogo12(char *L1,char *L2,char *L3,
                char *L4,char *L5,char *L6,char *L7,char *L8,char *L9,
                        char *L10){

        if(GetModule()->ExecDialog(new Caja12(this,"DIALOG_01"))==IDOK)
       {

        Desactiva();

  strcpy(L1,Transfiere12.c1);
  strcpy(L2,Transfiere12.c2);
  strcpy(L3,Transfiere12.c3);
  strcpy(L4,Transfiere12.c4);
  strcpy(L5,Transfiere12.c5);
  strcpy(L6,Transfiere12.c6);
  strcpy(L7,Transfiere12.c7);
  strcpy(L8,Transfiere12.c8);
  strcpy(L9,Transfiere12.c9);
  strcpy(L10,Transfiere12.c10);
  Guarda(L1,L2,L3,L4,L5,L6,L7,L8,L9,L10);

  recalcula=1;
  int q=Calcula(ArchNom,1);
  if(q==0) EXITO
  if(q==-1) E_ARGUMENTO
  if(q==-2) E_ARCHIVOSALIDA
  if(q==-3) E_OPCION
  if(q==-4) E_ARCHIVOENTRADA

return " ";

  }
}
```

La principal importancia de éste archivo radica en que continiene la función principal, además de contener más definiciones de funciones miembro de la clases antes definidas.

```
/*
                        GENERAL.CPP


*/

/*
#include"general.h"

/*#include "caja24.cpp"
#include "caja22.cpp"
#include "caja12.cpp"
#include "caja14.cpp"
#include "caja16.cpp"
#include "caja18.cpp"*/
```

```
void Miventana::CMUHelpIndex( RTMessage )
{
        WinHelp( HWindow, "CPLT2.HLP", HELP_INDEX, 0L );
}

void Miventana::CMUHelpHelp( RTMessage )
{
        WinHelp( HWindow, "CPLT2.HLP", HELP_HELPONHELP, 0L );
}

Miventana::Miventana(PTWindowsObject AParent,LPSTR ATitle, LPSTR AFileName)
  :TFileWindow(AParent, ATitle, AFileName) {
      recalcula=1;
      x=0;y=0; AssignMenu("FileCommands");
                escudo= LoadBitmap( GetApplication()->hInstance, "cplt" );

                BITMAP_1=LoadBitmap( GetApplication()->hInstance, "BITMAP_1" );
                BITMAP_2=LoadBitmap( GetApplication()->hInstance, "BITMAP_2" );
                BITMAP_3=LoadBitmap( GetApplication()->hInstance, "BITMAP_3" );
              BITMAP_4=LoadBitmap( GetApplication()->hInstance, "BITMAP_4" );
              BITMAP_5=LoadBitmap( GetApplication()->hInstance, "BITMAP_5" );
                BITMAP_6=LoadBitmap( GetApplication()->hInstance, "BITMAP_6" );

                strcpy(Transfiere12.c1,"");
                strcpy(Transfiere12.c2,"");
                strcpy(Transfiere12.c3,"");
                strcpy(Transfiere12.c4,"");
                strcpy(Transfiere12.c5,"");
                strcpy(Transfiere12.c6,"");
                strcpy(Transfiere12.c7,"");
                strcpy(Transfiere12.c8,"");
                strcpy(Transfiere12.c9,"");
                strcpy(Transfiere12.c10,"");


                strcpy(Transfiere14.c1,"");
                strcpy(Transfiere14.c2,"");
                strcpy(Transfiere14.c3,"");
                strcpy(Transfiere14.c4,"");
                strcpy(Transfiere14.c5,"");
                strcpy(Transfiere14.c6,"");
                strcpy(Transfiere14.c7,"");
                strcpy(Transfiere14.c8,"");
                strcpy(Transfiere14.c9,"");
                strcpy(Transfiere14.c10,"");
                strcpy(Transfiere14.c11,"");
                strcpy(Transfiere14.c12,"");
                strcpy(Transfiere14.c13,"");
                strcpy(Transfiere14.c14,"");

                strcpy(Transfiere16.c1,"");
                strcpy(Transfiere16.c2,"");
                strcpy(Transfiere16.c3,"");
                strcpy(Transfiere16.c4,"");
```

```
strcpy(Transfiere16.c5,"");
strcpy(Transfiere16.c6,"");
strcpy(Transfiere16.c7,"");
strcpy(Transfiere16.c8,"");
strcpy(Transfiere16.c9,"");
strcpy(Transfiere16.c10,"");
strcpy(Transfiere16.c11,"");
strcpy(Transfiere16.c12,"");
strcpy(Transfiere16.c13,"");
strcpy(Transfiere16.c14,"");
strcpy(Transfiere16.c15,"");
strcpy(Transfiere16.c16,"");


strcpy(Transfiere18.c1,"");
strcpy(Transfiere18.c2,"");
strcpy(Transfiere18.c3,"");
strcpy(Transfiere18.c4,"");
strcpy(Transfiere18.c5,"");
strcpy(Transfiere18.c6,"");
strcpy(Transfiere18.c7,"");
strcpy(Transfiere18.c8,"");
strcpy(Transfiere18.c9,"");
strcpy(Transfiere18.c10,"");
strcpy(Transfiere18.c11,"");
strcpy(Transfiere18.c12,"");
strcpy(Transfiere18.c13,"");
strcpy(Transfiere18.c14,"");
strcpy(Transfiere18.c15,"");
strcpy(Transfiere18.c16,"");


strcpy(Transfiere22.c1,"");
strcpy(Transfiere22.c2,"");
strcpy(Transfiere22.c3,"");
strcpy(Transfiere22.c4,"");
strcpy(Transfiere22.c5,"");
strcpy(Transfiere22.c6,"");
strcpy(Transfiere22.c7,"");
strcpy(Transfiere22.c8,"");
strcpy(Transfiere22.c9,"");
strcpy(Transfiere22.c10,"");
strcpy(Transfiere22.c11,"");
strcpy(Transfiere22.c12,"");
strcpy(Transfiere22.c13,"");
strcpy(Transfiere22.c14,"");
strcpy(Transfiere22.c15,"");
strcpy(Transfiere22.c16,"");
strcpy(Transfiere22.c17,"");
strcpy(Transfiere22.c18,"");
strcpy(Transfiere22.c19,"");
strcpy(Transfiere22.c20,"");
strcpy(Transfiere22.c21,"");
strcpy(Transfiere22.c22,"");
```

```
                strcpy(Transfiere24.c1,"");
                strcpy(Transfiere24.c2,"");
                strcpy(Transfiere24.c3,"");
                strcpy(Transfiere24.c4,"");
                strcpy(Transfiere24.c5,"");
                strcpy(Transfiere24.c6,"");
                strcpy(Transfiere24.c7,"");
                strcpy(Transfiere24.c8,"");
                strcpy(Transfiere24.c9,"");
                strcpy(Transfiere24.c10,"");
                strcpy(Transfiere24.c11,"");
                strcpy(Transfiere24.c12,"");
                strcpy(Transfiere24.c13,"");
                strcpy(Transfiere24.c14,"");
                strcpy(Transfiere24.c15,"");
                strcpy(Transfiere24.c16,"");
                strcpy(Transfiere24.c17,"");
                strcpy(Transfiere24.c18,"");
                strcpy(Transfiere24.c19,"");
                strcpy(Transfiere24.c20,"");
                strcpy(Transfiere24.c21,"");
                strcpy(Transfiere24.c22,"");
                strcpy(Transfiere24.c23,"");
                strcpy(Transfiere24.c24,"");
            strcpy(Transfiere24.c25,"");
                strcpy(Transfiere24.c26,"");
                Imagen();

 };


void Miventana::Recalcula(){
        int q=Calcula(ArchNom,recalcula);
   if(q==0) EXITO
   if(q==-1) E_ARGUMENTO
   if(q==-2) E_ARCHIVOSALIDA
   if(q==-3) E_OPCION
   if(q==-4) E_ARCHIVOENTRADA
}

void Miventana::WMLButtonDown(RTMessage Msg){

}
void Miventana::WMRButtonDown(RTMessage Msg){
        InvalidateRect(HWindow,NULL,TRUE);
}

void Ventana :InitMainWindow(){

MainWindow=new Miventana(NULL,Name,"");
}
```

```
int PASCAL WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
 LPSTR lpCmdLine, int nCmdShow){
 Ventana Ventanita("LINEAS DE TRANSMISION", hInstance, hPrevInstance,
 lpCmdLine, nCmdShow);
 Ventanita.Run();

 return Ventanita.Status;
}
```

Este archivo contiene todo el código necesario para el análisis del archivo de entrada que genera el programa. También tiene parte del código de la generación del archivo de salida

```
/*
        IMPARCHI.CPP
*/

#include<iostream.h>
#include<fstream.h>
#include<stdlib.h>
#include<conio.h>
#include<string.h>
#include<stdio.h>
#include "imp.cpp"
class archivo{
  protected:
        char *arch;
  public:
   archivo(char *l="C:\\datos.txt"){
        arch=new char[strlen(l)+1];
        strcpy(arch,l);
   }
   ~archivo(){ delete []arch; }
    int entrada(char *);
    char* salida(double &,char);
};

int archivo::entrada(char *p){

        fstream s(arch, ios::in|ios::out);
        if(!s){
                cout << "No puedo abrir el archivo";
                return 1;
        }
        s.seekp(0,ios::end);
        while(*p)
          s.put(*p++);
        s.close();
        return 0;
}

char * archivo::salida(double &m,char l='.'){
```

```
  ifstream aut(arch);
  if(!aut){
     cerr<<"No puedo abrir el archivo";
     exit(0);
   }
     char buffer[100];
     aut.seekg(m,ios::beg);
     aut.getline(buffer,sizeof(buffer),1);
     m=aut.tellg();
     aut.close();
     return buffer;
}

Linea ArmaL(char *p){
    archivo RGCA(p);
    double m=0,Rc,f,prt,rRMG,Rg,rg,x1,y1,x2,y2,x3,y3;
    RGCA.salida(m);
    Rc=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    f=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    prt=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    rRMG=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    x1=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    y1=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    x2=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    y2=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    x3=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    y3=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    Linea L(Rc,f,prt,rRMG,x1,y1,x2,y2,x3,y3);
    L.Cuadratura();
    L.ImpSecPos();
    L.ImpSecCero();
    return L;
}

LineaGuarda ArmaLG(char *p){
    archivo RGCA(p);
    double m=0,Rc,f,prt,rRMG,Rg,rg,x1,y1,x2,y2,x3,y3,gx,gy;
    RGCA.salida(m);
    Rc=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    f=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    prt=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
```

```
    rRMG=strtod(RGCA salida(m,'\n'),NULL);
    RGCA salida(m);
    Rg=strtod(RGCA salida(m,'\n'),NULL);
    RGCA salida(m),
    rg=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m),
    x1=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA salida(m);
    y1=strtod(RGCA salida(m,'\n'),NULL);
    RGCA salida(m);
    x2=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA salida(m),
    y2=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA salida(m),
    x3=strtod(RGCA salida(m,'\n'),NULL);
    RGCA.salida(m);
    y3=strtod(RGCA salida(m,'\n'),NULL);
    RGCA salida(m),
    gx=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA salida(m),
    gy=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA salida(m);
    LineaGuarda L(Rc,f,prt,rRMG,Rg,rg,x1,y1,x2,y2,x3,y3,gx,gy),
    L.Cuadratura();
    L Cuadraturag();
    L.ImpSecPos(),
    L ImpSecCero();
    return L;
}

LineaGuarda2 ArmaL2G(char *p){
    archivo RGCA(p),
    double m=0,Rc,f,prt,rRMG,Rg,rg,x1,y1,x2,y2,x3,y3,gx,gy,gx1,gy1,
    RGCA salida(m);
    Rc=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA salida(m);
    f=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    prt=strtod(RGCA salida(m,'\n'),NULL);
    RGCA.salida(m);
    rRMG=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m),
    Rg=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    rg=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    x1=strtod(RGCA salida(m,'\n'),NULL),
    RGCA.salida(m);
    y1=strtod(RGCA salida(m,'\n'),NULL);
    RGCA salida(m),
    x2=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m),
    y2=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m),
```

```
    x3=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    y3=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    gx=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    gy=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    gx1=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    gy1=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    LineaGuarda2 L(Rc,f,prt,rRMG,Rg,rg,x1,y1,x2,y2,x3,y3,gx,gy,gx1,gy1);
    L.Cuadratura();
    L.Cuadraturag();
    L.Cuadraturag2();
    L.ImpSecPos();
    L.ImpSecCero();
    return L;
}

LineaParalela ArmaLP(char *p){
    archivo RGCA(p);
    double m=0,Rc,f,prt,rRMG,Rg,rg,x1,y1,x2,y2,x3,y3,x11,y11,x22,y22,x33,y33;
    RGCA.salida(m);
    Rc=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    f=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    prt=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    rRMG=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    x1=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    y1=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    x2=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    y2=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    x3=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    y3=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    x11=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    y11=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    x22=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    y22=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    x33=strtod(RGCA.salida(m,'\n'),NULL);
```

```
    RGCA.salida(m);
    y33=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    LineaParalela L(Rc,f,prt,rRMG,x1,y1,x2,y2,x3,y3,x11,
            y11,x22,y22,x33,y33);
    L.Cuadratura();
    L.Cuadratura12();
    L.ImpSecPos();
    L.ImpSecCero();
    return L;
}

LineaParalelaGuarda ArmaLPG(char *p){
    archivo RGCA(p);
    double m=0,Rc,f,prt,rRMG,Rg,rg,x1,y1,x2,y2,x3,y3,gx,gy,x11,y11,x22,y22,x33,y33,gx1,gy1;
    RGCA salida(m);
    Rc=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    f=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    prt=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    rRMG=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    Rg=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    rg=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    x1=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    y1=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    x2=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    y2=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    x3=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    y3=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    gx=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    gy=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    x11=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    y11=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    x22=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    y22=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    x33=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
```

```
    y33=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    gx1=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    gy1=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    LineaParalelaGuarda L(Rc,f,prt,rRMG,Rg,rg,x1,y1,x2,y2,x3,y3,gx,
                gy,x11,y11,x22,y22,x33,y33,gx1,gy1);
    L.Cuadratura();
    L.Cuadratura12();
    L.Cuadraturag();
    L.Cuadraturag2();
        L.Cuadratura22();
    L.ImpSecPos();
    L.ImpSecCero();
    return L;
}

LineaParalelaGuarda2 ArmaLP2G(char *p){
    archivo RGCA(p);
        double m=0,Rc,f,prt,rRMG,Rg,rg,x1,y1,x2,y2,x3,y3,gx,gy,x11,
                y11,x22,y22,x33,y33,gx1,gy1,gx2,gy2,gx3,gy3;
    RGCA.salida(m);
    Rc=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    f=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    prt=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    rRMG=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    Rg=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    rg=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    x1=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    y1=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    x2=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    y2=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    x3=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    y3=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    x11=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    y11=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    x22=strtod(RGCA.salida(m,'\n'),NULL);
    RGCA.salida(m);
    y22=strtod(RGCA.salida(m,'\n'),NULL);
```

```
    RGCA.salida(m);
    x33=strtod(RGCA salida(m,'\n'),NULL);
    RGCA.salida(m);
    y33=strtod(RGCA.salida(m,'\n'),NULL),
    RGCA.salida(m),
    gx=strtod(RGCA.salida(m,'\n'),NULL),
    RGCA.salida(m),
    gy=strtod(RGCA.salida(m,'\n'),NULL);
        RGCA.salida(m);
        gx1=strtod(RGCA salida(m,'\n'),NULL);
    RGCA salida(m);
        gy1=strtod(RGCA.salida(m,'\n'),NULL);
        RGCA.salida(m);
        gx2=strtod(RGCA.salida(m,'\n'),NULL),
    RGCA.salida(m),
        gy2=strtod(RGCA.salida(m,'\n'),NULL);
        RGCA salida(m),
        gx3=strtod(RGCA salida(m,'\n'),NULL);
        RGCA salida(m),
        gy3=strtod(RGCA salida(m,'\n'),NULL);
        RGCA salida(m);
    LineaParalelaGuarda2  L(Rc,f,prt,rRMG,Rg,rg,x1,y1,x2,y2,x3,y3,gx,
                gy,x11,y11,x22,y22,x33,y33,gx1,gy1,gx2,gy2,gx3,gy3),
    L.Cuadratura();
    L Cuadratura12();
    L Cuadraturag();
    L Cuadraturag2();
        L.Cuadratura22();
        L ImpSecPos();
    L ImpSecCero(),
    return L;
}

int Calcula(char *FileName,int opcion){
        char *s=new char[strlen(FileName)+1];
        //strcpy(s,"c:\\salida.lin");
        strncpy(s,FileName,strlen(FileName)-3);
        s[strlen(FileName)-3]='\0',
        strncat(s,"clt",3);
    fstream salida(s,ios::trunc|ios::out);
    if(!salida)
            return -2;
    int a=opcion;
    switch(a){
        case 1:
                Linea A=ArmaL(FileName);
                A Salida(salida),
            break;
        case 2:
                LineaGuarda B=ArmaLG(FileName);
            B.Salida(salida);
            break;
        case 3:
            LineaGuarda2 C=ArmaL2G(FileName);
```

```
            C Salida(salida);
            break;
        case 4:
            LineaParalela D=ArmaLP(FileName);
            D.Salida(salida);
                break;
        case 5:
            LineaParalelaGuarda E=ArmaLPG(FileName);
                E.Salida(salida);
                break;
        case 6:
                LineaParalelaGuarda2 F=ArmaLP2G(FileName);
                F Salida(salida);
                break;
        default:
                return -3;
        }
        salida.close();
        return 0;
}
```

Este archivo contiene todo el código que logra hacer el cálculo de las Lineas de Transmisión en sus diferentes topologías. Veasé la herencia que se usa conveniente a una aplicación como la propuesta

```
/*
            IMP.CPP
*/
#include<iostream.h>
#include<math.h>
#include<COMPLEX h>
#include<conio h>
#include<stdlib.h>
#include<dos.h>
#include<fstream.h>

class Linea{  // 1 LINEA SIN C.G  **
  protected:
        complex Z11,Z22,Z00;
        double DMG,f,/*ha,hb,hc,*/ro,Rc,De,rc;
        double xa,ya,xb,yb,xc,yc;
        double ab,ac,bc;

public
    Linea(double pRc,double pf,double pro,double prc,
          double pxa,double pya,double pxb,double pyb,double pxc,double pyc):
         Rc(pRc),f(pf),ro(pro),rc(prc),xa(pxa),ya(pya),xb(pxb),yb(pyb),
         xc(pxc),yc(pyc){}
 //   virtual void Datos();
    void ImpSecPos();
    virtual void ImpSecCero();
    void Salida(fstream &);
    void Cuadratura();
```

```
    double modulo(complex Z);
};

double Linea  modulo(complex Z){
          return sqrt(norm(Z));

}
class LineaGuarda virtual public Linea{  // 1 LINEA CON 1 C.G **
 protected:
    double Rg,rg,xg,yg;
    double ag,bg,cg;
  public
    LineaGuarda(double pRc,double pf,double pro,double prc,
          double pRg, double prg, double px1, double py1, double px2,
          double py2, double px3, double py3, double pxg, double pyg)
          Linea(pRc,pf,pro,prc,px1,py1,px2,py2,px3,py3),Rg(pRg),rg(prg),
          xg(pxg),yg(pyg){}
   // void Datos(int=0);
    void ImpSecCero();
    void Cuadraturag();
};

class LineaGuarda2:public LineaGuarda{ // 1 LINEA CON 2 C G
   protected:
    double xg2,yg2,dgg2;   // Coordenadas de guarda 2
    double ag2,bg2,cg2;
  public:
    LineaGuarda2(double pRc,double pf,double pro,double prc,
          double pRg, double prg, double px1, double py1, double px2,
          double py2, double px3, double py3, double pxg, double pyg,
          double pxg2, double pyg2):
          LineaGuarda(pRc,pf,pro,prc,pRg,prg,px1,py1,px2,py2,px3,py3,pxg,pyg),
          Linea(pRc,pf,pro,prc,px1,py1,px2,py2,px3,py3),xg2(pxg2),yg2(pyg2){}
    void ImpSecCero();
    void Cuadraturag2();

};

class LineaParalela:virtual public Linea{ // 2 CIRCUITOS , SIN C G
  protected:
          double xa2,xb2,xc2,ya2,yb2,yc2;
          double aa2,bb2,cc2,ab2,ac2,ba2,bc2,ca2,cb2;
  public:
 //       void Datos();
          LineaParalela(double pRc,double pf,double pro,double prc,
            double pxa,double pya,double pxb,double pyb,double pxc,double pyc,
            double pxa2,double pya2,double pxb2,double pyb2,double pxc2,double pyc2):
          Linea(pRc,pf,pro,prc,pxa,pya,pxb,pyb,pxc,pyc),xa2(pxa2),xb2(pxb2),
          xc2(pxc2),ya2(pya2),yb2(pyb2),yc2(pyc2){}
          void ImpSecCero();
          void Cuadratura12();
};

class LineaParalelaGuarda:public LineaParalela,public LineaGuarda2 {// 2 CIRCUITOS CON 2 C G
```

```
  protected:
        double a2g,a2g2,b2g,b2g2,c2g,c2g2,a2b2,a2c2,b2c2,
  public:
        LineaParalelaGuarda(double pRc,double pf,double pro,double prc,
        double pRg, double prg, double px1, double py1, double px2,
        double py2, double px3, double py3, double pxg, double pyg,
        double pxa2,double pya2,double pxb2,double pyb2,double pxc2,double pyc2,
        double pxg2,double pyg2):
        LineaGuarda2(pRc,pf,pro,prc,pRg,prg,px1,py1,px2,py2,px3,py3,pxg,pyg,
        pxg2,pyg2),
        LineaParalela(pRc,pf,pro,prc,px1,py1,px2,py2,px3,py3,pxa2,
           pya2,pxb2,pyb2,pxc2,pyc2),
        /*      LineaGuarda(pRc,pf,pro,prc,pRg,prg,px1,py1,px2,py2,px3,py3,pxg,pyg),*/
        Linea(pRc,pf,pro,prc,px1,py1,px2,py2,px3,py3){}
    void ImpSecCero();
    void Cuadratura22();
};

void LineaParalelaGuarda::ImpSecCero(){

   De=658*sqrt(ro/f);
        complex Z0l,Z0g,Z0lg;

   double RMGl,RMGg,DMGlg;
   double
temp=pow(rc,6)*pow(ab,2)*pow(ac,2)*pow(aa2,2)*pow(ab2,2)*pow(ac2,2)*pow(bc2,2)*pow(ba2,2)*pow(b
b2,2)*pow(bc2,2)*pow(ca2,2)*pow(cb2,2)*pow(cc2,2)*pow(a2b2,2)*pow(a2c2,2)*pow(b2c2,2);
   RMGg=sqrt(rg*dgg2),
        RMGl=pow(temp,0.02777777777777778);
   DMGlg=pow(ag*ag2*bg*bg2*cg*cg2*a2g*a2g2*b2g*b2g2*c2g*c2g2,.0833333333333333),
        Z0l=complex(Rc/2+0 002964*f,0.008676*f*log10(De/RMGl)),
  // cout<<"No revento en Z0l ";
        Z0g=complex(1 5*Rg+0 002964*f,0.008676*f*log10(De/RMGg));
  // cout<<"No revento en Z0g" ;
        Z0lg=complex(0.002964*f,0.008676*f*log10(De/DMGlg));
  // cout<<"No revento en Z0lg ";

   Z00=Z0l-pow(Z0lg,2)/Z0g;
}

void LineaParalelaGuarda::Cuadratura22(){

   a2g=sqrt(pow(fabs(xa2-xg),2)+pow(fabs(ya2-yg),2));

   a2g2=sqrt(pow(fabs(xa2-xg2),2)+pow(fabs(ya2-yg2),2)) ;

   b2g=sqrt(pow(fabs(xb2-xg),2)+pow(fabs(yb2-yg),2)),

   b2g2=sqrt(pow(fabs(xb2-xg2),2)+pow(fabs(yb2-yg2),2));

   c2g=sqrt(pow(fabs(xc2-xg),2)+pow(fabs(yc2-yg),2)) ;

   c2g2=sqrt(pow(fabs(xc2-xg2),2)+pow(fabs(yc2-yg2),2));
```

```
    a2b2=sqrt(pow(fabs(xa2-xb2),2)+pow(fabs(ya2-yb2),2));

    a2c2=sqrt(pow(fabs(xa2-xc2),2)+pow(fabs(ya2-yc2),2)) ;

        b2c2=sqrt(pow(fabs(xb2-xc2),2)+pow(fabs(yb2-yc2),2)).

}

void LineaParalela::ImpSecCero(){
    complex Z0m,Z0p;

    double Inc,hl,Rc,Im,DMG,DMG12,


  //  Inc=1.015*f*((ha+hb+hc)/sqrt(ro/f))/1000000,
    Inc=1 015*f*((ya+yb+yc)/sqrt(ro/f))/1000000.
    Re=Rc+0.00294*f-Inc,
    De=658*sqrt(ro/f),

    DMG=pow(ab*ac*bc, 33333333333),
    DMG12=pow(aa2*bb2*cc2*ab2*ac2*ba2*bc2*ca2*cb2,.1111111111);

    Im=0.0086*f*log10(De/pow(pow(DMG,2)*rc,.33333333))+Inc;
    Z0p=complex(Re,Im),

    hl=(ya+yb+yc)/3;
    Inc=1.015*f*3*hl/(sqrt(ro/f)*1000000);
    Re=0.002964*f-Inc;
    Im=0 008676*f*log10(De/DMG12)+Inc.

    Z0m=complex(Re,Im),
    Z00=(Z0p+Z0m)/2;
}

void LineaParalela::Cuadratura12(){

    aa2=sqrt(pow(fabs(xa-xa2),2)+pow(fabs(ya-ya2),2));

    bb2=sqrt(pow(fabs(xb-xb2),2)+pow(fabs(yb-yb2),2)) ;

    cc2=sqrt(pow(fabs(xc-xc2),2)+pow(fabs(yc-yc2),2));

    ab2=sqrt(pow(fabs(xa-xb2),2)+pow(fabs(ya-yb2),2));

    ac2=sqrt(pow(fabs(xb-xc2),2)+pow(fabs(yb-yc2),2)) .

    ba2=sqrt(pow(fabs(xc-xa2),2)+pow(fabs(yc-ya2),2));

    bc2=sqrt(pow(fabs(xa-xc2),2)+pow(fabs(ya-yc2),2));

    ca2=sqrt(pow(fabs(xb-xa2),2)+pow(fabs(yb-ya2),2)) ;

    cb2=sqrt(pow(fabs(xc-xb2),2)+pow(fabs(yc-yb2),2));
```

```
                                }
void LineaGuarda2::ImpSecCero(){  //CALCULOS 1 CTO. 1 C.G

  double RMGl,DMGabc,RMGg,DMGlg;
  complex Z0l,Z0g,Z0lg;

    DMGabc=pow(ab*ac*bc,.333333333333);
    RMGl=pow(rc*pow(DMGabc,2),.33333333333);

    RMGg=sqrt(rg*dgg2);

    DMGlg=pow(ag*ag2*bg*bg2*cg*cg2,.16666666666666667);

    De=658*sqrt(ro/f);

    Z0l=complex(Rc+0.002964*f,0.008676*f*log10(De/RMGl));
    Z0g=complex(3*Rg/2+0.002964*f,0.008676*f*log10(De/RMGg));
    Z0lg=complex(0.002964*f,0.008676*f*log10(De/DMGlg));
    Z00=Z0l-pow(Z0lg,2)/Z0g;

}
void LineaGuarda2::Cuadraturag2(){

    ag2=sqrt(pow(fabs(xa-xg2),2)+pow(fabs(ya-yg2),2));

    bg2=sqrt(pow(fabs(xb-xg2),2)+pow(fabs(yb-yg2),2)) ;

    cg2=sqrt(pow(fabs(xc-xg2),2)+pow(fabs(yc-yg2),2));

    dgg2=sqrt(pow(fabs(xg-xg2),2)+pow(fabs(yg-yg2),2));
}


void LineaGuarda::ImpSecCero(){

  double RMGl,DMGlg,Im,Re,De;
    complex Z0l,Z0g,Z0lg;

    DMG=pow(ab*ac*bc,.33333333333);

    RMGl=pow(rc*pow(DMG,2),.3333333333);

    DMGlg=pow(ag*bg*cg,.33333333333);

    De=658*sqrt(ro/f);

    Rc=Rc+0.002964*f;

    Im=0.008676*f*log10(De/RMGl);

    Z0l=complex(Re,Im);

    Re=3*Rg+0.002964*f;
```

```
    Im=.008676*f*log10(De/rg);

    Z0g=complex(Re,Im);

    Rc=0.002964*f;

    Im=0.008676*f*log10(De/DMGlg);

    Z0lg=complex(Re,Im);

    Z00=Z0l-pow(Z0lg,2)/Z0g;
}

void LineaGuarda: Cuadraturag(){

    ag=sqrt(pow(fabs(xa-xg),2)+pow(fabs(ya-yg),2));

    bg=sqrt(pow(fabs(xb-xg),2)+pow(fabs(yb-yg),2)) ;

    cg=sqrt(pow(fabs(xc-xg),2)+pow(fabs(yc-yg),2));
}


void Linea::Cuadratura(){
        ab=sqrt(pow(fabs(xa-xb),2)+pow(fabs(ya-yb),2));

        ac=sqrt(pow(fabs(xa-xc),2)+pow(fabs(ya-yc),2));

        bc=sqrt(pow(fabs(xb-xc),2)+pow(fabs(yb-yc),2));

}
void Linea::ImpSecCero(){

   double Re,Im,Inc;

    //  Inc=1.015*f*((ha+hb+hc)/sqrt(ro/f))/1000000;
        Inc=1.015*f*((ya+yb+yc)/sqrt(ro/f))/1000000;
        Re=Rc+0.00294*f-Inc;
        De=658*sqrt(ro/f);
        DMG=pow(ab*ac*bc,.33333333333);
        Im=0.0086*f*log10(De/pow(pow(DMG,2)*rc,.33333333))+Inc;
        Z00=complex(Re,Im);
}
void Linea.:ImpSecPos(){
        double Im;
        Im=0.002892*f*log10(pow(ab*ac*bc,.33333)/rc);
        Z11=Z22=complex(Rc,Im);
}
void Linea::Salida(fstream &salida){

        salida<<"Impedancia de Secuencia positiva y negativa\n\n ";
        salida<<Z11<<"\tOhms/km\t\t";
        salida<<modulo(Z11)<<" |_"<<arg(Z11)<<"o Ohms/km\n";
        salida<<Z11/.621371192237<<"\tOhms/mi\t\t";
```

```
        salida<<modulo(Z11)/0.621371192237<<" |_"<<arg(Z11)<<"o Ohms/km\n\n\n\n";
        salida<<"Impedancia de Secuencia Cero\n\n";
        salida<<Z00<<"\tOhms/km\t\t";
        salida<<modulo(Z00)<<" |_"<<arg(Z00)<<"o Ohms/km\n";
        salida<<Z00/.621371192237<<"\tOhms/mi\t\t";
        salida<<modulo(Z00)/0.621371192237<<" |_"<<arg(Z00)<<"o Ohms/mi\n";

}

//*************** Aqui empieza el SHOW ************************

class LineaParalelaGuarda2:public LineaParalelaGuarda{// 4 CIRCUITOS CON 2 C.G
        double ag3,bg3,cg3,a2g3,b2g3,c2g3,gg3,g2g3,
                        ag4,bg4,cg4,a2g4,b2g4,c2g4,gg4,g2g4,g3g4,xg3,yg3,xg4,yg4,
  public:
        LineaParalelaGuarda2(double pRc,double pf,double pro,double prc,
        double pRg, double prg, double px1, double py1, double px2,
        double py2, double px3, double py3, double pxg, double pyg,
        double pxa2,double pya2,double pxb2,double pyb2,double pxc2,double pyc2,
        double pxg2,double pyg2,double pxg3,double pyg3,double pxg4,double pyg4)
        LineaParalelaGuarda(pRc,pf,pro,prc,pRg,prg,px1,py1,px2,py2,px3,py3,pxg,pyg,pxa2,
        pya2,pxb2,pyb2,pxc2,pyc2,pxg2,pyg2),
        xg3(pxg3),yg3(pyg3),xg4(pxg4),yg4(pyg4),Linea(pRc,pf,pro,prc,px1,py1,px2,py2,px3,py3){}
    void ImpSecCero();
    void Cuadratura22();
};

void LineaParalelaGuarda2: ImpSecCero(){

    De=658*sqrt(ro/f),
        complex Z0l,Z0g,Z0lg;

    double RMGl,RMGg,DMGlg,
        double
temp=pow(rc,6)*pow(ab,2)*pow(ac,2)*pow(aa2,2)*pow(ab2,2)*pow(ac2,2)*pow(bc2,2)*pow(ba2,2)*pow(b
b2,2)*pow(bc2,2)*pow(ca2,2)*pow(cb2,2)*pow(cc2,2)*pow(a2b2,2)*pow(a2c2,2)*pow(b2c2,2);

        RMGl=pow(temp,0.02777777777777778);

        double
tempg=pow(rg,4)*pow(dgg2,2)*pow(gg3,2)*pow(gg4,2)*pow(g2g3,2)*pow(g2g4,2)*pow(g3g4,2);

        RMGg=pow(tempg,0.0625),

        DMGlg=pow(ag*ag2*bg*bg2*cg*cg2*a2g*a2g2*b2g*b2g2*c2g*c2g2*ag3*
                        ag4*bg3*bg4*cg3*cg4*a2g3*a2g4*b2g3*b2g4*c2g3*c2g4,0 041666666667),

        Z0l=complex(Rc/2+0 002964*f,0 008676*f*log10(De/RMGl));


        Z0g=complex( 75*Rg+0 002964*f,0 008676*f*log10(De/RMGg));
```

```
        Z0lg=complex(0.002964*f,0.008676*f*log10(De/DMGlg));

    Z00=Z0l-pow(Z0lg,2)/Z0g;
}

void LineaParalelaGuarda2: Cuadratura22(){

        LineaParalelaGuarda.:Cuadratura22().
        ag3=sqrt(pow(fabs(xa-xg3),2)+pow(fabs(ya-yg3),2));
        bg3=sqrt(pow(fabs(xb-xg3),2)+pow(fabs(yb-yg3),2));
        cg3=sqrt(pow(fabs(xc-xg3),2)+pow(fabs(yc-yg3),2));

        a2g3=sqrt(pow(fabs(xa2-xg3),2)+pow(fabs(ya2-yg3),2));
        b2g3=sqrt(pow(fabs(xb2-xg3),2)+pow(fabs(yb2-yg3),2));
        c2g3=sqrt(pow(fabs(xc2-xg3),2)+pow(fabs(yc2-yg3),2));

        gg3=sqrt(pow(fabs(xg-xg3),2)+pow(fabs(yg-yg3),2));
        g2g3=sqrt(pow(fabs(xg2-xg3),2)+pow(fabs(yg2-yg3),2));


        ag4=sqrt(pow(fabs(xa-xg4),2)+pow(fabs(ya-yg4),2));
        bg4=sqrt(pow(fabs(xb-xg4),2)+pow(fabs(yb-yg4),2));
        cg4=sqrt(pow(fabs(xc-xg4),2)+pow(fabs(yc-yg4),2));

        a2g4=sqrt(pow(fabs(xa2-xg4),2)+pow(fabs(ya2-yg4),2));
        b2g4=sqrt(pow(fabs(xb2-xg4),2)+pow(fabs(yb2-yg4),2));
        c2g4=sqrt(pow(fabs(xc2-xg4),2)+pow(fabs(yc2-yg4),2));

        gg4=sqrt(pow(fabs(xg-xg4),2)+pow(fabs(yg-yg4),2));
        g2g4=sqrt(pow(fabs(xg2-xg4),2)+pow(fabs(yg2-yg4),2));

        g3g4=sqrt(pow(fabs(xg3-xg4),2)+pow(fabs(yg3-yg4),2));

}
```

Este archivo contiene toda la parte gráfica, necesaria para la creación del menú de cortina, las cajas de diálogo, los Bitmap, etc.

```
/*

                GENERAL2 RC
*/
#include"apoyo rc"

cplt   BITMAP cplt bmp

BITMAP_1 BITMAP "  /dibujosl/dibujo1.bmp"
BITMAP_4 BITMAP "  ./dibujosl/dibujo4.bmp"
BITMAP_5 BITMAP "../dibujosl/dibujo5.bmp"
BITMAP_6 BITMAP "../dibujosl/dibujo6.bmp"
BITMAP_3 BITMAP "  /dibujosl/dibujo3.bmp"
```

```
BITMAP_2 BITMAP ". /dibujosl/dibujo2 bmp"

DIALOG_01 DIALOG 13. 32, 261. 127
STYLE DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSMENU
CLASS "BorDlg"
CAPTION "Linea sin cables de Guarda"
FONT 8, "Arial Rounded MT Bold"
BEGIN
        EDITTEXT ID_C1. 74, 6, 40. 11. ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C2. 74. 28, 40. 11. ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER '
WS_TABSTOP
        EDITTEXT ID_C3. 74. 49. 40. 11. ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C4. 74. 71, 40, 11, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C5. 142, 18. 40, 12. ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C6. 213, 18. 40, 11. ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER '
WS_TABSTOP
        EDITTEXT ID_C7. 143, 41, 40, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C8, 215, 39, 40, 12. ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER '
WS_TABSTOP
        EDITTEXT ID_C9. 143. 66, 40, 12. ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C10. 215. 67. 40. 12. ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        LTEXT "R_conductor=". -1. 19, 9. 46. 8. WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "Frecuencia =", -1, 18, 29, 47. 11. WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "p_res_tierra =". -1. 17. 50, 49, 12. WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "r_ RMG_cond=". -1. 15, 73. 55. 11. WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "x1 =". -1. 123. 19. 16. 9. WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "y1 =". -1. 192. 18. 17. 12, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "x2 =". -1. 123. 42. 15. 11, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "y2 =". -1. 193. 40. 18. 10. WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "x3 =". -1. 122. 66. 17. 11. WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "y3 =". -1. 193. 66. 15. 10. WS_CHILD | WS_VISIBLE | WS_GROUP
        CONTROL "". IDOK. BUTTON_CLASS. BS_DEFPUSHBUTTON | WS_CHILD | WS_VISIBLE |
WS_TABSTOP. 44. 98. 37. 24
        CONTROL "". IDCANCEL. BUTTON_CLASS. BS_PUSHBUTTON | WS_CHILD | WS_VISIBLE
| WS_TABSTOP. 173. 96. 36. 24
END

DIALOG_02 DIALOG 3. 22. 280. 161
STYLE DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSMENU
CLASS "BorDlg"
CAPTION "Linea con 1 guarda"
FONT 8. "Arial Rounded MT Bold"
BEGIN
        EDITTEXT ID_C1. 76. 5. 40. 11. ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C2. 76. 22. 40. 11. ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
```

```
        EDITTEXT ID_C3, 76, 38, 40, 11, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C4, 76, 55, 40, 11, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C5, 76, 73, 40, 11, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C6, 76, 90, 40, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C7, 150, 17, 40, 11, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C8, 233, 15, 40, 11, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C9, 150, 48, 40, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C10, 231, 47, 40, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C11, 151, 77, 40, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C12, 230, 75, 40, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C13, 150, 106, 40, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C14, 229, 106, 40, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        LTEXT "R_conductor=", -1, 19, 6, 46, 8, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "Frecuencia =", -1, 22, 22, 45, 11, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "p_res_tierra =", -1, 21, 39, 49, 12, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "r_ RMG_cond=", -1, 16, 56, 55, 11, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "Rg_guarda =", -1, 26, 73, 44, 11, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "rg_RMG_guarda=", -1, 15, 89, 55, 11, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "x1 =", -1, 128, 18, 16, 9, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "y1 =", -1, 202, 15, 17, 12, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "x2 =", -1, 132, 49, 15, 11, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "y2 =", -1, 203, 47, 18, 10, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "x3 =", -1, 130, 77, 17, 11, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "y3 =", -1, 206, 76, 15, 10, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "gx =", -1, 131, 108, 17, 10, WS_CHILD | WS_VISIBLE | WS_GROUP
        CONTROL "gy =", -1, "STATIC", SS_LEFT | WS_CHILD | WS_VISIBLE | WS_GROUP |
WS_TABSTOP, 206, 107, 17, 10
        CONTROL "", IDOK, BUTTON_CLASS, BS_DEFPUSHBUTTON | WS_CHILD | WS_VISIBLE |
WS_TABSTOP, 55, 131, 37, 24
        CONTROL "", IDCANCEL, BUTTON_CLASS, BS_PUSHBUTTON | WS_CHILD | WS_VISIBLE
| WS_TABSTOP, 177, 130, 36, 24
END


DIALOG_03 DIALOG 13, 18, 275, 182
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CLASS "BorDlg"
CAPTION "Linea con 2 guardas"
FONT 8, "Arial Rounded MT Bold"
BEGIN
        EDITTEXT ID_C1, 78, 4, 40, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
```

```
        EDITTEXT ID_C2, 78, 21, 40, 11, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C3, 78, 39, 40, 11, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C4, 78, 58, 40, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C5, 78, 78, 40, 11, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C6, 78, 98, 40, 13, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C7, 151, 14, 40, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C8, 224, 13, 40, 11, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C9, 152, 38, 40, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C10, 222, 37, 40, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C11, 152, 62, 40, 13, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C12, 223, 61, 40, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C13, 152, 90, 40, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C14, 222, 92, 40, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C15, 152, 112, 40, 14, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C16, 223, 114, 40, 11, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        LTEXT "R_conductor=", -1, 23, 6, 46, 8, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "Frecuencia =", -1, 25, 21, 45, 11, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "p_res_tierra =", -1, 23, 39, 49, 12, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "r_ RMG_cond=", -1, 20, 57, 55, 11, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "Rg_guarda =", -1, 29, 77, 44, 11, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "rg_RMG_guarda=", -1, 17, 97, 55, 11, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "x1 =", -1, 132, 14, 19, 11, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "y1 =", -1, 203, 14, 19, 11, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "x2 =", -1, 130, 37, 17, 10, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "y2 =", -1, 201, 38, 18, 10, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "x3 =", -1, 130, 63, 17, 11, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "y3 =", -1, 203, 62, 15, 10, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "gx1 =", -1, 129, 115, 20, 9, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "gy1 =", -1, 200, 114, 20, 9, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "gx =", -1, 129, 91, 20, 8, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "gy =", -1, 201, 91, 20, 9, WS_CHILD | WS_VISIBLE | WS_GROUP
        CONTROL "", IDOK, BUTTON_CLASS, BS_DEFPUSHBUTTON | WS_CHILD | WS_VISIBLE |
WS_TABSTOP, 44, 144, 37, 24
        CONTROL "", IDCANCEL, BUTTON_CLASS, BS_PUSHBUTTON | WS_CHILD | WS_VISIBLE
| WS_TABSTOP, 166, 143, 36, 24
END

DIALOG_04 DIALOG 4, 25, 285, 178
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
```

```
CLASS "BorDlg"
CAPTION "Lineas Paralelas sin cables de Guarda"
FONT 8, "Arial Rounded MT Bold"
BEGIN
        EDITTEXT ID_C1, 80, 4, 40, 11, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C2, 80, 22, 40, 11, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C3, 80, 41, 40, 11, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C4, 80, 61, 40, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C5, 154, 10, 40, 11, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C6, 231, 9, 40, 13, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C7, 154, 31, 40, 11, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C8, 232, 31, 40, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C9, 155, 54, 40, 11, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C10, 233, 52, 40, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C11, 155, 78, 40, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C12, 234, 77, 40, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C13, 158, 102, 40, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C14, 234, 101, 40, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C15, 155, 127, 40, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C16, 233, 127, 40, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        LTEXT "R_conductor=", -1, 25, 6, 46, 8, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "Frecuencia =", -1, 27, 23, 45, 11, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "p_res_tierra =", -1, 25, 42, 49, 12, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "r_ RMG_cond=", -1, 21, 62, 55, 11, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "x1 =", -1, 131, 10, 19, 11, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "y1 =", -1, 209, 9, 19, 14, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "x2 =", -1, 133, 31, 20, 11, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "y2 =", -1, 206, 32, 18, 10, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "x3 =", -1, 132, 55, 17, 11, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "y3 =", -1, 207, 53, 15, 10, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "x11 =", -1, 133, 82, 20, 9, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "y11 =", -1, 205, 77, 21, 10, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "x22 =", -1, 129, 103, 20, 8, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "y22 =", -1, 205, 102, 18, 8, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "x33 =", -1, 132, 128, 19, 9, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "y33 =", -1, 205, 127, 18, 10, WS_CHILD | WS_VISIBLE | WS_GROUP
        CONTROL "", IDOK, BUTTON_CLASS, BS_DEFPUSHBUTTON | WS_CHILD | WS_VISIBLE |
WS_TABSTOP, 49, 149, 37, 24
```

```
        CONTROL "", IDCANCEL, BUTTON_CLASS, BS_PUSHBUTTON | WS_CHILD | WS_VISIBLE
| WS_TABSTOP, 170, 147, 36, 24
END


DIALOG_06 DIALOG 26, 10, 274, 211
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CLASS "BorDlg"
CAPTION "Lineas Paraletas con 2 guardas"
FONT 8, "Arial Rounded MT Bold"
BEGIN
        EDITTEXT ID_C1, 68, 11, 40, 11, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C2, 68, 33, 40, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C3, 68, 53, 40, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C4, 68, 75, 40, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C5, 68, 97, 40, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C6, 68, 118, 40, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C7, 145, 9, 40, 11, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C8, 221, 9, 40, 11, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C9, 144, 27, 40, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C10, 222, 27, 40, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C11, 145, 47, 40, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C12, 222, 46, 40, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C13, 144, 67  40, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C14, 223, 67, 40, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C15, 144, 86, 40, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C16, 223, 85, 40, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C17, 142, 109, 40, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C18, 223, 109, 40, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C19, 140, 131, 40, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C20, 224, 130, 40, 13, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C21, 140, 153, 40, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
```

```
        EDITTEXT ID_C22, 225, 155, 40, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        LTEXT "R_conductor=", -1, 13, 11, 46, 8, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "Frecuencia =", -1, 13, 34, 45, 11, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "p_res_tierra =", -1, 12, 54, 49, 12, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "r_ RMG_cond=", -1, 9, 75, 55, 11, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "Rg_guarda =", -1, 11, 97, 44, 11, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "rg_RMG_guarda=", -1, 7, 118, 55, 11, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "x1 =", -1, 121, 9, 19, 11, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "y1 =", -1, 202, 9, 19, 14, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "x2 =", -1, 124, 29, 20, 11, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "y2 =", -1, 199, 27, 18, 10, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "x3 =", -1, 123, 48, 17, 11, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "y3 =", -1, 200, 47, 15, 10, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "gx =", -1, 124, 68, 17, 10, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "gy =", -1, 198, 66, 20, 11, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "x11 =", -1, 121, 88, 20, 9, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "y11 =", -1, 198, 85, 21, 10, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "x22 =", -1, 119, 111, 20, 8, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "y22 =", -1, 199, 109, 18, 8, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "x33 =", -1, 118, 134, 19, 9, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "y33 =", -1, 198, 131, 18, 10, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "gx1 =", -1, 117, 154, 20, 9, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "gy1 =", -1, 199, 155, 20, 9, WS_CHILD | WS_VISIBLE | WS_GROUP
        CONTROL "", IDOK, BUTTON_CLASS, BS_DEFPUSHBUTTON | WS_CHILD | WS_VISIBLE |
WS_TABSTOP, 50, 176, 37, 24
        CONTROL "", IDCANCEL, BUTTON_CLASS, BS_PUSHBUTTON | WS_CHILD | WS_VISIBLE
| WS_TABSTOP, 151, 175, 36, 24
END

DIALOG_07 DIALOG 26, 10, 274, 261
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CLASS "BorDlg"
CAPTION "Lineas Paralelas con 2 guardas cada una"
FONT 8, "Arial Rounded MT Bold"
BEGIN
        EDITTEXT ID_C1, 68, 11, 40, 11, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C2, 68, 33, 40, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C3, 68, 53, 40, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C4, 68, 75, 40, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C5, 68, 97, 40, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C6, 68, 118, 40, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C7, 145, 9, 40, 11, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C8, 221, 9, 40, 11, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C9, 144, 27, 40, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
```

```
        EDITTEXT ID_C10, 222, 27, 40, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C11, 145, 47, 40, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C12, 222, 46, 40, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C15, 146, 76, 40, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C16, 225, 77, 40, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C17, 147, 98, 40, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C18, 222, 96, 40, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C19, 145, 118, 40, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C20, 226, 119, 40, 13, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C13, 142, 145, 40, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C14, 223, 147, 40, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C21, 142, 163, 40, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C22, 224, 164, 40, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C23, 142, 185, 40, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C24, 222, 183, 40, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C25, 139, 204, 41, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        EDITTEXT ID_C26, 221, 201, 41, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP
        LTEXT "R_conductor=", -1, 13, 11, 46, 8, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "Frecuencia =", -1, 13, 34, 45, 11, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "p_res_tierra =", -1, 12, 54, 49, 12, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "r_ RMG_cond=", -1, 9, 75, 55, 11, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "Rg_guarda =", -1, 11, 97, 44, 11, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "rg_RMG_guarda=", -1, 7, 118, 55, 11, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "x1 =", -1, 121, 9, 19, 11, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "y1 =", -1, 202, 9, 19, 14, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "x2 =", -1, 124, 29, 16, 11, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "y2 =", -1, 199, 27, 18, 10, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "x3 =", -1, 123, 48, 17, 11, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "y3 =", -1, 200, 47, 15, 10, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "gx =", -1, 118, 146, 17, 10, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "gy =", -1, 199, 148, 20, 11, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "x11 =", -1, 121, 80, 20, 9, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "y11 =", -1, 197, 77, 21, 10, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "x22 =", -1, 119, 98, 20, 8, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "y22 =", -1, 198, 96, 18, 8, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "x33 =", -1, 119, 117, 19, 9, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "y33 =", -1, 197, 118, 18, 10, WS_CHILD | WS_VISIBLE | WS_GROUP
```

```
        LTEXT "gx1 =", -1, 114, 165, 20, 9, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "gy1 =", -1, 198, 164, 20, 9, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "gx2 =", -1, 113, 185, 20, 9, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "gy2 =", -1, 196, 184, 20, 9, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "gx3 =", -1, 113, 205, 20, 9, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "gy3 =", -1, 194, 202, 20, 9, WS_CHILD | WS_VISIBLE | WS_GROUP
        CONTROL "", IDOK, BUTTON_CLASS, BS_DEFPUSHBUTTON | WS_CHILD | WS_VISIBLE |
WS_TABSTOP, 52, 227, 37, 24
        CONTROL "", IDCANCEL, BUTTON_CLASS, BS_PUSHBUTTON | WS_CHILD | WS_VISIBLE
| WS_TABSTOP, 164, 228, 36, 24
        CONTROL "", 126, "BorShade", BSS_GROUP | WS_CHILD | WS_VISIBLE, 10, 32, 49, 14
        CONTROL "", 127, "BorShade", BSS_GROUP | WS_CHILD | WS_VISIBLE, 11, 54, 51, 12
        CONTROL "", 128, "BorShade", BSS_GROUP | WS_CHILD | WS_VISIBLE, 9, 74, 56, 13
        CONTROL "", 129, "BorShade", BSS_GROUP | WS_CHILD | WS_VISIBLE, 10, 96, 48, 13
        CONTROL "", 130, "BorShade", BSS_GROUP | WS_CHILD | WS_VISIBLE, 6, 117, 58, 14
        CONTROL "", 131, "BorShade", BSS_GROUP | WS_CHILD | WS_VISIBLE, 122, 29, 19, 12
END


FILECOMMANDS MENU
BEGIN
        POPUP "&Archivo"
        BEGIN
                MENUITEM "&Nuevo", CM_FILENEW
                MENUITEM "&Abrir...", CM_FILEOPEN
                MENUITEM "&Salvar", CM_FILESAVE
                MENUITEM "Salvar c&omo...", CM_FILESAVEAS
                MENUITEM SEPARATOR
                MENUITEM "S&alir", CM_EXIT
        END

        POPUP "&Editar"
        BEGIN
                MENUITEM "&Deshacer\aAlt+BkSp", CM_EDITUNDO
                MENUITEM SEPARATOR
                MENUITEM "&Cortar\aShift+Del", CM_EDITCUT
                MENUITEM "C&opiar\aCtrl+Ins", CM_EDITCOPY
                MENUITEM "&Pegar\aShift+Ins", CM_EDITPASTE
                MENUITEM "&Borrar\aDel", CM_EDITDELETE
                MENUITEM "L&impiar todo\aCtrl+Del", CM_EDITCLEAR
        END

        POPUP "&Buscar"
        BEGIN
                MENUITEM "&Encontrar...", CM_EDITFIND
                MENUITEM "&Remplazar...", CM_EDITREPLACE
                MENUITEM "&Siguiente\aF3", CM_EDITFINDNEXT
        END

        POPUP "&Imagen"
        BEGIN
                MENUITEM "1 Linea", DM_LINEA
                MENUITEM "1 Linea Guarda", DM_LINEAGUARDA
                MENUITEM "1 Linea 2 Guardas", DM_LINEAGUARDA2
```

```
            MENUITEM "2 Linea Paralela", DM_LINEAPARALELA
            MENUITEM "2 Linea Paralela Guarda", DM_LINEAPARALELAGUARDA
            MENUITEM "2 Linea Paralela Guarda c/u", DM_LINEAPARALELAGUARDA2
    END
        POPUP "&Lineas"
        BEGIN
            MENUITEM "Selecciona Archivo.  ", CM_ESCOGE
            MENUITEM SEPARATOR
            MENUITEM "1 Linea", CM_LINEA, GRAYED
            MENUITEM "1 Linea  Guarda", CM_LINEAGUARDA, GRAYED
            MENUITEM "1 Linea 2 Guardas", CM_LINEAGUARDA2, GRAYED
            MENUITEM "2 Linea Paralela", CM_LINEAPARALELA, GRAYED
            MENUITEM "2 Linea Paralela Guarda", CM_LINEAPARALELAGUARDA, GRAYED
            MENUITEM "2 Linea Paralela Guarda c/u", CM_LINEAPARALELAGUARDA2,
GRAYED
            MENUITEM SEPARATOR
            MENUITEM "R&ecalcula", CM_RECALCULA, GRAYED
        END

        POPUP "A&yuda", HELP
        BEGIN
            MENUITEM "Acerca de ", CM_ABOUT
            MENUITEM SEPARATOR
            MENUITEM "&Indice\aShift+F1", CM_U_HELPINDEX
            MENUITEM "&Uso de Ayuda", CM_U_HELPHELP
        END

END


ABOUT DIALOG 18, 23, 142, 136
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CLASS "BorDlg"
CAPTION "Acerca de .."
BEGIN
        CONTROL "", IDOK, BUTTON_CLASS, BS_DEFPUSHBUTTON | WS_CHILD | WS_VISIBLE |
WS_TABSTOP, 55, 103, 37, 24
        CONTROL "Lorenzo Caballero Rosas", 103, "STATIC", SS_CENTER | WS_CHILD |
WS_VISIBLE, 24, 53, 99, 10
        CONTROL "Copyright \251 MCMXCIX", 104, "STATIC", SS_CENTER | WS_CHILD |
WS_VISIBLE, 22, 71, 101, 10
        CONTROL "UNAM FI DIE", 108, "STATIC", SS_CENTER | WS_CHILD | WS_VISIBLE, 23, 81,
103, 9
        CONTROL "Calculo de Parámetros de", -1, "STATIC", WS_CHILD | WS_VISIBLE |
WS_GROUP, 25, 7, 92, 8
        CTEXT "Carlos Daniel Ramirez Briseño", -1, 21, 63, 103, 8, WS_CHILD | WS_VISIBLE |
WS_GROUP
        CONTROL "ICON_1", -1, "STATIC", SS_ICON | WS_CHILD | WS_VISIBLE | WS_GROUP, 63,
29, 16, 16
        CONTROL "", 105, "BorShade", BSS_GROUP | WS_CHILD | WS_VISIBLE, 19, 48, 108, 46
        CONTROL "", 106, "BorShade", BSS_GROUP | WS_CHILD | WS_VISIBLE, 21, 5, 105, 21
        CONTROL "  Lineas de Transmisión", -1, "STATIC", WS_CHILD | WS_VISIBLE |
WS_GROUP, 25, 16, 92, 8 END
```

Bibliografía

VIQUEIRA jacinto, "Redes Eléctricas 1", Alfaomega, México 1993.

VIQUEIRA jacinto, "Redes Eléctricas 2", Representaciones y Servicios de Ingeniería, México 1987.

GRAINGER john y STEVENSON william, Análisis de Sistemas de Potencia", McGraw Hill, México 1996

RAULL josé, "Diseño de Subestaciones Eléctricas", McGraw-Hill, México 1994.

MARTIN james y ODELL james, "Análisis y Diseño Orientado a Objetos", Prentice Hall, México 1994.

CEBALLOS francisco, "Programación Orientada a Objetos con C++" , RAMA, 2o. edi., México 1998.

FAISON ted, " Borland C++ 3.1 Programación Orientada a Objetos" , Prentice Hall, 2o. edi. México 1993

ARRIETA norberto y NAVARRO edwin, "Notas de Programación Orientada a Objetos Usando C++", Facultad de Ingeniería, México 1994.

BOOCH, "Análisis y Diseño Orientado a Objetos",Addison-Wesley. 2o. edi., México 1996.

RUMBAUGH, "Object Oriented Modeling and Design", Prentice-Hall, EUA 1991.

CABALLERO ignacio y LOPEZ eduardo, "Sistema de Captura y Presentación Gráfica Bajo Ambiente Windows", Tesis FI UNAM, México 1995.