UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DESARROLLO DE HERRAMIENTAS DE SOFTWARE PARA EL ANÁLISIS DE SISTEMAS ESTRUCTURALES

T E S I S

QUE PARA OBTENER EL TÍTULO DE
INGENIERO EN COMPUTACIÓN

P R E S E N T A N :

PAULINA VENTURA AGUSTÍN
VICENTE JAVIER GUERRERO QUINTERO

DIRECTOR DE TESIS: DR. GUADALUPE MOISÉS ARROYO CONTRERAS

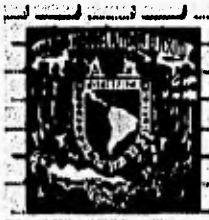
Ciudad Universitaria, México, D.F., 1996

AGRADECIMIENTOS

El presente trabajo no habría llegado a feliz término sin la ayuda de las siguientes instituciones:

División de Estudios de Posgrado de la Facultad de Ingeniería de la UNAM, Dirección General de Asuntos del Personal Académico de la UNAM (DGAPA), Consejo Nacional de Ciencia y Tecnología (CONACYT) y a la Fundación UNAM de Iniciación Temprana a la Investigación y la Docencia.

Agradecemos toda la ayuda proporcionada por nuestro director de tesis Dr. Guadalupe Moisés Arroyo Contreras y a nuestra gran institución la Universidad Nacional Autónoma de México que nos preparó para afrontar con valentía el duro futuro que se nos adviene y recordar que sin su valioso apoyo no se hubiera cumplido nuestro más grande deseo. El ser profesionistas.



CONFIA

ANALISIS DE
DE SISTEMAS
ESTRUCTURALES

UNAM- DEPFI

ÍNDICE GENERAL

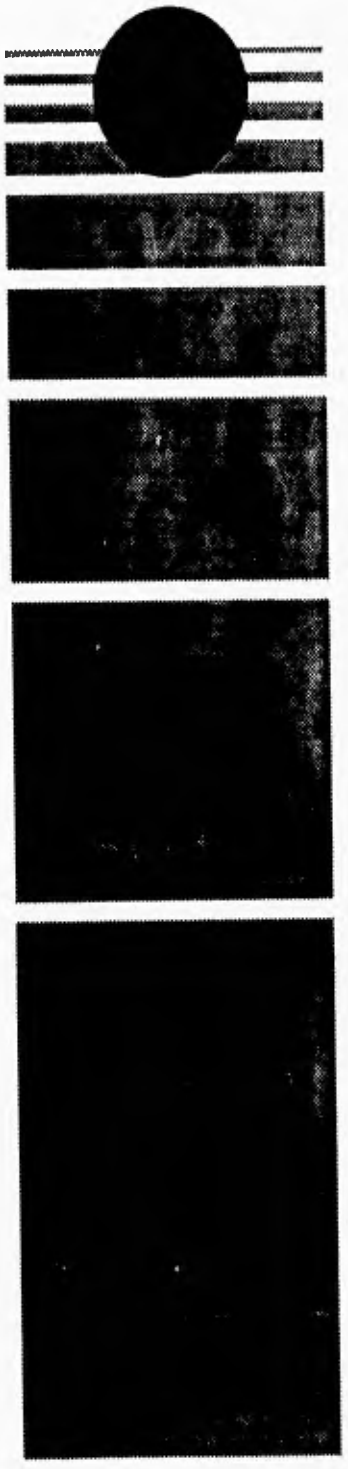
RESUMEN

En el presente trabajo se ha desarrollado una herramienta de software (CONFIA) para el análisis de sistemas estructurales con comportamiento lineal y elastoplástico. El proyecto fue implementado con el lenguaje de programación C++ y la tecnología orientada a objetos (TOO), para operar en ambientes de computadoras personales con el sistema operativo MS-DOS y compatibles.

El objetivo del sistema es apoyar a los alumnos de la carrera de Ingeniería Civil de nivel licenciatura y posgrado en el aprendizaje y aplicación de los análisis de sistemas estructurales de una manera más eficaz e interactiva.

Este trabajo de tesis forma parte de un proyecto más general de desarrollo de una herramienta de software para los análisis de confiabilidad de sistemas mecánicos. Este sistema se realizó en su totalidad en el Departamento de Mecánica de la División de Estudios de Posgrado de la Facultad de Ingeniería de la Universidad Nacional Autonóma de México.

INTRODUCCIÓN



En este proyecto de tesis se ha desarrollado una herramienta de software para realizar análisis de sistemas estructurales con comportamiento lineal y elastoplástico. Se hace la observación que esta tesis forma parte de un proyecto más general de desarrollo de herramientas de software para los análisis de confiabilidad de sistemas, que comprende tres grandes temas: 1) obtención de los estados límites de un sistema estructural con comportamiento elastoplástico, 2) obtención de las fuerzas a las que puede estar sometida la estructura y 3) evaluación de la estructura durante su periodo de existencia. Esta tesis se enfoca a resolver principalmente el problema del tema 1, dejando los otros 2 temas para futuros desarrollos.

El programa desarrollado funciona en computadoras personales y permite a los alumnos y profesores que tengan interés en el aprendizaje de los conceptos y técnicas del análisis estructural, tener acceso a una herramienta interactiva que los guíe y los familiarice con estos conceptos.

Una de las características principales de este programa es que permite al usuario observar el efecto de la variabilidad en la confiabilidad o seguridad de los elementos aislados y del sistema completo, debido a los cambios en los parámetros e incertidumbres que se asignen en el modelo. Los resultados obtenidos se podrán presentar en forma gráfica en cada etapa del análisis para una mejor visualización, comprensión e interpretación de la metodología utilizada.

En el capítulo uno se presentan los conceptos básicos usados en el análisis estructural, así como el procedimiento general del método de rigidez para encontrar los desplazamientos en los nodos de las estructuras y las fuerzas internas de los elementos que la componen.

La herramienta de software CONFIA, ha sido desarrollado en C++, utilizando la tecnología orientada a objetos (TOO), por lo que en el capítulo dos se proporciona una introducción a los conceptos básicos de la TOO y se muestran las características principales y los beneficios que se obtienen al usar esta metodología.

En el capítulo tres se presenta el análisis, diseño e implementación del sistema CONFIA, mostrándose los objetos componentes y su interelación para formar el programa, así como las características y comportamiento interno de cada objeto.

Para evaluar el sistema se tienen dos ejemplos en el capítulo cuatro, donde se desarrolla el análisis estructural manualmente y se muestran los resultados con el sistema CONFIA para su comprobación.

Finalmente se dan las conclusiones del presente trabajo así como las proyecciones que se tienen a futuro para mejorar y ampliar el sistema CONFIA.

En este trabajo se tienen además cuatro apéndices, el apéndice A es el manual de instalación, donde se tienen las características necesarias de hardware para poder usar el sistema CONFIA, los archivos necesarios para su ejecución, los archivos temporales que se crean, los límites del programa y demás especificaciones requeridas para un buen funcionamiento. El segundo apéndice es el manual de utilización, donde se muestra todos los comandos posibles de ejecución, así como las posibles combinaciones que proporciona el programa para realizar los diversos análisis de que consta. El tercer apéndice es un glosario de términos usados a lo largo

del presente trabajo referentes a la Ingeniería Civil y el cuarto apéndice es un glosario referente a los términos usados en la Ingeniería en Computación.

Anexo a este trabajo se proporciona un disco que contiene el código fuente del sistema CONFIA, así como el programa ejecutable con los diversos archivos de soporte y varios ejemplos de estructuras para ser probadas por el usuario.

TEORÍA DEL ANÁLISIS DE SISTEMAS ESTRUCTURALES

1.1 CONCEPTOS BÁSICOS DEL ANÁLISIS ESTRUCTURAL

El análisis estructural es el proceso mediante el cual el ingeniero estructurista determina la respuesta de una estructura a cargas u otras acciones específicas. Esta respuesta generalmente se mide estableciendo las fuerzas y desplazamientos en toda la estructura.

El objetivo del análisis estructural es crear una estructura segura y estable que satisfaga también un conjunto de diversos requisitos impuestos por factores tales como la función de la estructura, las condiciones del lugar, aspectos económicos, estéticos, posibilidades para construir y las restricciones legales.

El cálculo de las fuerzas internas y los desplazamientos es una parte integral del proceso de análisis de una estructura. Por lo regular interesan los esfuerzos internos producidos por las cargas, porque la finalidad es diseñar y dimensionar la estructura de manera que los esfuerzos no excedan los valores límites de seguridad.

Dentro del terreno del análisis estructural existe una gran diversidad de términos usados en la Ingeniería Civil, generalmente estos términos son exclusivos de los profesionistas de estas áreas, o por lo menos su interpretación es específica a estas áreas de aplicación por lo cual se explicarán algunos conceptos básicos usados a lo largo de este capítulo y subsecuentes para una mejor comprensión de toda la teoría expresada a lo largo de este trabajo.

1.1.1 TIPOS DE ESTRUCTURAS

Un sistema estructural es un conjunto de elementos estructurales unidos de tal forma que sirven para resistir las cargas a las que va estar sometido durante toda su existencia. Existen diversos tipos de estructuras: puentes, edificios, estadios, torres de transmisión de energía eléctrica, torres de radio y televisión y muchos más. Los modelos de sistemas estructurales más simples son los sistemas esqueletales formados de:

- Nodos - son los puntos de unión entre 2 o más barras.
- Barras o elementos longitudinales - son los componentes de que consta la estructura y que soportan las cargas o fuerzas externas aplicadas a la misma.
- Vigas - son elementos sometidos principalmente a flexión, se trata de elementos generalmente usados en posición horizontal y sujetos a cargas de gravedad o perpendiculares a su eje longitudinal.
- Columnas - son elementos sometidos principalmente a fuerzas axiales de comprensión.

Para el análisis de las estructuras se puede comenzar con el estudio de dos estructuras básicas:

- Armadura - es un sistema estructural formado de una serie de elementos (barras) rectos, conectados entre sí mediante nodos de tal manera que forman una estructura que actúa como una viga de gran tamaño. Una armadura se construye con formas triangulares en un solo plano de tal forma que las cargas externas se aplican sólo en los nodos, por lo que gracias a la configuración geométrica de las barras y a la condición de carga sólo están

sometidas a fuerzas (axiales) de tensión y comprensión.

- Marcos planos - son sistemas estructurales de forma rectangular que por el tipo de conexión que se tiene, hace que los elementos estén sometidos a fuerzas de flexión, fuerzas cortantes y fuerzas axiales. Además los nodos que unen a los elementos están conectados rígidamente en sus extremos por lo que los elementos son resistentes a momentos. Cuando se aplican cargas a este tipo de estructuras causarán deformación y sus nodos se desplazarán, sin embargo, se consideran los cambios de geometría en los elementos como despreciables. En este tipo de estructuras se basa prácticamente la totalidad del presente trabajo por lo que en posteriores capítulos se hará referencia a los marcos como las estructuras a analizar preferentemente.

En la figura 1.1 se pueden ver ejemplos de los dos tipos de estructuras y sus componentes principales: las barras y los nodos.

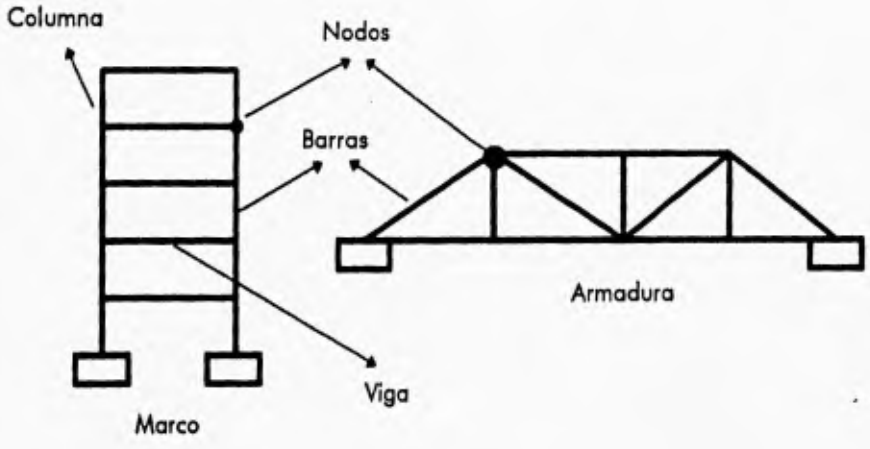


Figura 1.1 Tipos de estructuras y sus componentes.

1.1.2 TIPOS DE ESFUERZOS

En la naturaleza existen infinidad de fuerzas que actúan sobre los cuerpos para producir en ellos diferentes tipos de esfuerzos y reacciones. Dentro del campo de la Ingeniería Civil es muy importante el estudio de las fuerzas que actúan directamente sobre la estructura a diseñar. Como las provocadas por el peso propio de la estructura, las cargas muertas, las cargas vivas, las cargas accidentales producto del sismo, viento, oleaje, nieve, etc.

Las cargas son fuerzas externas que actúan sobre una estructura y causan diversos tipos de esfuerzos en ella, como las fuerzas internas axiales o fuerzas de flexión. Existen diversos tipos de cargas y sólo se mencionarán algunas de ellas para una mejor compresión de capítulos subsecuentes.

- Cargas muertas - son cargas de magnitud constante aplicadas a la estructura y que permanecen en un mismo lugar, constan del peso propio de la estructura y de atras cargas que están permanentemente unidas a ella. En un edificio con estructura de acero algunas de las cargas muertas son la estructura en sí, las paredes, los pisos, las tuberías y los accesorios que éste posee.
- Cargas vivas - son cargas aplicadas a la estructura y que pueden cambiar en magnitud y posición a través del tiempo, ejemplos de estas cargas serían: personas, camiones, viento, lluvia, sismo etc.
- Cargas concentradas - son las cargas que se concentran en un punto específico de la estructura y que por lo general se aplican en los nodos extremos a las barras para cuestiones de estudio.
- Cargas distribuidas - son las cargas aplicadas a la estructura y que se distribuyen a lo largo de toda la barra en la que están aplicadas y que para cuestiones de estudio se consideran constantes sobre toda la superficie en la que actúan.
- Cargas por sismo - son fuerzas que producen máximos esfuerzos o deformaciones en un elemento de la estructura durante un sismo o fuerzas equivalentes aplicadas.
- Cargas por viento - son las fuerzas máximas que el viento puede aplicar a una estructura en un intervalo de tiempo.

Los esfuerzos o fuerzas internas provocadas en los elementos estructurales de un sistema estructural plano son:

- Fuerzas axiales - son fuerzas longitudinales que causan sólo tensión o comprensión en los elementos (barras) de la estructura. Las fuerzas axiales de tensión tienden a agrandar los elementos de la estructura. Las fuerzas axiales de compresión tienden a acortar o pandear los elementos de la estructura.
- Fuerzas cortantes - son fuerzas perpendiculares a los elementos de la estructura y aplicadas en ellos.
- Fuerzas de flexión - son fuerzas que se aplican en los elementos de la estructura y que tienden a flexionarlos o curvearlos, ejemplos de estas fuerzas son momentos aplicados en los extremos de las barras de la estructura, lo que provoca giros en éstos.

En la figura 1.2 se pueden observar los diversos tipos de fuerzas internas y algunos tipos de cargas que se pueden presentar en una estructura, recordando que las cargas son fuerzas externas aplicadas a la misma estructura.

1.1.3 TIPOS DE APOYO

Las condiciones de apoyo o de fijación de la estructura deben definirse de manera que se simule en forma realista el comportamiento del soporte de la estructura. El apoyo es un elemento que puede restringir todos o algunos de los desplazamientos de los elementos que soporta y puede ser:

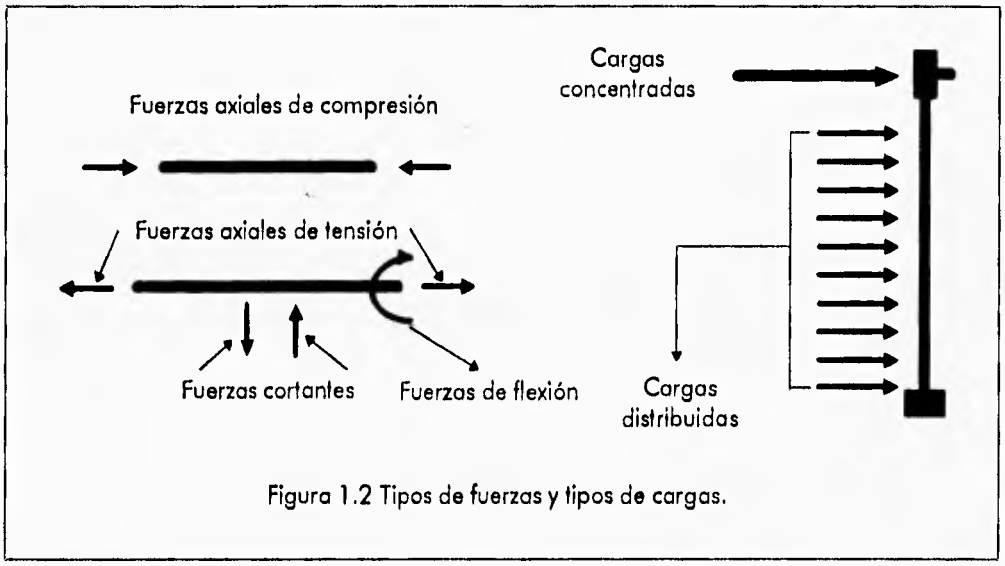


Figura 1.2 Tipos de fuerzas y tipos de cargas.

- Libre - en este tipo de unión el soporte externo no existe, por consiguiente el movimiento de los nodos no está restringido en ninguna dirección, de tal forma que al aplicar cargas los nodos con soporte libre podrán desplazarse en las dos direcciones (X y Y), así como sufrir desplazamientos angulares.
- Articulado - en este soporte el movimiento en el plano está restringido y sólo pueden aparecer desplazamientos angulares o giros.
- Empotrado - este tipo de soporte no permite ningún tipo de desplazamiento ni giro.

El número de posibilidades de desplazamientos independientes que tienen los nodos de un elemento en el espacio se le designa como grados de libertad y son seis en total, sean éstos lineales o angulares.

1.1.4 TIPOS DE ANÁLISIS

Dentro del análisis de las estructuras, para fines de modelación del comportamiento del sistema estructural, se trabaja con diversos tipos de modelos de comportamiento de los elementos de la estructura, aquí sólo se mencionan los tipos de análisis que se utilizan en este trabajo:

- Análisis lineal elástico - los análisis lineales elásticos son aquellos en los que la relación esfuerzo-deformación es proporcional a la elasticidad de los elementos de la estructura. Donde se sabe que la elasticidad es la propiedad de los materiales que permite deformaciones al aplicar cargas dentro de cierto límite a un elemento y al eliminarlas el elemento recupere su estado original.
- Análisis elastoplástico - en estos análisis se consideran las condiciones límites de esfuerzo antes de que los elementos fluyan, es decir antes de que los elementos se plastifiquen o se deformen permanentemente. Otra característica de estos análisis es que son recurrentes,

debido a que es necesario cambiar las propiedades de los elementos de la estructura y volver a realizar el análisis para observar las formaciones de los mecanismos de falla del sistema así como las magnitudes de las cargas que ocasionan éstos.

En la figura 1.3 se muestra una gráfica en la que se indican los rangos en que cada uno de los análisis mencionados actúa. Para el análisis lineal elástico el campo de interacción es en el rango 0-1, en donde la relación esfuerzo-deformación es lineal y es válida la ley de Hooke. Para el caso del análisis elastoplástico su campo de acción o de interés es en el rango de 1-6, en donde se observa que la relación esfuerzo-deformación no es lineal y por lo tanto no obedece a la ley de Hooke.

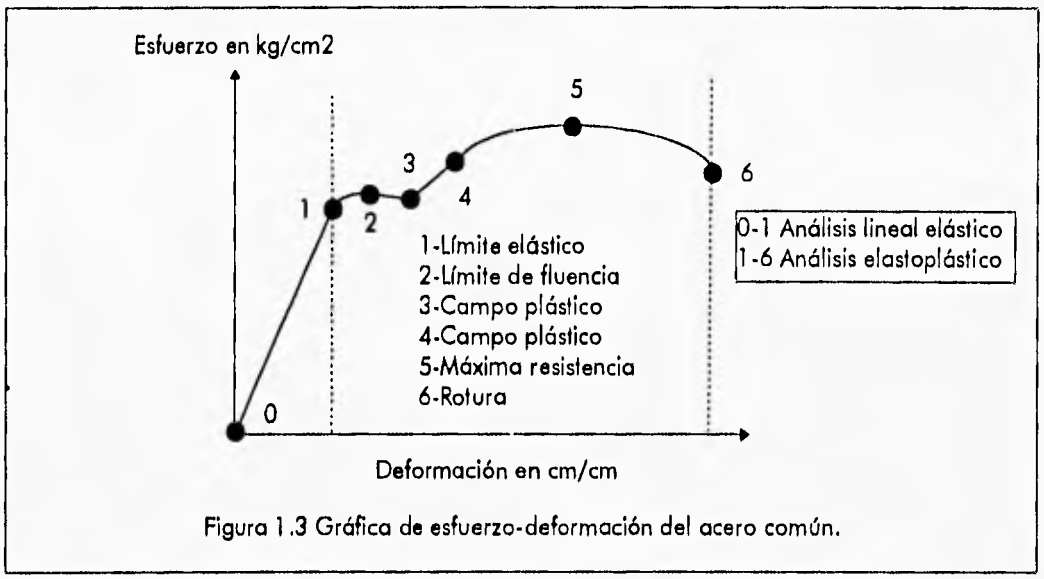


Figura 1.3 Gráfica de esfuerzo-deformación del acero común.

1.2 INTRODUCCIÓN AL ANÁLISIS ESTRUCTURAL

Existen diversos métodos para el análisis estructural, entre ellos están los métodos matriciales de análisis que proporcionan un lenguaje matemático muy adecuado para la descripción de estructuras muy complejas y que pueden ser ejecutadas fácilmente por las computadoras.

El método matricial que puede ser programado más fácilmente en una computadora es el método de los desplazamientos, por lo que es el más ampliamente usado. En el método de los desplazamientos también llamado método de las rigideces o del equilibrio, se obtienen los desplazamientos de cada uno de los nodos provocado por las cargas aplicadas en la estructura para describir y apreciar la forma en que ésta queda deformada. Estos desplazamientos se obtienen mediante la solución de un conjunto de ecuaciones simultáneas de equilibrio del sistema estructural. Después de resolver estas ecuaciones y determinar los desplazamientos, éstos se sustituyen en las relaciones fuerza-deformación de cada elemento para determinar las diversas fuerzas o momentos internos que existen en la estructura.

La ley que relaciona las fuerzas internas de los elementos con los desplazamientos nodales de la estructura es la ley de Hooke descrita mediante la relación:

$$\{ f \} = [K] \{ d \}$$

Donde [K] es la rigidez, sabiendo que la rigidez da una medida de las fuerzas que están asociadas con un conjunto dado de desplazamientos. En su forma más simple, la rigidez es la fuerza que debe aplicarse en algún punto para producir un desplazamiento unitario en ese punto. Los parámetros { f } y { d } son los vectores de fuerza y desplazamientos nodales respectivamente.

En una estructura cada barra componente está referenciada a unas coordenadas locales, en la figura 1.4 se muestra una estructura ficticia con la finalidad de visualizar las diferentes coordenadas locales presentes.

Sin embargo para poder hacer el análisis de la estructura todos los elementos deben estar referenciados a unas coordenadas comunes globales, por lo que es necesario hacer una rotación de vectores o de fuerzas aplicadas a los nodos extremos de las barras y de los desplazamientos involucrados como lo muestra la figura 1.5 para poder obtener un sistema de fuerzas o desplazamientos equivalente en un sistema de referencia global o referenciado a las coordenadas globales X-Y a través de una matriz de rotación de ejes coordenados como se muestra en las ecuaciones 1.1a y 1.1b.

$$\begin{Bmatrix} F_1 \\ F_2 \\ F_3 \end{Bmatrix}_{X-Y} = \begin{bmatrix} \text{Cos}\theta & -\text{Sen}\theta & 0 \\ \text{Sen}\theta & \text{Cos}\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} f_1 \\ f_2 \\ f_3 \end{Bmatrix}_{x-y} \qquad (1.1a)$$

$$\{F\}_{X-Y} = [R]\{f\}_{x-y}$$

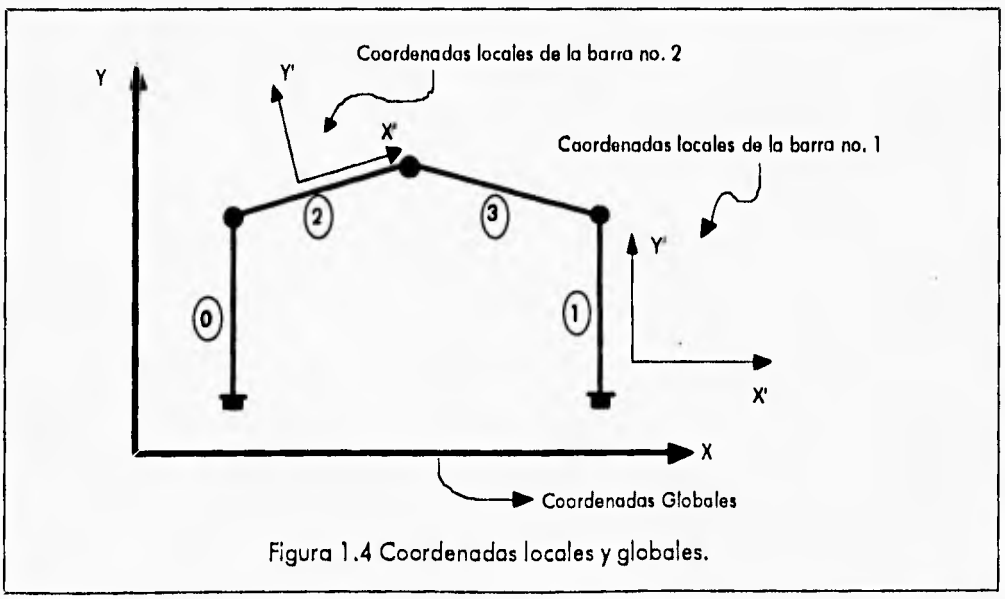


Figura 1.4 Coordenadas locales y globales.

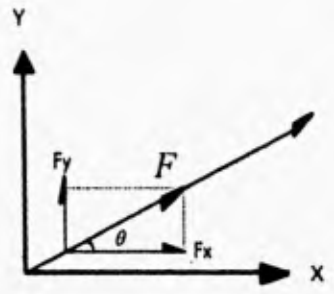


Figura 1.5 Rotación de vectores de coordenados locales a globales.

$$\begin{Bmatrix} D_1 \\ D_2 \\ D_3 \end{Bmatrix}_{X-Y} = \begin{bmatrix} \text{Cos}\theta & -\text{Sen}\theta & 0 \\ \text{Sen}\theta & \text{Cos}\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} d_1 \\ d_2 \\ d_3 \end{Bmatrix}_{x-y} \qquad (1.1b)$$

$$\{D\}_{X-Y} = [R]\{d\}_{x-y}$$

Donde [R] es la matriz de rotación.

Para un vector de fuerzas o desplazamientos nodales aplicados en los extremos de un elemento, la transformación de fuerzas se obtiene a través de la matriz de transformación [T] siguiente:

$$\begin{Bmatrix} F_i \\ F_j \end{Bmatrix}_{X-Y} = [T]\begin{Bmatrix} f_i \\ f_j \end{Bmatrix}_{x-y} = \begin{bmatrix} R & 0 \\ 0 & R \end{bmatrix} \begin{Bmatrix} f_{xi} \\ f_{yi} \\ f_{zi} \\ f_{xj} \\ f_{yj} \\ f_{zj} \end{Bmatrix} \qquad (1.2)$$

$$\begin{Bmatrix} D_i \\ D_j \end{Bmatrix}_{X-Y} = [T]\begin{Bmatrix} d_i \\ d_j \end{Bmatrix}_{x-y} = \begin{bmatrix} R & 0 \\ 0 & R \end{bmatrix} \begin{Bmatrix} d_{xi} \\ d_{yi} \\ d_{zi} \\ d_{xj} \\ d_{yj} \\ d_{zj} \end{Bmatrix}$$

No sólo la fuerzas y desplazamientos deben estar referencias a unas coordenadas globales, sino también se debe calcular la matriz de rigidez global de una barra mediante la ecuación:

$$[K_n]_{global} = [T_n][k_n][T_n]^T \qquad (1.3)$$

Donde $[K_n]_{global}$ se llama la matriz de rigidez del elemento n en coordenadas globales, y relaciona las fuerzas con los desplazamientos del elemento en coordenadas globales.

La ecuación de equilibrio de la estructura está dada por:

$$\{F\} = [K]_{global}\{D\} \qquad (1.4)$$

Donde:

{ F } es el vector de fuerzas a la que está sometida la estructura.
[K] es la matriz de rigidez global del sistema.
{ D } es el vector de desplazamientos de los nodos del sistema estructural y la incógnita a encontrar.

La matriz [K] se obtiene al ensamblar las matrices de rigidez global de cada barra considerando los grados de libertad que tengan los nodos extremos de la barra.

Una vez obtenidos los desplazamientos nodales resolviendo el sistema de ecuaciones en 1.4, se pueden calcular las fuerzas internas que se encuentran en las barras con la siguiente expresión:

$$\{f_n\} = [k_n]\{d_n\} = [k_n][T_n]^T\{D_n\} \qquad (1.5)$$

1.3 PROCEDIMIENTO GENERAL DEL ANÁLISIS ESTRUCTURAL

Habiendo presentado una pequeña introducción al análisis estructural ahora se presenta el procedimiento general que se sigue para realizar el análisis estructural por el método de las rigideces. Los pasos para realizar el análisis son los siguientes:

A)-Establecer la información básica general del sistema estructural

En esta primera etapa del análisis estructural se definen los materiales, la geometría, las propiedades mecánicas, la topología de los elementos y del sistema estructural en general como se puede observar en la figura 1.6.

La geometría general de un sistema estructural, como el mostrado en la figura 1.6 consiste en definir:

- Las coordenadas de los nodos, en función de un sistema de coordenadas globales.
- La topología del sistema, que consiste en definir la orientación y conectividad de las barras. Como todas las barras o elementos estructurales tienen dos nodos extremos, uno de éstos será el nodo origen y el restante el nodo destino o el nodo final. Ésto se realiza con el fin de poder tener una orientación en los elementos e interpretar los resultados de las fuerzas internas, así como para determinar las secciones críticas de los elementos en base a esta convención.

Para las propiedades mecánicas del material, si se consideran materiales con comportamiento lineal elástico, basta con definir el módulo de elasticidad E; el área de la sección transversal A; y el momento de inercia I.

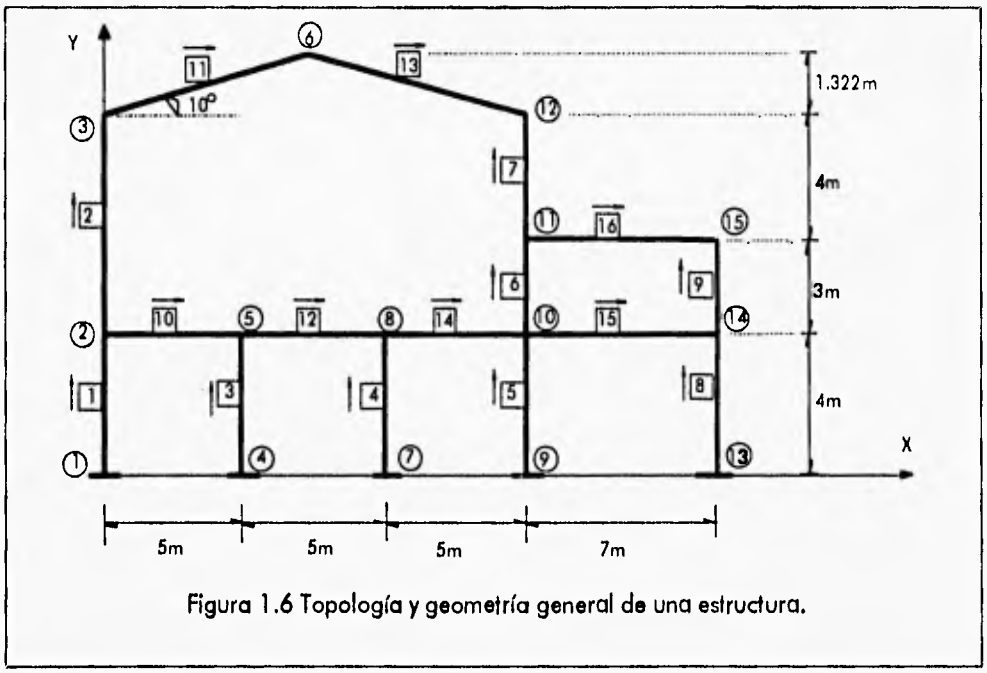


Figura 1.6 Topología y geometría general de una estructura.

Las cargas pueden ser modeladas como fuerzas estáticas, que se representan por un lado, como fuerzas concentradas o como fuerzas distribuidas; por otro lado, son las fuerzas a las que puede estar sometida una estructura, provocadas por los efectos más comunes de la carga muerta, carga viva, empujes de material o de fluidos y por los efectos accidentales del sismo, viento y oleaje.

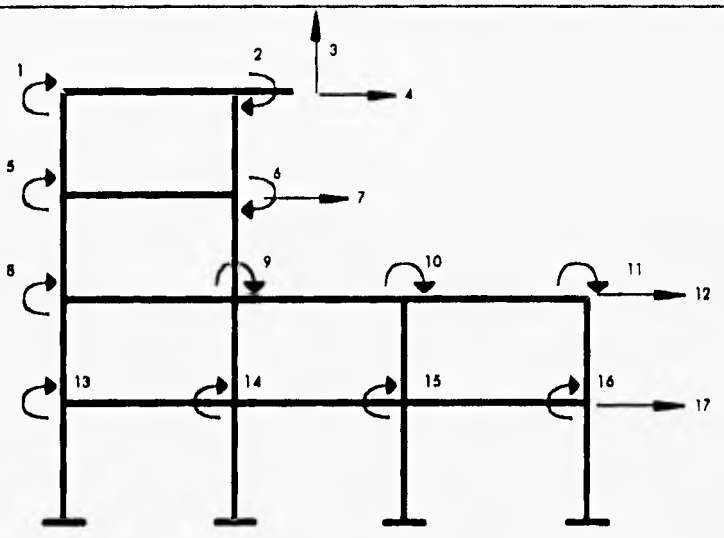
B)-Definir los vectores y matrices estructurales

En esta etapa del análisis estructural se definen arreglos estructurales referidos a un sistema de coordenadas generalizadas. Un sistema de coordenadas generalizadas es el conjunto de fuerzas o desplazamientos, numerados en forma secuencial y localizados en una estructura en posición y dirección. Para el método de las rigideces las coordenadas generalizadas son la posición y

dirección de los desplazamientos incógnitas, como se muestra en la figura 1.7. Por fuerzas y desplazamientos se entiende no sólo los fuerzas y desplazamientos lineales sino también los momentos y rotaciones, respectivamente. Los términos de un vector de fuerzas o desplazamientos se agrupan en el orden secuencial de las coordenadas generalizadas.

C)-Realizar la rotación de vectores y determinar las matrices de rigidez de los elementos en coordenadas locales y globales

Anteriormente ya se ha definido como se establece la rotación de coordenadas locales a coordenadas globales de las fuerzas y desplazamientos que intervienen en una estructura para su homogeneización. Existen diferentes tipos de matrices de rigidez de los elementos estructurales, que dependen de: la geometría (por ejemplo, si son elementos de sección constante o elementos de sección variable), del comportamiento mecánico del material, de las simplificaciones adoptadas para fines prácticos, etc. Para elementos de sección constante y homogéneos, empotrados en ambos extremos, formados con materiales de comportamiento elástico lineal, se puede obtener fácilmente su matriz de rigidez, haciendo uso del sistema de referencia local del elemento, como se muestra en las figuras 1.7a y 1.7b.



Coordenadas generalizadas y grados de libertad

Figura 1.7a Sistema de coordenados generalizadas y vectores estructurales.

$$[k]_{local} = \begin{bmatrix} EA/L & 0 & 0 & -EA/L & 0 & 0 \\ & 12EI/L^3 & 6EI/L^2 & 0 & -12EI/L^3 & 6EI/L^2 \\ & & 4EI/L & 0 & -6EI/L^2 & 2EI/L \\ & & & EA/L & 0 & 0 \\ & \text{simétrica} & & & 12EI/L^3 & -6EI/L^2 \\ & & & & & 4EI/L \end{bmatrix} \qquad (1.6)$$

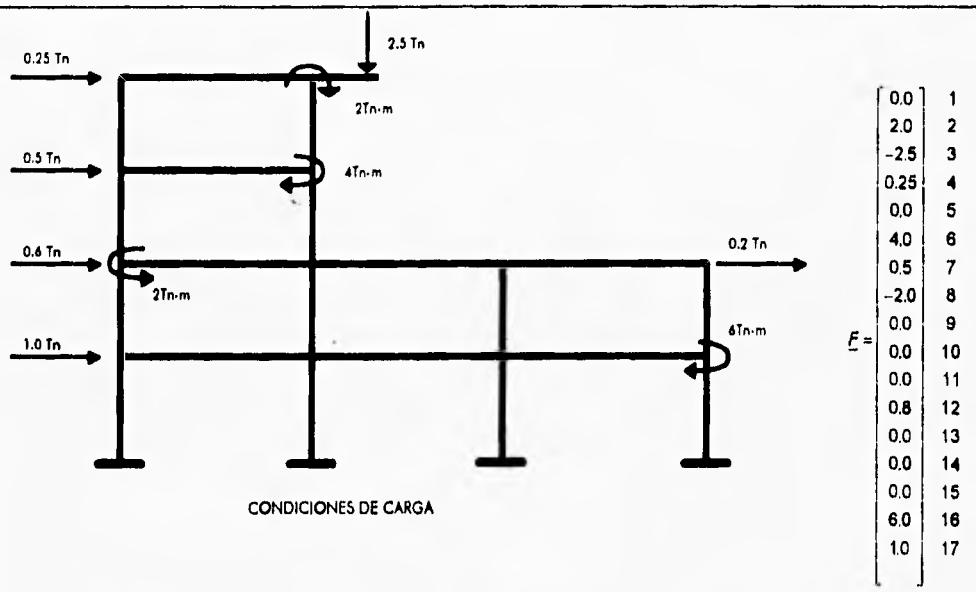


Figura 1.7b Sistema de coordenadas generalizadas y vectores estructurales.

La matriz de rigidez de un elemento articulado en ambos extremos, tipo armadura en coordenadas locales se puede escribir como sigue:

$$[k] = \begin{bmatrix} 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \frac{EA}{l} \qquad (1.7)$$

La matriz de rigidez de un elemento, referenciada al sistema de coordenadas global del sistema estructural, se puede obtener mediante la ecuación 1.3.

D)-Ensamblar la matriz de rigidez global del sistema estructural

Cada matriz de rigidez de los elementos en coordenadas globales, tiene asociadas unas coordenadas generalizadas, correspondientes a los grados de libertad de los nodos extremos del elemento (a este grupo de coordenadas generalizadas se le llama también vector de localización). A partir de las matrices de rigidez de los elementos en coordenadas globales y de los vectores estructurales asociados a estas matrices, se ensambla la matriz de rigidez global del sistema estructural; es decir, ensamblando cada matriz de un elemento en coordenadas globales, en la matriz global del sistema, en las posiciones indicadas por el vector de localización:

$$[K]_{sistema} = \sum_{i=1}^{n} [K_i] \qquad (1.8)$$

Donde n es el número de elementos que integran el sistema estructural y la sumatoria indica "ensamble de las matrices de rigidez de los elementos".

E)-Obtener los desplazamientos de los nodos de la estructura resolviendo el sistema de ecuaciones:

$$\{ F \} = [K] \{ D \} \qquad (1.9)$$

F)-Obtener las fuerzas internas en los extremos de los elementos mediante la ecuación:

$$\{f_n\} = [k_n]\{d_n\} = [k_n][T_n]^T \{D_n\} \qquad (1.10)$$

1.4 ANÁLISIS ELASTOPLÁSTICO

El análisis elástico de una estructura es importante para estudiar su comportamiento, en especial por lo que respecta al servicio que debe prestar, bajo la carga para la cual es diseñada la estructura. Sin embargo, si se aumenta la carga hasta que la estructura cede en algunos lugares, ésta sufre deformaciones elastoplásticas y con un aumento adicional de carga se alcanza una condición completamente plástica, donde se forma un número suficiente de articulaciones plásticas para transformar la estructura en un mecanismo o en un sistema estructural inestable.

El objetivo principal del análisis elastoplástico es encontrar los posibles mecanismos de colapso de la estructura y las cargas críticas que dan origen a la formación de tales mecanismos.

El análisis elastoplástico de estructuras es básicamente el mismo que el análisis elástico, siendo la diferencia principal que este proceso se tiene que aplicar repetidamente a medida que las propiedades de la estructura cambian con el aumento de carga debido a la formación de articulaciones plásticas.

1.4.1 PROCEDIMIENTO GENERAL PARA EL ANÁLISIS ELASTOPLÁSTICO

El procedimiento general para realizar el análisis elastoplástico de estructuras es el siguiente:

1) Partiendo de un estado inicial de estabilidad de la estructura donde todos los elementos tienen un comportamiento lineal, se identifica la sección o elemento más esforzado y el incremento de carga necesario para llegar a su estado límite de resistencia o de fluencia del material. El cálculo de los estados límites de seguridad consta de:

- Cálculo de la condición de plasticidad de un elemento o una de sus secciones.
- Cálculo de la función de estado de seguridad límite del elemento expresada como una combinación lineal de las fuerzas nodales de resistencia y de las cargas aplicadas a la estructura.

2) Se realiza un nuevo análisis lineal del sistema estructural, plastificando la sección más esforzada encontrada en el análisis anterior. La matriz de rigidez local del elemento plastificado se sustituye por una matriz de rigidez reducida y se calcula el vector de fuerzas de resistencia residual nodal equivalente del elemento plastificado. A continuación se identifica la nueva articulación plástica, a partir de la función de estado límite de la nueva sección más esforzada.

3) Se repite el paso 2, hasta alcanzar un estado de falla de la estructura o hasta identificar un mecanismo plástico de falla del sistema estructural.

4) A partir de la función de estado límite de la última sección más esforzada, se puede obtener la carga crítica que produce un estado de inestabilidad o posible mecanismo de colapso de la estructura.

Para realizar el procedimiento anterior se deben considerar las hipótesis siguientes para disminuir la complejidad del análisis:

- Las barras y los elementos estructurales se consideran rectos, homogéneos y sólo existen cargas concentradas.
- El comportamiento mecánico del material se supone elastoplástico ideal. Es decir la deformación plástica es ilimitada.
- Una sección del elemento cede o se plastifica cuando la función de estado límite de seguridad se anula (esta función está determinada por las dimensiones y la fuerza del límite elástico, así como por la fuerza interna demandada) y se forma por consiguiente una articulación plástica.
- Las secciones críticas sólo aparecen en los extremos de los elementos, o en los lugares donde las cargas concentradas están aplicadas.
- El cambio de geometría es despreciable.

1.4.2 MATRIZ DE RIGIDEZ REDUCIDA Y VECTOR DE FUERZAS EQUIVALENTES

Con el fin de realizar los análisis paso a paso, con carga constante, degradación de la estructura y con el mismo número de nodos del sistema; en cada plastificación de las secciones potencialmente críticas, se pueden deducir unas matrices de rigidez reducidas para los elementos plastificados y unos vectores de fuerzas nodales equivalentes. La matriz de rigidez inicial de un elemento empotrado en ambos extremos está dada por:

$$[k]_k = \begin{bmatrix} EA/L & 0 & 0 & -EA/L & 0 & 0 \\ & 12EI/L^3 & 6EI/L^2 & 0 & -12EI/L^3 & 6EI/L^2 \\ & & 4EI/L & 0 & 6EI/L^2 & 2EI/L \\ & & & EA/L & 0 & 0 \\ & SIM. & & & 12EI/L^3 & -6EI/L^2 \\ & & & & & 4EI/L \end{bmatrix} \qquad (1.11)$$

Cuando se forma una articulación plástica en el extremo izquierdo (2k-1) del elemento k, la matriz de rigidez del elemento a flexión empotrado en ambos extremos es reemplazada por una matriz de rigidez reducida de un elemento a flexión, articulado a la izquierda y empotrado a la derecha y en los nodos extremos del elemento k-ésimo se aplica un vector $\{f_k\}$ de fuerzas nodales equivalentes:

$$[k_k]^I \begin{bmatrix} EA/L & 0 & 0 & -EA/L & 0 & 0 \\ & 3EI/L^3 & 0 & 0 & -3EI/L^3 & 3EI/L^2 \\ & & 0 & 0 & 0 & 0 \\ & & & EA/L & 0 & 0 \\ & SIM. & & & 3EI/L^3 & -3EI/L^2 \\ & & & & & 3EI/L \end{bmatrix}$$

$$\{f_k\}^I = \{0, (3/2l_k)M_{p_k}, M_{p_k}, 0, -(3/2l_k)M_{p_k}, M_{p_k}/2\} \qquad (1.12)$$

Donde M_{pk} es el momento plástico del extremo izquierdo del elemento (k) y l_k es la longitud del elemento (k).

De la misma manera, cuando una articulación plástica se forma en el extremo derecho (2k) del elemento k-ésimo, la matriz de rigidez del elemento a flexión empotrado en ambos extremos es reemplazada por una matriz de rigidez reducida de un elemento a flexión articulado a la derecha y empotrado a la izquierda y el vector de fuerzas nodales equivalente $\{f_k\}$ aplicadas en los nodos extremos del elemento k-ésimo es:

$$[k_k]^D = \begin{bmatrix} EA/L & 0 & 0 & -EA/L & 0 & 0 \\ & 3EI/L^3 & 3EI/L^2 & 0 & -3EI/L^3 & 0 \\ & & 3EI/L & 0 & -3EI/L^2 & 0 \\ & & & EA/L & 0 & 0 \\ & SIM & & & 3EI/L^3 & 0 \\ & & & & & 0 \end{bmatrix}$$

$$\{f_k\}^D = \{0, (3/2l_k)M_{p_k}, M_{p_k}/2, 0, -(3/2l_i)M_{p_k}, M_{p_k}\} \qquad (1.13)$$

Finalmente, cuando se forman dos articulaciones plásticas en los extremos del elemento, se tiene una matriz de rigidez reducida equivalente a un elemento barra bi-articulado (tipo armadura) y un vector de fuerzas nodales equivalentes aplicado en los extremos:

$$[k_k]^{ID} = \begin{bmatrix} EA/L & 0 & 0 & -EA/L & 0 & 0 \\ & 0 & 0 & 0 & 0 & 0 \\ & & 0 & 0 & 0 & 0 \\ & & & EA/L & 0 & 0 \\ & SIM. & & & 0 & 0 \\ & & & & & 0 \end{bmatrix}$$

$$(f_k)^{ID} = \left\{0, \left(M_{p_{2k-1}} + M_{p_{2k}}\right)/l_k, M_{p_{2k-1}}, 0, -\left(M_{p_{2k-1}} + M_{p_{2k}}\right)/l_k, M_{p_{2k}}\right\} \qquad (1.14)$$

1.4.3 CRITERIO DE FALLA DE UNA BARRA FLEXIONADA (FLEXIÓN PURA)

Sea una estructura con barras sujetas principalmente a flexión, compuesta de n elementos (barras) y sujeta a 3m cargas concentradas en los nodos, donde m es el número de nodos, las secciones extremas i de las barras k (k=1, 2, ..., n) están numeradas a la izquierda por (2k-1) y a la derecha por (2k). El análisis de fuerzas de la estructura se hace por el método matricial de rigidez. Por tanto, las fuerzas en el extremo i de un elemento están dadas por:

$$S_i = M_i = \sum_{j=1}^{3m} b_{ij} L_j \qquad (i = 1,2,\ldots,2n) \qquad (1.15)$$

Donde L_j son las cargas aplicadas a la estructura y b_{ij} coeficientes que están en función de las propiedades geométricas y mecánicas de los materiales de los elementos (E-módulo de elasticidad, A-área de la sección transversal del elemento, L-longitud del elemento, I-momento de inercia en el elemento). Los momentos resistentes límites de flexión de las zonas plastificadas del elemento o de las secciones críticas son:

$$R_{2k-1} = R_{2k} = M_{pk} = Z_{ok}\,\sigma_{yk} \qquad (k = 1,\ldots,n) \qquad (1.16)$$

Donde Z_{ok} es el módulo de sección plástico (momento estático, se deriva de las propiedades geométricas de la sección transversal del elemento) y σ_{yk} es el esfuerzo límite elástico o esfuerzo de fluencia (es una propiedad del material) del elemento i-ésimo respectivamente.

La función de estado de seguridad de las secciones extremas de los elementos está dada por:

$$\begin{aligned} Z_i &= R_i - \text{signo}(S_i)S_i \\ &= M_{pi} - \text{signo}(M_i)M_i \qquad (i = 1,2,\ldots,2n) \\ &= \text{Resistencia del elemento - Demanda del elemento} \end{aligned} \qquad (1.17)$$

El criterio de falla (o de fluencia) de una sección está entonces dado por:

$$Z_i \leq 0 \qquad \text{ó} \quad Z_i = M_{pi} - \lambda_i \, \text{signo}(M_i) M_i = 0 \tag{1.18}$$

Donde λ_i es un parámetro mayor o igual que la unidad e indica el aumento de carga necesario para llegar al estado límite de seguridad del elemento.

1.4.4 CRITERIO DE ESTADO LÍMITE DE UN SISTEMA ESTRUCTURAL

El estado límite del sistema está definido por la formación de un mecanismo plástico en la estructura. Cuando una sección de un elemento se plastifica, los esfuerzos se redistribuyen en la estructura de la que forman parte.

Un mecanismo de falla se determina a través de un análisis paso a paso, donde se van identificando en cada etapa los estados de plastificación parcial en los elementos de la estructura hasta la formación de un mecanismo. Es decir, cuando una sección de un elemento se plastifica, una redistribución de esfuerzos ocurre en los elementos restantes, para determinar en seguida una nueva plastificación.

Para el cálculo de esfuerzos en cada paso del análisis, las matrices de rigidez de elementos plastificados son reemplazadas por unas matrices de rigidez reducidas y unos vectores de fuerzas nodales equivalentes son aplicados en los nodos de la estructura. Se llega a la falla de la estructura cuando se alcanza un número determinado de plastificaciones y se tiene que:

$$\text{DET} \left| K^p \right| = 0 \tag{1.19}$$

1.4.5 EXPRESIONES PARA LOS ESTADOS DE SEGURIDAD LÍMITE DE UN SISTEMA ESTRUCTURAL

Después de que se forma una serie de articulaciones $r_1, r_2, \ldots, r_{p-1}$, las matrices de rigidez de los elementos dañados se reemplazan por matrices de rigidez reducidas y las fuerzas residuales se aplican en los nodos del sistema como fuerzas nodales equivalentes. En estas condiciones se efectúa un nuevo análisis del sistema, donde las fuerzas internas de los elementos restantes son:

$$S^{(p)}_{i(r_1, r_2, \ldots, r_{p-1})} = \sum_{j=1}^{3l} b^{(p)}_{ij} L^{(p)}_j = \sum_{j=1}^{3l} b^{(p)}_{ij} L_j - \sum_{k=1}^{p-1} \alpha^{p-1}_{ir_k} R_{r_k} \tag{1.20}$$

Donde $r_1, r_2, \ldots, r_{p-1}$ significa un conjunto de secciones plastificadas, arreglados por orden cronológico de aparición de falla y a_{ir} son unos coeficientes de influencia.

Dadas las resistencias límite de los elementos R_i y las fuerzas internas (ec. 1.20), las ecuaciones de estado de seguridad límite de los elementos restantes pueden escribirse:

$$Z_{i(.)}^{(p)} \cong R_i - \text{signo}(S_{i(.)}^{(p)}) S_{i(.)}^{(p)} = R_i + \sum_{k=1}^{p-1} a_{ir}^{(p)} R_k - \sum_{j=1}^{ml} b_{ij}^{(p)} L_i \qquad (1.21)$$

Donde $(\cdot)$ representa $(r_1, r_2, \ldots, r_{p-1})$. Se tiene la falla del sistema cuando las secciones r_1, $r_2, \ldots, r_{p-1}$ están plastificadas. El criterio de falla del sistema estructural puede ser representado por la función de seguridad de las secciones de las barras más esforzadas, como sigue:

$$Z_{r_p(r_1, r_2, \ldots, r_{p-1})}^{(p)} \leq 0 \qquad p = 1, 2, \ldots, p_q \qquad (1.22)$$

1.5 ANÁLISIS DE CONFIABILIDAD DE SISTEMAS ESTRUCTURALES

En el análisis y diseño estructural intervienen varias variables cuyo valor no puede predecirse con exactitud, por ejemplo, la magnitud de las cargas que actúan sobre la estructura y las propiedades mecánicas del material. Si todas las variables que intervienen en el diseño de un sistema estructural fueran deterministas, es decir, si su valor pudiera predecirse con absoluta precisión, para lograr la seguridad deseada de la estructura, bastaría diseñarla para que su resistencia ante todos los posibles estados límite de falla fuera superior que la acción correspondiente.

En realidad existen incertidumbres en todo el proceso de diseño que hacen que no pueda fijarse con precisión el valor de las variables involucradas. La incertidumbre puede ser reducida pero no eliminada. Por muy conservador que sea el diseño, siempre habrá una probabilidad mayor de cero de que la acción máxima que se vaya a presentar exceda a la resistencia para algún estado límite de falla. El diseño debe procurar que esta probabilidad sea muy pequeña dentro de los límites que permita la economía.

Dentro del análisis de sistemas estructurales, el tratamiento de problemas que se refieren a la seguridad en sí, se denomina confiabilidad estructural, donde se hace uso de la teoría de las probabilidades para su desarrollo. En este contexto debe definirse un concepto importante para el análisis de confiabilidad de los sistemas estructurales: Estado límite.

Se llama estado límite de una estructura a cualquier etapa de su comportamiento a partir de la cual su respuesta se considera inaceptable. Se distinguen dos tipos de estados límite. Aquellos relacionados con la seguridad o con los estados límite de falla que corresponden a situaciones en las que la estructura sufre una falla total o parcial, o presenta daños que afectan su capacidad para resistir nuevas acciones. El otro tipo de estado límite se relaciona con aquellas situaciones que, aún sin poner en riesgo la seguridad de la estructura, afectan su correcto funcionamiento, éstos se denominan estados límite de servicio.

Por otro lado, el objetivo del diseño estructural es proporcionar una seguridad adecuada ante la aparición de estados límite de falla para las acciones más desfavorables que puedan presentarse durante la vida útil de la estructura y procurar que en las condiciones normales de operación no se sobrepasen los estados límite de servicio.

La teoría de confiabilidad para los sistemas estructurales tratada en este proyecto, se enfoca a problemas de confiabilidad del tipo de estado límite donde pueden distinguirse tres estados de comportamiento en un sistema estructural, definidos a través de una función de estado tal que:

$$g(\{X\}) \begin{cases} < 0 & \text{estado de falla} \\ = 0 & \text{estado límite} \\ > 0 & \text{estado de seguridad} \end{cases}$$

Una estructura está un estado de falla, si las cargas externas aplicadas en la estructura son mayores a la resistencia que ofrece la estructura. Una estructura se encuentra en el estado límite si la carga externa es igual a la resistencia de la estructura, y finalmente una estructura está en el estado de seguridad si la resistencia de la estructura es mayor a las cargas externas que afectan a la estructura.

Si el problema se simplifica a dos variables de demanda y resistencia de la estructura, el objetivo es encontrar la probabilidad de que la estructura falle o la probabilidad de que la estructura sea segura, que se pueden definir como:

$$P_{falla} = \text{Prob}[F>R \text{ al menos una vez en } (0,t)] = \text{Prob}[Fmax>R] \qquad (1.23)$$

$$P_{seguridad} = 1 - P_{falla} = \text{Prob}[F<R \text{ siempre para todo el intervalo } (0,t)] = \text{Prob}[Fmax<R] \qquad (1.24)$$

F - es una variable aleatoria que representa la demanda a la que está sujeto el sistema estructural.
R - es la resistencia del sistema estructural.
Fmax - es la carga más grande que actúa sobre la estructura en el intervalo (0,t).

La determinación numérica de la probabilidad de falla es una tarea difícil, ya que normalmente son valores muy pequeños, por lo que se ha creado una técnica muy sencilla para simplificar el problema y medir la probabilidad de falla de un sistema estructural utilizando sólo las medias y desviaciones estándar de las variables aleatorias involucradas.

La función de estado límite del sistema estructural puede definirse como una función lineal de la forma: $Z = g(X) = b_0 + b^T X$, donde X es un vector de variables aleatorias que en nuestro contexto son las resistencias y solicitaciones (parámetros de carga) que intervienen en la estructura, $b = (b_1, b_2, \ldots, b_n)$ un vector de constantes y b_0 también parámetro constante, b^T es la transpuesta de b. Si $\bar{Z}$ es la media de Z y σ_z su desviación estándar, entonces:

$$\bar{Z} = b_0 + b^T \bar{X}; \quad \sigma_z = \left(b^T [C_X] b\right)^{1/2} \qquad (1.25)$$

Donde $\bar{X}$ es el vector de los valores medios de X y $[C_X]$ su matriz de covarianza. La

probabilidad de falla P_{falla} del sistema se obtiene entonces como sigue:

$$P_{falla} = p\ (Z < 0) = \Phi\left[-\frac{\bar{Z}}{\sigma_z}\right] = \Phi\left[-\frac{b_o + b^T\bar{X}}{(b^T[C_x]b)^{1/2}}\right] = \Phi\ (\ -\beta\ ) \qquad (1.26)$$

Donde $\beta = -\bar{Z}/\sigma_Z$ representa el índice de confiabilidad del sistema. Este índice de confiabilidad puede interpretarse como el número de desviaciones estándar σ_Z que separa la media $\bar{Z}$ del origen o frontera con la región de falla. Por regla general valores grandes de β implican probabilidades de falla pequeñas; $\Phi\ (\ -\beta\ )$ es la función de distribución normal acumulada.

En la figura 1.8 se puede visualizar gráficamente el significado de este índice β; para este caso la variable aleatoria Z sigue una función de distribución normal estándar p(Z).

$$P_{falla} = Pr\,ob(Z < 0) = \int_{-\infty}^{0} p(Z)dZ \qquad (1.27)$$

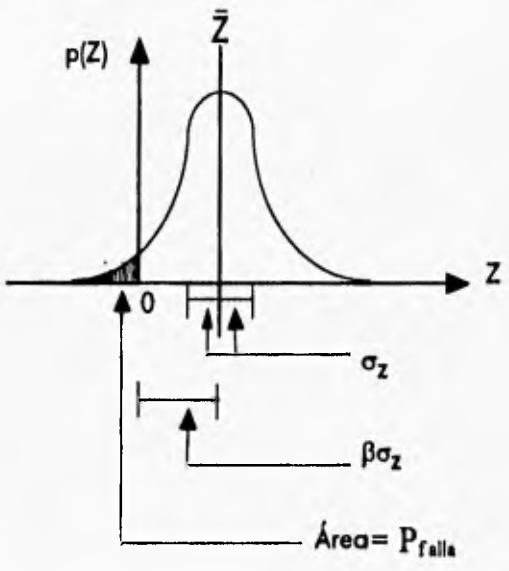


Figura 1.8 Índice de confiabilidad β.

1.6 SOLICITACIONES

Las solicitaciones son fuerzas externas de magnitud variable que pueden aparecer sobre la estructura en cualquier período de su existencia; debido a esto el ingeniero estructurista debe de tomar en cuenta este tipo de fenómenos al diseñar la estructura, para de esta forma no declinar el ciclo de vida de la construcción y brindar una seguridad adecuada.

Las solicitaciones a las que una estructura puede estar sometida a lo largo de su período de vida son diversas, como por ejemplo: los sismos, el viento, el oleaje, la lluvia, la nieve, etc. En el presente trabajo sólo se analizan y desarrollan dos tipos de solicitaciones: sismo y viento. Estas fuerzas se obtienen de acuerdo al Reglamento de Construcciones del D.F.

1.6.1 SISMO

Las fuerzas sísmicas en los diferentes niveles de una estructura pueden evaluarse suponiendo un conjunto de fuerzas horizontales que actúan sobre cada uno de los puntos donde se concentran las masas. La fuerza actuante donde se concentra una masa i es igual al peso de la misma (Wi), por un coeficiente proporcional a la altura (Hi), de la masa en cuestión sobre el desplante (o nivel a partir del cual las deformaciones estructurales pueden ser apreciables). En general la aplicación del método estático para encontrar la fuerza horizontal Pi aplicada en el centro de masas del nivel i ocasionada por un sismo está dada por la fórmula:

$$P_i = \frac{W_i H_i}{\sum W_i H_i} C_s \sum W_i$$

Donde:

Wi - es el peso de la masa i donde se aplica la fuerza sísmica.
Hi - es un coeficiente proporcional a la altura del nivel i.
Cs - es igual a c / Q.
Q - es un factor de comportamiento sísmico que depende de las características de la estructura.
c - es un coeficiente sísmico que depende de las características del suelo, los valores de este coeficiente dependen del tipo de zona, que para el Distrito Federal son las siguientes:

Zona	c	Tipo de terreno
I	0.16	Firme
II	0.32	De transición
III	0.40	Compresible

Tabla 1.1, Valores de c para el tipo de suelo.

1.6.2 VIENTO

El efecto que ocasiona la fuerza del viento en una estructura se puede considerar equivalente a una presión (empuje o succión) que actúa en forma estática en dirección perpendicular a la superficie expuesta. Su intensidad se puede determinar por la siguiente expresión:

$$P = C_p C_z K P_o$$

Donde:

Po - es la presión básica de diseño y se toma igual a 30 kg / m^2 para las estructuras comunes y 35 kg / m^2 para las estructuras clasificadas como del tipo A, basada en el artículo 174 del Título Sexto del Reglamento de Construcción del D.F.

K - es un factor correctivo por condiciones de exposición del predio en que se ubica la construcción, se determina según la sección A.

Cz - es un factor correctivo por la altura, sobre la superficie del terreno de la zona expuesta, se calcula según la sección A.

Cp - es el factor de presión que depende de la forma de construcción y de la posición de la superficie expuesta; sus valores se indican más adelante en la sección B.

SECCIÓN A - Los valores de K y Cz de la ecuación anterior dependen de las condiciones de exposición de la estructura en estudio; para su determinación se consideran tres zonas de ubicación:

A - zona de gran densidad de edificios altos; por lo menos la mitad de las edificaciones que se encuentran en un radio de 500 mts. alrededor de la estructura tienen altura superior a 20 mts.

B - zona típica urbana y suburbana; el sitio está rodeado predominantemente por construcciones de mediana y baja altura o por áreas arboladas y no se cumplen las condiciones de la zona A.

C - zona de terreno abierto; existen pocas o nulas obstrucciones al viento.

El factor Cz se toma igual a uno para alturas hasta de 10 mts. sobre el nivel del terreno y para alturas mayores es igual a:

$$C_z = \left(\frac{Z}{10}\right)^{2/a}$$

Donde Z es la altura del área expuesta sobre el nivel del terreno y el coeficiente "a" junto con el factor K se indican en la tabla siguiente según la zona de ubicación.

	ZONA		
	A	B	C
K	0.65	1	1.6
a	3.6	4.5	7.0

Tabla 1.2, Valores de K y a para zona de ubicación.

SECCIÓN B - Los factores de presión Cp se determinan según el tipo y forma de la construcción, de acuerdo con la clasificación siguiente:

	Cp
Pared de barlovento	0.8
Pared de sotavento	-0.5
Paredes laterales	-0.7
Techos planos	-0.7
Techos inclinados	-0.7
Techos inclinados, lado sotavento	-0.7
Techos inclinados, lado de barlovento	-0.8

Tabla 1.3, Factores de presión Cp.

TECNOLOGÍA ORIENTADA A OBJETOS (TOO)

2.1 INTRODUCCIÓN A LA TECNOLOGÍA ORIENTADA A OBJETOS (TOO)

2.1.1 PANORAMA DE LA TOO

Durante los años 60's y 70's, la programación estructurada marcó una revolución en la forma de realizar programas y sobre todo lenguajes. Como casi todas las revoluciones, fue buena en esencia pero se excedió en algunos términos. La programación estructurada se basa en el "divide y vencerás", y estructura los programas según operaciones más pequeñas que se dividen a su vez en suboperaciones más pequeñas todavía. El mayor logro de la programación estructurada es el concepto de procedimiento y la abstracción que ello supone. La estructuración estricta consiste en la transcripción a código fuente de un diagrama de flujo limpio (sin cruces entre las líneas de flujo). En muchos casos todo este conjunto de técnicas puede ser tediosa e inflexible y todavía más cuando los sistemas a realizar son muy grandes y complicados.

La tecnología orientada a objetos nace de la necesidad de crear formas alternativas para el desarrollo de complejos sistemas de una forma más eficiente y rápida. A medida que los sistemas de cómputo comenzaron a ser más grandes, paralelamente surgieron dificultades para la creación y mantenimiento de tales sistemas. Por ejemplo en décadas anteriores el software diseñado con lenguajes de tercera generación no excedía de 50 mil líneas de código lo que no presentaba necesidades de buscar nuevas técnicas y métodos de desarrollo de software, debido a que el diseño estructurado y los lenguajes estructurados resolvían el problema de una manera aceptable. Actualmente los requerimientos y las exigencias de los sistemas son mucho mayores, lo que ha ocasionado el surgimiento de la tecnología orientada a objetos que viene a resolver en gran medida los problemas encontrados en las técnicas antecesoras principalmente en el diseño estructurado.

En la actualidad esta tecnología está teniendo gran auge debido a la creciente demanda de software a más bajo costo, de rápido desarrollo y de gran confiabilidad. La tecnología orientada a objetos debido a sus características innovadoras permite aplicar una verdadera Ingeniería de Software para el desarrollo de sistemas con los requerimientos y las exigencias que se solicitan actualmente.

2.1.2 HISTORIA DE LA TOO

El surgimiento de la computación orientada a objetos en los 80's es uno de los pasos más significativos en la historia de la computación. Las ideas centrales de la computación orientada a objetos comienza a finales de los 60's con la introducción de SIMULA, un lenguaje de programación diseñado para escribir programas de simulación. SIMULA fue el primer lenguaje en mostrar el énfasis de que un programa hará lo que esté modelado, sin cambiar el paradigma de la transición del análisis al diseño y del diseño a la codificación. Las ideas se fueron desarrollando más con la introducción de la serie de lenguajes como SMALLTALK, comenzando con SMALLTALK-72 y culminado en SMALLTALK-80. En este tiempo muchas de las ideas detrás de la computación orientada a objetos fueron madurando, sin embargo el uso de las técnicas orientadas a objetos permanecieron restringidas a pequeños pero activos campos.

Fue hasta 1986 cuando el interés en la computación orientada a objetos comenzó a ampliarse significativamente. Todo se debió a dos conferencias en este campo: "Object Oriented Programming Workshop" en IBM y "the Ist Internacional Conference on Object-Oriented Programming Languages, Systems and Aplications" en Portland, Oregon. Ambas conferencias presentaron gran información en la materia.

Del interés de estos eventos, fue claro que muchos investigadores fueron encaminándose hacia la computación orientada a objetos para ayudar a resolver un amplio rango de problemas. Hay muchas razones para este fenómeno, quizá la razón más grande fue la crisis del software. Esta crisis se refiere al conjunto de problemas encontrados en el software. Estos problemas no estaban limitados al hecho de que los programas no funcionaban bien sino que también abarcaba los problemas asociados a como desarrollar el software de una manera eficaz, como mantener un volumen existente de software y como satisfacer la demanda creciente de este software. Los ingenieros y desarrolladores de sistemas se dieron cuenta que todos estos problemas podían corregirse y la clave estaba en darle un enfoque de ingeniería al desarrollo de software y que mejor con las técnicas y métodos de la TOO.

Otra importante influencia para el desarrollo de la TOO fue el rápido desarrollo de la tecnología de estaciones de trabajo. Las estaciones de trabajo modernas tienen la habilidad de soportar un ambiente de programación personal sofisticado y también proveen una interfaz de computación al usuario basada en gráficos. Para desarrollar aplicaciones en dichos sistemas se necesitaban herramientas para que la programación de ventanas, botones e iconos no fuera tan compleja y lenta, además que el programador no tuviera que comenzar desde cero con una nueva aplicación, por lo que la TOO ha sido adoptada por muchos sistemas con ambiente gráfico gracias a las características que presenta.

En la actualidad un gran porcentaje del software para computadoras personales se está desarrollando con la tecnología orientada a objetos y el resultado ha sido bastante satisfactorio tanto para las empresas involucradas en el desarrollo y la comercialización como para los usuarios.

2.1.3 EVOLUCIÓN DE LAS TÉCNICAS DE DESARROLLO DE SOFTWARE HACIA LA TOO

Las tecnologías de desarrollo de software han ido evolucionando desde sus inicios hasta la fecha y las tendencias indican que en el futuro continuará esta transformación, en seguida se muestran los diferentes modelos de programación que han existido y como han ido evolucionando de forma natural hacia el modelo orientado a objetos.

Se puede observar en la figura 2.1 como en la primera generación de lenguajes, los programas estaban divididos en pequeños subprogramas que compartían un área de datos global, si ocurría un error en una parte del programa podía repercutir en el resto del programa. Cuando las modificaciones son hechas en un sistema grande, es difícil de mantener la integridad del diseño original. Ejemplos de lenguajes de programación de esta generación fueron:

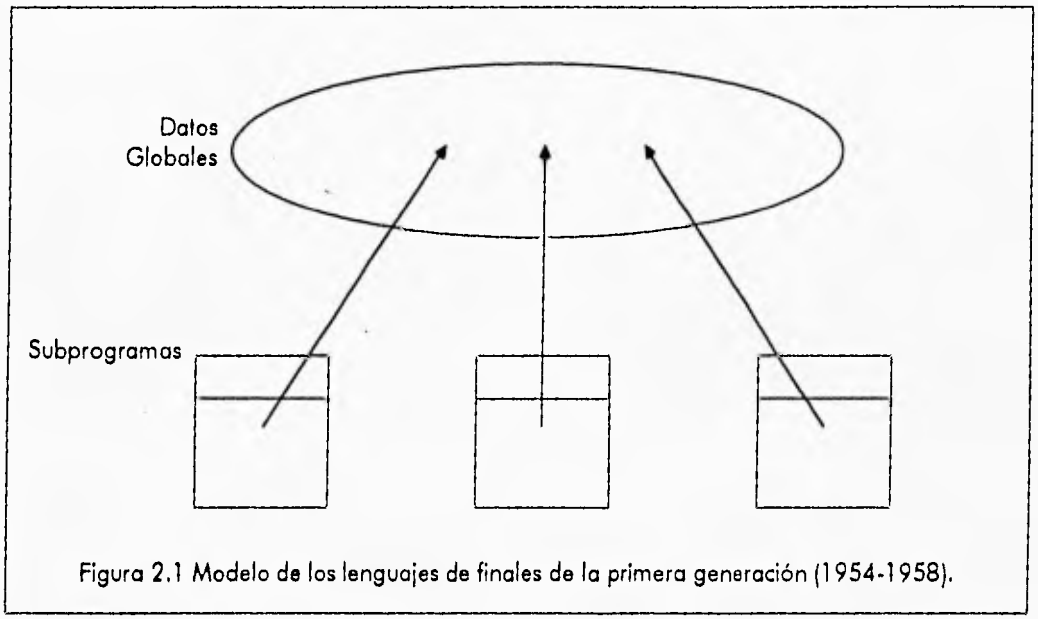


Figura 2.1 Modelo de los lenguajes de finales de la primera generación (1954-1958).

- Primera generación (1954-1958)

Lenguaje	Principales características
FORTRAN I	expresiones matemáticas
ALGOL 58	expresiones matemáticas

Cabe mencionar que la primera generación de lenguajes de programación se remonta a los días en que se codificaba en lenguaje máquina, el código máquina y el lenguaje ensamblador representaron los inicios de la primera generación de lenguajes acarreando todos los problemas que representaba programar de esta forma, lo que a su vez ocasionó que rápidamente surgieran los primeros lenguajes de alto nivel como el FORTRAN y el ALGOL. En la actualidad aunque ya han quedado obsoletos los lenguajes ensambladores, todavía en algunos casos de programación especial son muy útiles sus servicios principalmente para aumentar la eficiencia del código ejecutable y tener un completo control del hardware de la máquina.

Se pueden ver en la figura 2.2 los modelos de programación de la segunda generación. Se ve que los subprogramas ya contenían otros subprogramas y que se permitía el paso de parámetros entre ellos, pero seguían compartiendo una zona de datos global lo que ocasionaba los mismos errores de la generación anterior, aunque la idea importante fue el surgimiento del diseño estructurado permitiendo construir grandes sistemas usando subprogramas que a su vez contenían otros subprogramas dentro. Ejemplos de lenguajes de programación de esta generación son:

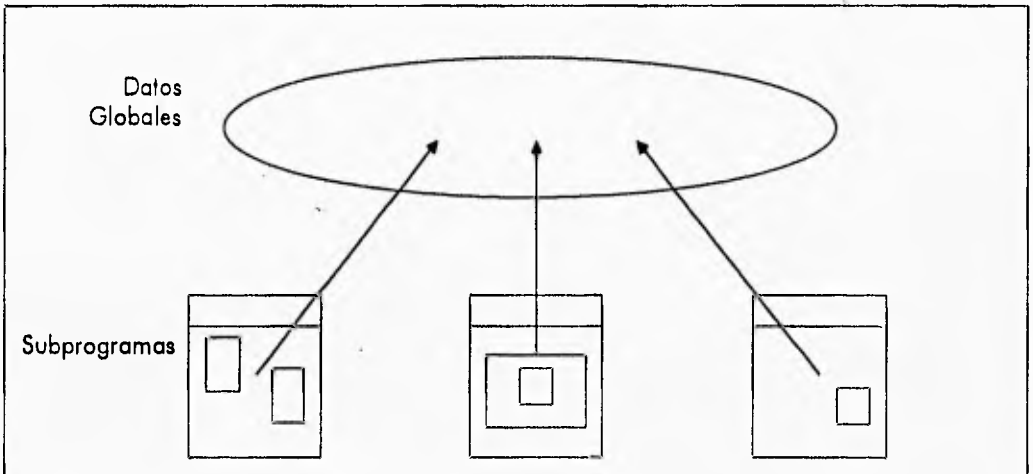


Figura 2.2 Modelo de programación de lenguajes de la segunda generación (1959-1962).

- Segunda generación (1959-1962)

Lenguaje	Principales características
FORTRAN II	subrutinas, compilación separada
ALGOL 60	estructuras de datos
COBOL	manejo de archivos
LISP	manejo de listas, apuntadores, recolección de basura

La importancia del surgimiento de esta generación de lenguajes no fue su amplio uso y desarrollo de sistemas con estos lenguajes, sino el legado que dejaron principalmente el ALGOL 60 que con su amplio repertorio de construcciones procedimentales y su tipificación de datos, para que apartir de ellos nacieran una gran variedad de lenguajes que hasta nuestros días todavía perduran, claro que con sus respectivos cambios y evoluciones que han sufrido cada uno de ellos.

En la figura 2.3 se puede apreciar el modelo de programación de los lenguajes de tercera generación, se observa que los datos continuaban siendo globales, pero hubo un gran avance; comenzaron a surgir grandes necesidades de software lo que se tradujo en grandes proyectos de programación y significó grandes equipos de desarrollo y la necesidad de desarrollar diferentes partes del mismo programa independientemente. La respuesta a esta necesidad fue la separación de módulos de compilación. La mayoría de los lenguajes de esta generación como PASCAL y C soportaban alguna forma de estructura modular, pero únicamente funcional, aún no existía la modularidad de datos. En esta generación surgen con gran fuerza los algoritmos y la programación estructurada. Ejemplos de lenguajes de programación que aparecieron en esta generación fueron:

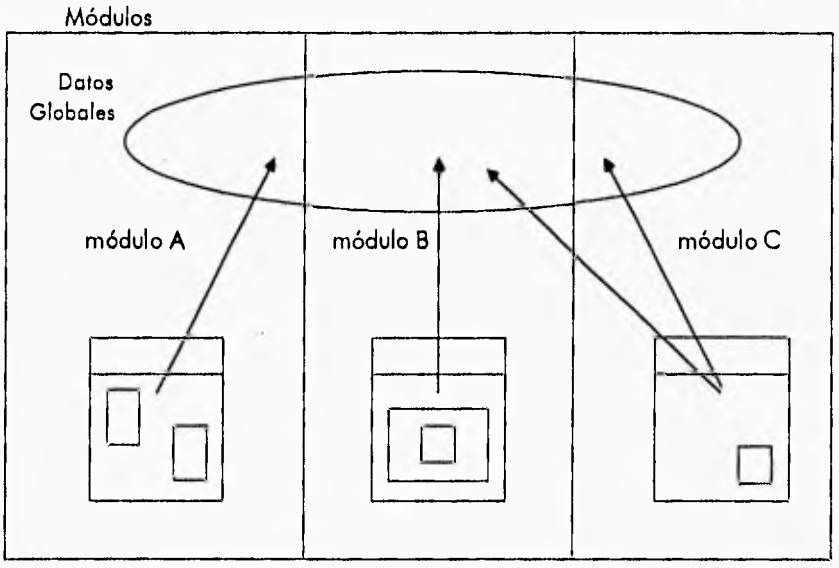


Figura 2.3 Modelo de programación de los lenguajes de la tercera generación (1963-?).

- Tercera generación (1963-?)

Lenguaje	Principales características
ALGOL 68	modificación del ALGOL 60
PASCAL	sucesor de ALGOL 60
MODULA-2	muy parecido a PASCAL
ADA	desarrollado para propósitos generales
C	desarrollado para programación de sistemas
SIMULA	clases y abstracción de datos

Los lenguajes de la tercera generación se caracterizaron por sus potentes posibilidades procedimentales y de estructuración de datos. Apartir de esta generación los lenguajes de programación se comenzaron a dividir en dos grandes categorías, los lenguajes especializados y los lenguajes de propósito general. Los lenguajes especializados que surgieron fueron diseñados para satisfacer requerimientos especiales y en consecuencia tenían una formulación y sintaxis únicas. Los lenguajes de propósito general han sido adoptados como lenguajes con potencial para un gran espectro de aplicaciones como por ejemplo para áreas de ciencias, ingeniería, comerciales y de sistemas.

En la Figura 2.4 se muestra como se ha llegado en la actualidad al desarrollo de sistemas mediante la TOO, se observa que el sistema está dividido en módulos independientes, que se pueden compilar por separado. Estos módulos a su vez están internamente constituidos por uno o varios objetos que a su vez están formados por datos y métodos que desarrollan ciertas

funciones dentro del objeto y modifican su comportamiento. Se observa que casi no existen los datos globales, debido a que la TOO permite no sólo la modularidad funcional sino también la modularidad de datos. Ejemplos de lenguajes orientados a objetos son: C++, SMALLTALK, EIFFEL, SIMULA, etc.

Cabe mencionar que actualmente existe una generación de lenguajes de programación llamados no procedurales o de 4 generación (4GL) que nacieron a finales de los 70´s con la intención de elevar los niveles de abstracción existentes en los lenguajes procedurales o de tercera generación, pero elevando los requerimientos de equipo de cómputo y perdiendo eficiencia en el sistema resultante.

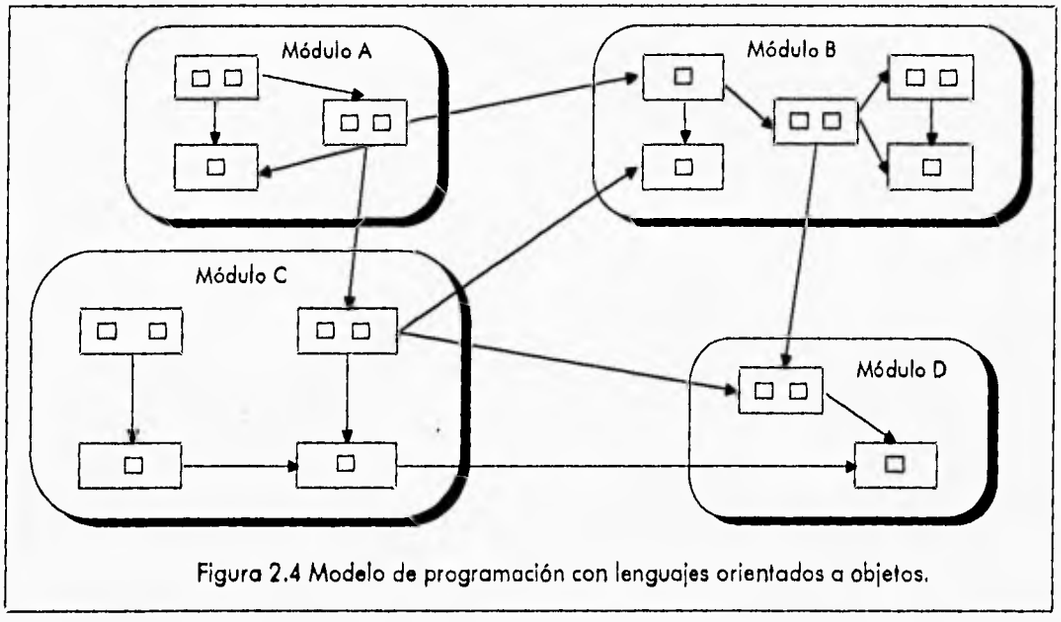


Figura 2.4 Modelo de programación con lenguajes orientados a objetos.

2.2 CARACTERÍSTICAS DE LA TECNOLOGÍA ORIENTADA A OBJETOS (TOO)

La tecnología orientada a objetos reúne cuatro características básicas en el desarrollo de sistemas: abstracción, encapsulación, herencia y modularidad, la ausencia de alguna de estas cuatro características limitaría su potencialidad. A continuación se explica cada una de estas características así como los conceptos que engloban cada una ellas.

2.2.1 ABSTRACCIÓN

Es la presentación de las características esenciales de algo sin incluir detalles innecesarios. Una abstracción se ocupa de la vista exterior de un objeto, y sirve para separar su comportamiento esencial de su implementación. Un sistema en general puede tener diferentes

niveles de abstracción, estar formado por varias capas, cada persona puede elegir su propio nivel de abstracción de acuerdo a sus necesidades.

Un buen ejemplo de abstracción con varios niveles es una computadora, la cual está compuesta por: una unidad de proceso central (CPU), un monitor, un teclado, y alguna forma de almacenamiento secundario, usualmente floppys o discos duros. Cada una de estas partes es una abstracción que puede también descomponerse en otros componentes. Por ejemplo: un CPU típicamente está compuesto de memoria primaria, una unidad aritmética lógica (ALU), y un bus por donde los dispositivos periféricos son atendidos. Cada una de estas partes puede a su vez ser descompuesta: una ALU puede estar dividida en registros y lógica de control aleatoria, las cuales están constituidas por elementos más primitivos, tales como compuertas NAND, inversores, etc.

La mayoría de los usuarios de un equipo de cómputo sólo les interesa o necesitan saber su comportamiento externo, el manejo de información. No tienen que conocer como funcionan los componentes internos. Sin embargo, si se tiene un problema en el reloj de la computadora, un usuario debe bajar su nivel de abstracción hasta la arquitectura física de la máquina.

La abstracción es importante para que los aspectos internos, no interfieran u obstaculicen el entendimiento del comportamiento general de un sistema.

2.2.2 ENCAPSULACIÓN

La encapsulación es el proceso de esconder todos los detalles de implementación de una abstracción. Con la encapsulación sólo se deja ver el comportamiento exterior de un objeto, no se tiene acceso a la implementación de tal comportamiento, es decir a sus datos y funciones.

La abstracción y la encapsulación son conceptos complementarios: la abstracción se encarga de la definición del comportamiento observable de un objeto, mientras que la encapsulación se ocupa de ocultar la implementación que da lugar a ese comportamiento.

Con la encapsulación se logra una mayor seguridad e integridad de los sistemas, ya que al no tener acceso a las estructuras internas de los objetos, su valor no se puede modificar involuntariamente por otros objetos. Además como los objetos no dependen de la estructura interna de otros objetos, son libres de cambiar el nombre de sus variables, el tipo de información que contienen, la cantidad de espacio que toman para almacenarse en memoria.

Con la encapsulación un objeto sólo necesita conocer como se puede comunicar con otro. Un ejemplo de encapsulación es un cajero automático: el cajero tiene diversas funciones, entre ellas sacar dinero y meter dinero. El usuario normal sólo conoce el comportamiento general del cajero. La caja metálica protege al cajero de cualquier acceso de una persona no calificada. El usuario no conoce la cantidad de dinero que tiene el cajero, no conoce su funcionamiento interno. La única forma de comunicación con el cajero es mediante la pantalla y botones que presenta al usuario.

2.2.3 MODULARIDAD

La modularidad fue una de las primeras evoluciones que los sistemas de cómputo tuvieron, ya que se divide un sistema muy grande en partes más pequeñas, que pueden ser creadas de forma independiente y que combinadas formen un sistema completo.

Los módulos representan una partición a nivel físico, separando el código en diferentes archivos que pueden ser compilados separadamente. Cada módulo puede ser desarrollado por una persona diferente, de esta forma los sistemas pueden desarrollarse más fácil y rápidamente ya que el programador sólo se encarga de una pequeña vista del sistema general.

Una característica que deben presentar los módulos es que puedan adaptarse fácilmente a sistemas para los que no fueron originalmente diseñados. Con lo anterior se puede lograr una mayor reusabilidad de código.

Otra característica importante de la modularidad es la protección, ya que las condiciones de error o excepción permanecen confinadas al módulo en el cual ocurren o se propaga sólo a los módulos más relacionados.

Para tener una buena modularidad se debe tener una interfaz pequeña y lo más simple posible, ésto es, cada módulo debe comunicarse con otros módulos lo menos posible y pasar la menor cantidad de datos.

A continuación se ve un ejemplo de un programa modular. Existen tres módulos cada uno de los cuales tiene definidos objetos que desarrollan una actividad específica, se tiene el módulo de Aplicación que se encarga mediante un menú de llamar a cada uno de los módulos del programa, en este caso el objeto "Archivo" se encarga de llamar al objeto "Principal" del módulo "Archivo" y el objeto "Edición" se encargará de llamar al objeto "Principal" del módulo "Edición" que a su vez llama a cada uno de los objetos de los módulos a los que pertenezcan para desarrollar funciones específicas del módulo correspondiente.

2.2.4 HERENCIA

La herencia es un mecanismo mediante el cual las clases (subclases) pueden hacer uso de variables y métodos definidos en otras clases (clases bases). La herencia se ha usado desde siempre, en forma de clasificación, un simple esquema de este tipo de clasificación es el reino animal, en el que existe todo un mecanismo de herencia.

En la figura 2.6 se observa un tipo de clasificación que nos ha ayudado para ir de conceptos generales a conceptos específicos. Un mamífero tiene o hereda todas las características de los vertebrados, sin embargo los mamíferos tienen características que los distinguen de otros vertebrados, a su vez los felinos tienen características particulares que los distinguen de otros mamíferos.

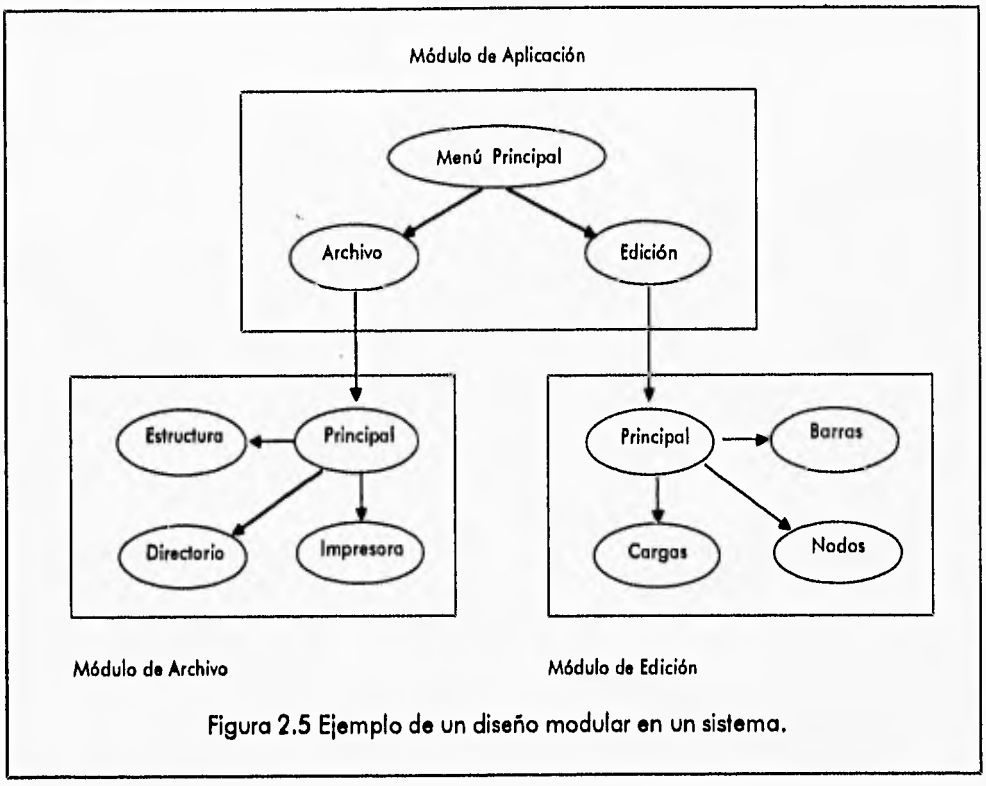


Figura 2.5 Ejemplo de un diseño modular en un sistema.

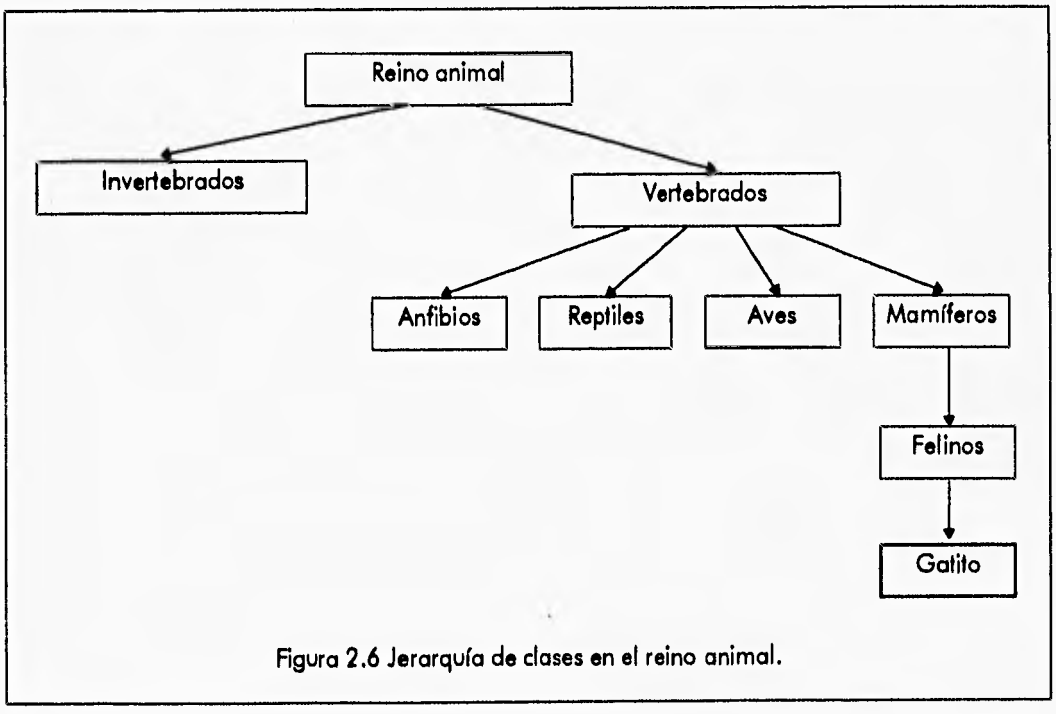


Figura 2.6 Jerarquía de clases en el reino animal.

Se puede usar el mismo mecanismo mencionado anteriormente para la creación de sistemas diseñando las abstracciones de tal forma que existan clases que encapsulen comportamientos generales y de ahí definir clases que presentan características especiales. Lo anterior es muy importante ya que en ocasiones cuando se crea una nueva clase se puede hacer un esfuerzo innecesario ya que pueden existir clases similares, resultando por lo tanto atractiva la idea de construir nuevas clases en base a otras ya establecidas. La nueva clase puede heredar el comportamiento de otras o modificarlo, de acuerdo al significado que deba adquirir.

La herencia es una de las características más importantes de la TOO, ya que con ella se logra una mayor reusabilidad de código, una mayor rapidez en la elaboración de sistemas y menor número de errores, ya que se construye en base a componentes probados con anterioridad. Aunque hay que aclarar que lo anterior sólo se logra si se diseña una buena jerarquía de clases lo más flexible y genérica posible, para que se pueda adaptar a la unión de nuevas clases o posibles cambios en su estructura.

2.2.5 OBJETOS

Un objeto es una colección de datos y funciones, diseñado para emular una entidad física o abstracta. Un objeto tiene un estado, un comportamiento, y una identidad. El estado de un objeto abarca todas sus propiedades (usualmente estáticas) y los valores actuales de cada una de sus propiedades (usualmente dinámicas).

Una propiedad es una característica distintiva o inherente, que hace al objeto único, por ejemplo una propiedad esencial de un elevador es que está construido para viajar hacia arriba o hacia abajo pero no horizontalmente. Las propiedades son usualmente estáticas, porque tales atributos son incambiables y son fundamentales para la naturaleza del objeto. Todas las propiedades tienen algún valor; el valor puede ser una simple cantidad o puede denotar a otro objeto. Este valor puede cambiar a lo largo de la vida del objeto por lo tanto es dinámico.

El hecho que cada objeto tiene un estado, implica que cada objeto toma alguna cantidad de espacio, ser una palabra física o memoria en la computadora. El comportamiento de un objeto es el conjunto de actividades que puede llevar a cabo. Tal comportamiento es influenciado por su historia y por la intervención de otros objetos.

La identidad es la propiedad de un objeto lo cual lo distingue de otros objetos. Cada objeto tiene un identificador único para todo su tiempo de vida. Su tiempo de vida se extiende desde que es creado (consumiendo espacio) hasta que es destruido y su espacio es reclamado.

Un objeto usa los conceptos de abstracción y encapsulación ya que se esconde su implementación; externamente sólo se conoce lo que puede hacer el objeto. Los objetos modelan entidades del mundo real, son construcciones genéricas, lo que los hace reusables para subsecuentes proyectos, aún si los objetivos del nuevo proyecto son muy diferentes.

Los objetos pueden estar formados a su vez por otros objetos, originando con ello posibles objetos de gran complejidad, por ejemplo se puede crear un objeto "coche" que está formado por objetos tipo "llanta", tipo "puerta", tipo "motor" etc.

2.2.6 CLASES

Una clase es un patrón por medio del cual los objetos pueden ser creados. Contiene una definición completa de las estructuras de datos, algoritmos internos y la vista externa que pueden tener los objetos que pertenecen a esa clase. La clase debe especificar explícitamente que funciones forman la interfaz con el exterior y que funciones son usadas sólo por la clase o por sus clases derivadas, para controlar de alguna forma el acceso a ella y su integridad, por lo tanto un dato o un método puede caer en una de estas categorías:

- público - una declaración que es accesible a todo el sistema
- protegido - una declaración que es accesible sólo en la clase misma y las subclases
- privado - una declaración que es accesible sólo a la clase misma

Las clases y objetos son dos conceptos separados pero íntimamente relacionados. Específicamente cada objeto es la instancia de alguna clase, y cada clase tiene cero o más instancias. La diferencia radical es que un objeto es una entidad concreta que existe en tiempo y espacio. Una clase representa sólo una abstracción, la esencia de un objeto. Por ejemplo se puede hablar de la clase mamíferos, la cual representa las características comunes de todos los mamíferos. Para identificar un mamífero particular de esta clase, se debe hablar de este mamífero o de aquel mamífero. Además los objetos de una misma clase son diferentes ya que cada uno tiene su propio estado e identidad.

Las clases se pueden relacionar con otras clases mediante la herencia, usando instancias de una clase en otra clase, o mediante clases abstractas. Con la herencia una clase (clase derivada o subclase) puede heredar las estructuras y comportamiento de otra (superclase o clase base). La herencia puede darse en forma simple o múltiple, en el primer caso una subclase hereda de una sola superclase, en el segundo case puede heredar de dos o más superclases.

La relación entre clases mediante clases abstractas, se desarrolla cuando se define una clase especial que será la clase base para varias clases derivadas, pero con dicha clase no se pueden crear objetos, este tipo de clases sirve para declarar un comportamiento común que tendrán sus clases derivas. Pero para tal clase no se han definido los métodos.

Las clases pueden modelar aspectos del mundo real, por el ejemplo se puede crear una clase "persona" que define todas las características de las personas: edad, peso, dirección, sexo y sus acciones principales: caminar, bailar, comer, dormir. Una vez definida la clase se pueden crear objetos de esa clase por ejemplo: persona "Juan", persona "Lola".

Si se quiere hacer una diferencia más tangible entre los hombres y mujeres, se pueden crear dos clases separadas: "Hombre" y "Mujer" y en cada clase definir características específicas de hombres y mujeres. Pero ¿ qué se hace con las características comunes ?, la respuesta es usar la herencia, se define una clase que tenga dichas características comunes y de ella se heredan las clases especializadas "Hombre" y "Mujer". Del ejemplo anterior, se deja ver que con el uso de la herencia, no hay que redefinir código, lo que trae como consecuencia que no exista redundancia en los programas y evitar así errores que se generan al crear nuevos módulos de programación.

2.3 DESARROLLO DE SISTEMAS CON LA TECNOLOGÍA ORIENTADA A OBJETOS (TOO)

En el desarrollo de un sistema con la tecnología orientada a objetos existen tres fases bien definidas para la implementación del programa: Análisis, Diseño y Codificación. De forma general se pueden definir así:

- Análisis: Definir claramente el problema a resolver
- Diseño: Crear una estructura para resolver el problema
- Codificación: Escribir y probar el código

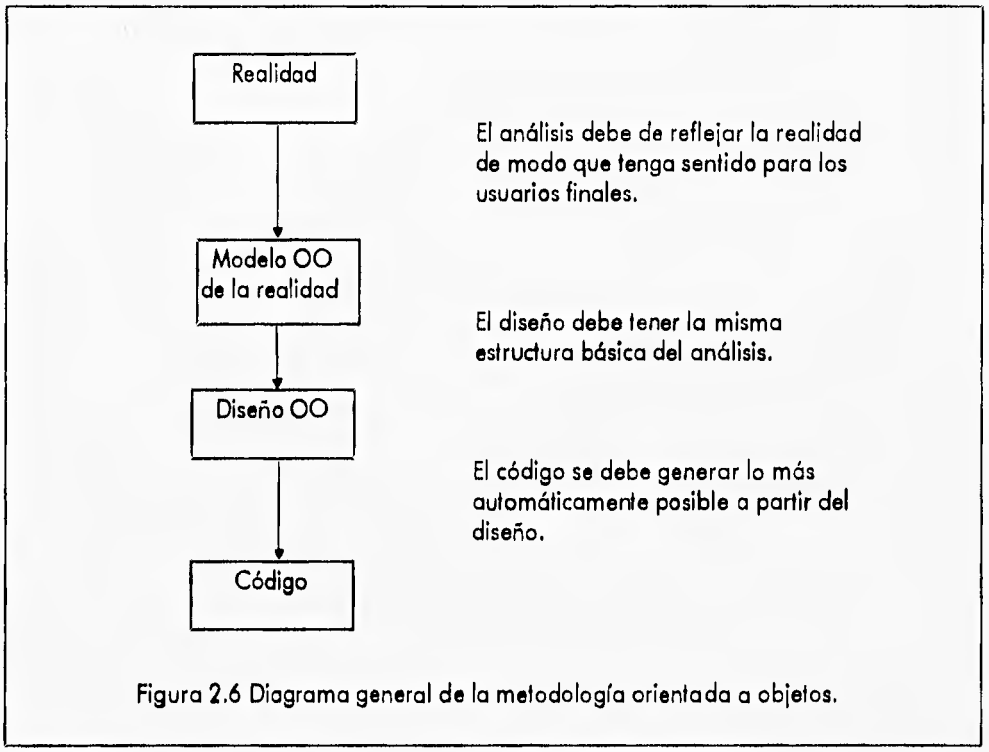


Figura 2.6 Diagrama general de la metodología orientada a objetos.

2.3.1 ANÁLISIS ORIENTADO A OBJETOS

El análisis es la descomposición de un problema en sus partes componentes. En el análisis orientado a objetos se pretende tener un modelo de la realidad identificando los objetos y clases claves que componen al sistema, así como definir el comportamiento global que se desea del sistema a construir.

La diferencia entre el análisis orientado a objetos y el análisis tradicional es que en el análisis tradicional se analiza por un lado el comportamiento y los datos por separado, y por otro lado, el sistema se descompone en funciones donde los datos son pasados a dichas funciones. En el

análisis orientado a objetos se combinan datos y funciones al encapsularlos en los objetos y el sistema se forma a su vez de estos objetos.

Las actividades principales a desarrollar en la etapa de análisis son:

- Identificar los objetos y clases del sistema
- Describir como se relacionan los objetos
- Definir la clasificación de los objetos y las clases

La identificación de las clases y objetos puede ser llevada a cabo en sus fases de inicio del sistema, de forma muy fácil y natural, ya que el propio contexto del problema a solucionar puede dar la pauta para encontrar los objetos principales del sistema.

En la etapa de diseño se puede enriquecer la lista de objetos al descubrir nuevos objetos, sobre todo los objetos auxiliares, que no fueron percibidos en la primera fase. Los desarrolladores pueden estar pasando del análisis al diseño y del diseño al análisis hasta lograr un diseño que satisfaga los requerimientos iniciales en mayor grado.

Al observar el comportamiento real del sistema, se puede hacer la descripción de la relación que tienen los objetos y que papel representa cada uno dentro del proyecto.

También es necesario comenzar a definir la jerarquía de herencia que se va a tener, que clases tienen características comunes y cuales son las clases más generales.

2.3.2 DISEÑO ORIENTADO A OBJETOS

El diseño orientado a objetos se encarga de definir la parte interna de cada objeto, así como su representación física. Un buen diseño de objetos y clases se alcanza si tiene por lo menos las siguientes características: inicialmente se debe buscar que las abstracciones sean lo menos dependientes de otras, aquí el diseñador debe pensar muy bien que es lo mejor para la aplicación que está desarrollando ya que la herencia crea una mayor dependencia entre abstracciones. También se debe buscar que exista cierta coherencia al momento de agrupar los objetos. Por ejemplo, considérese una clase agrupando las abstracciones de perros y pelotas, las cuales tienen comportamientos que no están relacionados. La forma más deseable de cohesión es una cohesión funcional, en la que los elementos de una clase o módulo trabajan entre ellos para proporcionar un comportamiento bien definido. Ésto es, la clase "Perro" es funcionalmente cohesiva si únicamente abarca el comportamiento de un perro. Otra característica que deben tener las abstracciones es que deben definir por los menos el comportamiento mínimo necesario para el buen funcionamiento de esa abstracción. Si se construye una pila sin una función que saque valores de ella, no se podrá usar normalmente. Lo ideal es que se definan todos los aspectos necesarios de una abstracción.

Para llevar a cabo un buen análisis y diseño es necesario manejar ciertas herramientas que auxilien para la mayor comprensión del desarrollo del sistema, sobre todo cuando el sistema es

grande y desarrollado por un equipo de trabajo, por lo tanto, es necesario además tener una visión clara y estándar del sistema.

Se debe tener una vista general del sistema, donde se identifiquen las abstracciones claves que forman al sistema y una vista específica de cada abstracción en la que se identifican las operaciones y métodos que utilizarán las instancias de las clases. Para la vista general se usan diagramas de clases. Para la vista específica se usan diagramas de funciones.

Los diagramas anteriores son estáticos. Sin embargo, los eventos son dinámicos en la mayoría de los sistemas. Describir un evento dinámico en un medio estático tal como una hoja de papel es un problema difícil. En desarrollos orientados a objetos, se expresa la dinámica de un problema a través de diagramas de transición de estado de los objetos.

DIAGRAMA DE CLASES

Un diagrama de clases es usado para mostrar la existencia de las clases y su relación en la vista lógica de un sistema. Un simple diagrama de clases representa toda o parte de la estructura de clases de un sistema. Los dos elementos esenciales de una diagrama de clases son las clases y sus relacianes básicas.

Las clases se pueden representar como rectángulos, en el interior del rectángulo se pone el nombre de la clase y las relaciones existentes como flechas que van de una clase a otra y varían de acuerdo al tipo de relación que se establece entre ellas. Para una relación tipo herencia, la línea es trazada en forma discontinua, para una relación de uso la línea es trazada en forma continua.

Por ejemplo, la figura 2.7 muestra un diagrama de clases, donde se tiene una clase base llamada "Menú" en las que se pueden definir estructuras de datos y métodos que son comunes a las dos clases derivadas llamadas "Submenú" y "Menú Principal". Por otro lado se observa que "Menú" hace uso de las clases "Ratón" y "Control".

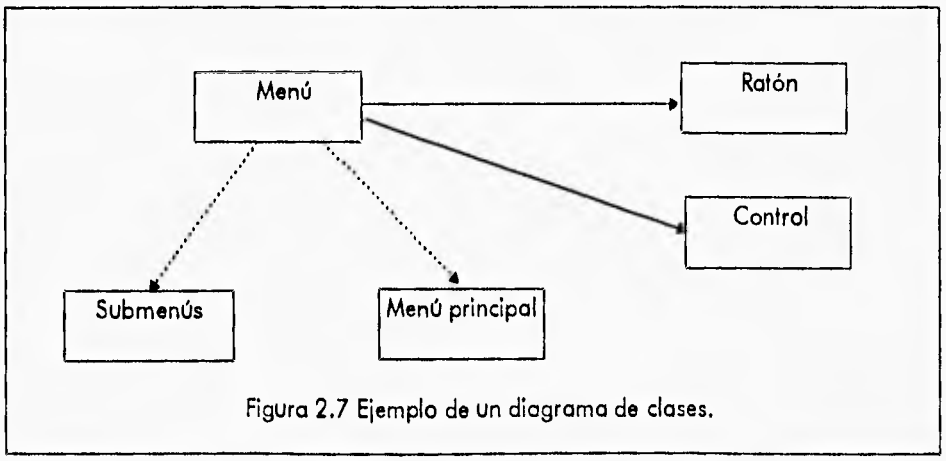


Figura 2.7 Ejemplo de un diagrama de clases.

DIAGRAMAS DE FUNCIONES

Los diagramas de clases ayudan a una fácil comprensión del funcionamiento del sistema, pero no son suficientes, por lo tanto, se debe incluir una especificación documentada más detallada con las características de la clase. Este documento debería incluir el nombre de la clase, las funciones que contiene especificando si son propias de la clase, o si son heredadas de otras clases. Las funciones se identifican con un rectángulo con bordes redondeados y para el caso de funciones heredadas se representan con líneas discontinuas, indicando las relaciones que existen entre cada una de ellas dentro de la clase.

En la figura 2.8 se observa un diagrama funcional de la clase LEÓN, en la que tiene 5 funciones de las cuales, dos funciones han sido heredadas de la clase padre MAMÍFEROS como son CORRE y MUEVE BOCA, también se muestra la relación que existe entre cada una de ellas.

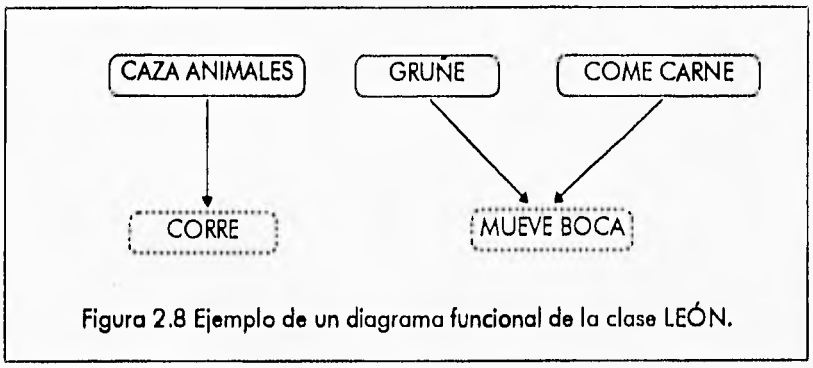


Figura 2.8 Ejemplo de un diagrama funcional de la clase LEÓN.

DIAGRAMA DE TRANSICIÓN DE ESTADO

El diagrama de clases no nos dice como las instancias individuales de las clases se comportan dinámicamente. El diagrama de transición de estado se usa para mostrar los eventos que causan una transición de un estado a otro y las acciones que resultan de un cambio de estado. En este tipo de diagrama se representa una vista del modelo dinámico de una sola clase o de todo el sistema. Las partes esenciales de un diagrama de transición de estado del objeto son los estados y las transiciones que sufre el objeto.

El estado es la fase de operación en la que se puede encontrar un objeto en un momento específico. Un evento es alguna ocurrencia que puede causar que el estado de un sistema cambie. Este cambio de estado es llamado una transición de estado.

Un estado se puede representar como un círculo dentro del cual aparece el nombre de la operación que se está ejecutando en ese momento, cada estado se une a otro mediante una flecha discontinua que va dirigida a uno de los probables estados siguientes. Por ejemplo se puede crear un diagrama de transición de estados de un objeto tipo Menú. Las operaciones que se pueden hacer con un "Menú" son crearlo, visualizarlo, obtener datos de entrada y borrarlo.

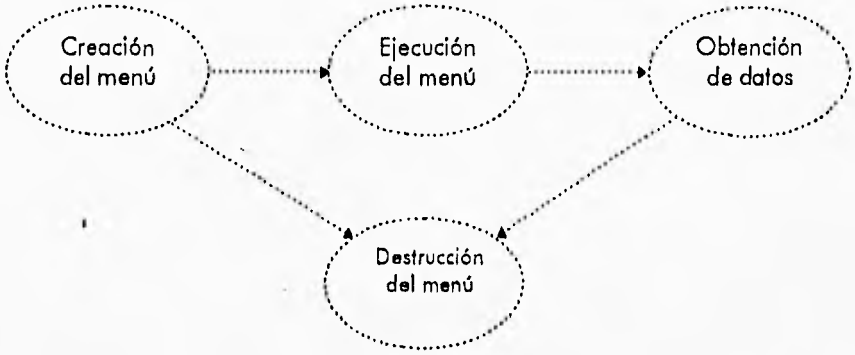


Figura 2.9 Ejemplo de un diagrama de transición de estado de un objeto.

2.3.3 LA PROGRAMACIÓN ORIENTADA A OBJETOS

La programación orientada a objetos es un método de implementación en la cual los programas son organizados como una colección cooperativa de objetos, cada uno de éstos representa una instancia de alguna clase y tales clases son miembros de una jerarquía de clases unificadas vía relación de herencia. De esta definición se puede establecer que un lenguaje es orientado a objetos sólo si tienen estas tres características:

- Hay una encapsulación de los datos y métodos en los objetos. Los objetos son los bloques fundamentales de construcción, no los algoritmos.
- Los objetos tienen un tipo asociado llamado clase. Todos los objetos con características comunes pueden ser agrupados en clases comunes.
- Las clases se relacionan con otras mediante una jerarquía de herencia. Las clases pueden heredar comportamiento de otras clases, llamadas superclases.

La programación orientada a objetos es más natural que la estructurada porque permite organizar la información de forma más familiar ya que se piensa a nivel del mundo real del sistema, no a nivel de un lenguaje de programación tradicional. Este nivel de abstracción es posible ya que se permite definir nuevos tipos de datos que describen objetos reales.

Los lenguajes tradicionales ya contaban en cierta medida con mecanismos para la manipulación de tipos de datos abstractos, pero de forma muy limitada. Sólo con la programación orientada a objetos se pueden crear tipos de datos abstractos y el lenguaje los usará como si fueran propios del lenguaje.

Existen lenguajes de programación que aunque usen objetos no son necesariamente orientados a objetos, por lo que es importante hacer su clasificación:

- Basado en objetos: definen objetos y realizan operaciones sobre ellos, ejemplo ADA.
- Basado en clases: están basados en objetos y permiten definir clases, ejemplo CLU.
- Orientados a objetos: están basados en objetos y además permiten definir clases y hacer uso de la herencia, ejemplo SMALLTALK y C++.

Los lenguajes orientados a objetos se puede dividir a su vez como puros o híbridos. Los lenguajes orientados a objetos puros, como el SMALLTALK o EIFFEL, han sido creados desde su inicio completamente con la TOO. Los lenguajes orientados a objetos híbridos, como C++, son una extensión de lenguajes tradicionales, es decir tales lenguajes se han enriquecido con la TOO.

Otra característica de los lenguajes orientados a objetos es el polimorfismo; como ya se ha mencionado anteriormente, los objetos se comunican entre ellos a través del paso de mensajes. El polimorfismo es la capacidad que tienen los lenguajes de que un mismo mensaje pueda ser interpretado por los objetos de diferentes formas, es decir se pueden tener comportamientos denominados con un nombre genérico, sin embargo cada clase puede implementar dicho comportamiento de diferente manera. Por ejemplo se puede tener una clase Figuras, que sea la clase base para subclases como Cuadrado, Rectángulo, Triángulo, etc., se puede declarar una función Área en la clase Figuras, pero como cada figura calcula su área de diferente forma, se deja la implementación específica a cada subclase. El polimorfismo es importante, ya que se pueden crear clases que establezcan un comportamiento muy general y dejar los detalles de implementación a posteriores clases derivadas a desarrollar.

La elección del lenguaje, se determina de acuerdo a las necesidades de determinado proyecto, no se puede afirmar que un lenguaje es mejor que otro en forma global, ya que depende de la aplicación a desarrollar.

2.4 BENEFICIOS DE LA TECNOLOGÍA ORIENTADA A OBJETOS (TOO)

La calidad de un sistema de software se evalúa de acuerdo a diversos criterios. La TOO en la actualidad es el método más adecuado para cubrir los requisitos que a continuación se enuncian:

- **Exactitud** - los programas deben de conocer sus especificaciones correctamente para que el producto obtenido cumpla con las mismas características.
- **Confiabilidad o fiabilidad** - los programas son fiables si funcionan bien aún en condiciones anormales, esta condición es conocida como robustez o tolerancia a fallas.
- **Mantenibilidad** - los programas deben ser fáciles de desarrollar, mantener y extender, así como también deben de ser fáciles de actualizar sin tener que realizar trabajo extra.
- **Reusabilidad** - los programas deben de ser construidos con módulos reusables, lo que permite un ahorro de tiempo y dinero, así como una mayor confiabilidad.
- **Interoperatividad** - los programas deben ser con facilidad compatibles con otros sistemas. Deben ser "sistemas abiertos".

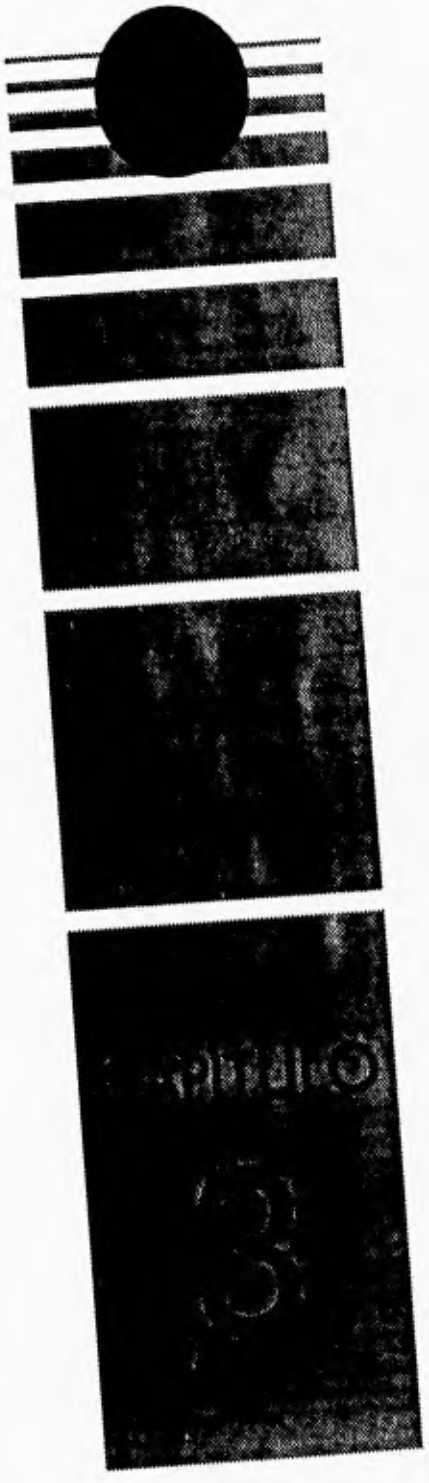
- **Eficiencia** - los programas deben de ser lo más eficientes posible, en cuanto a velocidad y rendimiento, así mismo deben de cubrir todas las necesidades para las cuales fueron creados.
- **Portabilidad** - los programas deben ser portables a través de las diversas plataformas existentes, así como de las nuevas plataformas que vienen llegando.
- **Integridad** - los sistemas necesitan protección contra actualizaciones inconsistentes, corrupción de datos y acceso no permitido a funciones. Los sistemas deben de poner de manifiesto la independencia entre módulos para conservar la integridad.
- **Amigabilidad** - los sistemas deben ser fáciles de desarrollar, de mantener, así mismo, deben de ser fáciles de usar por la mayoría de los usuarios y deben ser interactivos.
- **Descriptible** - debe ser posible crear y mantener documentación de un sistema de la manera más fácil y rápida posible.

Recientemente ha venido surgiendo la TOO que proporciona a los diseñadores y programadores diversos beneficios en el desarrollo de software y cubre más ampliamente que cualquier técnica conocida todas las necesidades y requerimientos antes mencionados.

He aquí porque la TOO ha ido ganando popularidad en la implementación de software de cualquier nivel.

- Para la mayoría de las personas, la forma de pensar orientada a objetos es más natural que las técnicas del análisis y diseño estructurado. Después de todo el mundo está formado por objetos.
- Los sistemas suelen construirse a partir de objetos ya existentes. Esto lleva a un alto grado de reutilización del código, un ahorro de dinero, un menor tiempo de desarrollo y una mayor confiabilidad del sistema.
- La creación de sistemas con un funcionamiento correcto es más fácil con la TOO debido a que las clases están autocontenidas y divididas en métodos que se pueden construir, depurar y modificar con relativa facilidad.
- La TOO a diferencia de otros métodos da como resultado un diseño que interconexiona las estructuras de datos y las operaciones o métodos de procesamiento, de forma que modulariza la información y el procesamiento, en vez de sólo modularizar el procesamiento, como sucede en el análisis y diseño estructurado.
- La TOO modela el problema a resolver de una manera más cercana a la realidad que lo que se logra con el análisis tradicional. El análisis orientado a objetos permite una transición de manera directa hacia el diseño y la codificación del sistema. Por el contrario en las técnicas convencionales, el paradigma cambia cuando se pasa del análisis al diseño y del diseño a la programación. Con las técnicas orientadas a objetos, el análisis, diseño y codificación utilizan el mismo paradigma y lo particularizan de manera sucesiva.
- La naturaleza única de la TOO está ligada a su habilidad para construir y desarrollar software, basándose en cuatro conceptos importantes de diseño del software: abstracción, ocultamiento de la información, herencia y modularidad. Todos los métodos de diseño buscan la creación de software que presente estas cuatro características fundamentales, pero sólo la TOO da un mecanismo que facilita a los diseñadores y programadores adquirir estas características sin ninguna complejidad.

DESARROLLO ORIENTADO A OBJETOS DEL SISTEMA CONFIA



3.1 ANÁLISIS ORIENTADO A OBJETOS DEL SISTEMA CONFIA

El objetivo del sistema CONFIA es crear un ambiente gráfico, lo más amigable posible para el usuario, para realizar análisis de confiabilidad de sistemas estructurales; donde se pueda modelar la estructura, dándole sus propiedades a través de un sistema de menús y poder realizar análisis de esfuerzos y confiabilidad. Este sistema debe ser además capaz de visualizar las estructuras captadas, así como los resultados de los análisis realizados, a través de imágenes que muestren por ejemplo la deformación que sufre la estructura, producto de la carga a la que está siendo sometida, a través de pequeños menús que muestren las fuerzas internas que actúan en cada uno de los elementos de la estructura y los índices de confiabilidad asociados a estas fuerzas.

El método que se seguía para realizar el análisis de una estructura era:

- Crear un archivo de datos, donde se tenían que dar las coordenadas de cada nodo, los nodos extremos de una barra, los propiedades de los materiales de las barras, las condiciones de carga, las cargas aleatorias. Todo lo anterior con un formato muy estricto y un poco engorroso, sobre todo cuando se deseaba analizar estructuras muy grandes.
- Ejecutar un programa que realizaba todo el cálculo del análisis, asignando como archivo de entrada, el descrito en el punto anterior.
- Interpretar los resultados del análisis numéricamente, lo que era complejo especialmente para gente que no está tan familiarizada con los análisis de confiabilidad.
- Si se deseaba cambiar algunas propiedades de la estructura, se tenía que volver a editar el archivo de datos, volver a ejecutar el programa y nuevamente interpretar el archivo de resultados. Todo lo anterior representaba una gran pérdida de tiempo y terminaba en el desánimo, lo que traía con sigo que se perdiera el objetivo que realmente se pretendía con el programa.

Originalmente todo el cálculo para el análisis estructural, estaba desarrollado con el lenguaje de programación FORTRAN, lenguaje muy usado todavía en el campo científico e ingenieril, pero debido a todas las limitaciones que este lenguaje presenta, así como la falta de una ingeniería de software en el programa desarrollado fue necesario su cambio a otro lenguaje de programación, en este caso el C++.

Con el sistema el usuario puede ver gráficamente la deformación que sufre determinada estructura de acuerdo a ciertas condiciones. Aprovechando la idea de que una imagen dice más que mil palabras. También se pueden estar cambiando las propiedades que se deseen, en los menús correspondientes y con sólo elegir la opción de Analizar dentro del menú indicado, el usuario puede visualizar los cambios de una forma rápida y amigable, sin tener que editar forzosamente archivos y ejecutar programas externos.

El lenguaje seleccionado para el desarrollo del sistema es el C++. Este lenguaje es una extensión del C, heredando las virtudes principales de este lenguaje como eficiencia y flexibilidad entre otras. Además, es un lenguaje orientado a objetos incrementando su potencialidad para el desarrollo de sistemas.

El sistema tiene dos partes principales: la interfaz gráfica y el cálculo numérico. Con C++ es posible crear una interfaz gráfica muy rica, debido a la flexibilidad que presenta este lenguaje. Cuenta con un amplio grupo de funciones para el manejo de gráficos que puede ser enriquecido con la construcción de tipos de datos abstractos (clases), lo que permite pasar con gran transparencia de un diseño orientados a objetos a la codificación del programa.

Como se dijo anteriormente el cálculo numérico estaba desarrollado en FORTRAN, sin embarga tal lenguaje presenta ciertas desventajas que se explican a continuación:

FORTRAN, así como los programas escritos en él, no son fácilmente extendibles, FORTRAN es muy fácil de aprender pero un programa escrito en FORTRAN no es fácil de usar y leer.

FORTRAN es deficiente en muchos aspectos, muchos de ellos son decisivos para producir programas eficientes. Por ejemplo su carencia de soporte para un uso dinámico de la memoria (ésto es, la posibilidad de introducir, durante una fase del proceso, un vector o una matriz del tamaño exacto requerido y usar la memoria por el tiempo requerido solamente). Esta deficiencia involucra el uso de vectores de trabajo que son de longitud fija y presentes durante toda la ejecución del programa. Su imposibilidad de crear 'bloques' en los que las variables estén escondidas (con la seguridad de posibles conflictos en las definiciones) y la carencia de control de la visibilidad de las variables entre diferentes funciones.

C++ supera en mucho las deficiencias encontradas en FORTRAN para el manejo numérico, se puede tener un verdadero uso dinámico de la memoria, se pueden crear vectores y matrices del tamaño estrictamente necesario, usarlas el tiempo que sean requeridos como verdaderos objetos y liberar la memoria que usan cuando ya no son necesarios. Lo anterior lo logra gracias al uso de apuntadores y de las características propias de los lenguajes orientados a objetos.

Otra importante razón por la que se eligió a C++ como lenguaje para realizar el sistema, es que existen en el mercado compiladores muy completos que presentan características de optimización de código, editores integrados, depuradores, documentación variada y una amplia gama de funciones.

Los recursos con los que se cuenta para desarrollar el sistema CONFIA son los siguientes:

Compilador Borland C++ ver 3.1.
Procesador de textos Word 6.0.
Computadora PC 80486DX2-66MHZ

Se pretende que el sistema funcione de la forma más simple posible y se consideran los siguientes requisitos para que el programa funcione de una manera adecuada.

Los requerimientos mínimos del sistema CONFIA son:

- Computadora 80286 a 12mhz con coprocesador matemático.
- Ratón compatible con Microsoft o IBM.
- Monitor VGA, con resolución de 640 X 480 pixeles monocromático.

- MS-DOS versión 5.0.
- Disco Duro con 1Mb de espacio libre.
- 450Kb libres de memoria RAM.

Los requerimientos recomendados del sistema CONFIA son:

- Computadora 80386DX a 40mhz con coprocesador matemático o superior.
- Ratón compatible con Microsoft o IBM.
- Monitor VGA, con resolución de 640 X 480 pixeles a colores.
- MS-DOS versión 6.2.
- Disco Duro con 2.5Mb de espacio libre.
- 600Kb libres de memoria RAM.

3.1.1 DIAGRAMA GENERAL DEL SISTEMA CONFIA

El sistema CONFIA contiene siete módulos principales:

- ARCHIVO
- MECÁNICO
- EDICIÓN
- SOLICITACIONES
- CONFIABILIDAD
- VISUALIZACIÓN
- AYUDA

Para interactuar con estos módulos es necesario manipularlos directamente mediante una clase especial llamada APLICACIÓN, esta clase es operada por la clase MENÚ PRINCIPAL que interactúa directamente con el usuario y permite seleccionar las diversas opciones que existen en la pantalla principal. De tal forma que la clase MENÚ PRINCIPAL es la interfaz entre el usuario y la aplicación, a su vez la clase APLICACIÓN es la interfaz entre las opciones del menú principal y las operaciones que cada módulo contiene. La figura 3.1 muestra como es la interfaz principal del programa y de que manera se llaman las operaciones y métodos que cada módulo contiene para realizar una acción específica y volver nuevamente al MENÚ PRINCIPAL.

3.1.2 MÓDULOS QUE COMPONEN EL SISTEMA CONFIA

- Módulo de Archivo - en este módulo se pueden realizar todas las operaciones referentes a manipulación de archivos y directorios, así como de la impresora. Se puede empezar una nueva estructura, eliminando toda la información que el programa tenía almacenado hasta el momento. Cargar una estructura de un archivo ya existente. Guardar una estructura que se halla modificado o editado. Ver los archivos de un directorio específico, para posteriormente poder cargar algún archivo en el programa. Cambiarse de directorio así como de unidad para poder visualizar otros archivos de estructuras que pudieran encontrarse en otros

directorios. Imprimir toda o parte de la información de la estructura que se está editando o analizando y finalmente salir del programa. Las principales clases que constituyen este módulo son:

- ARCHIVO - interfaz principal para la manipulación de todas las operaciones que hay en el módulo de Archivo, como cargar o guardar una estructura, visualizar o cambiarse a otro directorio, imprimir información de la estructura y salir del programa.
- ESTRUCTURA - se tienen las operaciones para cargar o guardar en el programa un archivo de datos con información de una estructura.
- IMPRESORA - controla el paso de información a la impresora, además permite seleccionar que archivos de datos se mandan a impresión.
- VER DIRECTORIO - visualiza los archivos que se encuentren en un directorio específico mediante algún filtro seleccionado, para posteriormente elegir uno de los archivos presentados en la pantalla para cargarlo en el programa mediante la clase Estructura y la información que ahí se encuentre.

En la figura 3.2 se observa el diagrama de clases y la manera en que éstas se relacionan dentro del módulo de Archivo.

- Módulo Mecánico - en este módulo se pueden definir características generales que va a poseer la estructura como son el tipo de análisis a realizar, el tipo de estructura, el tipo de interacción, así como la opción de analizar la estructura una vez terminada la edición de la misma. También se pueden especificar algunos datos generales del programa para su correcto funcionamiento. Las principales clases que constituyen este módulo son:

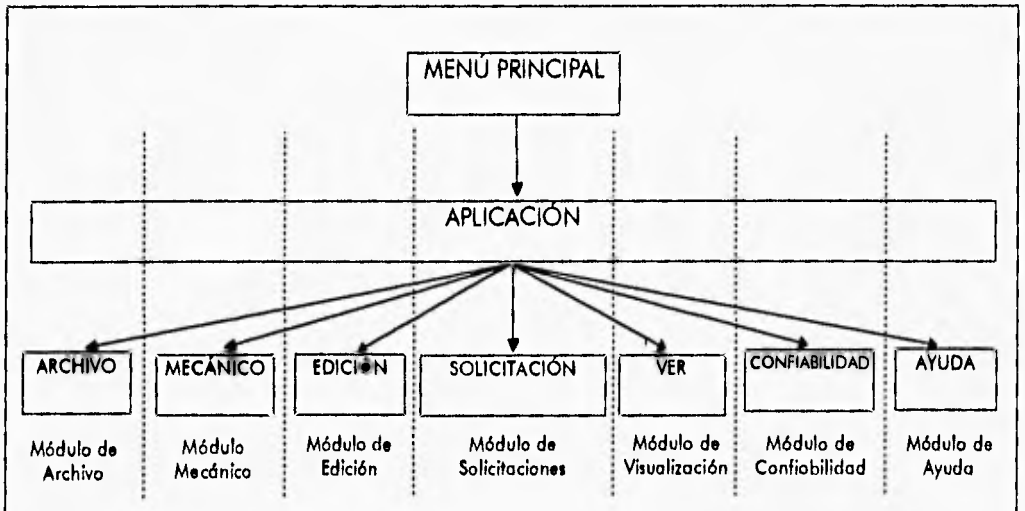


Figura 3.1 Diagrama principal de interfaz del programa.

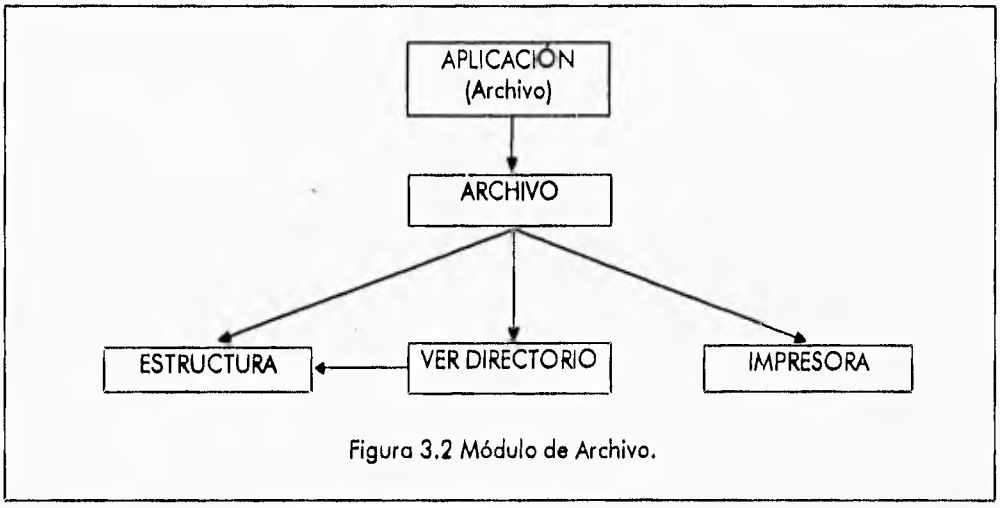


Figura 3.2 Módulo de Archivo.

- **CONTROL** - esta clase es una clase genérica que entre sus funciones principales tiene la tarea de operar en gran parte el módulo Mecánico para definir las diversas características que puede poseer la estructura a editar o analizar. También posee operaciones para crear y manipular gráficos en la pantalla, así como inicializar los diversos dispositivos de entrada y salida como son, el ratón, la impresora y la pantalla gráfica.
- **ANÁLISIS MECÁNICO** - esta clase está compuesta por varias subclases que trabajan conjuntamente para poder realizar el análisis mecánico de la estructura, para posteriormente realizar otros análisis y observar los resultados en otros módulos implementados específicamente para ello.

En la figura 3.3 se observa el diagrama de clases y la manera en que éstas se relacionan dentro del módulo Mecánico.

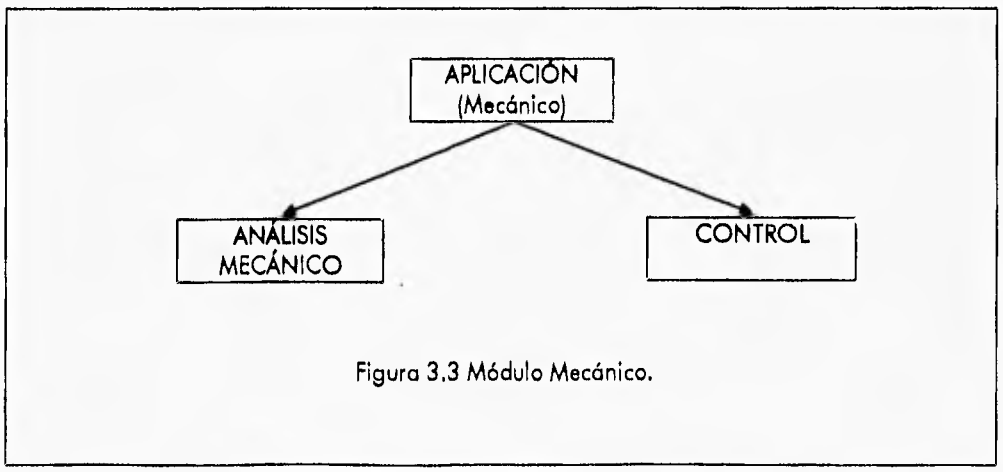


Figura 3.3 Módulo Mecánico.

- Módulo de Edición - este módulo es la parte medular del programa, se edita la estructura con todas sus características: nodos, barras, condicianes de carga, cargas aleatorias, generación automática de nodos, acercamiento o alejamiento de la imagen (zoom), así como guardar y cargar los archivos de configuración del programa. Las principales clases que constituyen este módulo son:

    - CONFIGURACIÓN DE PANTALLA - esta clase tiene las funciones necesarias para activar los límites de la pantalla de trabajo, activar o desactivar el snap y generar acercamientos o alejamientos de la pantalla de edición (zoom).
    - NODOS - contiene las operaciones necesarias para manipular los nodos de la estructura, como por ejemplo: agregar nodos, eliminar nodos, mover nodos y generar nodos.
    - CARGAS - se tienen las operaciones de identificación de los nodos que tienen determinadas cargas, ya sea cargas aleatorias o condiciones de carga y el menú de selección de estas cargas.
    - BARRAS - tiene las operaciones necesarias para poder manipular las barras de la estructura como por ejemplo: agregar barras, eliminar barras, mover barras.

  En la figura 3.4 se observa el diagrama de clases y la manera en que éstas se relacionan dentro del módulo de Edición.

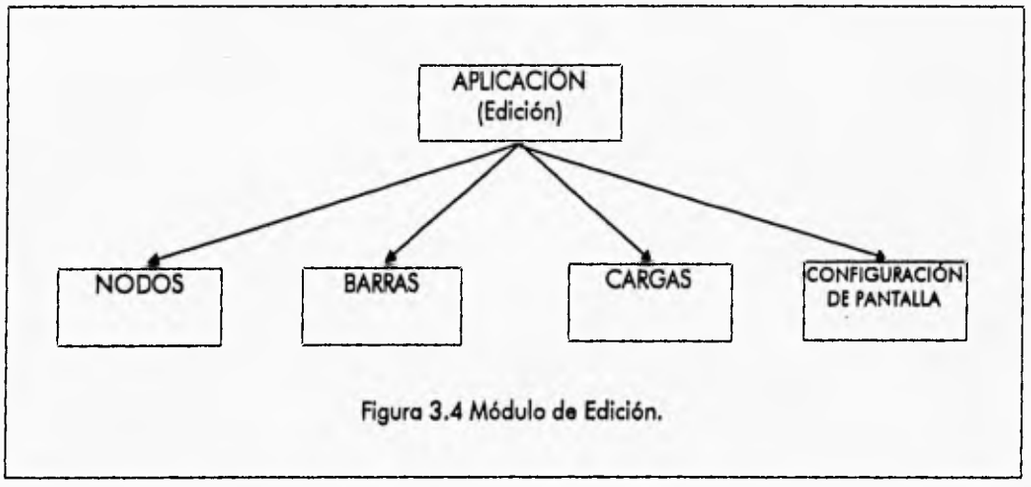


Figura 3.4 Módulo de Edición.

- Módulo de Solicitaciones - en este módulo se pueden definir fuerzos que actúan sobre la estructura y que intervienen en su análisis, como son las fuerzos sísmicas y fuerzos ocasionados por la occión del viento. Lo close principol que compone este módulo es:

    - SOLICITACIÓN - contiene las funciones necesarios pora locolizor e identificor los fuerzos debido a solicitaciones sísmicas o de viento.

En la figura 3.5 se observa el diagrama de clases y la manera en que éstas se relacionan dentro del módulo de Solicitaciones.

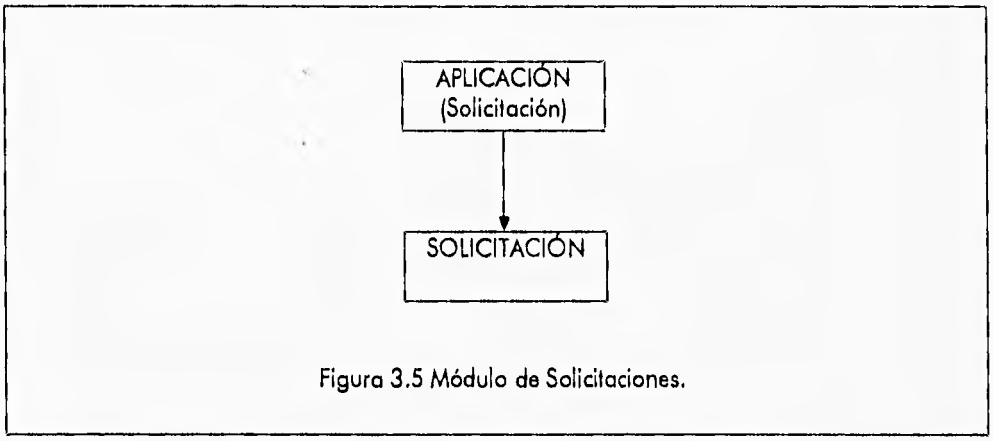


Figura 3.5 Módulo de Solicitaciones.

- Módulo de Confiabilidad - en este módulo se especifican algunas propiedades necesarias para los análisis de confiabilidad de la estructura y se observan los mecanismos plásticos analizados hasta el momento, así mismo se puede observar el resultado final de los análisis completos después de haber pasado por varias etapas de procesamiento. Las clases principales que componen este módulo son:

    - SIMULACIÓN - esta clase contiene todas las funciones y procesos para poder realizar los análisis de confiabilidad.
    - VISUALIZACIÓN - la clase visualización encapsula las funciones necesarias para poder visualizar gráficamente los resultados a través de la estructura deformada, así como en forma numérica como lo son los índices betas y las fuerzas internas.

En la figura 3.6 se observa el diagrama de clases y la manera en que éstas se relacionan dentro del módulo de Confiabilidad.

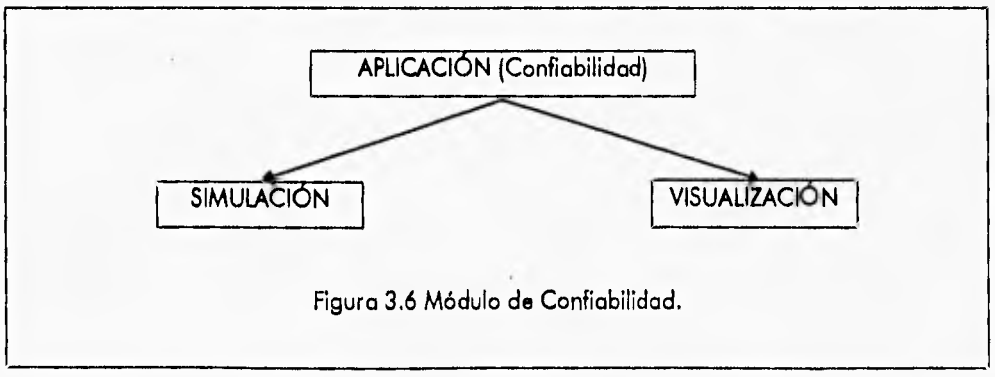


Figura 3.6 Módulo de Confiabilidad.

- Módulo de Visualización - en este módulo se pueden visualizar los resultados del análisis mecánico tanto de manera gráfica mediante la visualización de la estructura deformada así como en forma numérica mostrando los valores de las fuerzas internas e índices betas que se obtuvieron del análisis realizado. La clase principal que compone este módulo es:

    - VISUALIZACIÓN - esta clase es la misma que ya fue explicada en el módulo de Confiabilidad.

En la figura 3.7 se observa el diagrama de clases y la manera en que éstas se relacionan dentro del módulo de Visualización.

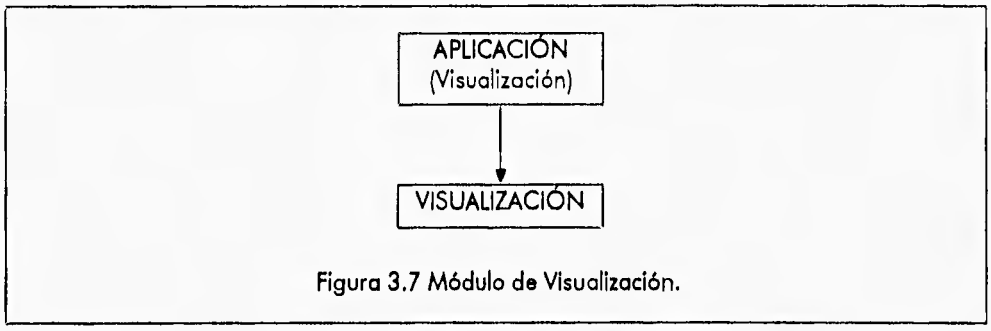


Figura 3.7 Módulo de Visualización.

- Módulo de Ayuda - este módulo sirve como auxilio en el manejo del programa tanto para resolver dudas que se pudieran presentar en el proceso de análisis de una estructura como en la forma de usar la interfaz gráfica del mismo programa. La clase que compone este módulo es la siguiente:

    - AYUDA - en esta clase se tienen las funciones básicas tanto para la presentación de los textos de la ayuda, como de la búsqueda y lectura de los archivos correspondientes, así como la elección de algún tópico específico en los menús que contiene la ayuda.

En la figura 3.8 se observa el diagrama de clases y la manera en que éstas se relacionan dentro del módulo de Ayuda.

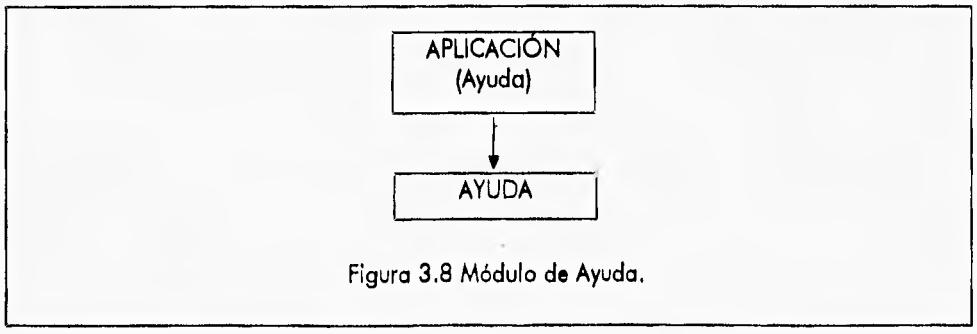


Figura 3.8 Módulo de Ayuda.

3.1.3 IDENTIFICACIÓN DE CLASES GENÉRICAS Y SUBCLASES

Una vez definida la estructura de los módulos y sus clases principales, se deben identificar las subclases y clases genéricas que están presentes en el sistema. Se entiende por subclase a una clase que es llamada dentro de una clase principal con lo que actúa directamente al mando del programa y una clase genérica es aquella clase que se usa por la mayoría de las clases que componen el programa. Las clases genéricas y las subclases son las siguientes:

- RATÓN - encapsula el comportamiento del ratón y permite su control por parte del usuario para poder interactuar con el programa y ser capaz de manipular todas las opciones que el programa posee.

- DIBUJA - contiene los métodos genéricos para dibujar y localizar nodos, dibujar barras, dibujar la estructura completo, así como las coordenadas de los nodos y barras editados

- CONFIGURACIÓN - esta clase sirve para cambiar de coordenadas de pantalla a coordenadas reales en metros, así como también redondear ciertas cantidades necesarias para las conversiones, así mismo tiene una función específica para generar una malla que sirve de guía en la pantalla gráfica para la edición de la estructura

- MENÚ PRINCIPAL - sirve para crear y manipular el menú principal del programa, así como, el menú interactivo, que a su vez, es la base para manipular todas las opciones que contiene el programa.

- SUBMENÚ - clase necesaria para crear los menús que aparecen por toda la pantalla y en los cuales se puede introducir información, así como también desplegarla y manipularla, para que posteriormente si es el caso, sea procesada en alguna de las [illegible]

- MATRIZ - encapsula las operaciones básicas de las matrices como son: suma de matrices, multiplicación de matrices, transpuesta de una matriz y demás operaciones, para poder llevar a cabo los análisis mecánicos y de confiabilidad.

- VECTOR - contiene operaciones básicas que se pueden efectuar sobre vectores. Al igual que la clase matriz, la clase vector es usada para los análisis mecánicos y de confiabilidad.

- BUFFER - esta pequeña clase sirve para guardar en el disco duro imágenes generadas en la pantalla cuando la memoria RAM es insuficiente para almacenar toda la porción de la pantalla especificada en el momento de programa, esto con el objeto de [illegible]

3.1.4 RELACIÓN DE HERENCIA ENTRE LAS CLASES

[illegible]

3.1.3 IDENTIFICACIÓN DE CLASES GENÉRICAS Y SUBCLASES

Una vez definida la estructura de los módulos y sus clases principales, se deben identificar las subclases y clases genéricas que están presentes en el sistema. Se entiende por subclase a una clase que es llamada dentro de una clase principal con la que actúa directamente el usuario del programa y una clase genérica es aquella clase que se usa por la mayoría de las clases que componen el programa. Las clases genéricas y las subclases son las siguientes:

- RATÓN - encapsula el comportamiento del ratón y permite su control por parte del usuario para poder interactuar con el programa y ser capaz de manipular todas las opciones que el programa posee.

- DIBUJA - contiene los métodos genéricos para dibujar y localizar nodos, dibujar barras, dibujar la estructura completa, así como las coordenadas de los nodos y barras editadas.

- CONFIGURACIÓN - esta clase sirve para cambiar de coordenadas de pantalla a coordenadas reales en metros, así como también redondear ciertas cantidades necesarias para las conversiones, así mismo tiene una función específica para generar una malla que sirve de guía en la pantalla gráfica para la edición de la estructura.

- MENÚ PRINCIPAL - sirve para crear y manipular el menú principal del programa, así como el menú interactivo, que a su vez, es la base para manipular todas las opciones que contiene el programa.

- SUBMENÚ - clase necesaria para crear los menús que aparecen por toda la pantalla y en los cuales se puede introducir información, así como también desplegarla y manipularla, para que posteriormente si es el caso, sea procesada en alguno de los análisis.

- MATRIZ - encapsula las operaciones básicas de las matrices como son suma de matrices, multiplicación de matrices, transpuesta de una matriz y demás operaciones, para poder llevar a cabo los análisis mecánicos y de confiabilidad.

- VECTOR - contiene operaciones básicas que se pueden efectuar sobre vectores. Al igual que la clase matriz, la clase vector es usada para los análisis mecánicos y de confiabilidad.

- BUFFER - esta pequeña clase sirve para guardar en el disco duro imágenes generadas en la pantalla cuando la memoria RAM es insuficiente para albergar toda la porción de la pantalla especificada en el transcurso del programa, ésto con el objeto de no llevar al programa a una terminación fuera de control y que pudiera ocasionar la pérdida de información.

3.1.4 RELACIÓN DE HERENCIA ENTRE LAS CLASES

Algunas clases comparten ciertos comportamientos generales, por lo que para disminuir la redundancia de funciones se aplica la siguiente jerarquía de herencia mostrada en las figuras 3.10 y 3.11.

El primer esquema de herencia es en el campo de la edición y configuración, tanto de la estructura a editar como del programa en general y es de la siguiente forma:

- La clase CONFIGURACIÓN contiene funciones genéricas para la conversión de coordenadas de pantalla a coordenadas reales, así como funciones para redondear cantidades enteras y de punto flotante de acuerdo a las necesidades específicas que se requieran. La clase DIBUJA hereda todas las funciones de la clase CONFIGURACIÓN ya que son necesarias para dibujar nodos, barras y calcular las posiciones de cada uno de estos elementos en la pantalla gráfica.

- La clase CONFIGURACIÓN DE PANTALLA hereda todas las funciones o métodos de la clase CONFIGURACIÓN ya que son necesarias para especificar y calcular los límites mínimos y máximos del plano real y el de pantalla, así como activar el snap con sus correspondientes escalas en cada uno de los ejes coordenados y el mallador basado en las escalas del snap.

- Las clases NODOS, BARRAS, CARGAS Y VISUALIZACIÓN heredan de la clase DIBUJA todos sus atributos para la edición y presentación de la estructura en forma gráfica con sus atributos correspondientes.

El segundo esquema de herencia es el referente a la creación y manipulación de los menús que aparecen en la pantalla gráfica y que es la principal vía de comunicación entre el usuario y el programa, de esta forma la jerarquía de clases es de la siguiente forma:

- La clase MENÚ PRINCIPAL tiene los atributos generales para la creación, ejecución y eliminación de los menús en pantalla.

- La clase SUBMENÚ hereda todos los atributos y métodos de la clase MENÚ PRINCIPAL y a su vez posee otros atributos para crear los menús en cualquier parte de la pantalla, así como poder introducir y desplegar información dentro de estos menús.

- La clase VER DIRECTORIO hereda todos los atributos y métodos de la clase MENÚ PRINCIPAL y posee otras funciones específicas para poder visualizar la lista de archivos dentro del menú y poder seleccionar uno de ellos mediante el ratón.

- La clase AYUDA tiene los mismos atributos y métodos de la clase padre VER DIRECTORIO, así mismo contiene una función adicional, para desplegar la información del texto de la ayuda en la pantalla gráfica.

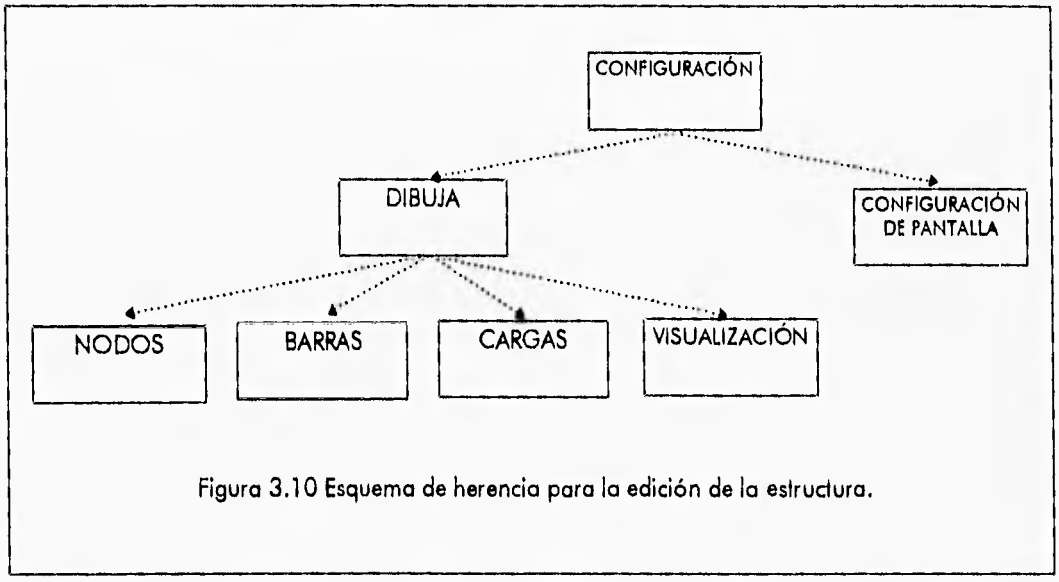


Figura 3.10 Esquema de herencia para la edición de la estructura.

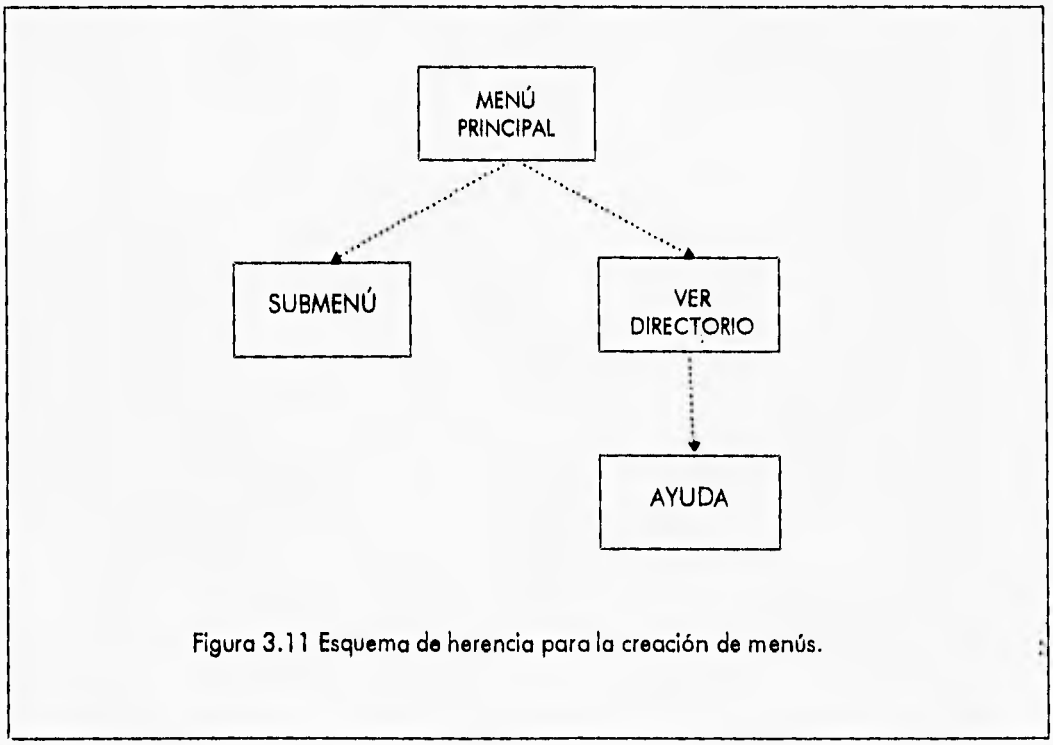


Figura 3.11 Esquema de herencia para la creación de menús.

3.2 DISEÑO ORIENTADOS A OBJETOS DEL SISTEMA CONFIA

El diseño del sistema CONFIA está basado en el análisis previamente realizado, la explicación en forma general de cada una de las clases que componen el sistema ya ha sido descrita. En esta sección sólo se presenta una explicación interna de cada clase, así como la forma en que interactúa cada una de ellas con el exterior.

3.2.1 CLASE APLICACIÓN

La clase APLICACIÓN consta de las siguientes funciones:

- MÓDULO_ARCHIVO - Función interfaz con el módulo de Archivo.
- MÓDULO_MECÁNICO - Función que crea y visualiza el menú del módulo Mecánico y permite modificar las características generales de la estructura, y realizar el análisis mecánico.
- MÓDULO_EDICIÓN - Función que permite el manejo del módulo de Edición.
- MÓDULO_SOLICITACIÓN - Función interfaz con el módulo de Solicitaciones.
- MÓDULO_VER - Función que permite el manejo del módulo Ver.
- MÓDULO_CONFIABILIDAD - Función que permite el manejo del módulo de Confiabilidad.
- MENÚ_CARGAS - Función que crea y visualiza los menús de condiciones de carga y cargas aleatorias, permite definir el número de condiciones de carga y cargas aleatorias que tendrá la estructura y editarlas ya sea de forma individual o de forma consecutiva.
- MENÚ_VER - Define los submenús específicos para desplazamientos, fuerzas internas e índices beta para su posterior visualización.

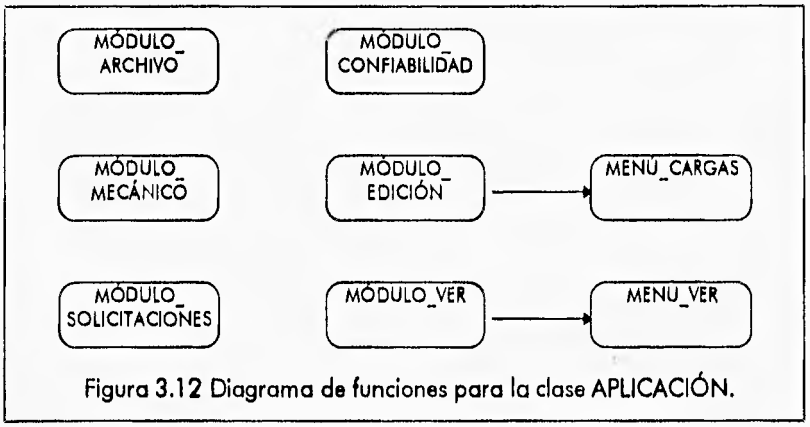


Figura 3.12 Diagrama de funciones para la clase APLICACIÓN.

La clase APLICACIÓN hace uso de la mayoría de las clases del sistema ya que es la que controla todo el programa.

- SOLICITACIÓN
  - PRINCIPAL

- SUBMENÚ
    - CREAR_MENÚ
    - EJECUTA_MENÚ
    - COMPONE
    - BORRA_MENÚ
- CONTROL
    - TIPO_ESTRUC
    - TIPO_ANAL
    - TIPO_INTER
    - CHECA_DRIVE
    - PON_TIPO_ESTRUC
- MENÚ PRINCIPAL
    - BORRA_MENÚ
    - CREAR_MENÚ
- CONFIGURACIÓN DE PANTALLA
    - ZOOM_VENTANA
    - ZOOM_PREVIO
    - ZOOM_TOTAL
    - ABRIR_CONFIG
    - PON_SNAP
    - PON_LÍMITES
    - GUARDAR_CONFIG
- DIBUJA
    - REDIBUJAR
- MECÁNICO
    - CREA_AUX1
    - ANALIZAR
- CARGAS
    - EDITAR_CARGAS
- NODOS
    - EDICIÓN NODOS
    - REDIBUJAR
    - GENERA_NODOS
    - BORRAR_NODOS
- BARRAS
    - EDITAR_BARRAS
    - BORRAR_BARRAS
    - REDIBUJAR
- SIMULACIÓN
    - INICIA_CÁLCULO
- VISUALIZACIÓN
    - MECANISMOS
    - INTERNAS
    - DESPLAZA

- ARCHIVO
    - PRINCIPAL

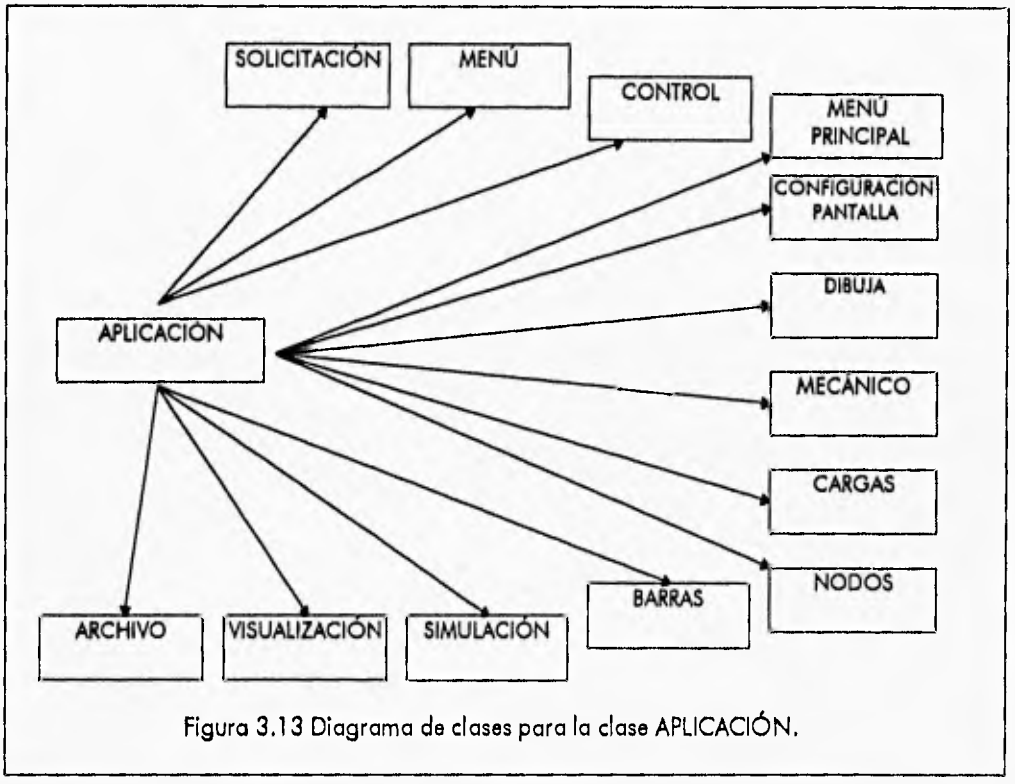


Figura 3.13 Diagrama de clases para la clase APLICACIÓN.

DIAGRAMA DE TRANSICIÓN DE ESTADO DEL OBJETO APLICACIÓN

El objeto APLICACIÓN se crea al inicio del programa y se encarga de manipular los módulos principales del sistema durante toda su ejecución.

3.2.2 CLASE MENÚ PRINCIPAL

La clase MENÚ PRINCIPAL consta de las siguientes funciones:

- POS_TEXTO - Esta función se encarga de proporcionar las coordenadas del texto que se van a desplegar en la pantalla.
- GUARDA_MENÚ - Esta función se encarga de guardar cierta porción de la pantalla gráfica en la memoria RAM, para posteriormente restaurar esta parte de la pantalla mediante la función BORRA_MENÚ.

Figura 3.14 Diagrama de transición de estado del objeto APLICACIÓN.

- TEXTO - Esta función se encarga de desplegar cierto texto en la pantalla gráfica en las coordenadas proporcionadas por la función POS_TEXTO.
- SOMBRA - Esta función se encarga de realizar el efecto de oprimir un botón en cualquier menú seleccionado.
- MENÚ_SEL - Esta función se encarga de seleccionar alguna de las opciones del menú creado en un momento específico.
- EJECUTA_MENÚ - Esta función se encarga de ejecutar el menú creado mediante la función CREAR_MENÚ y a su vez inicializar ciertas variables para su posterior manipulación.
- BORRA_MENÚ - Esta función se encarga de restaurar la porción de la pantalla gráfica usada por el menú.

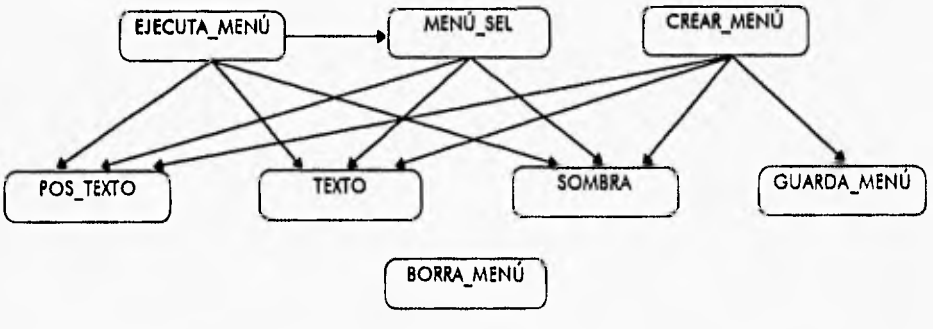


Figura 3.15 Diagrama de funciones de la clase MENÚ PRINCIPAL.

Las clases y funciones que usa la clase MENÚ PRINCIPAL son las siguientes:

- RATÓN
  - OBT_COORDS
  - MOSTRAR

  - OCULTAR
  - CAJA
  - BOTÓN_PRES
  - BOTÓN_REL
- BUFFER
  - GETIMAGE_FILE
  - PUTIMAGE_FILE

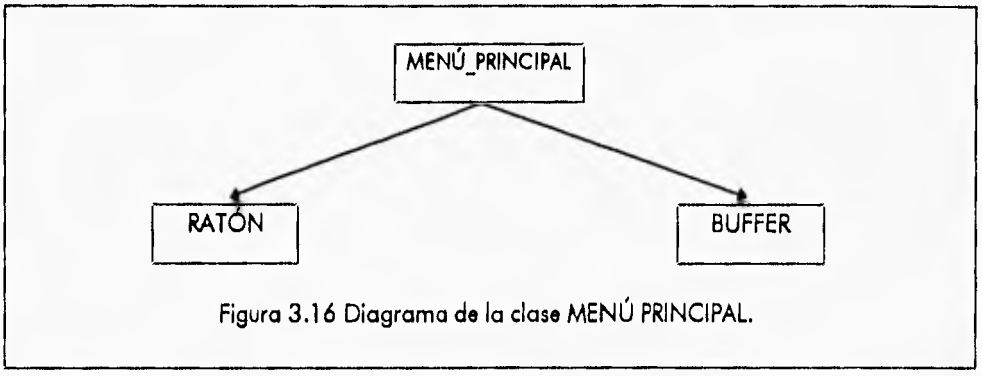


Figura 3.16 Diagrama de la clase MENÚ PRINCIPAL.

DIAGRAMA DE TRANSICIÓN DE ESTADO DEL OBJETO MENÚ PRINCIPAL

En el proceso de operación del objeto MENÚ PRINCIPAL se debe de crear primeramente el menú mediante la función CREAR_MENÚ, posteriormente ejecutarlo y elegir una opción con las funciones EJECUTA_MENÚ y MENÚ_SEL, para finalmente borrarlo de la pantalla con la función BORRAR_MENÚ y proceder a ejecutar otras operaciones.

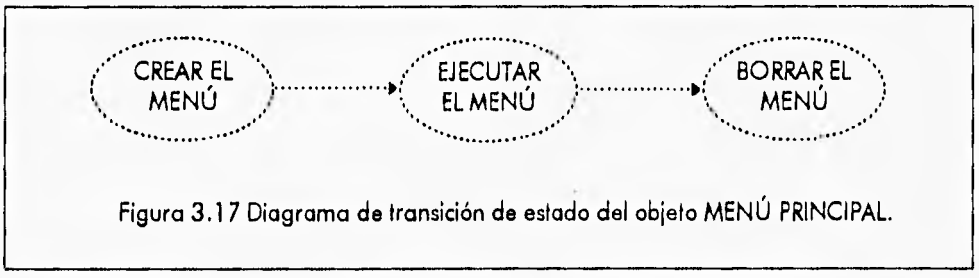


Figura 3.17 Diagrama de transición de estado del objeto MENÚ PRINCIPAL.

3.2.3 CLASE ARCHIVO

Las funciones que pertenecen a la clase ARCHIVO son:

- NUEVO - Prepara al sistema para editar una nueva imagen, limpia la pantalla e inicializa las variables globales.

- CARGAR_O_GUARDAR - En esta función se crean los submenús para cargar o guardar una estructura y se checan las unidades, si se quiere hacer uso de una unidad de disco flexible.
- CAMBIAR_DIR - Con esta función se crean los submenús para cambiar el directorio de trabajo o la unidad de disco.
- VER_DIR - En está función se crea un submenú para indicar algún directorio que se desee ver y presenta la lista de los archivos contenidos en el directorio especificado, además carga en el programa el archivo de datos que se haya señalado.
- IMPRIMIR - Crea el submenú que indica los archivos que se pueden imprimir y llama a la función indicada de acuerdo a la opción que se haya escogido.
- SALIR - Crea el submenú para salir del programa y realiza las acciones que se escojan.
- PRINCIPAL - Crea y manipula el menú del módulo Archivo.

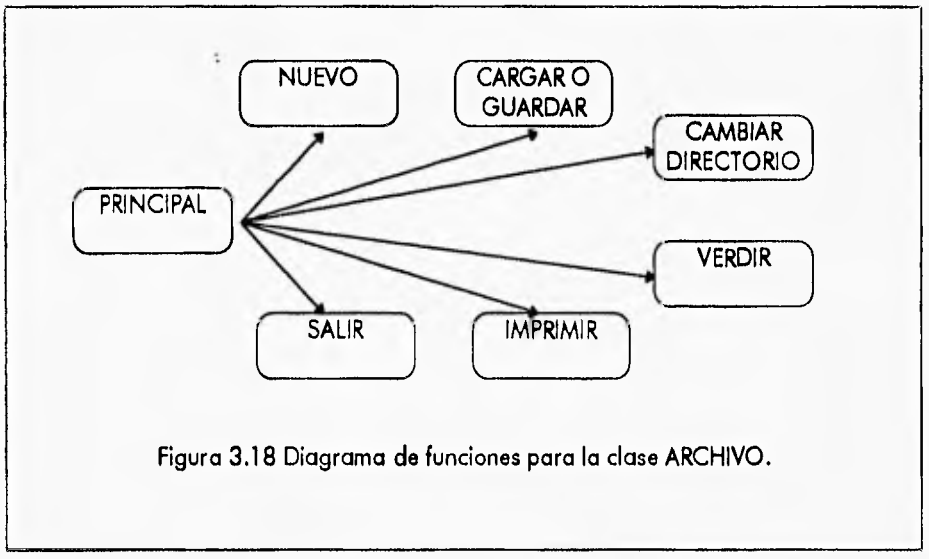


Figura 3.18 Diagrama de funciones para la clase ARCHIVO.

La clase ARCHIVO hace uso de varias clases para su funcionamiento:

- RATÓN
- MENÚ PRINCIPAL
- CONTROL
  - INIT_GLOBALES
  - PON_TIPO_ESTRUC
  - CHECA_DRIVE
- SUBMENÚ
  - CREAR_MENÚ
  - EJECUTA_MENÚ
  - COMPONE
  - BORRA_MENÚ
- CONFIGURACIÓN DE PANTALLA

    - PON_LÍMITES
    - PON_SNAP
    - GUARDAR_CONFIG
    - DIBUJA
    - REDIBUJAR
- ESTRUCTURA
    - CARGAR_ESTRUCTURA
    - GUARDAR_ESTRUCTURA
- NODOS
    - ELIMINAR_NODO
- BARRAS
    - ELIMINAR_BARRA
- VERDIR
    - DIRECTORIO
- IMPRESORA
    - ESTADO_IMPRESORA
    - IMPRIMIR_ARCH
    - IMPRIMIR_DES
    - IMPRIMIR_FUE

DIAGRAMA DE TRANSICIÓN DE ESTADO DEL OBJETO ARCHIVO

En el proceso de operación del objeto ARCHIVO se debe comenzar con elegir que operación se va a realizar en el módulo de Archivo, para lo cual se deberá llamar a la función PRINCIPAL, posteriormente ejecutar una de las operaciones elegidas con cualquiera de las funciones creadas para tal efecto como por ejemplo:

VERDIR
SALIR
CAMBIAR_DIR

Para finalmente abandonar el objeto.

3.2.4 CLASE IMPRESORA

Las funciones que pertenecen a la clase IMPRESORA son:

- ESTADO_IMPRESORA - Antes de mandar a imprimir, esta función verifica si la impresora está conectada, que tenga papel y esté lista para recibir datos.
- IMPRIMIR_ARCH - Imprime los datos de la estructura como son las coordenadas de los nodos, las cargas aplicadas a cada nodo, los nodos extremos de las barras, las resistencias y el tipo de material de la barra, las características de todos los materiales aplicados a la

estructura, el tipo de estructura que es, el tipo de análisis que se le aplica y el tipo de interacción.

- IMPRIMIR_DES - Imprime los desplazamientos nodales que se encontraron en el análisis mecánico.
- IMPRIMIR_FUE - Manda a impresión las fuerzas internas obtenidas en el análisis mecánico.

Esta clase no hace uso de otras clases para su funcionamiento.

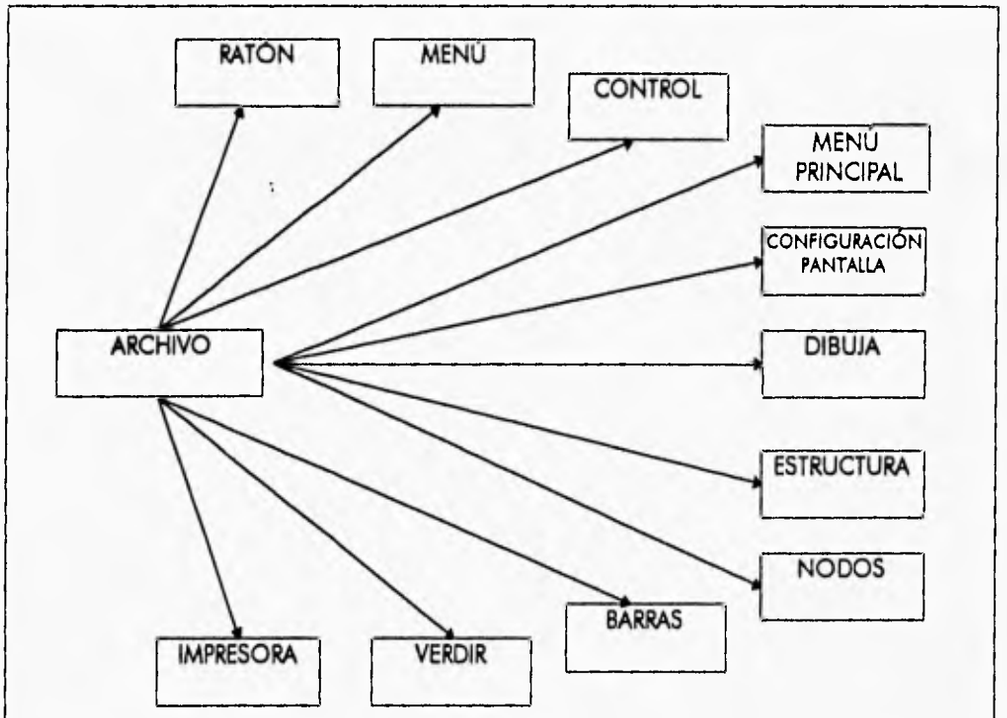


Figura 3.19 Diagrama de clases para la clase ARCHIVO.

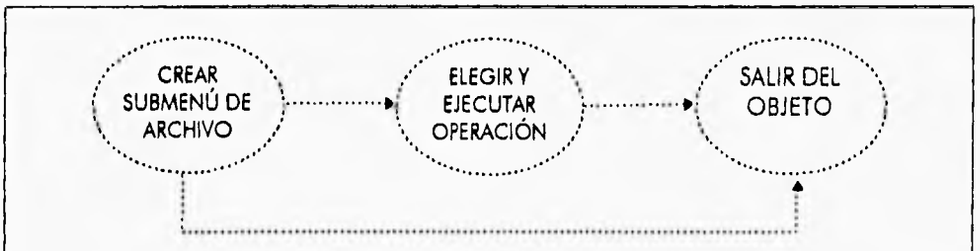


Figura 3.20 Diograma de transición de estado del objeto ARCHIVO.

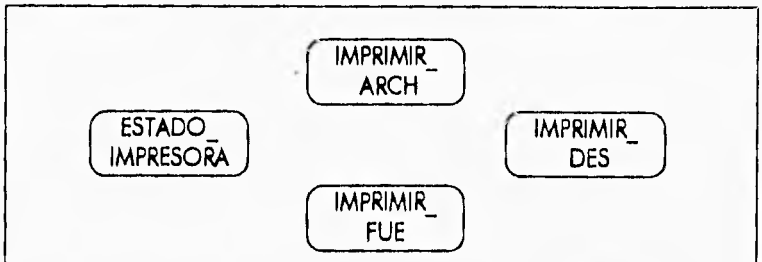


Figura 3.21 Diagrama de funciones para la clase IMPRESORA.

Figura 3.22 Diagrama de clases para la clase IMPRESORA.

DIAGRAMA DE TRANSICIÓN DE ESTADO DEL OBJETO IMPRESORA

Al crear este objeto el primer paso es checar el estado de la impresora siempre, para posteriormente, mandar la información a ella, con alguna de las tres opciones elegidas si el resultado de la verificación fue satisfactorio y finalmente abandonar el objeto.

Figura 3.23 Diagrama de transición de estado del objeto IMPRESORA.

3.2.5 CLASE ESTRUCTURA

Las funciones que pertenecen a la clase ESTRUCTURA son:

- INTER_GUARDAR - Se puede guardar la estructura con la que se está trabajando, tecleando F2, pero antes de llamar directamente a la función de GUARDAR ESTRUCTURA, se tienen que hacer algunos pasos que realiza esta función, como encontrar el nombre de la estructura que se está editando y checar la unidad donde se va a guardar la estructura.

- CARGAR_ESTRUCTURA - Carga en la memoria del programa una estructura guardada en un archivo de datos.
- GUARDAR_ESTRUCTURA - Guarda la estructura editada, en un archivo de datos.

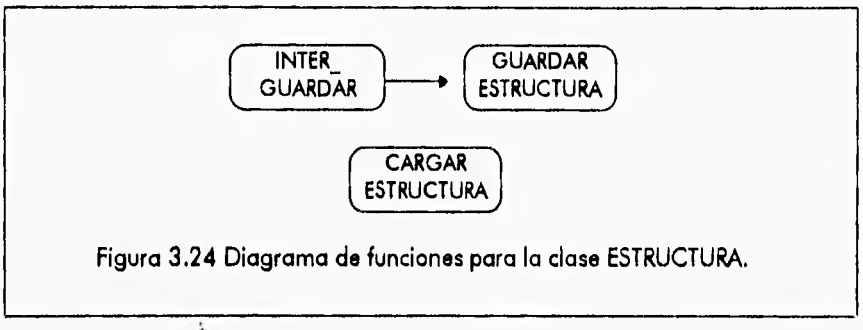


Figura 3.24 Diagrama de funciones para la clase ESTRUCTURA.

La clase ESTRUCTURA utiliza varias clases para su funcionamiento:

- RATÓN
  - OCULTAR
  - MOSTRAR
- CONTROL
  - INIT_GLOBALES
  - PON_TIPO_ESTRUC
  - CHECA_DRIVE
- CONFIGURACIÓN DE PANTALLA
  - PON_SNAP
  - PON_LÍMITES
  - PON_TIPO_ESTRUC
- NODOS
  - ELIMINAR_NODO
  - AGREGAR_NODO
- BARRAS
  - ELIMINAR_BARRA
  - AGREGAR_BARRA

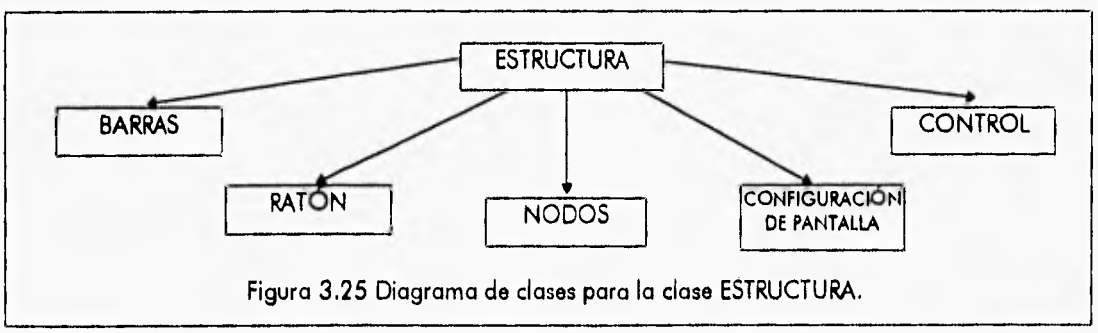


Figura 3.25 Diagrama de clases para la clase ESTRUCTURA.

3.2.6 CLASE VER DIRECTORIO

Las funciones que pertenecen a la clase VER DIRECTORIO son las siguientes:

- POSICIÓN_ARCHIVO - Esta función proporciona las coordenadas en la pantalla gráfica del nombre del archivo que pertenece a un subdirectorio específico y que además es desplegado en un recuadro dentro de la misma pantalla gráfica.
- PON_ARCHIVO - Esta función sirve para desplegar los nombres de una lista de archivos de un directorio específico, dentro de un recuadro en la pantalla gráfica.
- BARRA - Esta función sirve para resaltar en modo de video invertido el nombre de un archivo que aparece en el recuadro dentro de la pantalla gráfica e indica que es el archivo que se está seleccionando para cargarlo posteriormente en el programa.
- BOTÓN - Esta función sirve para dibujar un pequeño recuadro en las partes superior e inferior derechas dentro del recuadro principal que muestra los nombres de los archivos disponibles listos para cargarse en la memoria del programa.
- CUADRO - Esta función dibuja el recuadro principal donde irá la lista de los archivos seleccionados de un subdirectorio específico.
- LOCALIZA - Esta función localiza la posición de un archivo en la lista de todos los archivos seleccionados en la función DIRECTORIO.
- AGREGA - Esta función genera una lista dinámica para almacenar los nombres de todos los archivos seleccionados.
- DIRECTORIO - Esta función lee la lista de archivos que pertenecen a un subdirectorio específico y dentro de los cuales, se selecciona uno para cargar la estructura que esté almacenada en él.
- EJECUTA_MENÚ, CREAR_MENÚ, BORRA_MENÚ, MENÚ_SEL - Estas funciones están sobrecargadas y desempeñan la misma función que en la clase padre MENÚ_PRINCIPAL.
- GUARDA_MENÚ - Función heredada de la clase padre MENÚ_PRINCIPAL.

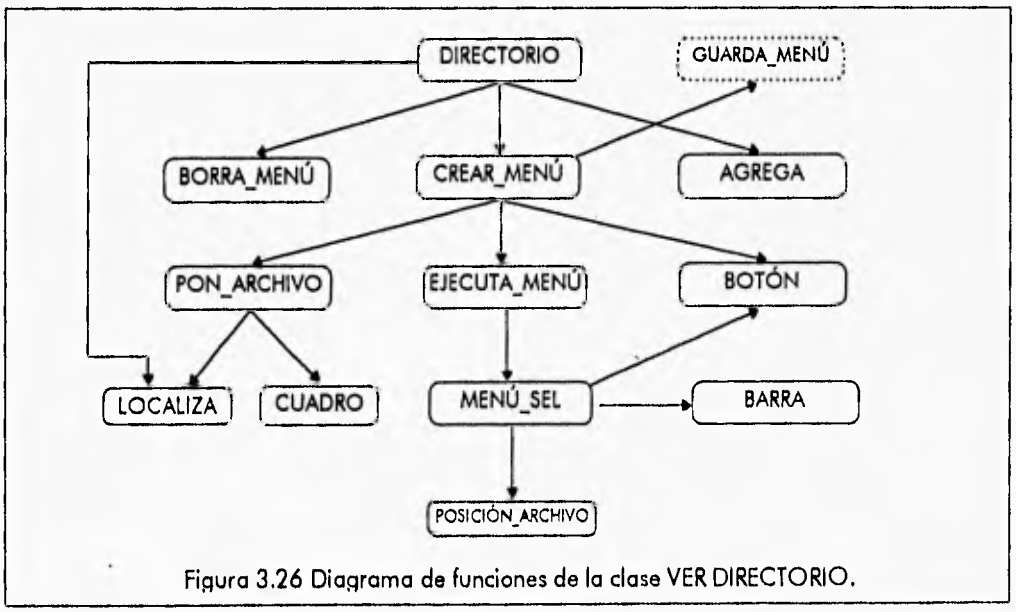


Figura 3.26 Diagrama de funciones de la clase VER DIRECTORIO.

Las clases y funciones que usa la clase VER DIRECTORIO son las siguientes:

- CONTROL
  - CHECA_DRIVE
- RATÓN
  - OBT_COORDS
  - MOSTRAR
  - OCULTAR
  - CAJA
  - BOTÓN_PRES
  - BOTÓN_REL
- BUFFER
  - GETIMAGE_FILE
  - PUTIMAGE_FILE

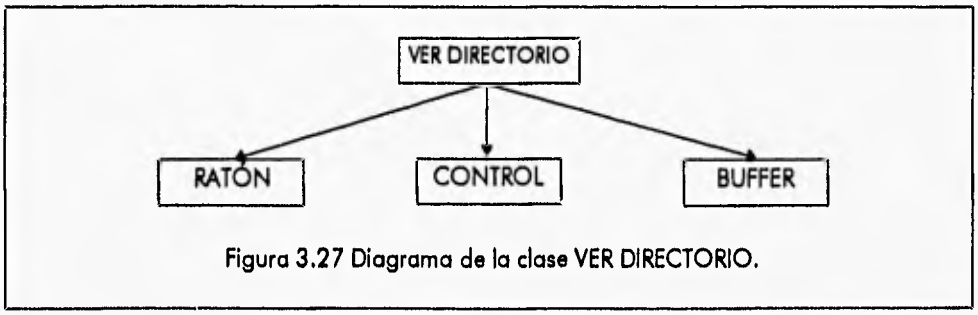


Figura 3.27 Diagrama de la clase VER DIRECTORIO.

DIAGRAMA DE TRANSICIÓN DE ESTADO DEL OBJETO VER DIRECTORIO

La transición de estados que sigue el objeto VER DIRECTORIO es como sigue: primero se efectúa la lectura de los archivos del directorio especificado mediante la función DIRECTORIO, posteriormente se despliegan en la pantalla a manera de menú y se elige uno de los archivos seleccionados para cargar la estructura en la memoria del programa, para finalmente borrar el menú y salir de la opción.

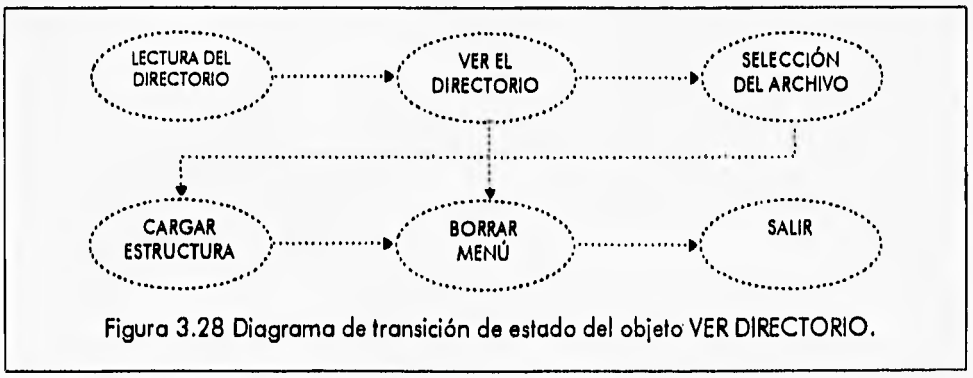


Figura 3.28 Diagrama de transición de estado del objeto VER DIRECTORIO.

3.2.7 CLASE CONTROL

Las funciones que pertenecen a la clase CONTROL son:

- CONTROL - Constructor de la clase CONTROL que se llama automáticamente al crear un objeto de la clase control y que verifica algunos aspectos importantes para el funcionamiento del sistema.
- ICONO - Verifica la existencia de cada icono en el archivo de iconos, si se encuentran se leen y se dibujan en el lugar que les corresponde, de lo contrario se genera un error.
- VERIFICA_NÚMERO_DRIVES - Verifica el número de unidades con las que cuenta la computadora en la que se está corriendo el programa.
- INITGRAPH - Cambia la pantalla de video a modo gráfico tipo VGA.
- PON_TIPO_ESTRUC - Pone los valores de las características generales de la estructura: tipo de análisis, tipo de estructura y tipo de interacción, en el menú derecho inferior.
- TIPO_ESTRUC - Crea, visualiza y permite manipular el submenú donde se indica el tipo de estructura a editar.
- TIPO_ANAL - Crea, visualiza y permite manipular el submenú donde se indica el tipo de análisis de la estructura a editar.
- TIPO_INTER - Crea, visualiza y permite manipular el submenú donde se indica el tipo de interacción de la estructura a editar.
- INTERACTIVO - Función que permite el manejo del sistema de una forma interactiva.
- CHECA_DRIVE - Verifica el estado de las unidades de disco flexible, esto sirve para controlar los errores que genera el sistema operativo al no estar lista cierta unidad o tener protección contra escritura.
- INTERFACE - Carga los valores de las variables de configuración general: límites, snap, mallador, tipo de estructura del archivo de configuración y verifica la existencia del ratón.
- INIT_GLOBALES - Inicializa las variables globales de todo el programa.

Esta clase hace uso de varias clases para su funcionamiento:

- RATÓN
  - OCULTAR
  - MOSTRAR
  - RESET
- MENÚ PRINCIPAL
  - MENÚ_SEL
  - SOMBRA
  - BORRA_MENÚ
- SUBMENÚ
  - CREAR_MENÚ
  - EJECUTA_MENÚ
  - COMPONE
  - BORRA_MENÚ
- DIBUJA
  - REDIBUJAR

  - DESMARCA_NODO
  - DESMARCA_BARRA
- CONFIGURACIÓN DE PANTALLA
  - ZOOM_VENTANA
  - ZOOM_PREVIO
  - INIT_SNAP
  - INIT_LÍMITES
  - ABRIR_CONFIG
  - PON_SNAP
  - PON_LÍMITES
  - GRID

DIAGRAMA DE TRANSICIÓN DE ESTADO DEL OBJETO CONTROL

El objeto control se crea desde el principio del programa, primeramente se hace una verificación de los elementos necesarios para la ejecución del programa mediante el propio constructor del objeto. Después dibuja la pantalla principal del sistema con la función INTERFACE, para posteriormente permitir un manejo interactivo del programa con la función INTERACTIVO.

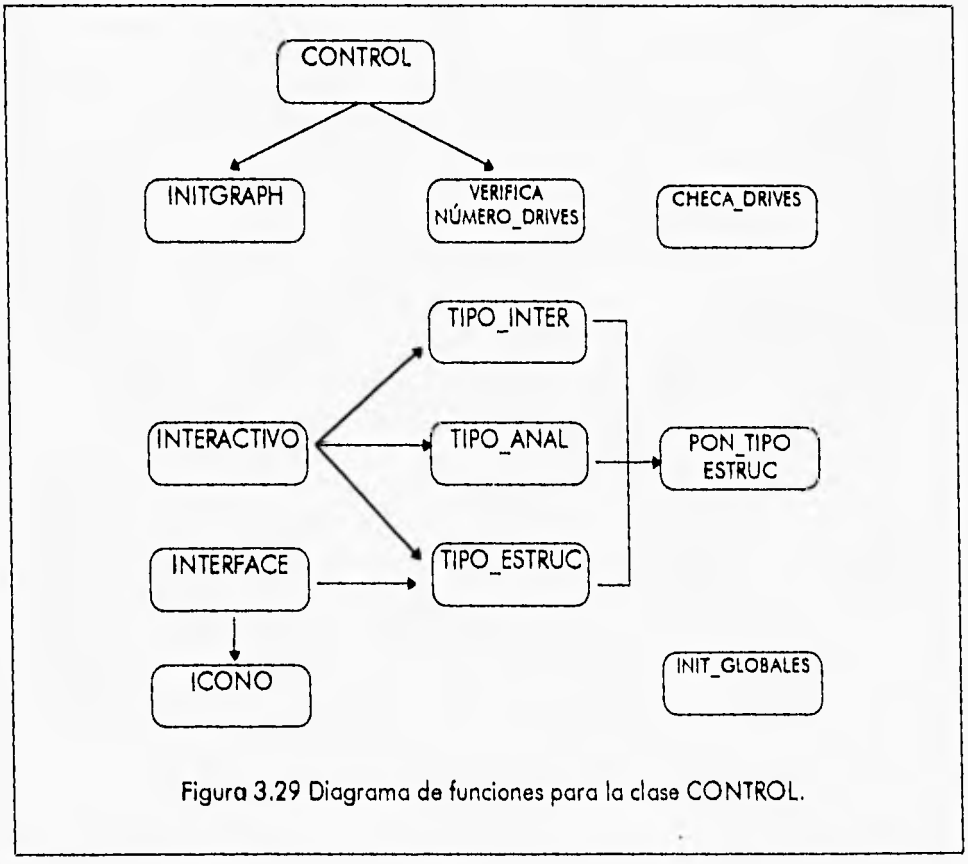


Figura 3.29 Diagrama de funciones para la clase CONTROL.

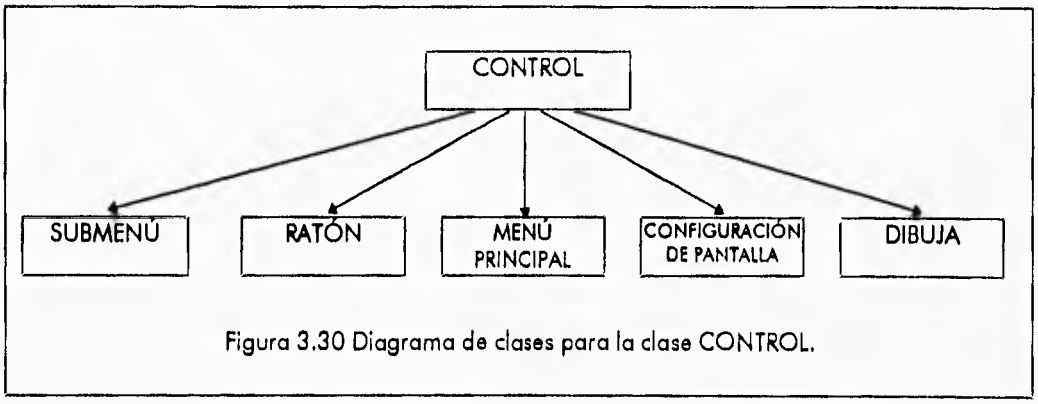


Figura 3.30 Diagrama de clases para la clase CONTROL.

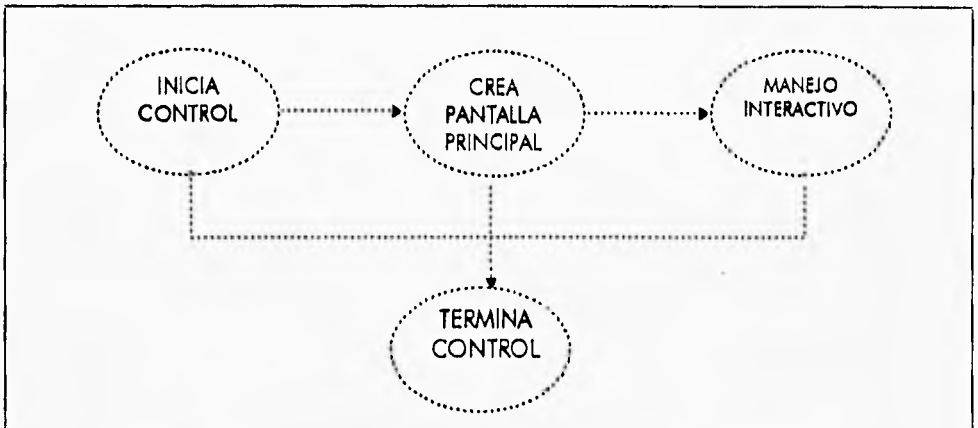


Figura 3.31 Diagrama de transición de estado del objeto CONTROL.

3.2.8 CLASE MECÁNICO

Las funciones de la clase MECÁNICO son:

- ASIGNAK - Con esta función se le asignan valores a la matriz de rigidez [K] local de cada barra para elementos tipo viga.
- ASIGNAT - Se calcula la matriz de traslación [T] de cada barra.
- FACTORIZACIÓN - Para resolver el sistema de ecuaciones {F} = [K]{D}, el primer paso a desarrollar es la factorización de la matriz [K] general del sistema.
- LYB - Encuentra el vector {V} para resolver el sistema de ecuaciones {F} = [K]{D}.
- DLTX - Realiza la sustitución hacia atrás para resolver el sistema {F} = [K]{D}.
- FUERZAS_MIEMBRO - Cálculo de las fuerzas internas de las barras.
- ASIGNA_CONST_F - Asigna valores a las constantes de las barras C1, C2, C3, C4, C5, C6 cuando se lleva a cabo un análisis lineal crítico, para elementos a pura flexión.

- ASIGNA_CONST_FM - Asigna valores a las constantes de las barras C1, C2, C3, C4, C5, C6 cuando se lleva a cabo un análisis lineal crítico, para elementos sujetos a esfuerzos combinados.
- ASIGNA_MÁRGENES - Asignación de los términos de los márgenes de seguridad de los extremos de las barras.
- ASIGNA_CA - Asignación de las cargas aleatorias a un vector.
- INICIAL_MCEL - Inicializa la primera columna de la matriz de coeficientes de los estados límite de las secciones más esforzadas y seleccionadas.
- SELEC_LC - Selecciona las secciones más esforzadas para el análisis lineal crítico.
- SELEC_EC - Selecciona las secciones más esforzadas para el análisis elastoplástico crítico.
- ORDENA - Ordena en orden creciente los vectores que guardan las secciones más esforzadas y los índices lambda.
- CALCULA_MÁRGENES - Encuentra e imprime los términos de los márgenes de seguridad de los extremos de las barras.
- NÚMERO_FALLAS - Calcula el número de fallas de un sistema elastoplástico o elastoplástico crítico.
- MATRK_12 - Forma una matriz de rigidez de miembro reducida: plastificación en el extremo izquierdo de la barra o plastificación en el extremo derecho de la barra.
- MATRIK_3 - Forma una matriz de rigidez de miembro reducida y dos vectores de fuerzas nodales equivalentes, referenciadas a las coordenadas globales del sistema, para barras que están plastificadas en los dos extremos.
- ÍNDICE - Encuentra el índice de confiabilidad para análisis lineal crítico o elastoplástico crítico.
- DESPLAZAMIENTOS - Función donde se calculan los desplazamientos de los nodos.
- FUERZAS_INTERNAS - Obtiene las fuerzas internas de las barras.
- ANALIZAR - Primer paso para realizar cualquier tipo de análisis, se crean e inicializan algunos vectores para cálculos posteriores.
- CREA_AUX1 - Crea un archivo de datos que puede servir de interfaz, para desarrollar el mismo análisis que se lleva a cabo en este sistema con un programa desarrollado en FORTRAN. También sirve para corroborar los resultados obtenidos en los dos programas.

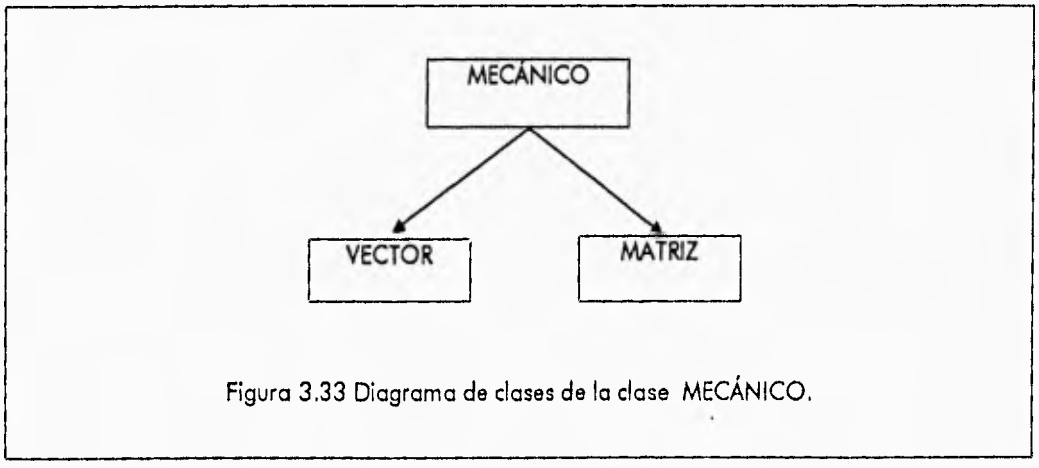


Figura 3.33 Diagrama de clases de la clase MECÁNICO.

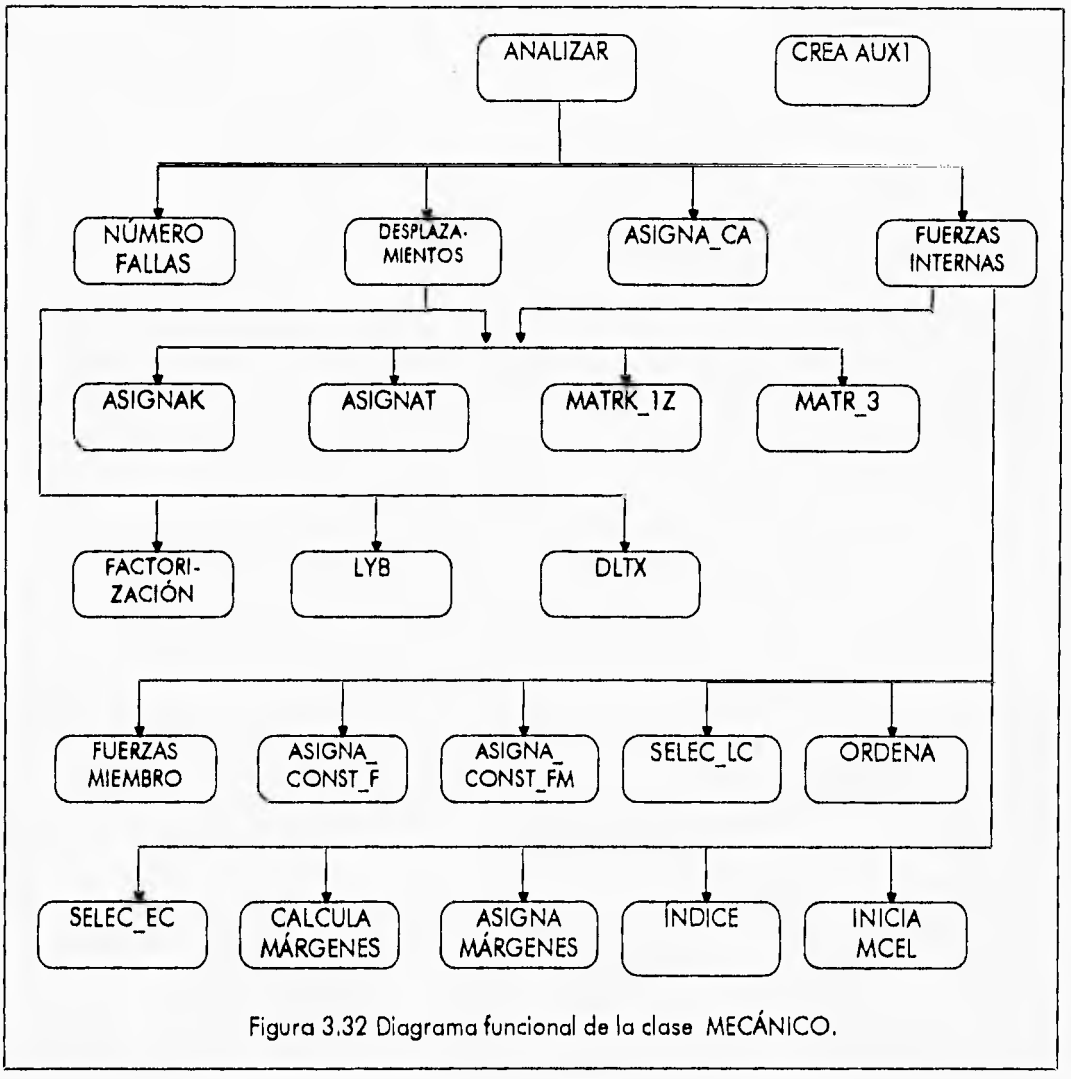


Figura 3.32 Diagrama funcional de la clase MECÁNICO.

Esta clase hace uso de todas las funciones de las clases VECTOR y MATRIZ ya que la mayor parte del análisis mecánico se apoya en el uso de vectores y matrices.

DIAGRAMA DE TRANSICIÓN DE ESTADO DEL OBJETO MECÁNICO.

Los pasos más importantes que lleva a cabo el objeto MECÁNICO son: crea matrices y vectores y se les asigna valores que son indispensables para todo el análisis mecánico. Después se hace el cálculo de los desplazamientos nodales mediante la función DESPLAZAMIENTOS y cuando ya se tienen los desplazamientos nodales se pueden calcular las fuerzas internas que actúan sobre las barras con la función FUERZAS_INTERNAS. Durante el desarrollo del análisis pueden ocurrir errores que tengan como consecuencia que el análisis no se pueda continuar por lo que se debe terminar inmediatamente.

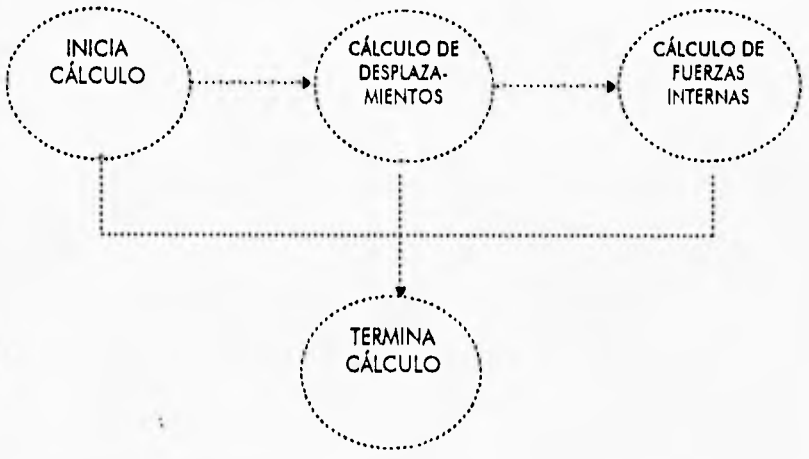


Figura 3.34 Diagrama de transición de estado del objeto MECÁNICO.

3.2.9 CLASE CONFIGURACIÓN DE PANTALLA

La clase CONFIGURACIÓN DE PANTALLA contiene las siguientes funciones:

- PON_LÍMITES - Esta función se encarga de actualizar los valores de los límites mínimos y máximos reales en la pantalla.
- PON_SNAP - Esta función se encarga de actualizar en la pantalla gráfica las escalas de los snaps y el mallador.
- ZOOM_VENTANA - Esta función permite realizar acercamientos de la pantalla de edición.
- ZOOM_PREVIO - Permite regresar a la pantalla anterior después de haber realizado un zoom de ventana.
- ZOOM_TOTAL - Esta función permite ver la pantalla de edición con sus límites mínimos y máximos posibles.
- INIT_SNAP - Permite cambiar la configuración de los snaps y el mallador.
- INIT_LÍMITES - Permite cambiar la configuración de los límites reales de la pantalla de edición.
- ABRIR_CONFIG - Esta función permite cargar de un archivo los valores de la configuración del programa previamente almacenados.
- GUARDAR_CONFIG - Esta función permite guardar en un archivo los valores actuales de la configuración del programa.
- PANTALLA_REAL, SNAP, REDONDEAR - Estas funciones son heredadas de la clase padre CONFIGURACIÓN.

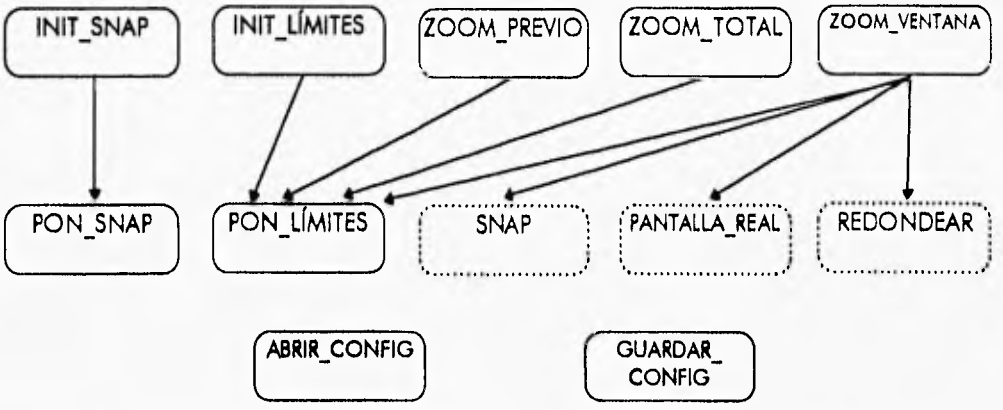


Figura 3.35 Diagrama de funciones de la clase CONFIGURACIÓN DE PANTALLA.

Las clases y funciones que usa la clase CONFIGURACIÓN DE PANTALLA son las siguientes:

- RATÓN
  - OBT_COORDS
  - MOSTRAR
  - OCULTAR
  - MOVER
  - LÍMITES
  - BOTÓN_PRES
  - BOTÓN_REL
- SUBMENÚ
  - CREAR_MENÚ
  - EJECUTA_MENÚ
  - COMPONE
  - BORRA_MENÚ

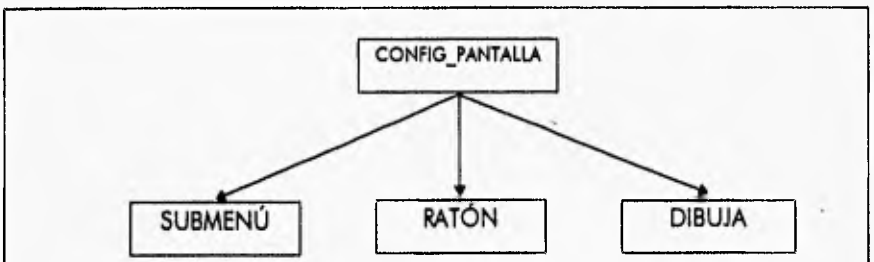


Figura 3.36 Diagrama de la clase CONFIGURACIÓN DE PANTALLA.

DIAGRAMA DE TRANSICIÓN DE ESTADO DEL OBJETO CONFIGURACIÓN DE PANTALLA

El objeto CONFIGURACIÓN DE PANTALLA debe ser inicializado al principio del programa mediante la llamada a la función ABRIR_CONFIG, posteriormente puede ser operado mediante la llamada a las funciones diseñadas para tal efecto, como por ejemplo: ZOOM_VENTANA, PON_LÍMITES, etc. Para finalmente guardar la configuración del programa al finalizar. La figura 3.37 muestra este proceso.

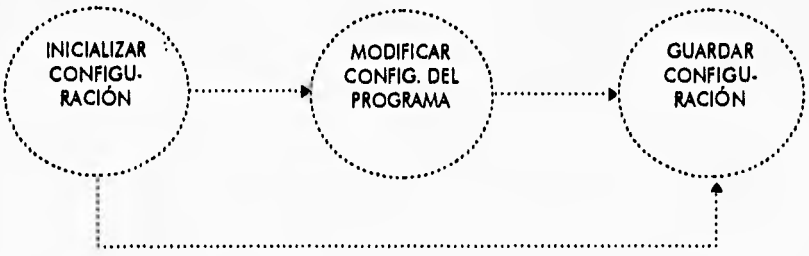


Figura 3.37 Diagrama de transición de estado del objeto CONFIGURACIÓN DE PANTALLA.

3.2.10 CLASE NODOS

La clase NODOS contiene las siguientes funciones:

- AGREGA_NODO - Esta función agrega la información del nodo recién editado en el arreglo dinámico de la memoria del programa.
- ELIMINAR_NODO - Esta función es la inversa de AGREGAR_NODO.
- EDICIÓN_NODOS - Esta función permite editar nodos en la pantalla de edición y asignarles características particulares como son: posición y grado de libertad.
- BORRAR_NODOS - Esta función permite borrar nodos de la estructura en edición o cambiar sus características, ya sea posición o grado de libertad.
- GENERAR_NODOS - Esta función permite generar nodos automáticamente con un cierto orden y una dirección específica.
- SNAP, DIBUJA_NODO, REDIBUJAR, PON_NUM_NODO, REAL_PANTALLA, INIT_FUN, NODO_PRÓXIMO, MARCA_NODO, DESMARCA_NODO - Estas funciones son heredadas de la clase padre DIBUJA.

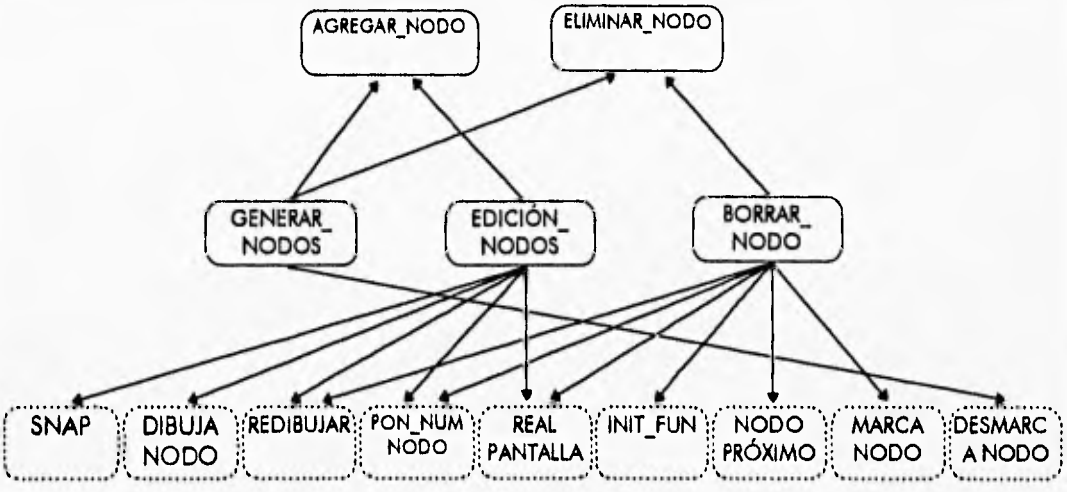


Figura 3.38 Diagrama de funciones de la clase NODOS.

Las clases y funciones que usa la clase NODOS son las siguientes:

- CONTROL
  - INTERACTIVO
- RATÓN
  - OBT_COORDS
  - MOSTRAR
  - OCULTAR
  - LÍMITES
  - BOTÓN_PRES
  - BOTÓN_REL
  - MOVER
- SUBMENÚ
  - CREAR_MENÚ
  - EJECUTA_MENÚ
  - COMPONE
  - BORRA_MENÚ

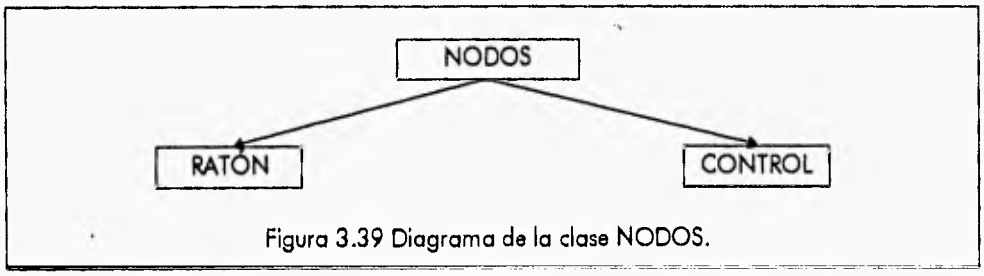


Figura 3.39 Diagrama de la clase NODOS.

DIAGRAMA DE TRANSICIÓN DE ESTADO DEL OBJETO NODO

El objeto NODO básicamente tiene tres opciones diferentes que permiten manipular o editar los nodos de la estructura, por lo que al principio de la creación del objeto se debe especificar el tipo de función a realizar, ya sea GENERAR NODOS, EDITAR NODOS o REEDITAR. a continuación dentro de cada una de estas funciones se realiza el proceso de localizar los nodos más cercanos al ratón para posteriormente borrarlos, moverlos, cambiar el grado de libertad o crear nodos nuevos para finalmente salir del objeto.

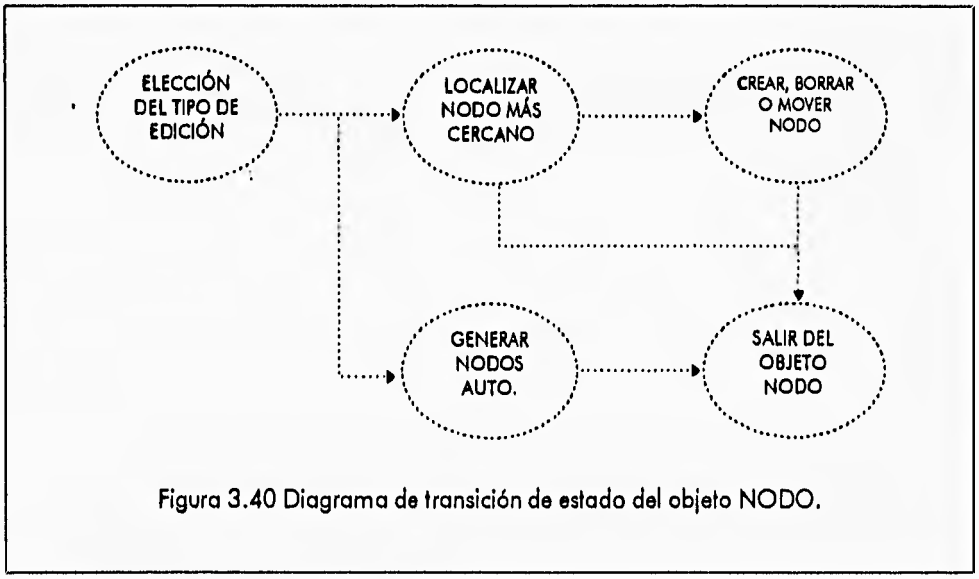


Figura 3.40 Diagrama de transición de estado del objeto NODO.

3.2.11 CLASE BARRAS

Las funciones que contiene la clase BARRAS son:

- AGREGAR_BARRA - Aparta espacio de memoria RAM para crear una nueva barra e inicializa todos los valores de la nueva barra a cero.
- ELIMINAR_BARRA - Libera la memoria RAM ocupada por una barra.
- EDITAR_BARRAS - Permite crear o editar barras ya existentes. Crea el submenú que se visualiza cuando se edita una barra, donde se presentan y asignan características para cada barra; así como manipular todas las acciones que se pueden realizar dentro de ese submenú.
- BORRAR_BARRAS - Función que permite escoger una barra para que sea eliminada de la estructura.

Funciones heredadas de la clase DIBUJA que se utilizan en esta clase:

CHECA_NODOS, INIT_FUN, NODO_PRÓXIMO, MARCA_NODO, DIBUJA_NODO, DESMARCA_NODO, DESMARCA_BARRA, PON_NUM_NODO, REAL_PANTALLA.

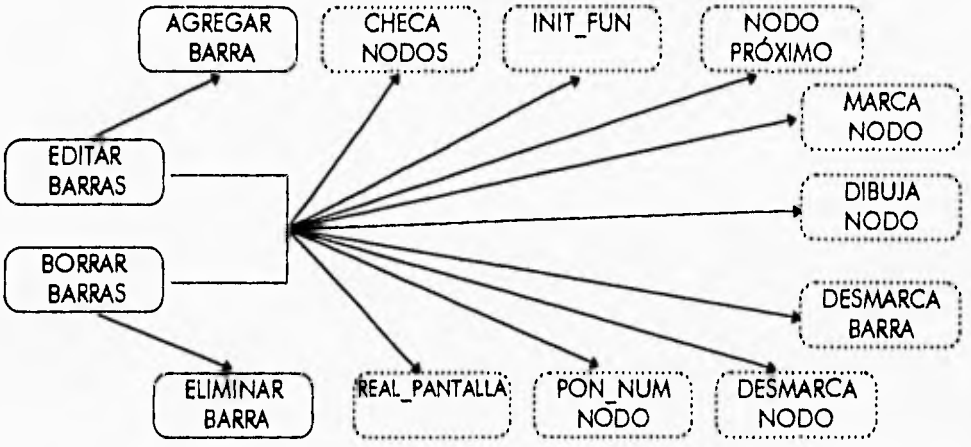


Figura 3.41 Diagrama de funciones de la clase BARRAS.

La clase BARRAS utiliza las siguientes clases para su funcionamiento:

- MENÚ PRINCIPAL
  - SOMBRA
- CONTROL
  - INTERACTIVO
- RATÓN
  - OBT_COORDS
  - MOSTRAR
  - OCULTAR
  - BOTÓN_PRES
  - BOTÓN_REL
- SUBMENÚ
  - CREAR_MENÚ
  - EJECUTA_MENÚ
  - COMPONE
  - BORRA_MENÚ

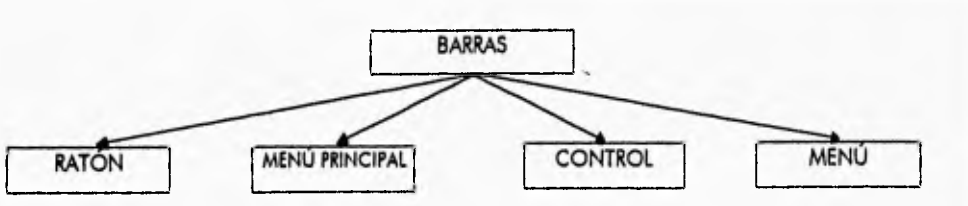


Figura 3.42 Diagrama de clases para la clase BARRAS.

DIAGRAMA DE TRANSICIÓN DE ESTADO DEL OBJETO BARRA

En el proceso de transición de estado del objeto BARRA, el primer paso es crear o editar la barra mediante la función EDITAR_BARRAS, posteriormente se pueden modificar las características de la barra mediante las funciones heredadas y propias de la clase como son: AGREGAR_BARRA, DESMARCA_BARRA, ELIMINAR_BARRA, etc., para finalmente abandonar el objeto.

Figura 3.43 Diagrama de transición de estado del objeto BARRA.

3.2.12 CLASE CARGAS

La clase CARGAS contiene las siguientes funciones:

- FLECHA - Esta función sirve para colocar una flecha al lado de cada nodo de la estructura, si existe una condición de carga presente en el nodo.
- CÍRCULO - Esta función coloca un círculo alrededor de cada nodo de la estructura si existe una carga aleatoria presente en el nodo en cuestión.
- CARGAS - Esta función permite asignar cargas aleatorias y condiciones de carga a cada uno de los nodos que existen en la estructura para su posterior análisis, para poder distinguir entre condiciones de carga o cargas aleatorias se debe especificar en los parámetros que recibe la función.
- CHECA_NODOS, DESMARCA_BARRA, PON_NUM_NODO, REAL_PANTALLA, INIT_FUN, NODO_PRÓXIMO, MARCA_NODO, DESMARCA_NODO - Estas funciones son heredadas de la clase padre DIBUJA.

Las clases y funciones que usa la clase CARGAS son las siguientes:

- CONTROL
  - INTERACTIVO
- RATÓN
  - OBT_COORDS
  - MOSTRAR
  - OCULTAR
  - BOTÓN_PRES

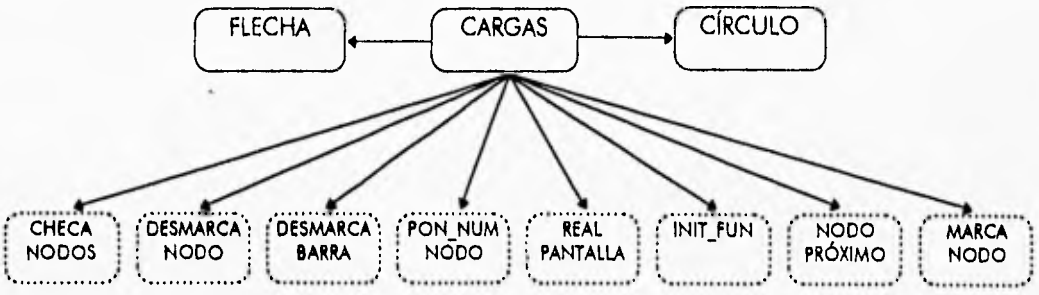


Figura 3.44 Diagrama de funciones de la clase CARGAS.

- SUBMENÚ
  - CREAR_MENÚ
  - EJECUTA_MENÚ
  - COMPONE
  - BORRA_MENÚ

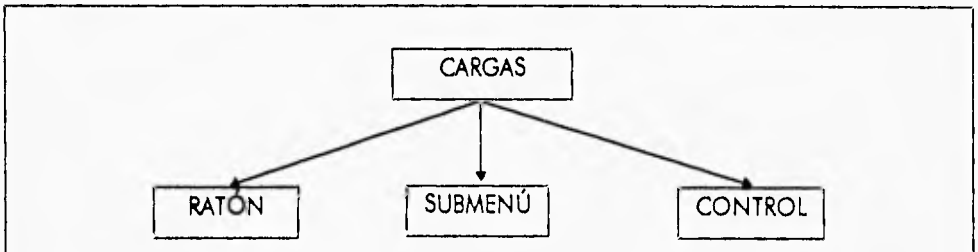


Figura 3.45 Diagrama de la clase CARGAS.

DIAGRAMA DE TRANSICIÓN DE ESTADO DEL OBJETO CARGAS

El objeto CARGAS es un objeto local, es decir se crea en cualquier función del programa y se destruye ahí mismo, y comienza con especificar cual de las 2 opciones se va a utilizar, ya sea editar condiciones de carga o cargas aleatorias, en seguida se indican cuales son los nodos que contiene cargas mediante las funciones FLECHA y CÍRCULO según sea el caso, luego se emplean los procedimientos ya mencionados para localizar los diferentes nodos de la estructura y a continuación se editan las cargas en cada uno de los nodos deseados, para posteriormente abandonar esta opción y regresar a la anterior.

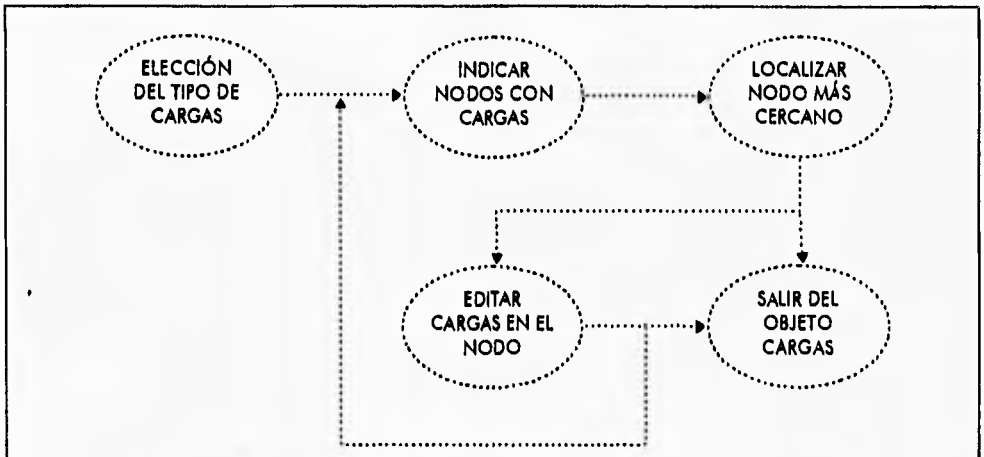


Figura 3.46 Diagrama de transición de estado del objeto CARGAS.

3.2.13 CLASE SOLICITACIÓN

Las funciones de la clase SOLICITACIÓN son:

- SOLICITACIÓN - Constructor de la clase SOLICITACIÓN, que se llama automáticamente al crear un objeto de esta clase e inicializa las variables de la misma.
- SISMO - Calcula las fuerzas que intervendrán en el análisis debido a solicitaciones sísmicas.
- VIENTO - Calcula las fuerzas que intervendrán en el análisis debido a solicitaciones de viento.
- COMPARA - Función que asigna las fuerzas calculadas (sísmicas y de viento) a los nodos que estén en el extremo izquierdo de cada entrepiso.
- PRINCIPAL - Función que crea los menús para el módulo de Solicitaciones.

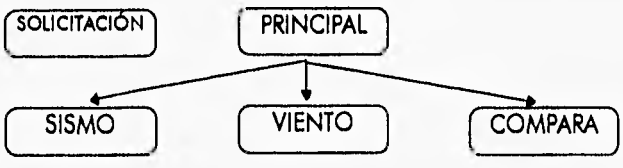


Figura 3.47 Diagrama funcional de la clase SOLICITACIÓN.

SOLICITACIÓN hace uso de la siguiente clase:

- SUBMENÚ

- CREAR_MENÚ
- EJECUTA_MENÚ
- COMPONE
- BORRA_MENÚ

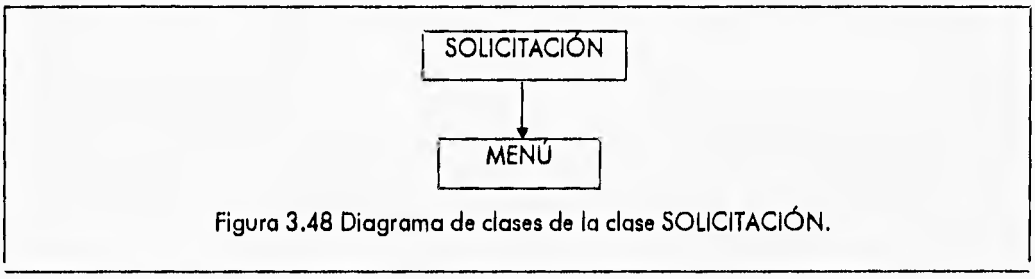


Figura 3.48 Diagrama de clases de la clase SOLICITACIÓN.

DIAGRAMA DE TRANSICIÓN DE ESTADO DEL OBJETO SOLICITACIÓN

El objeto SOLICITACIÓN crea un submenú donde se selecciona el tipo de solicitación y se asignan valores a algunas variables importantes para su cálculo, todo lo anterior mediante la función PRINCIPAL, Dependiendo del tipo de solicitación elegida se realiza el cálculo y posteriormente se asignan las fuerzas obtenidas a ciertos nodos. Una vez hecho lo anterior se termina y abandona el objeto SOLICITACIÓN.

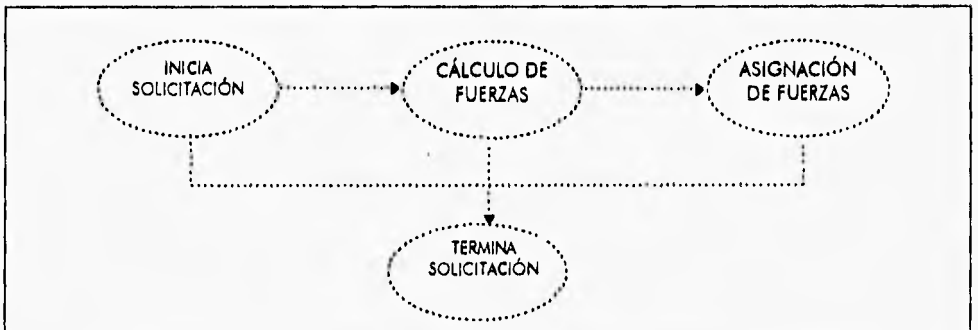


Figura 3.49 Diagrama de transición de estado del objeto SOLICITACIÓN.

3.2.14 CLASE SIMULACIÓN

Las funciones de la clase SIMULACIÓN son:

- INICIA_CÁLCULO - Primera función que se llama al hacer el análisis de confiabilidad, se crean vectores y matrices y se les asignan valores para posteriores cálculos.

- DESVIACIÓN_ESTÁNDAR - Encuentra la desviación estándar del vector de resistencias de las barras.
- COVARIANZA_RES - Calcula la matriz de covarianza para las resistencias de las barras.
- CAL_INDSEG - Cálculo del vector de índices de seguridad de estado límite.
- MARGINAL_Z - Cálculo del vector de probabilidades marginales Z.
- ORDEMA - Ordena de menor a mayor el vector de estado límite de seguridad.
- PROFA - Realiza el análisis de confiabilidad por simulación Monte Carlo.
- VERIFI - Función para verificar si los modos de falla están ordenados de mayor a menor de acuerdo a la probabilidad de falla marginal.
- RAND - Genera números aleatorios entre 0 y 1.
- PNMAYOR4 - Función para calcular la función normal de probabilidad si el valor absoluto de la variable "Z" es más grande o igual que 4.
- PNMENOR4 - Función para calcular la función normal de probabilidad si el valor absoluto de la variable "Z" es menor que 4.
- INV - Cálculo para encontrar la inversa de una matriz.
- MEDIAS - Función para calcular la media condicional.
- VARIAS - Función para calcular la varianza condicional.
- INBETA - Función para calcular el índice beta de la probabilidad de ruina acumulada.

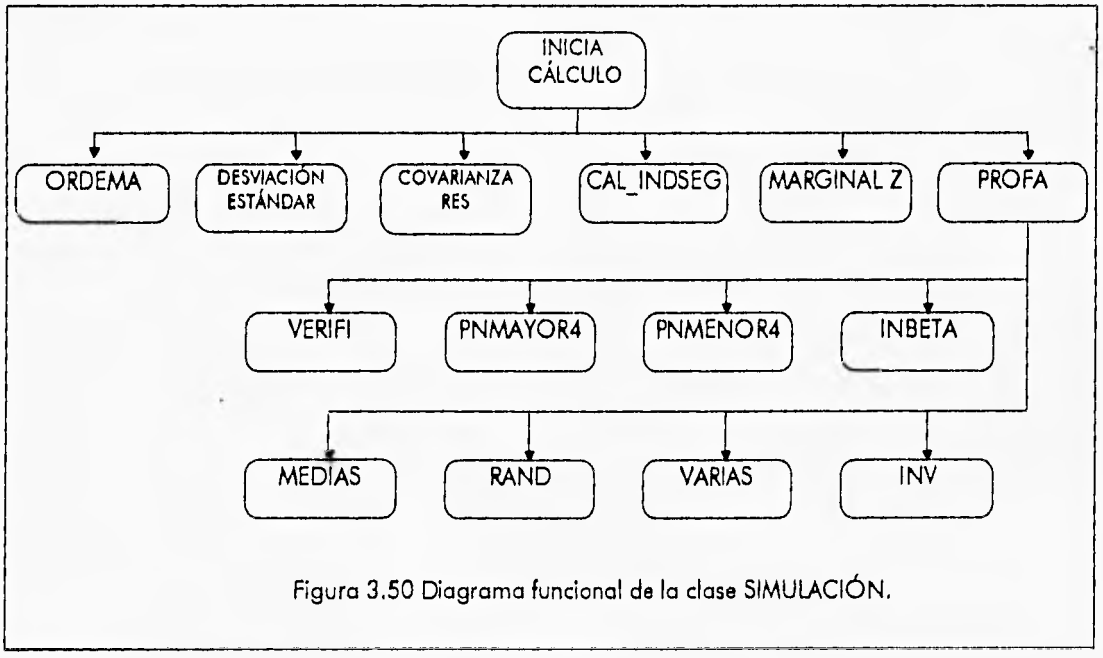


Figura 3.50 Diagrama funcional de la clase SIMULACIÓN.

Esta clase utiliza todas las funciones de las clases VECTOR y MATRIZ, ya que el análisis de simulación Monte Carlo se basa en objetos de tal tipo.

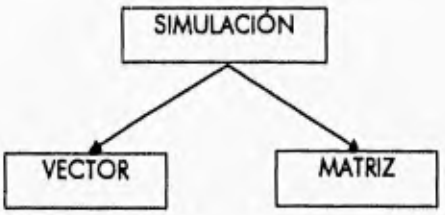


Figura 3.51 Diagrama de clases de la clase SIMULACIÓN.

DIAGRAMA DE TRANSICIÓN DE ESTADO DEL OBJETO SIMULACIÓN

El objeto SIMULACIÓN primero debe crear algunos vectores y matrices y asignarles ciertos valores con la función INICIA_CÁLCULO antes de pasar al cálculo de simulación Monte Carlo en sí. En cualquier etapa del análisis se puede generar algún error que no permita continuar, por lo tanto se debe terminar el análisis.

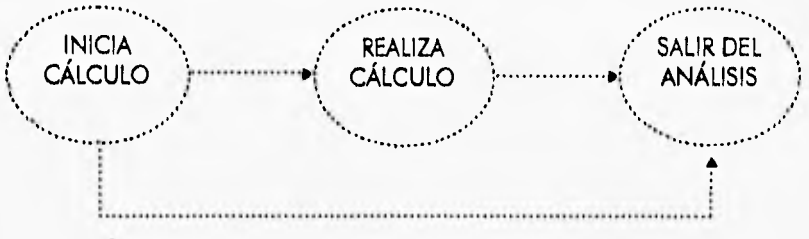


Figura 3.52 Diagrama de transición de estado del objeto SIMULACIÓN.

3.2.15 CLASE VISUALIZACIÓN

La clase VISUALIZACIÓN contiene las siguientes funciones:

- BARRA_DEFORMADA - Esta función gráfica la barra deformada de cada una de las barras de la estructura y es llamada por la función DEFOR para generar toda la deformada de la estructura completa así como aplicar el factor de curvatura y desplazamiento a la deformada.
- DEFOR - Esta función genera la deformada de la estructura completa y llama a BARRA_DEFORMADA para tal objetivo.

- INTERNAS - Esta función permite visualizar las fuerzas internas y los esfuerzos [illegible] de la estructura después de haberse realizado el análisis.
- DEFORMADA - Esta función permite visualizar la forma deformada de la estructura y obtener los desplazamientos de cada uno de los nodos.
- MECANISMOS - Esta función permite visualizar todos los mecanismos almacenados hasta un momento específico.
- CHECA_NODOS, DESMARCA_BARRA, PON_NUM_NODO, REAL_PANTALLA, [illegible] NODO_PRÓXIMO, MARCA_NODO, DESMARCA_NODO, DIBUJA_NODO. Estas funciones son heredadas de la clase padre DIBUJA.

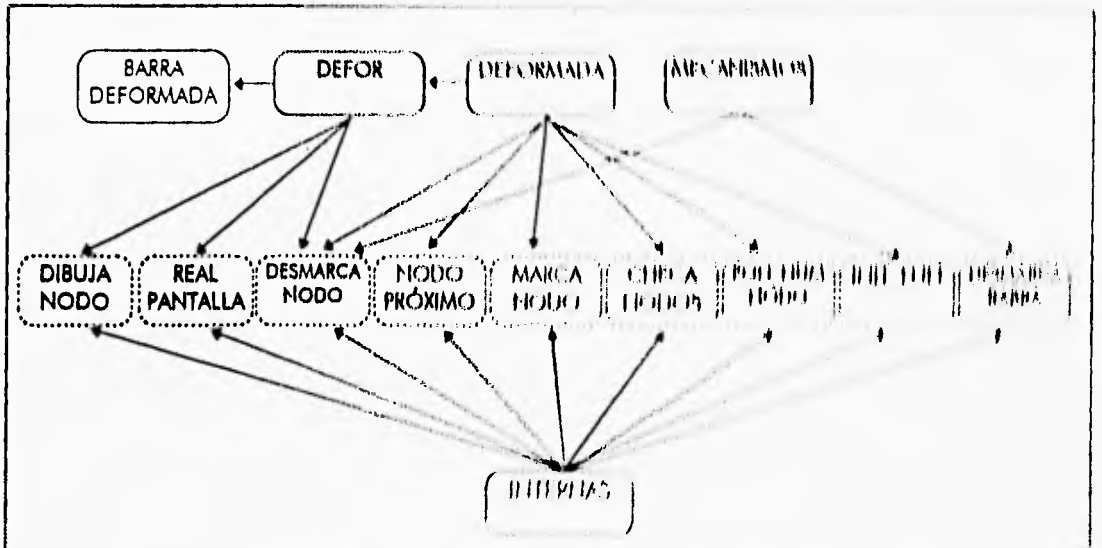


Figura 3.53 Diagrama de funciones de la clase [illegible]

Las clases y funciones que usa la clase [illegible] son las siguientes:

- CONTROL
  - INTERACTIVO
- RATON
  - OBT_COORDS
  - MOSTRAR
  - OCULTAR
  - MOVER
  - BOTON_PRES
  - BOTON_REL
  - LIMITES
- SUBMENU
  - CREA_MENU

- INTERNAS - Esta función permite visualizar las fuerzas internas y los índices beta de la estructura después de haberse realizado el análisis.
- DEFORMADA - Esta función permite visualizar la barra deformada de la estructura y observar los desplazamientos de cada uno de los nodos.
- MECANISMOS - Esta función permite visualizar tados los mecanismos almacenados hasta un momento específico.
- CHECA_NODOS, DESMARCA_BARRA, PON_NUM_NODO, REAL_PANTALLA, INIT_FUN, NODO_PRÓXIMO, MARCA_NODO, DESMARCA_NODO, DIBUJA_NODO - Estas funciones son heredadas de la clase padre DIBUJA.

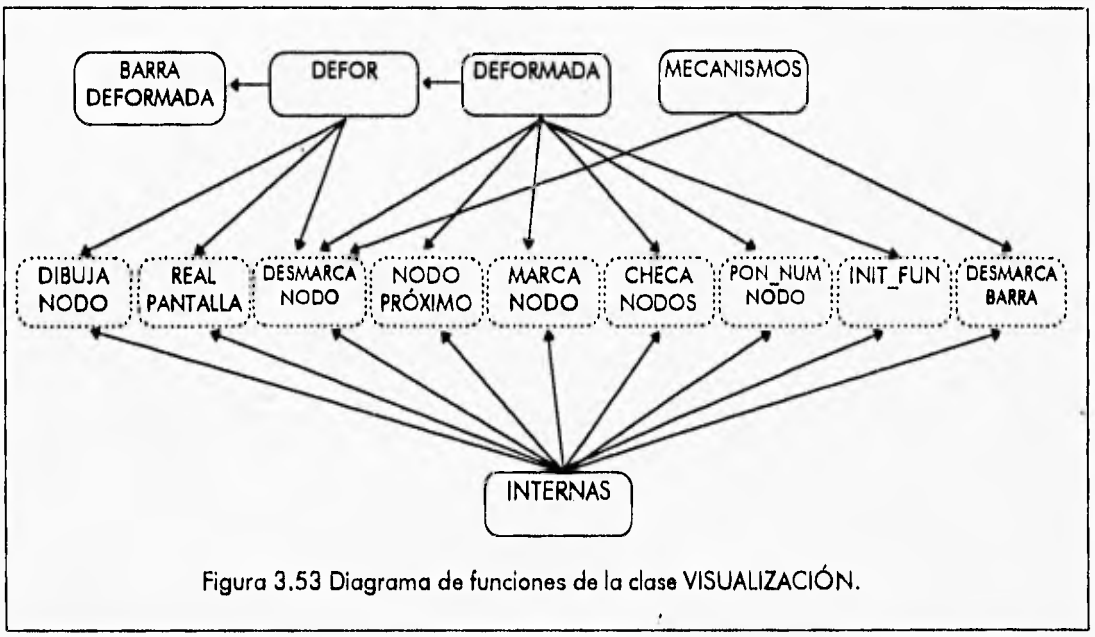


Figura 3.53 Diagrama de funciones de la clase VISUALIZACIÓN.

Las clases y funciones que usa la clase VISUALIZACIÓN son las siguientes:

- CONTROL
  - INTERACTIVO
- RATÓN
  - OBT_COORDS
  - MOSTRAR
  - OCULTAR
  - MOVER
  - BOTÓN_PRES
  - BOTÓN_REL
  - LÍMITES
- SUBMENÚ
  - CREAR_MENÚ

- EJECUTA_MENÚ
- COMPONE
- BORRA_MENÚ

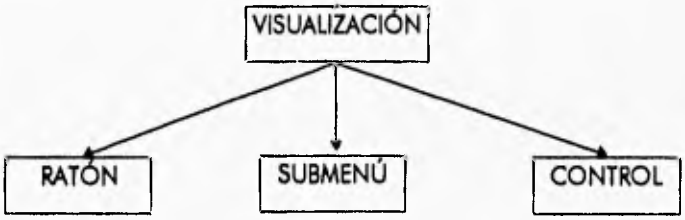


Figura 3.54 Diagrama de la clase VISUALIZACIÓN.

DIAGRAMA DE TRANSICIÓN DE ESTADO DEL OBJETO VISUALIZACIÓN

El objeto VISUALIZACIÓN posee tres métodos diferentes de acceso: DESPLAZAMIENTOS, FUERZAS INTERNAS y MECANISMOS, los tres poseen los mismos métodos internos para localizar nodos, barras y cambiar las condiciones de carga aplicadas y así observar los resultados. Primeramente debemos seleccionar el método a operar de los tres mencionados, luego seleccionar un nodo o una barra según sea el caso para observar los resultados, para finalmente salir del objeto.

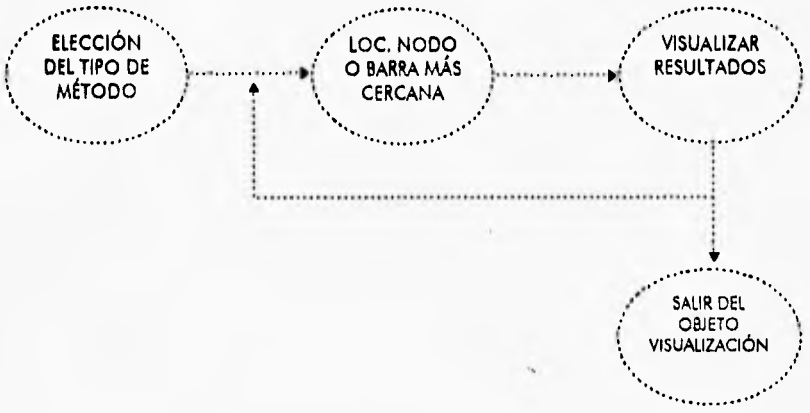


Figura 3.55 Diagrama de transición de estado del objeto VISUALIZACIÓN.

3.2.16 CLASE AYUDA

Las funciones de la clase AYUDA son la siguientes:

- MENÚ_AYUDA - Esta función crea el submenú del módulo de Ayuda y permite seleccionar una de las opciones que ahí se encuentran: ÍNDICE, BUSCAR TÓPICO, ACERCA DE.
- ÍNDICE - Esta función despliega en la pantalla una lista de los tópicos de ayuda disponibles en el programa para que el usuario los pueda consultar y poder resolver algunas dudas que se pudieran presentar en el transcurso de operación del programa.
- VISUALIZA_AYUDA - Esta función despliega en la pantalla gráfica el texto correspondiente al tópico de ayuda seleccionado, ya sea mediante la tecla F1 o por medio del índice de la ayuda.
- LOCALIZA, AGREGA, EJECUTA_MENÚ, CREAR_MENÚ, BORRA_MENÚ - Estas funciones son heredadas de la clase padre VER DIRECTORIO.

Las clases y funciones que usa la clase AYUDA son las siguientes:

- CONTROL
  - CHECA_DRIVE
- RATÓN
  - OBT_COORDS

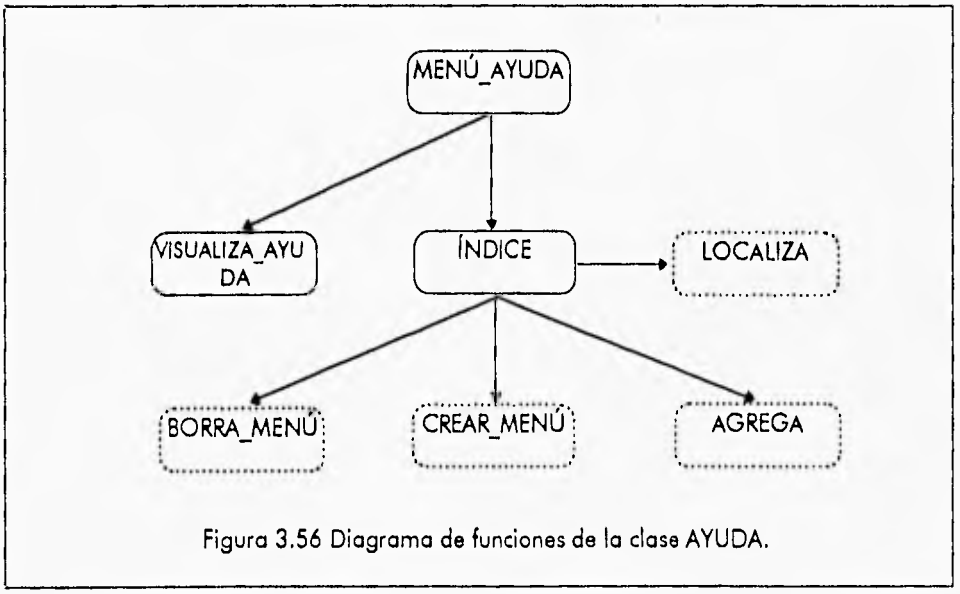


Figura 3.56 Diagrama de funciones de la clase AYUDA.

- MOSTRAR
- OCULTAR
- CAJA
- BOTÓN_PRES

    - BOTÓN_REL
- SUBMENÚ
    - CREAR_MENÚ
    - EJECUTA_MENÚ
    - COMPONE
    - BORRA_MENÚ
- BUFFER
    - GETIMAGE_FILE
    - PUTIMAGE_FILE

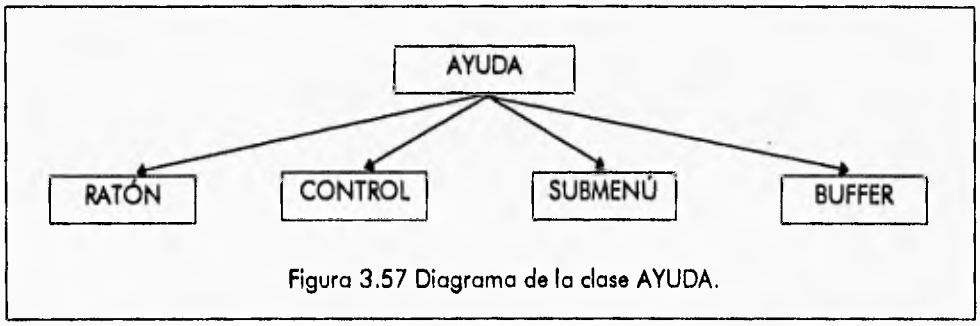


Figura 3.57 Diagrama de la clase AYUDA.

DIAGRAMA DE TRANSICIÓN DE ESTADO DEL OBJETO AYUDA

El proceso de transición de estado del objeto AYUDA es como sigue: primero se selecciona una de las opciones del menú de la ayuda mediante la función MENÚ_AYUDA, posteriormente se procede a visualizar los textos de la ayuda de la misma forma en que ocurre en VER DIRECTORIO y finalmente se borra el recuadro de la ayuda para salir del objeto.

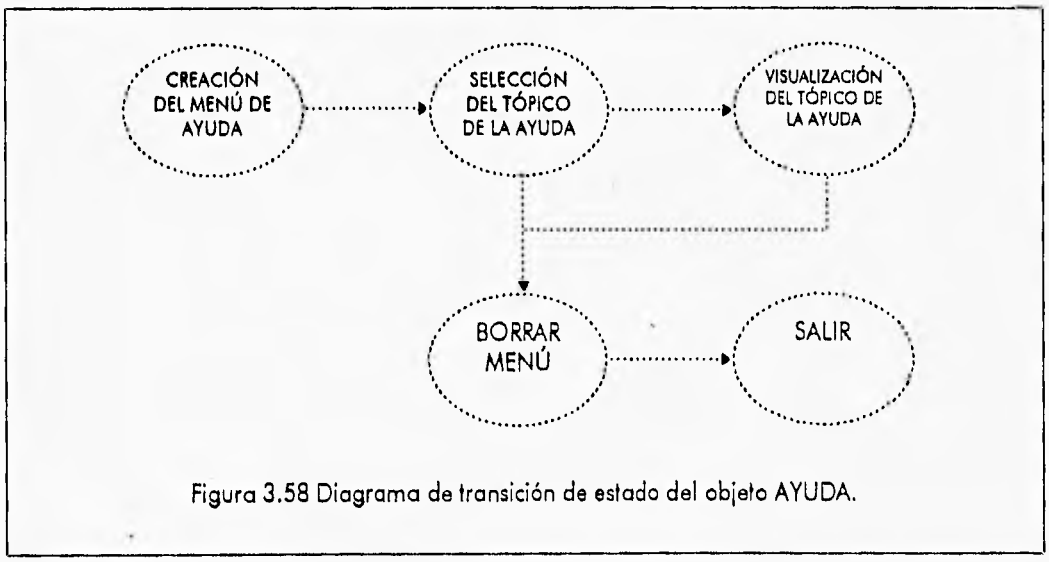


Figura 3.58 Diagrama de transición de estado del objeto AYUDA.

3.2.17 CLASE SUBMENÚ

La clase SUBMENÚ consta de las siguientes funciones:

- TECLA_OPRIMIDA - Esta función se encarga de validar las teclas oprimidas durante la edición de textos dentro de los submenús en la pantalla gráfica.
- CELDA_TXT - Esta función se encarga de editar celdas de tipo texto dentro de un submenú.
- CELDA_NUM - Esta función se encarga de editar celdas de tipo numérico dentro de un submenú.
- COMPONE_CELDA - Esta función se encarga de actualizar las diferentes opciones del menú.
- GUARDA_MENÚ, SOMBRA - Estas funciones son heredadas de la clase padre MENÚ PRINCIPAL.
- TEXTO, MENÚ_SEL, EJECUTA_MENÚ, CREAR_MENÚ, BORRA_MENÚ - Estas funciones están sobrecargadas de la clase padre MENÚ PRINCIPAL y desarrollan las mismas funciones.

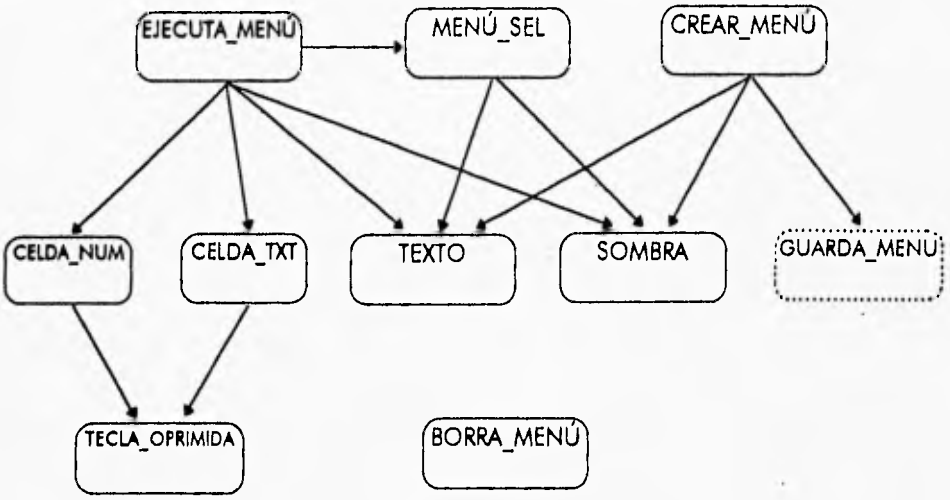


Figura 3.59 Diagrama de funciones de la clase SUBMENÚ.

Las clases y funciones que usa la clase SUBMENÚ son las siguientes:

- RATÓN
  - TECLA
  - OBT_COORDS
  - MOSTRAR
  - OCULTAR

  - CAJA
  - BOTÓN_PRES
  - BOTÓN_REL
- BUFFER
  - GETIMAGE_FILE
  - PUTIMAGE_FILE

DIAGRAMA DE TRANSICIÓN DE ESTADO DEL OBJETO SUBMENÚ

En el proceso de operación del objeto SUBMENÚ se debe de crear primeramente el menú mediante la función CREAR_MENÚ, posteriormente ejecutarlo y elegir una opción con las funciones EJECUTA_MENÚ y MENÚ_SEL, después si es el caso se deberá introducir información en la opción elegida, para finalmente eliminar el submenú de la pantalla con la función BORRAR_MENÚ y proceder a ejecutar otras operaciones.

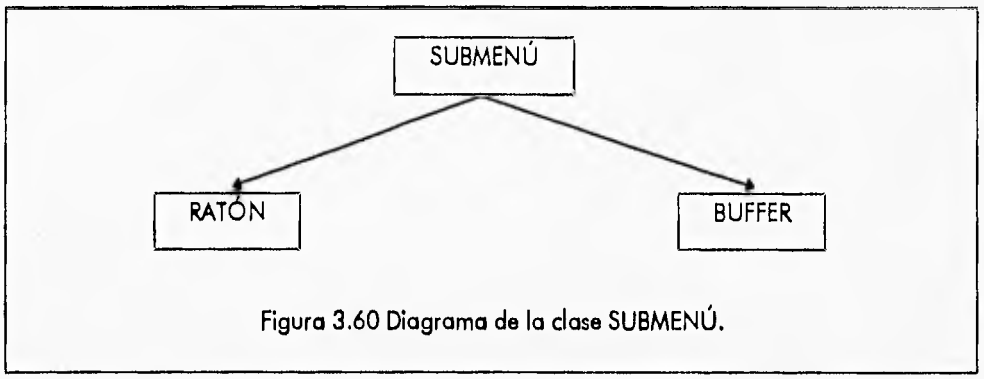


Figura 3.60 Diagrama de la clase SUBMENÚ.

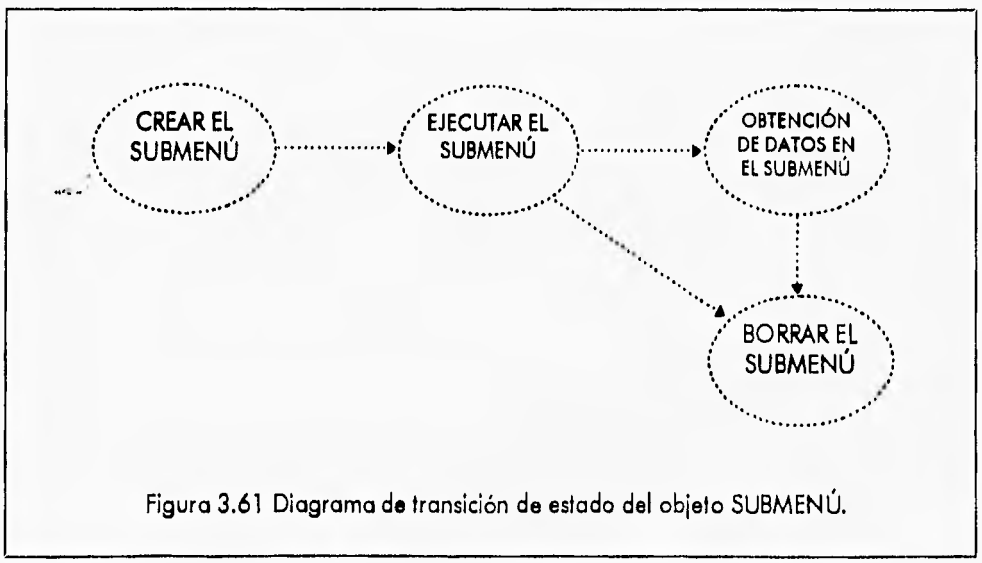


Figura 3.61 Diagrama de transición de estado del objeto SUBMENÚ.

3.2.18 CLASE CONFIGURACIÓN

La clase CONFIGURACIÓN contiene las siguientes funciones:

- REDONDEAR_INT - Esta función redondea cualquier cantidad al número entero más cercano.
- REDONDEAR - Esta función redondea cualquier cantidad a un número de simple precisión con 2 dígitos decimales.
- PANTALLA_REAL - Esta función se emplea para convertir las coordenadas de la pantalla gráfica (pixeles), en coordenadas reales en metros.
- REAL_PANTALLA - Esta función es la inversa de PANTALLA_REAL.
- SNAP - Esta función sirve para colocar el apuntador del ratón en un punto específico que corresponda a la escala del snap activado.
- GRID - Esta función pone en pantalla el mallador que sirve como referencia para el uso de las herramientas de edición de la estructura.

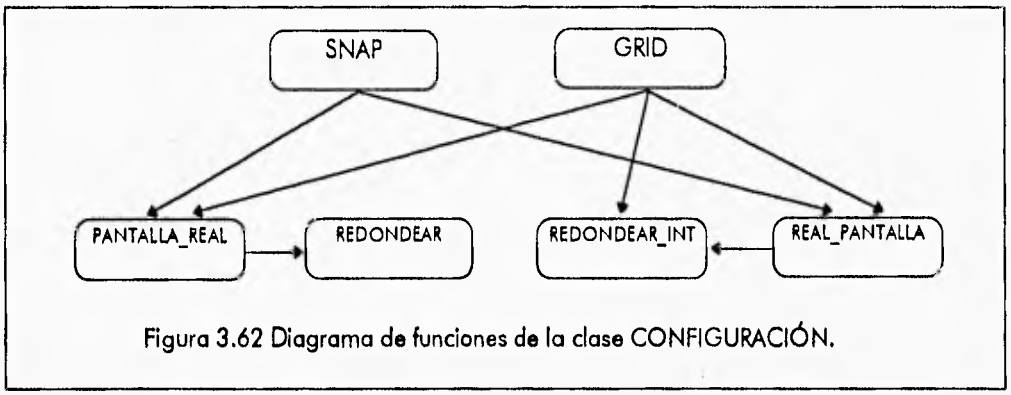


Figura 3.62 Diagrama de funciones de la clase CONFIGURACIÓN.

Las clases y funciones que usa la clase CONFIGURACIÓN son las siguientes:

- RATÓN
  - OBT_COORDS
  - MOSTRAR
  - OCULTAR
  - MOVER
  - LÍMITES
  - BOTÓN_PRES
  - BOTÓN_REL
- SUBMENÚ
  - CREAR_MENÚ
  - EJECUTA_MENÚ
  - COMPONE
  - BORRA_MENÚ

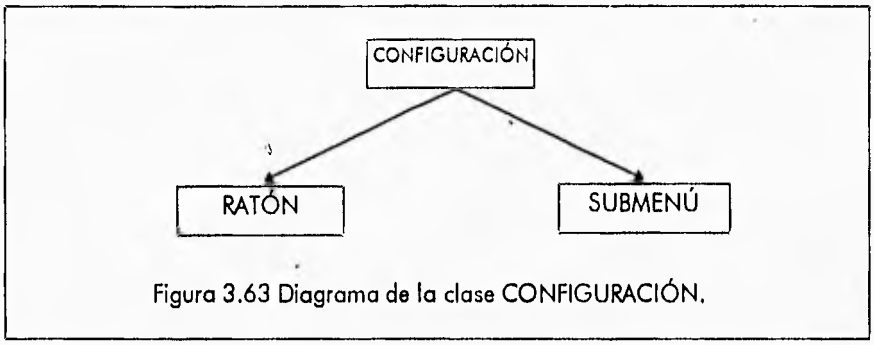


Figura 3.63 Diagrama de la clase CONFIGURACIÓN.

DIAGRAMA DE TRANSICIÓN DE ESTADO DEL OBJETO CONFIGURACIÓN

El objeto CONFIGURACIÓN es un objeto que encapsula las operaciones básicas de conversión de coordenadas y redondeo de números decimales, factoriza las funciones comunes a las clases descendientes de DIBUJA y CONFIGURACIÓN DE PANTALLA. El diagrama de transición de estado del objeto no está bien definido debido a que cualquier función puede ser llamada por cualquier función de los objetos descendientes.

3.2.19 CLASE DIBUJA

La clase DIBUJA contiene las siguientes funciones:

- DIBUJA_NODO - Esta función se encarga de dibujar los nodos de la estructura en la pantalla gráfica.
- PON_NUM_NODO - Esta función pone en la parte inferior el número de nodo que se está editando si es el caso.
- NODO_PRÓXIMO - Esta función localiza tanto en la memoria como en la pantalla el nodo más próximo al apuntador del ratón.
- MARCA_NODO - Esta función marca de cierto color en la pantalla el nodo más cercano al ratón y desmarca el nodo anterior seleccionado.
- INIT_FUN - Esta función se encarga de llamar a las funciones: RATÓN.MOSTRAR, RATÓN.LÍMITES, BARRA_STATUS e inicializa ciertas variables de la clase.
- CHECA_NODOS - Esta función se encarga de checar si existe algún nodo dentro de los límites de la pantalla gráfica.
- DIBUJA_SECCIÓN - Esta función es la encargada de dibujar en los extremos de las barras el tipo de interacción para cada una de las secciones de la barra.
- DESMARCA_NODO - Esta función dibuja de cierto color un nodo en la pantalla, agregándolo a la memoria del programa.
- DESMARCA_BARRA - Esta función dibuja una barra en la pantalla y la agrega a la memoria del programa.
- REDIBUJAR - Esta función redibuja la estructura, así como la pantalla de edición del programa.

- REDONDEAR_INT, PANTALLA_REAL, REAL_PANTALLA, GRID - Estas funciones son heredadas de la clase padre CONFIGURACIÓN.

Las clases y funciones que usa la clase DIBUJA son las siguientes:

- RATÓN
  - MOSTRAR
  - OCULTAR
  - OBT_COORDS
  - MOVER

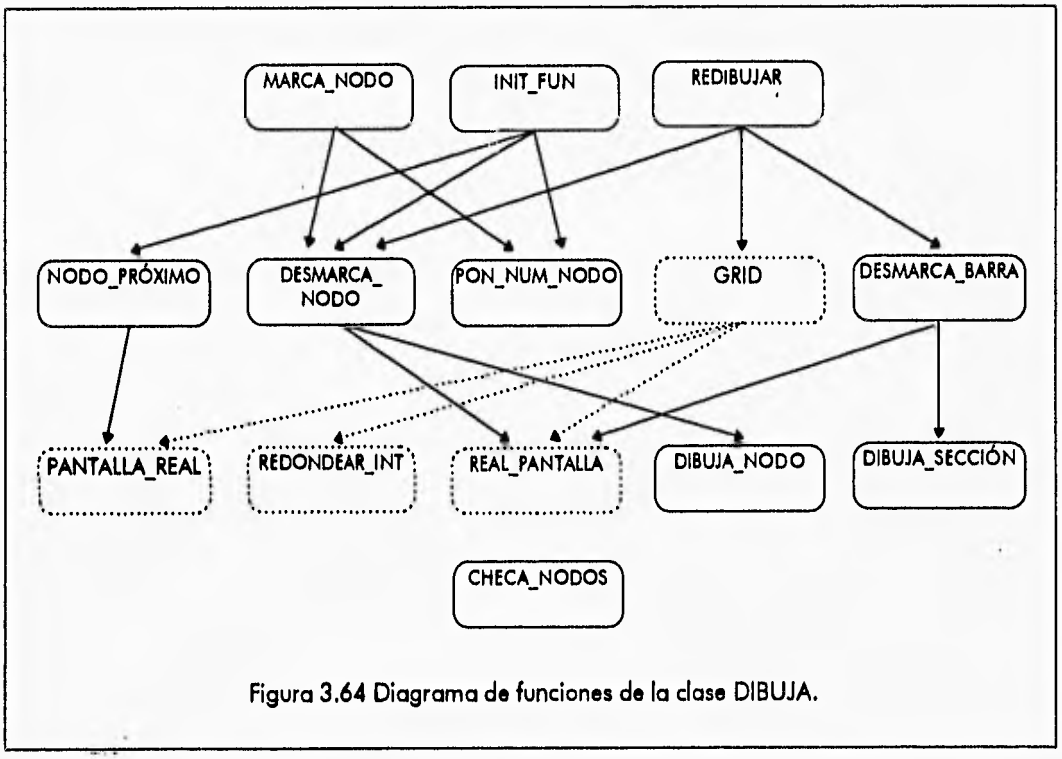


Figura 3.64 Diagrama de funciones de la clase DIBUJA.

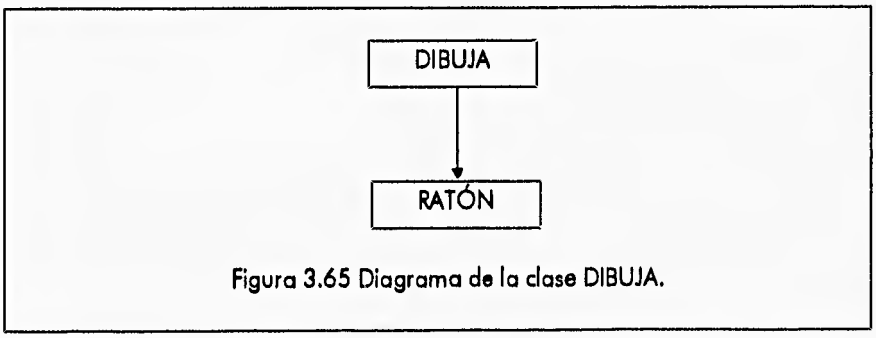


Figura 3.65 Diagrama de la clase DIBUJA.

DIAGRAMA DE TRANSICIÓN DE ESTADO DEL OBJETO DIBUJA

El objeto DIBUJA no tiene un proceso de cambio de estado a lo largo del programa bien definido, debido a que puede ser creado en cualquier parte del programa y únicamente ser llamada una sola función del objeto para realizar cierta operación y después ser destruido.

3.2.20 CLASE VECTOR

Las funciones de la clase VECTOR son:

- VECTOR - Constructor de la clase, que se llama automáticamente al momento de crear un objeto de este tipo.
- CREA_ESPACIO - Aparta memoria RAM para la creación de un VECTOR.
- INICIAL - Le asigna a todo el vector algún valor específico.
- BORRA - Libera la memoria RAM ocupada por el VECTOR.
- ASIGNACIÓN - Función que permite a los objetos de la clase VECTOR usar el operador "=" para poder realizar asignaciones del tipo a=b, donde a y b son vectores.
- DESTRUCTOR - Función que se llama automáticamente al abandonar el alcance donde el vector fue creado, para liberar la memoria ocupada por tal vector.

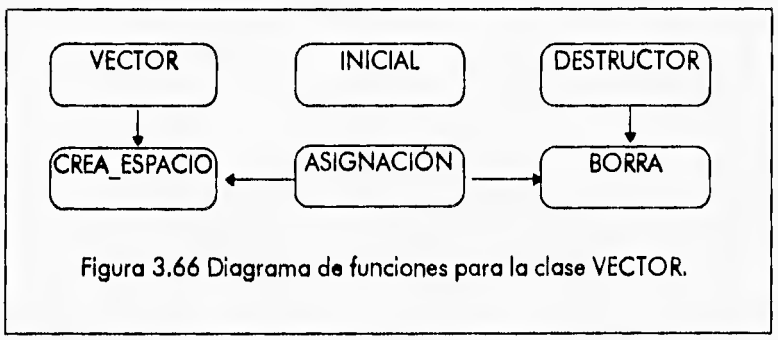


Figura 3.66 Diagrama de funciones para la clase VECTOR.

Esta clase no hace uso de otras clases para su funcionamiento.

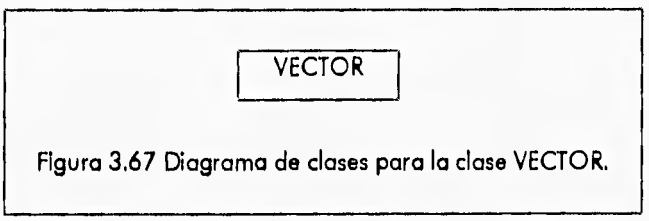


Figura 3.67 Diagrama de clases para la clase VECTOR.

DIAGRAMA DE TRANSICIÓN DE ESTADO DEL OBJETO VECTOR

Los vectores son creados automáticamente al hacer la declaración de objetos de la clase Vector, posteriormente se pueden manipular los vectores asignándoles valores u otros vectores. Cuando se terminan de usar los vectores se deben borrar mediante la función BORRA para liberar la memoria ocupada y usarla para otras operaciones. Cuando se termina de usar una función y no se ha borrado un vector, automáticamente se llama el destructor correspondiente al vector para desocupar la memoria asignada.

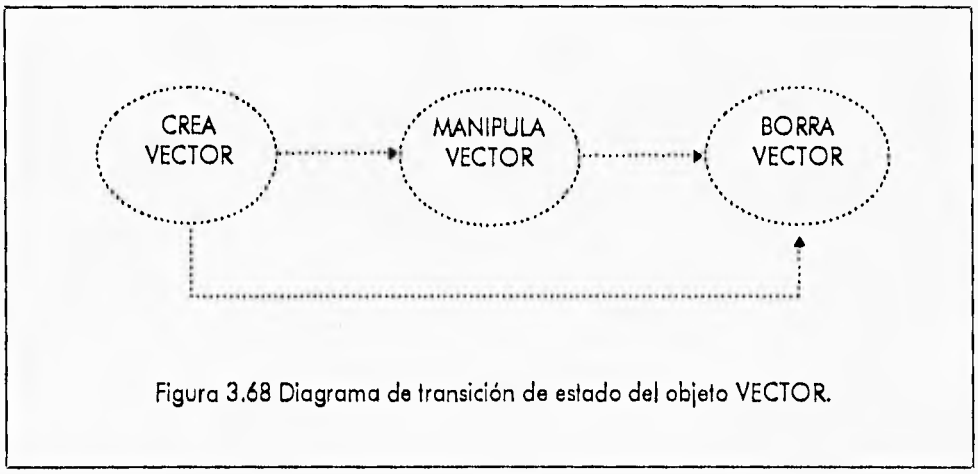


Figura 3.68 Diagrama de transición de estado del objeto VECTOR.

3.2.21 CLASE MATRIZ

Las funciones que implementan a la clase MATRIZ son:

- MATRIZ - Constructor que se llama automáticamente al crear un objeto de esta clase.
- CREA_ESPACIO - Aparta memoria RAM para la creación de una MATRIZ.
- INICIAL - Por defecto la matriz se inicializa con cero, sin embargo se le puede asignar cierto valor a toda la matriz no sólo durante su inicio, si no también en cualquier momento que se requiera.
- BORRA - Libera la memoria RAM ocupada por la matriz, para darle otros usos.
- ASIGNACIÓN - Función que permite a los objetos de la clase MATRIZ usar el operador "=" para poder realizar asignaciones del tipo A=B, donde A y B son matrices.
- SUMA - Función que permite a los objetos de tipo MATRIZ usar el operador "+", para realizar sumas de la forma: A=B+C, A=B+A, A=A+B, A=A+A; donde A, B y C son objetos tipo MATRIZ.
- MULTIPLICACIÓN - Función que permite a los objetos de tipo MATRIZ usar el operador "*", para realizar multiplicaciones de la forma: A=B*C, A=B*A, A=A*B, A=A*A; donde A, B y C son objetos tipo MATRIZ.
- TMULTIPLICACIÓN - Función que permite a los objetos de tipo MATRIZ usar el operador "%", para realizar multiplicaciones de la forma: A=BT*C, A=BT*A, A=AT*B, A=AT*A; donde A,

B, C, AT, BT y CT son objetos tipo MATRIZ y AT, BT, CT, representan sus matrices transpuestas correspondientes.

- TRANSPUESTA - Encuentra la transpuesta de una matriz.
- DESTRUCTOR - Función que se llama automáticamente al abandonar el alcance donde la matriz fue creada, para liberar la memoria ocupada por dicha matriz.

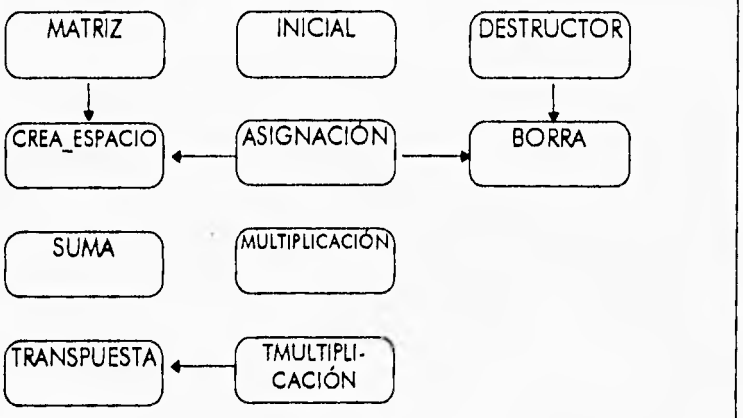


Figura 3.69 Diagrama de funciones para la clase MATRIZ.

Esta clase no hace uso de otras clases para su funcionamiento.

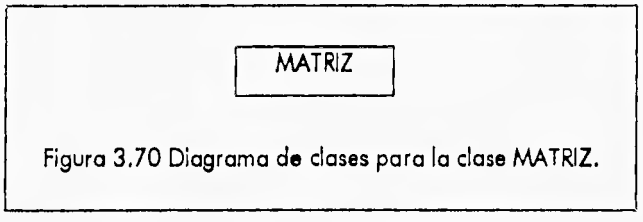


Figura 3.70 Diagrama de clases para la clase MATRIZ.

DIAGRAMA DE TRANSICIÓN DE ESTADO DEL OBJETO MATRIZ

Los objetos de la clase MATRIZ deben ser creados y apartar memoria para su uso antes de intentar cualquier operación sobre ellos, lo anterior se logra automáticamente al hacer su declaración sin tener que usar una función específica. Después de ello se pueden efectuar operaciones tales como suma y multiplicación de matrices, usando los operadores +, *, respectivamente. Cuando ya no sea necesario hacer uso de las matrices creadas deben ser borradas ya sea explícitamente mediante la función BORRA o implícitamente al abandonar la función donde se crearon tales matrices.

Figura 3.71 Diagrama de transición de estado del objeto MATRIZ.

3.2.22 CLASE BUFFER

Las funciones de la clase BUFFER son:

- GETIMAGE_FILE - Cuando la memoria del sistema no es suficiente para guardar en memoria RAM imágenes generadas en la pantalla. Esta función crea un archivo temporal en el disco duro "C" para guardarla.
- PUTIMAGE_FILE - Esta función dibuja en pantalla la imagen guardada en un archivo temporal.

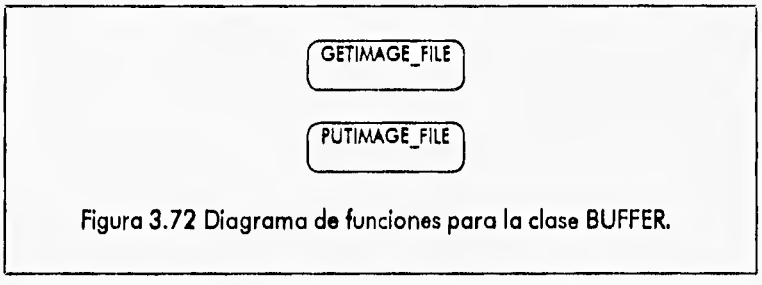


Figura 3.72 Diagrama de funciones para la clase BUFFER.

Esta clase no hace uso de otras clases para su funcionamiento.

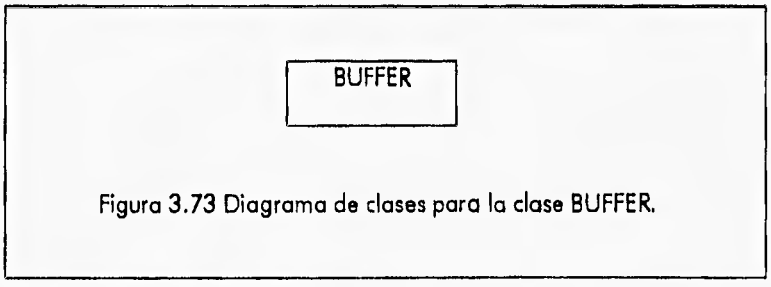


Figura 3.73 Diagrama de clases para la clase BUFFER.

DIAGRAMA DE TRANSICIÓN DE ESTADO DEL OBJETO BUFFER

El objeto BUFFER debe primero GUARDAR la imagen que se desee en un archivo de datos mediante la función GETIMAGE_FILE, para posteriormente leerla y ponerla en pantalla con la función PUTIMAGE_FILE, no se puede hacer uso de esta última función sin antes haber guardado la imagen.

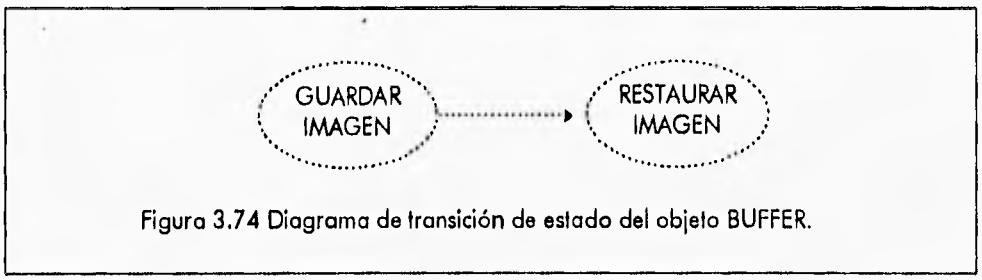


Figura 3.74 Diagrama de transición de estado del objeto BUFFER.

3.2.23 CLASE RATÓN

Descripción de funciones de la clase RATÓN:

- CAJA - Esta función determina si las coordenadas recibidas como parámetros, están contenidas dentro de un rectángulo formado por otro conjunto de parámetros recibidos.
- MOSTRAR - Esta función muestra el apuntador del ratón en la pantalla gráfica.
- OCULTAR - Esta función oculta el apuntador del ratón en la pantalla gráfica.
- MOVER - Mueve el apuntador del ratón a las coordenadas recibidas como parámetros.
- OBT_COORDS - Regresa las coordenadas del apuntador del ratón.
- BOTÓN_PRES - Regresa el estado de los 2 botones del ratón, si se oprimió algún botón regresa un valor verdadero, de lo contrario regresa un valor falso.
- BOTÓN_REL - Igual que el anterior sólo que checa si se liberó algún botón del ratón, funciona de la misma forma que la anterior.
- TECLA - Si se pulsó alguna tecla, esta función regresa un número que relaciona a la tecla que se oprimió.
- RESET - Esta función inicializa el apuntador del ratón y lo coloca en el centro de la pantalla.
- LIMITES - Esta función nos permite poner los límites máximos y mínimos de movimiento del apuntador del ratón.

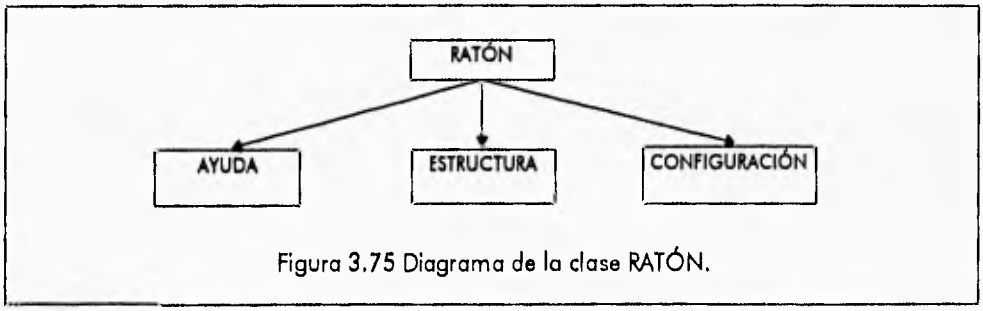


Figura 3.75 Diagrama de la clase RATÓN.

Las clases y funciones que usa la clase RATÓN son las siguientes:

- CONFIGURACIÓN
  - SNAP
- AYUDA
  - VISUALIZA_AYUDA
- ESTRUCTURA
  - INTER_GUARDAR
  - GUARDAR_ESTRUCTURA

DIAGRAMA DE TRANSICIÓN DE ESTADO DEL OBJETO RATÓN

El objeto RATÓN debe ser inicializado al principio del programa, posteriormente puede ser operado mediante la llamada de las funciones diseñadas para tal efecto, como por ejemplo: mostrarlo en pantalla, moverlo a una posición determinada, obtener las coordenadas, etc., para finalmente desaparecerlo de la pantalla al terminar la aplicación. La figura 3.77 muestra este proceso.

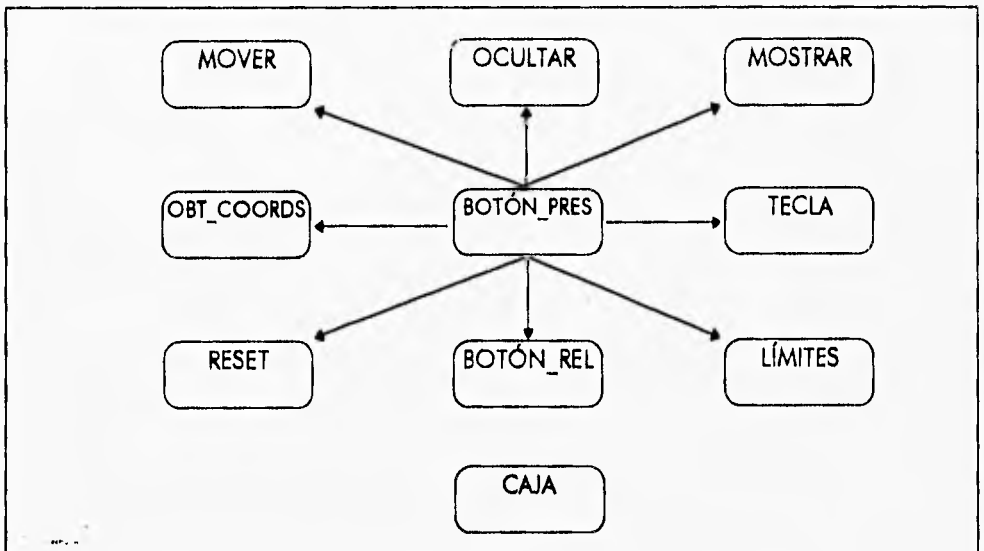


Figura 3.76 Diagrama de funciones de la clase RATÓN.

Figura 3.77 Diagrama de transición de estado del objeto RATÓN.

3.3 IMPLEMENTACIÓN ORIENTADA A OBJETOS DEL SISTEMA CONFIA

La codificación del sistema está basada en el diseño previamente implementado, la explicación de cada una de las funciones y su operación ya ha sido descrita, en esta sección sólo se presentan las estructuras de datos y variables globales del sistema, junto con una breve explicación; también se presentan los prototipos de las 23 clases que conforman el sistema CONFIA, recordando que el listado completo del sistema CONFIA viene en el disco anexo a esta tesis.

Las estructuras de datos generales que se utilizan en el sistema CONFIA son las siguientes:

```
struct Nodos
{
 float x,y;  int gl;
 struct { float FX,FY,MZ; } CARGA[Carga_maxima];
 struct { float DX,DY,GZ; } DESPLA[Carga_maxima];
} *NODO[Nodo_maximo];

struct Barras
{
 int A,B,MATE,SECC;
 float RA,CA,RB,CB;
 float C1,C2,C3,C4,C5,C6,RP;
 struct { float NA,VA,MA,NB,VB,MB; } INTER[Carga_maxima];
 float BETA[2];
 float LANDA[2];
 int  SECB;
} *BARRA[Barra_maxima];
struct { float E,A,IZ,NU,AC,J; } MATE[Mate_maximo];
struct { float ME,CV; int nodo;} CARGA_ALE[Carga_maxima];
```

Las estructuras NODOS y BARRAS nos sirven para almacenar toda la información referente a la estructura que se planea editar en el programa, como por ejemplo nodos, barras, cargas, desplazamientos, fuerzas internas, índices beta, factores lambda, tipos de material que componen la estructura etc.

Las variables globales del sistema que sirven para contabilizar la cantidad de elementos de la estructura son las siguientes:

```
int Num_nodo,
   Num_barra,
   Num_car_ale,
   Num_con_car:
```

Las variables siguientes nos sirven para definir las características específicas que tendrá la estructura, como por ejemplo el tipo de fluencia, el tipo de estructura etc.

```
int  Fluencia,
     Tipo_estruc,
     Tipo_anal_est,
     Cardet,
     Tipvar,
     Tcargas,
     Secfor;
float  Fy;
```

Los prototipos de las 23 clases que componene el sistema CONFIA ordenadas por módulos son las siguientes:

```
class APLICACION
{
 public:
        void MODULO_ARCHIVO();
        void MODULO_MECANICO();
        void MODULO_EDICION();
        void MODULO_SOLICITACION();
        void MODULO_VER();
        void MODULO_AYUDA();
        void MODULO_CONFIABILIDAD();
        void MENU_CARGAS(int,int,int);
        void MENU_VER(int,int,int);
};

class MENUPR
{
   void POS_TEXTO(int,int&,int&,int&,int&);
 protected:
   int izq,der,arriba,abajo;
   void *bitmap1, *bitmap2;
   int bandera_bitmap;
   int GUARDA_MENU(int,int,int,int);
   void TEXTO(int,int,int);
 public:
   int  status;
   void SOMBRA(int,int,int,int,int,int);
   int opcion;
   void MENU_SEL(int,int);
   void EJECUTA_MENU(void);
   void BORRA_MENU(void);
   int  CREAR_MENU(void);
};

class ARCHIVO
{
```

```
  void NUEVO(void);
  void CARGAR_O_GUARDAR(int);
  void CAMBIAR_DIR(void);
  void VER_DIR(void);
  void IMPRIMIR(int,int);
  int  SALIR(int,int);
public:
  void PRINCIPAL(void);
};

class IMPRESORA
{
 public:
  int  ESTADO_IMPRESORA(void);
  void IMPRIMIR_ARCH(void);
  void IMPRIMIR_DES(void);
  void IMPRIMIR_FUE(void);
};

class ESTRUCTURA
{
 public:
 void INTER_GUARDAR();
 int  CARGAR_ESTRUCTURA(char *);
 int  GUARDAR_ESTRUCTURA(char *);
};

class VERDIR : public MENUPR
{
 protected:
   struct VER_DIR
   {
     char archivo[14]; int ind; struct VER_DIR *sig;
   } *ver,*ini,*indicador,*fin;
   int  num_files,pagina_final,pagina;
   void POSICION_ARCHIVO(int);
   void PON_ARCHIVO(void);
   void BARRA(int,int,int);
   void CUADRO(void);
   void BOTON(int,int);
   char *LOCALIZA(int);
   int AGREGA(void);
 public:
   char *DIRECTORIO(char *);
   void EJECUTA_MENU(void);
   int  CREAR_MENU(void);
   void BORRA_MENU(void);
```

```
    void MENU_SEL(void);
};

class CONTROL
{
  void ICONO(char *,int,int);
  void VERIFICA_NUMERODRIVES(void);
  void INITGRAPH(void);
 public:
  CONTROL();
  int total_unidades;
  int num_floppys;
  void PON_TIPO_ESTRUC(void);
  void TIPO_ESTRUC(int,int);
  void TIPO_ANAL(int,int);
  void TIPO_INTER(int,int);
  int INTERACTIVO(void);
  int CHECA_DRIVE(int,int);
  void INTERFACE(void);
  void INIT_GLOBALES(void);
};

class MECANICO
{
 FILE* archivo;
 int seccion_guardar;
 void ASIGNAK(MATRIZ<float> &K,int,float);
 void ASIGNAT(MATRIZ<float> &T,float,float,float);
 int  FACTORIZACION(MATRIZ<float> &akg,int nc);
 void LYB(MATRIZ<float> &ak,VECTOR<float> &fu,int nc1,int L,int tcargas);
 int  DLTX(MATRIZ<float> &akg,VECTOR<float> &fue,int nc1,int L,int tcargas);
 void FUERZAS_MIEMBRO(MATRIZ<float> &T,VECTOR<float> &fue,VECTOR<float>
                      &lm,VECTOR<float> &fb,int ncc);
 void ASIGNA_CONST_F(VECTOR<float> &fb,int nr);
 void ASIGNA_CONST_FM(VECTOR<float> &fb,int nr);
 void ASIGNA_MARGENES(MATRIZ<float> &FBBE,MATRIZ<float> &AUX,int tipo,int seccion,int
                      columna_tomar);
 void ASIGNA_CA(VECTOR<float> &cce);
 void INICIA_MCEL(MATRIZ<float> &cels,int nc1);
 int  SELEC_LC(VECTOR<float> &fb,VECTOR<int> &nsec,VECTOR<float> &flam,int nel);
 int  SELEC_EC(MATRIZ<float> &fbbe, VECTOR<float> &flam, VECTOR<int> &nsec_int,
              MATRIZ<float> &CELS,int tcargas,int paso,int i,int secciones);
 void ORDENA(VECTOR<float> &flam,VECTOR<int> &nsec,float fl1,int nl1,int secciones);
 int  CALCULA_MARGENES(VECTOR<float> &fb,VECTOR<float> &cij,VECTOR<float>
                      &cc,MATRIZ<float> &fbbe,int ncc,int nhl);
 int  NUMERO_FALLAS(void);
 int  MATRK_12(MATRIZ<float> &K,MATRIZ<float> &T,MATRIZ<float> &Fl,int i);
```

```
 int  MATRK_3(MATRIZ<float> &K,MATRIZ<float> &T,MATRIZ<float> &FI, MATRIZ<float>
              &FJ,int i);
 void INDICE(MATRIZ<float> &CELS,VECTOR<int> &nsec,int,int,int);
 int  DESPLAZAMIENTOS(VECTOR<int> &lgl,VECTOR<float> &fue,VECTOR<float> &cc,
                      int grado_libertad,int tcargas);
 int  FUERZAS_INTERNAS(VECTOR<int> &lgl,VECTOR<float> &fue,VECTOR<float>
                      &cc,VECTOR<int> &nsec,
 int tcargas,int secciones);
public:
 int ANALIZAR(int secc_guardar=-1);
 void CREA_AUX1();
};

class CONFIG : public CONFIGURACION
{
   struct
    {
       float ant_Xmin, ant_Ymin, ant_Xmax, ant_Ymax;
    } previo[20];
 public:
   void PON_LIMITES(void);
   void PON_SNAP(void);
   int indice;
   int ZOOM_VENTANA(void);
   void ZOOM_PREVIO(void);
   void ZOOM_TOTAL(void);
   int INIT_SNAP(int,int);
   int INIT_LIMITES(int,int);
   int  ABRIR_CONFIG(void);
   void GUARDAR_CONFIG(void);
};

class NODOS : public DIBUJA
{
 public:
    int  AGREGAR_NODO(int);
    void ELIMINAR_NODO(int);
    void EDICION_NODOS(void);
    void BORRAR_NODOS(void);
    void GENERA_NODOS(void);
};

class BARRAS : public DIBUJA
{
 public:
   int AGREGAR_BARRA(int);
   void ELIMINAR_BARRA(int);
```

```
    void EDITAR_BARRAS(void);
    void BORRAR_BARRAS(void);
};

class CARGAS : public DIBUJA
{
    void FLECHA(int,int);
    void CIRCULO(int,int);
 public:
    int  EDITAR_CARGAS(int,int);
};

class SOLICITACION
{
struct
  {
     int z, n;
     float q;
  } VAR_SISMO;
  struct alp
  {
     float w, h;
  } *ALT_PES[100];
  struct
  {
    char ge,ze;
    int nni,nma;
    float alt,ant;
  } VAR_VIENTO;
  float p[100];
  void SISMO(void);
  void VIENTO(void);
  void COMPARA(int);
  public:
    SOLICITACION();
    void PRINCIPAL();
};

class SIMULACION
{
 public:
   void INICIA_CALCULO(double,double,double&,double&);
   double pf,vpf;
 private:
   void DESVIACION_ESTANDAR(VECTOR<double> &A,MATRIZ<double> &B,
                            VECTOR<double> &C,int);
  void COVARIANZA_RES(VECTOR<double>&A,MATRIZ<double>&B,MATRIZ<double>&C,int);
```

```
  int CAL_INDSEG(MATRIZ<double> &A,MATRIZ<double> &B, int, VECTOR<double> &C);
  void MARGINAL_Z(VECTOR<double> &A,int, VECTOR<double> &C);
  void ORDEMA(MATRIZ<double> &A,MATRIZ<double> &B, VECTOR<double> &C,
              MATRIZ<double>  &D,VECTOR<double> &E,int);
  int PROFA(MATRIZ<double> &A,VECTOR<double> &B,int,int,int,double,double&,int);
  int VERIFI(int, VECTOR<double> &A, MATRIZ<double> &B);
  double RAND(void){return (random(32001)*3.1249e-05);};
  void PNMAYOR4(double, double &PZ);
  void PNMENOR4(double, double &);
  int INV(MATRIZ<double> &A, int);
  void MEDIAS(double &AMEC,MATRIZ<double>&A,VECTOR<double>&B,VECTOR<double>
              &C,int);
  void VARIAS(double &VARIA, MATRIZ<double> &A, MATRIZ<double> &B, int);
  int INBETA(double, double &BIDB);
};

class VISUALIZACION : public DIBUJA
{
   void BARRA_DEFORMADA(int,int,int,int,double,double,double);
   void DEFOR(int,double,double);
 public:
   int  INTERNAS(int,int,int);
   int  DESPLAZA(int,int,double,double);
   int  MECANISMOS(int,int);
};

class AYUDA : public VERDIR
{
 void INDICE(void);
 public:
 void VISUALIZA_AYUDA(char *);
 void MENU_AYUDA(void);
};

class MENU : public MENUPR
 {
    class CELDA
    {
        int  pos_cur,ancho,x_cel,y_cel;
        char texto[50],cursor[2];
        int  TECLA_OPRIMIDA(int);
       public:
         double CELDA_NUM(double,int,int,int,char *);
         char  *CELDA_TXT(char *,int,int,int);
    };
    int long_celda,menu_tipo,posx,posy,num_menus,espacios;
    int posicion[9],formato_int[9];
```

```
    char *texto[9],*formato[9];
    void TEXTO(int,int);
  public:
    double cel_num[9];
    char cel_txt[9][45];
    void COMPONE(int);
    int menu_x,menu_y;
    void MENU_SEL(void);
    void EJECUTA_MENU(void);
    void BORRA_MENU(void);
    int  CREAR_MENU(int, int, int, int, int, char *[], char *[], int []);
};

class CONFIGURACION
{
 protected:
   int  REDONDEAR_INT(double);
   double REDONDEAR(double);
   void PANTALLA_REAL(int,int);
   void REAL_PANTALLA(void);
   int Xp,Yp;
   double Xr,Yr;
 public:
   void SNAP(int&,int&);
   void GRID(void);
};

class DIBUJA : public CONFIGURACION
{
 protected:
      void DIBUJA_NODO(int,int,int,int);
      void PON_NUM_NODO(int);
      int  NODO_PROXIMO(void);
      void MARCA_NODO(int);
      void INIT_FUN(void);
      int  CHECA_NODOS(void);
      void DIBUJA_SECCION(int,int,int,int,int,int);
 public:
      void DESMARCA_NODO(int,int);
      void DESMARCA_BARRA(int,int);
      void REDIBUJAR(void);
      static int bandera_betas;
};

template <class T>
class VECTOR
{
```

```
  static contador;
  int identificador;
  int tam;
  void CREA_ESPACIO(int dimension);
  public:
   T *vec;
   VECTOR(int dimension);
   VECTOR(const VECTOR &vector_der);
   ~VECTOR(void);
   void INICIAL(T);
   void BORRA(void);
   VECTOR &operator =(const VECTOR &vector_der);
   T &operator [] (int i){return vec[i];}
};

template <class R>
class MATRIZ
{
  int renglones,columnas;
  static contador;
  int tam;
  R **mt;
  int identificador;
  void CREA_ESPACIO(int valor_renglon,int valor_columna);
 public:
    MATRIZ(int ren, int col);
    MATRIZ(const MATRIZ &matriz_der);
    ~MATRIZ(void);
    void INICIAL(R);
    void BORRA(void);
    friend MATRIZ<R> TRANSPUESTA(MATRIZ<R> &A);
    MATRIZ &operator = (const MATRIZ &matriz_der);
    R *operator [] (int renglon) { return mt[renglon]; }
    MATRIZ operator + (MATRIZ &matriz_der);
    MATRIZ operator * (MATRIZ &matriz_der);
    MATRIZ operator % (MATRIZ &matriz_der);
    friend MATRIZ<R> operator * (float escalar,MATRIZ<R> &matriz_der);
    friend MATRIZ<R> operator * (MATRIZ<R> &matriz_izq,float escalar);
};

class BUFFER
{
 public:
  int  GETIMAGE_FILE(int,int,int,int,char *);
  void PUTIMAGE_FILE(int,int,int,int,char *);
};
```

```
class MOUSEOBJ
{
  union REGS regs;
 public:
   void LIMITES(int x1,int y1);
   int RESET(void);
   void OCULTAR(void);
   void MOSTRAR(void);
   void MOVER(int x,int y);
   void OBT_COORDS(int &x,int &y);
   int BOTON_PRES(int boton);
   int TECLA(void);
   int BOTON_REL(int boton);
   int CAJA(int izq,int arriba,int der,int abajo,int x,int y);
};

void BARRA_STATUS(char *,char *);
void MENSAJE(char *,char *,char *,char *,char *);
```

EVALUACIÓN DE RESULTADOS

La evaluación del sistema CONFIA se ha dividido en dos partes principales. El primer aspecto a evaluar es la comprobación de las resultados obtenidos en el análisis mecánico y en el análisis de confiabilidad por medio del programa CONFIA con los métodos tradicionales de cálculo.

Para ilustrar los resultados se tienen dos ejemplos sencillos, el primero de ellos es una estructura tipo marco plano, al que se le aplica un análisis lineal elástico. Este tipo de análisis es el más simple que se puede efectuar en un sistema estructural con el programa CONFIA.

Para efectuar la evaluación primero se realiza el cálculo manual paso a paso de los desplazamientos nodales y después se muestran los resultados obtenidos con el sistema CONFIA tanto en forma gráfica como numérica.

El segundo ejemplo es un sistema estructural tipo marco al que se le aplica un análisis elastoplástico crítico a flexión. Este tipo de análisis es el más complejo que se puede realizar con el programa CONFIA, donde se consideran los efectos aleatorios para el análisis. En este ejemplo también se realiza el análisis paso a paso y después se presentan los resultados obtenidos con el sistema CONFIA.

La otra forma de evaluar el sistema CONFIA ha sido mediante su utilización por parte de los alumnos de los siguientes cursos:

- Diseño estructural de la Facultad de Ingeniería de la Universidad Nacional Autónoma de México.
- Análisis de confiabilidad de sistemas estructurales de la División de Estudios de Posgrado de la misma Facultad.
- Diseño de Estructuras de Acero de la División de Estudios de Posgrado de la Facultad de Ingeniería de la Universidad Autónoma de Querétaro.

Quienes además de evaluar el sistema CONFIA y probar su correcto funcionamiento, han aportado numerosas ideas en la presentación del programa, debido a esto los estudiantes y usuarios del sistema se han mostrado interesados y más motivados en aprender el análisis de sistemas estructurales gracias ha esta nueva herramienta que permite comprobar y observar los resultados de los análisis de una manera más rápida y adecuada debido a su interfaz gráfica y a sus herramientas de diseño.

4.1 EJEMPLO 1 - ANÁLISIS LINEAL ELÁSTICO

Dada la estructura de la figura 4.1 y mostrando su topología en la figura 4.2, se presenta el análisis de la estructura de forma manual y se comprueban las fuerzas internas y desplazamientos nodales obtenidos con este procedimiento, con los resultados obtenidos aplicando el sistema CONFIA.

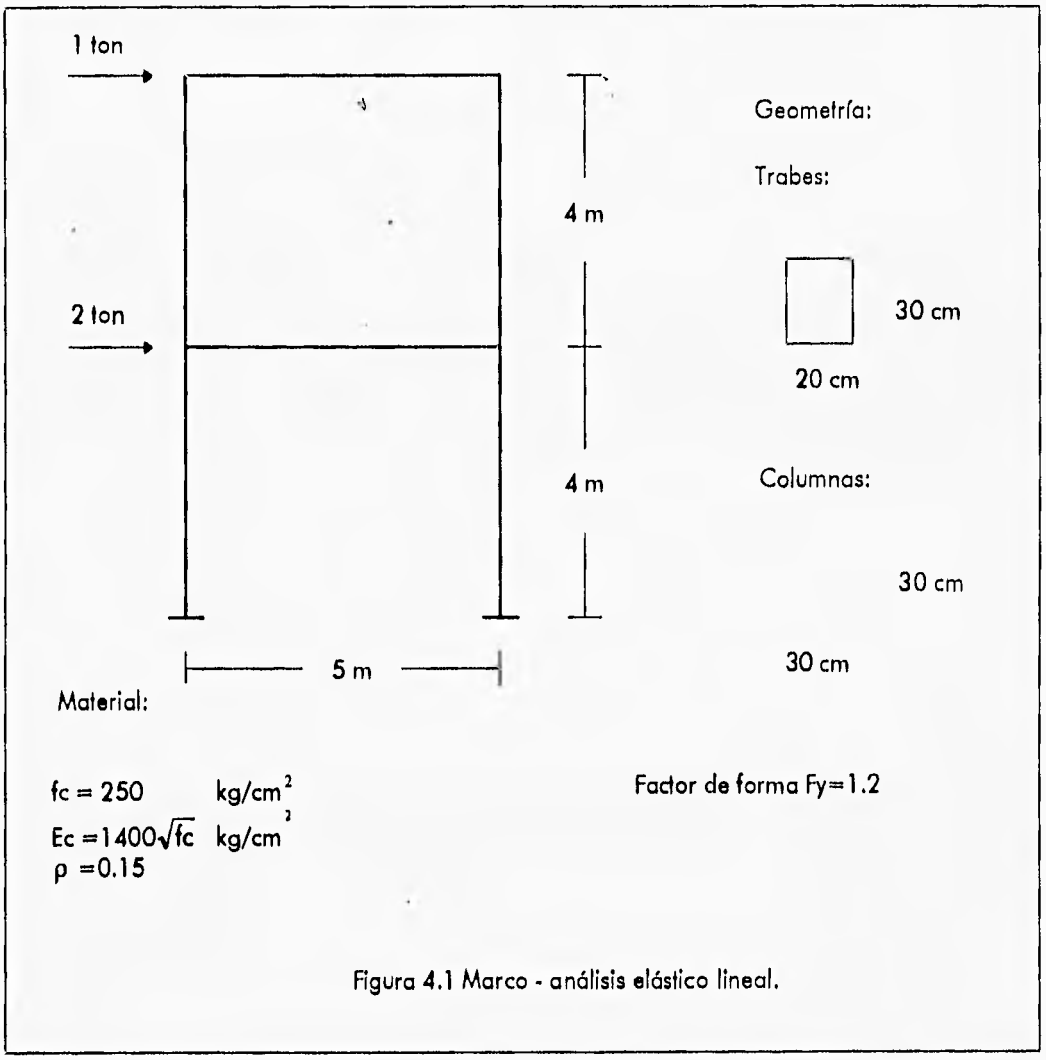


Figura 4.1 Marco - análisis elástico lineal.

Una vez establecida la topología de la estructura se debe obtener la matriz de rigidez local de cada barra, esta matriz está definida de la siguiente forma:

$$[k_{local}] = \begin{bmatrix} k_{11} & 0 & 0 & -k_{11} & 0 & 0 \\ 0 & k_{22} & k_{32} & 0 & -k_{22} & k_{32} \\ 0 & k_{32} & k_{33} & 0 & -k_{32} & k_{36} \\ -k_{11} & 0 & 0 & k_{11} & 0 & 0 \\ 0 & -k_{22} & -k_{32} & 0 & k_{22} & -k_{32} \\ 0 & k_{32} & k_{36} & 0 & -k_{32} & k_{33} \end{bmatrix}$$

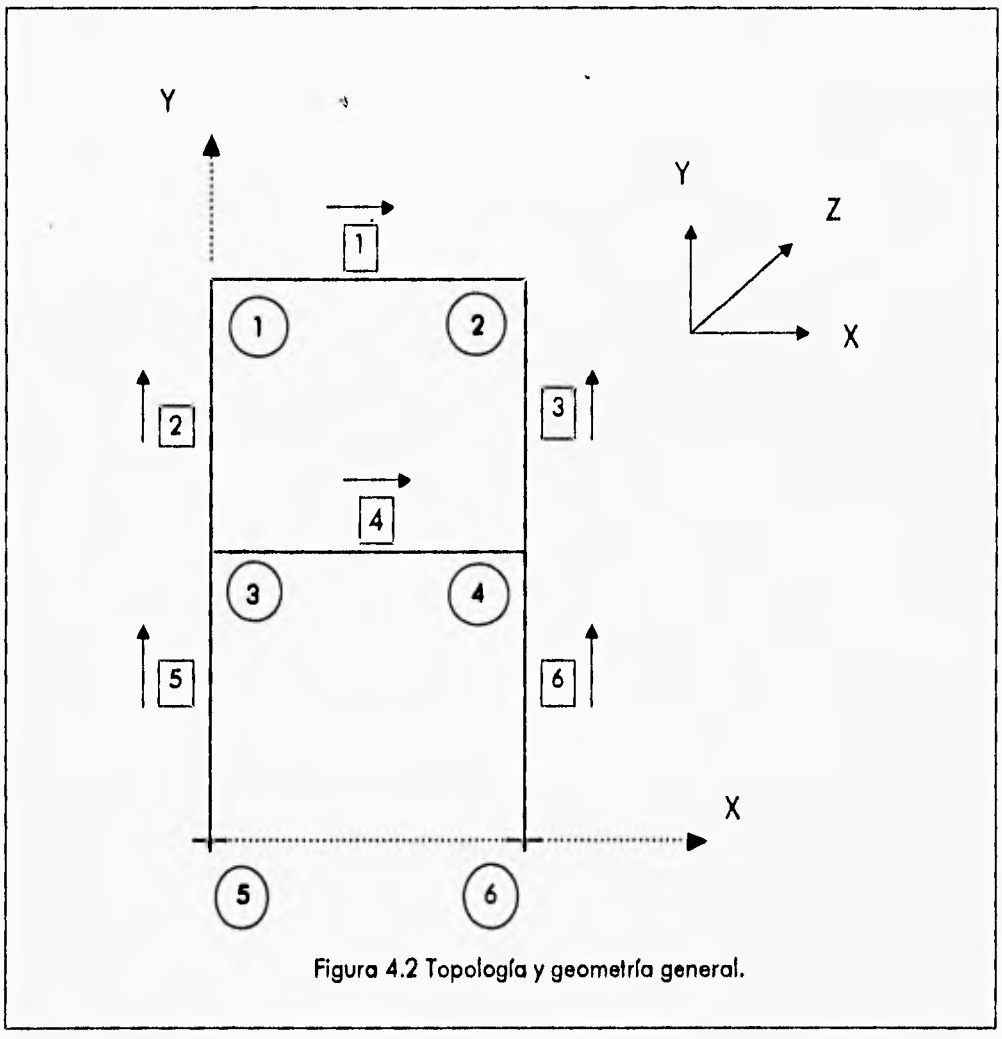


Figura 4.2 Topología y geometría general.

Donde:

$k_{11}=(AE)/L$

$k_{22}=(12EI)/L^3$

$k_{32}=(6EI)/L^2$

$k_{33}=(4EI)/L$

$k_{36}=(2EI)/L$

Por la geometría y mismo material, las matrices de rigidez en referencia local son iguales para todas las columnas:

$$[K^2]=[K^3]=[K^5]=[K^6]$$

Cálculo de constantes:

$A=(0.30)(0.30)=0.090m^2$
$L=4.0m$
$I_z=((0.30)(0.30)^3)/12=0.00068m^4$
con $fc=250kg/cm^2$
$E=14000\sqrt{250}$
$=221359.4\ kg/cm^2$
$=2213594\ ton/m^2$

Valores de los coeficientes de las matrices de rigidez para las columnas:

$k_{11}=$ 49805.867
$k_{22}=$ 282.233
$k_{32}=$ 564.466
$k_{33}=$ 1505.243
$k_{36}=$ 752.621

Por geometría y mismo material, se tiene que las matrices de rigidez en referencia local son iguales para las dos trabes:

$$[K^1]=[K^4]$$

Constantes:

$A=(0.20)(0.30)=0.060m^2$
$L=5.0m$
$I_z=0.00045m^4$
$E=2213594\ ton/m^2$

Valores de los coeficientes de las matrices de rigidez para las trabes:

$k_{11}=$ 26563.127
$k_{22}=$ 95.627
$k_{32}=$ 239.068
$k_{33}=$ 796.893
$k_{36}=$ 398.446

El siguiente paso es encontrar las matrices de rigidez de las barras en referencia global. La expresión que define la matriz de rigidez global de una barra es:

$$[K_{global}]=[T][k_{local}][T]^T$$

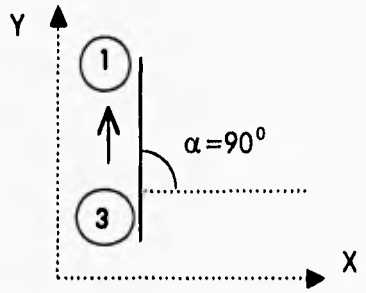
Donde:

$$[T]=\begin{bmatrix} R & 0 \\ 0 & R \end{bmatrix} \quad \text{siendo } [R]=\begin{bmatrix} \cos\alpha & -\text{sen}\alpha & 0 \\ \text{sen}\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Las matrices de rigidez globales para las columnas resultan iguales, debido a que todas ellas tienen una misma inclinación con respecto al eje X_{global}, sea:

$$[K^2]=[K^3]=[K^5]=[K^6]$$

Para la columna 2:



coseno 90=0
seno 90=1

La matriz [T] es igual a:

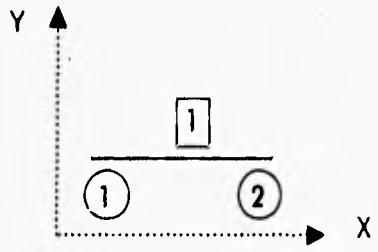
$$[T]=\begin{bmatrix} 0 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Y finalmente la matriz de rigidez global para las columnas queda de la forma:

$$[K_{global}]=\begin{bmatrix} 282.233 & 0 & -564.466 & -282.233 & 0 & -564.466 \\ 0 & 49805.867 & 0 & 0 & -49805.867 & 0 \\ -564.466 & 0 & 1505.243 & 564.466 & 0 & 752.621 \\ -282.233 & 0 & 564.466 & 282.233 & 0 & 564.466 \\ 0 & -49805.867 & 0 & 0 & 49805.867 & 0 \\ -564.466 & 0 & 752.621 & 564.466 & 0 & 1505.243 \end{bmatrix}$$

Como los ejes de referencia, local y global de las trabes son paralelos, se tiene:

$$[K^1_{global}] = [K^1_{local}] \text{ y } [K^4_{global}] = [K^4_{local}]$$



Por lo que la matriz de rigidez global para las trabes es:

$$[K_{global}] = \begin{bmatrix} 26563.127 & 0 & 0 & -26563.127 & 0 & 0 \\ 0 & 95.627 & 239.068 & 0 & -95.627 & 239.068 \\ 0 & 239.068 & 796.893 & 0 & -239.068 & 398.446 \\ -26563.127 & 0 & 0 & 26563.127 & 0 & 0 \\ 0 & -95.627 & -239.068 & 0 & 95.627 & -239.068 \\ 0 & 239.068 & 398.446 & 0 & -239.068 & 796.893 \end{bmatrix}$$

El siguiente paso es establecer la correspondencia de las matrices de rigidez globales de las barras de acuerdo a las coordenadas generalizadas de la estructura:

Trabe 1:

$$[K^1] = \begin{array}{c} \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 \end{matrix} \\ \begin{bmatrix} 26563.127 & 0 & 0 & -26563.127 & 0 & 0 \\ 0 & 95.627 & 239.068 & 0 & -95.627 & 239.068 \\ 0 & 239.068 & 796.893 & 0 & -239.068 & 398.446 \\ -26563.127 & 0 & 0 & 26563.127 & 0 & 0 \\ 0 & -95.627 & -239.068 & 0 & 95.627 & -239.068 \\ 0 & 239.068 & 398.446 & 0 & -239.068 & 796.893 \end{bmatrix} \end{array} \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{matrix}$$

Columna 2:

$$[K^2] = \begin{array}{c} \begin{array}{cccccc} 7 & 8 & 9 & 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 282.233 & 0 & -564.466 & -282..233 & 0 & -564.466 \\ 0 & 49805.867 & 0 & 0 & -49805.867 & 0 \\ -564.466 & 0 & 1505.243 & 564.466 & 0 & 752.621 \\ -282.233 & 0 & 564.466 & 282.233 & 0 & 564.466 \\ 0 & -49805.867 & 0 & 0 & 49805.867 & 0 \\ -564.466 & 0 & 752.621 & 564.466 & 0 & 1505.243 \end{bmatrix} \begin{array}{c} 7 \\ 8 \\ 9 \\ 1 \\ 2 \\ 3 \end{array} \end{array}$$

Columna 3:

$$[K^3] = \begin{array}{c} \begin{array}{cccccc} 10 & 11 & 12 & 4 & 5 & 6 \end{array} \\ \begin{bmatrix} 282.233 & 0 & -564.466 & -282..233 & 0 & -564.466 \\ 0 & 49805.867 & 0 & 0 & -49805.867 & 0 \\ -564.466 & 0 & 1505.243 & 564.466 & 0 & 752.621 \\ -282.233 & 0 & 564.466 & 282.233 & 0 & 564.466 \\ 0 & -49805.867 & 0 & 0 & 49805.867 & 0 \\ -564.466 & 0 & 752.621 & 564.466 & 0 & 1505.243 \end{bmatrix} \begin{array}{c} 10 \\ 11 \\ 12 \\ 4 \\ 5 \\ 6 \end{array} \end{array}$$

Trabe 4:

$$[K^4] = \begin{array}{c} \begin{array}{cccccc} 7 & 8 & 10 & 11 & 12 & 13 \end{array} \\ \begin{bmatrix} 26563.127 & 0 & 0 & -26563.127 & 0 & 0 \\ 0 & 95.627 & 239.068 & 0 & -95.627 & 239.068 \\ 0 & 239.068 & 796.893 & 0 & -239.068 & 398.446 \\ -26563.127 & 0 & 0 & 26563.127 & 0 & 0 \\ 0 & -95.627 & -239.068 & 0 & 95.627 & -239.068 \\ 0 & 239.068 & 398.446 & 0 & -239.068 & 796.893 \end{bmatrix} \begin{array}{c} 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \end{array} \end{array}$$

Columna 5:

$$[K^5]=\begin{array}{c} \begin{matrix} 0 & 0 & 0 & 7 & 8 & 9 \end{matrix} \\ \left[\begin{matrix} 282.233 & 0 & -564.466 & -282..233 & 0 & -564.466 \\ 0 & 49805.867 & 0 & 0 & -49805.867 & 0 \\ -564.466 & 0 & 1505.243 & 564.466 & 0 & 752.621 \\ -282.233 & 0 & 564.466 & 282.233 & 0 & 564.466 \\ 0 & -49805.867 & 0 & 0 & 49805.867 & 0 \\ -564.466 & 0 & 752.621 & 564.466 & 0 & 1505.243 \end{matrix}\right] \begin{matrix} 0 \\ 0 \\ 0 \\ 7 \\ 8 \\ 9 \end{matrix} \end{array}$$

Columna 6:

$$[K^6]=\begin{array}{c} \begin{matrix} 0 & 0 & 0 & 10 & 11 & 12 \end{matrix} \\ \left[\begin{matrix} 282.233 & 0 & -564.466 & -282..233 & 0 & -564.466 \\ 0 & 49805.867 & 0 & 0 & -49805.867 & 0 \\ -564.466 & 0 & 1505.243 & 564.466 & 0 & 752.621 \\ -282.233 & 0 & 564.466 & 282.233 & 0 & 564.466 \\ 0 & -49805.867 & 0 & 0 & 49805.867 & 0 \\ -564.466 & 0 & 752.621 & 564.466 & 0 & 1505.243 \end{matrix}\right] \begin{matrix} 0 \\ 0 \\ 0 \\ 10 \\ 11 \\ 12 \end{matrix} \end{array}$$

Una vez calculado las matrices globales de rigidez, se deben ensamblar en la matriz de rigidez general de la estructura, la cual se presenta en la página siguiente. La dimensión de esta matriz depende del número de grados de libertad de la estructura y es cuadrada, como se visualizó en la figura 4.2. El presente ejemplo tiene 12 grados de libertad.

Después de haber calculado la matriz de rigidez general de la estructura, se puede resolver el sistema de ecuaciones:

$$\{F\} = \left[K_{general}\right]\{D\}$$

MATRIZ DE RIGIDEZ GLOBAL DEL EJEMPLO 1 - ANÁLISIS LINEAL ELÁSTICO

$$[K_{general}] = \begin{bmatrix}
26845.3 & 0 & 564.446 & -26563.1 & 0 & 0 & -282.2 & 0 & 564.4 & 0 & 0 & 0 \\
0 & 49901.4 & 239.0 & 0 & -95.6 & 239.0 & 0 & -49805.9 & 0 & 0 & 0 & 0 \\
564.4 & 239.0 & 2302.1 & 0 & -239.0 & 398.4 & -564.4 & 0 & 752.6 & 0 & 0 & 0 \\
-26563.1 & 0 & 0 & 26845.3 & 0 & 564.6 & 0 & 0 & 0 & -282.2 & 0 & 564.4 \\
0 & -95.6 & -239.0 & 0 & 49901.4 & -239.0 & 0 & 0 & 0 & 0 & -49805.9 & 0 \\
0 & 239.0 & 398.4 & 564.4 & -239.0 & 2302.1 & 0 & 0 & 0 & -564.4 & 0 & 752.6 \\
-282.2 & 0 & -564.4 & 0 & 0 & 0 & 27127.5 & 0 & 0 & -26563.1 & 0 & 0 \\
0 & -49805.9 & 0 & 0 & 0 & 0 & 0 & 99707.3 & 239.7 & 0 & -95.6 & 239.0 \\
564.4 & 0 & 752.6 & 0 & 0 & 0 & 0 & 239.0 & 3807.3 & 0 & -239.0 & 398.4 \\
0 & 0 & 0 & -282.2 & 0 & -564.4 & -26563.1 & 0 & 0 & 27127.5 & 0 & 0 \\
0 & 0 & 0 & 0 & -49805.9 & 0 & 0 & -95.6 & -239.0 & 0 & 99707.3 & -239.0 \\
0 & 0 & 0 & 564.4 & 0 & 752.6 & 0 & 239.0 & 398.4 & 0 & -239.0 & 3807.3
\end{bmatrix}$$

Donde {F} representa al vector de fuerzas de la estructura y debido a que sólo se tienen cargas concentradas en los nodos 1 y 3 (ver figuras 4.1 y 4.2) resulta:

$$\{F\} = \begin{Bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 2 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{Bmatrix}$$

{D} es el vector de desplazamientos, que al resolver el sistema queda determinado como:

Δ_1	=	0.0187352	m
Δ_2	=	0.0000446	m
θ_3	=	-0.0011944	rad
Δ_4	=	0.0187163	m
Δ_5	=	-0.0000446	m
θ_6	=	-0.0012017	rad
Δ_7	=	0.0099682	m
Δ_8	=	0.0000333	m
θ_9	=	-0.0023001	rad
Δ_{10}	=	0.0099307	m
Δ_{11}	=	-0.0000333	m
θ_{12}	=	-0.0022954	rad

Con el sistema CONFIA, se diseña la estructura y el usuario no tiene que incluir la topología, ya que se encuentra definida por el programa. Los datos que se proporcionan al programa son los valores de las constantes E, A, I, para los dos tipos de elementos: columnas y trabes; y las condiciones de carga, que son fuerzas concentradas en los nodos respectivos. Una vez ejecutada la opción de Analizar en el módulo Mecánico, podemos ver gráficamente la deformación que sufre toda la estructura (figura 4.3) y numéricamente los desplazamientos en cada uno de los nodos en el submenú Desplazamientos del módulo Ver (Figura 4.4). Los valores de las fuerzas internas de las barras pueden ser también vistas en el submenú Fuerzas internas del módulo Ver (también llamado Visualización).

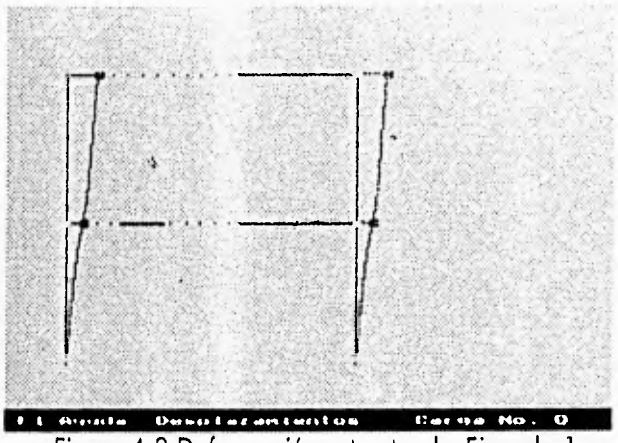


Figura 4.3 Deformación estructural - Ejemplo 1.

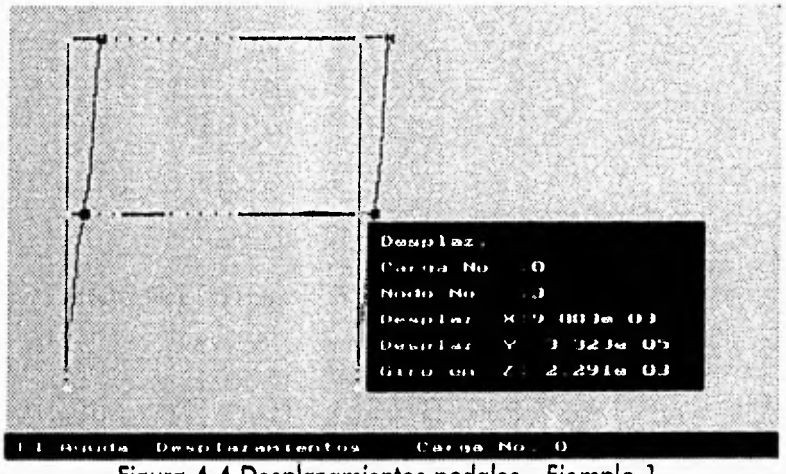


Figura 4.4 Desplazamientos nodales - Ejemplo 1.

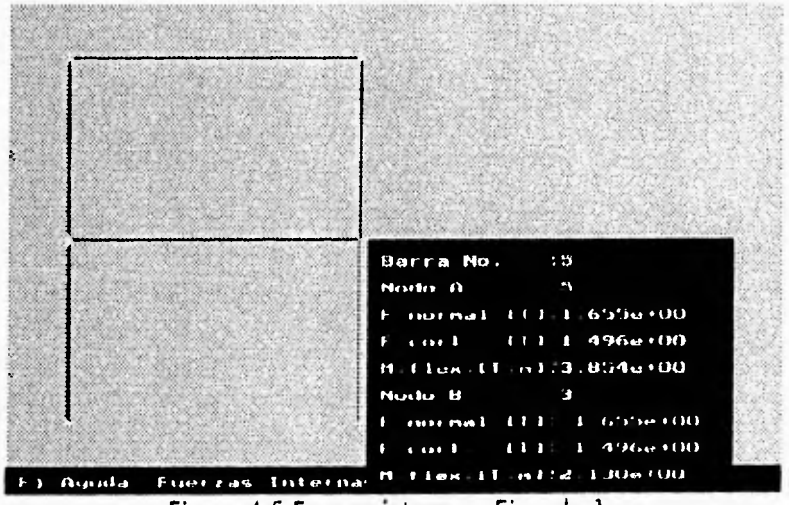


Figura 4.5 Fuerzas internas - Ejemplo 1.

Además, cada vez que se desarrolla un análisis se genera un archivo de resultados, "RRESULT.DAT", que puede ser visualizado cuando se termina el programa, donde se tienen los desplazamientos nodales y las fuerzas internas de los elementos del último análisis. También se puede realizar una impresión de los resultados sin abandonar el programa mediante el submenú Imprimir del módulo Archivo.

El siguiente es el formato del archivo de resultados "RRESULT.DAT" y la forma en que aparecenen las impresiones, si se elige esta opción dentro del programa CONFIA.

DESPLAZAMIENTOS NODALES

NODO	CARGA	DES.X	DES.Y	GIRO Z
0	0	1.86669e-02	4.45652e-05	-1.19526e-03
	1	1.02736e-02	2.92982e-05	-9.37885e-04
	2	8.39329e-03	1.52670e-05	-2.57374e-04
1	0	1.86480e-02	-4.45652e-05	-1.20264e-03
	1	1.02548e-02	-2.92983e-05	-9.33143e-04
	2	8.39319e-03	-1.52670e-05	-2.69493e-04
2	0	9.92013e-03	3.32265e-05	-2.29564e-03
	1	4.19665e-03	2.04298e-05	-1.21352e-03
	2	5.72348e-03	1.27966e-05	-1.08212e-03
3	0	9.88270e-03	-3.32265e-05	-2.29088e-03
	1	4.19660e-03	-2.04298e-05	-1.21145e-03
	2	5.68610e-03	-1.27966e-05	-1.07943e-03
4	0	0	0	0
	1	0	0	0
	2	0	0	0
5	0	0	0	0
	1	0	0	0
	2	0	0	0

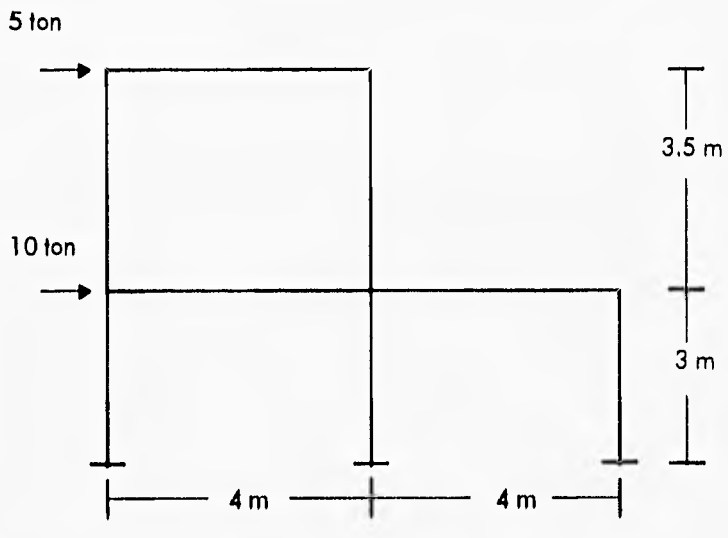
FUERZAS DE MIEMBRO DE ELEMENTOS TIPO VIGA

	NODO:I			NODO:J		
BARRA	FUERZA AXIAL	FUERZA CORTANTE	MOMENTO DE FLEXIÓN	FUERZA AXIAL	FUERZA CORTANTE	MOMENTO DE FLEXIÓN
0	5.01858e-01	-5.64737e-01	-1.4103e+00	-5.01858e-01	5.64737e-01	-1.4133e+00
0	4.99268e-01	-4.41700e-01	-1.1051e+00	-4.99268e-01	4.41700e-01	-1.1033e+00
0	2.59565e-03	-1.23037e-01	-3.05179e-01	-2.59565e-03	1.23037e-01	-3.10007e-01
1	-5.64737e-01	4.98144e-01	5.82204e-01	5.64737e-01	-4.98144e-01	1.4103e+00
1	-4.41700e-01	5.00735e-01	8.97748e-01	4.41700e-01	-5.00735e-01	1.1051e+00
1	-1.23037e-01	-2.59123e-03	-3.15544e-01	1.23037e-01	2.59123e-03	3.05179e-01
2	5.64737e-01	5.01896e-01	5.94274e-01	-5.64737e-01	-5.01896e-01	1.4133e+00
2	4.41700e-01	4.99287e-01	8.93844e-01	-4.41700e-01	-4.99287e-01	1.1033e+00
2	1.23037e-01	2.60934e-03	-2.99569e-01	-1.23037e-01	-2.60934e-03	3.10008e-01
3	9.94182e-01	-1.0901e+00	-2.7262e+00	-9.94182e-01	1.0901e+00	-2.7243e+00
3	1.30072e-03	-5.75825e-01	-1.4399e+00	-1.30072e-03	5.75825e-01	-1.4391e+00
3	9.92887e-01	-5.14311e-01	-1.2863e+00	-9.92887e-01	5.14311e-01	-1.2852e+00
4	-1.6548e+00	1.5039e+00	3.8718e+00	1.6548e+00	-1.5039e+00	2.1440e+00
4	-1.0175e+00	4.99444e-01	1.4555e+00	1.0175e+00	-4.99444e-01	5.42228e-01
4	-6.37348e-01	1.0045e+00	2.4162e+00	6.37348e-01	-1.0045e+00	1.6018e+00

5	1.6548e+00	1.4961e+00	3.8542e+00	-1.6548e+00	-1.4961e+00	2.1301e+00
5	1.0175e+00	5.00596e-01	1.4570e+00	-1.0175e+00	-5.00596e-01	5.45309e-01
5	6.37348e-01	9.95505e-01	2.3972e+00	-6.37348e-01	-9.95505e-01	1.5848e+00

4.2 EJEMPLO 2 - ANÁLISIS ELASTOPLÁSTICO CRÍTICO A PURA FLEXIÓN

En las figuras 4.6a y 4.6b se muestra la estructura para analizar con sus características genera'es, nuevamente se desarro"a e' aná'isis manua'mente para comparar 'os resu'tados con el programa CONFIA.



Material:

Columnas:	Trabes:
$E=2.1\ 10^{7}$	$E=2.1\ 10^{7}$
Área$=1.11\ 10^{-1}$	Área$=4.6\ 10^{-3}$
Iz$=6.4\ 10^{-5}$	Iz$=1.4\ 10^{-5}$
R=19.5	R=5.60
CV=0.10	CV=0.10

Figura 4.6a Ejemplo 2 - Análisis elastoplástico crítico.

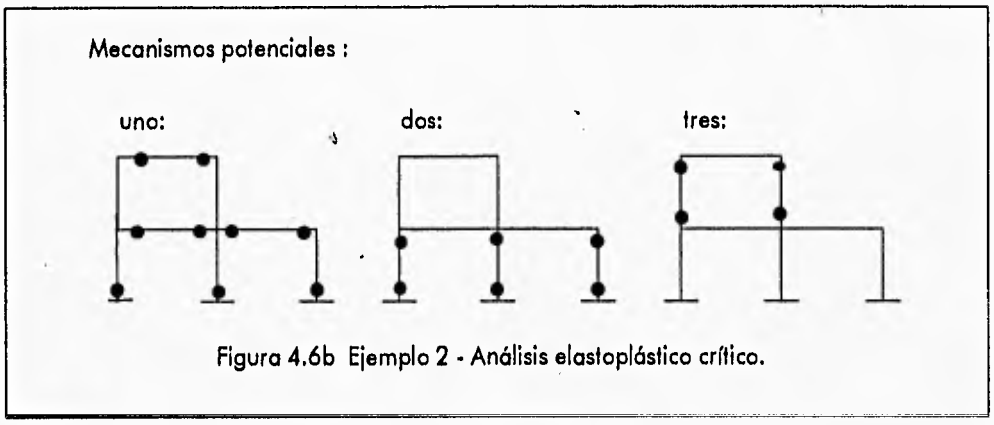


Figura 4.6b Ejemplo 2 - Análisis elastoplástico crítico.

Antes de comenzar el análisis se deben numerar los nodos y barras, así como darles una dirección a las barras para poder identificar las conectividades de éstas, como muestra la figura 4.7. (por defecto, de izquierda a derecha y de abajo hacia arriba).

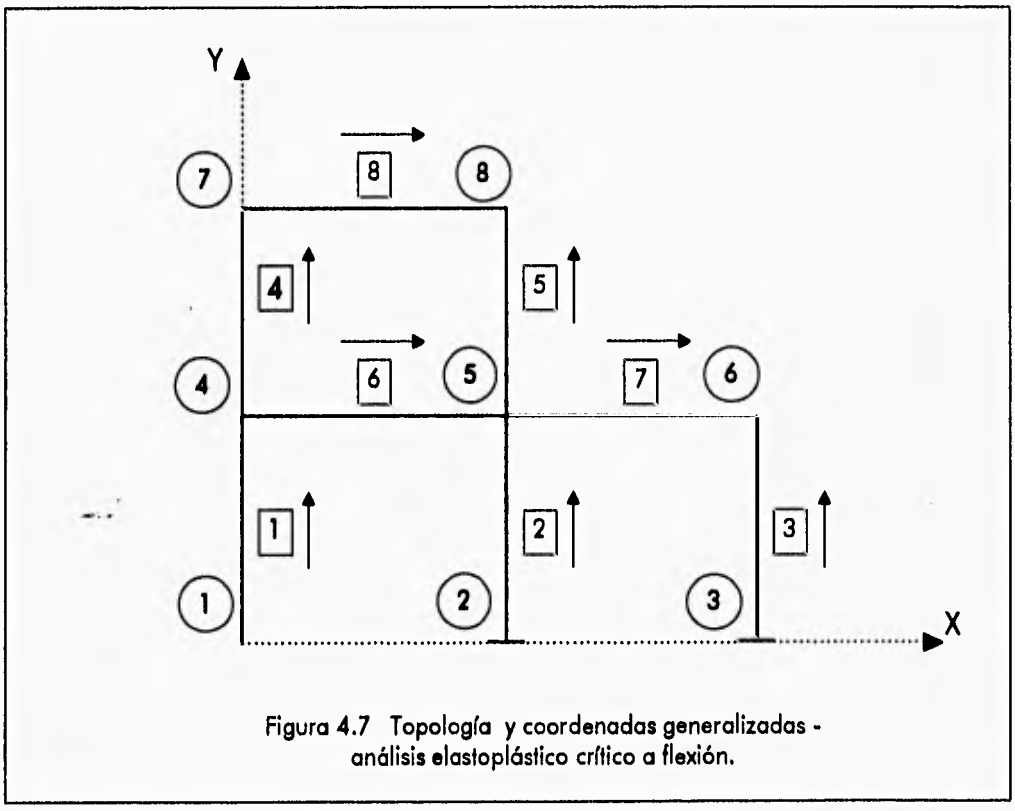
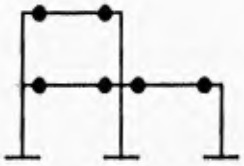


Figura 4.7 Topología y coordenadas generalizadas - análisis elastoplástico crítico a flexión.

Para simplificar la comparación sólo se muestra el análisis para una estructura parcialmente plastificada.



El primer paso para el análisis es calcular la matriz de rigidez local de cada barra.

Columnas:

Las columnas no tienen plastificaciones en ninguno de sus extremos por lo que la matriz de rigidez local de las columnas es de la forma:

$$[k_{local}]=\begin{bmatrix} k_{11} & 0 & 0 & -k_{11} & 0 & 0 \\ 0 & k_{22} & k_{32} & 0 & -k_{22} & k_{32} \\ 0 & k_{32} & k_{33} & 0 & -k_{32} & k_{36} \\ -k_{11} & 0 & 0 & k_{11} & 0 & 0 \\ 0 & -k_{22} & -k_{32} & 0 & k_{22} & -k_{32} \\ 0 & k_{32} & k_{36} & 0 & -k_{32} & k_{33} \end{bmatrix}$$

Donde:

$k_{11}=(AE)/L$
$k_{22}=(12EI)/L^3$
$k_{32}=(6EI)/L^2$
$k_{33}=(4EI)/L$
$k_{36}=(2EI)/L$

Para las columnas 1,2,3 se tienen los siguientes valores:

$k_{11}=777000.000000$
$k_{22}=597.333313$
$k_{32}=896.000000$
$k_{33}=1792.000000$
$k_{36}=896.000000$

Para las columnas 4 y 5:

$k_{11}=666000.000000$
$k_{22}=376.163269$

$k_{32} = 658.285706$
$k_{33} = 1536.000000$
$k_{36} = 768.000000$

Trabes:

En el caso de las trabes la matriz de rigidez local es de la forma siguiente, ya que se tienen plastificaciones en ambos extremos, ver ecuación 1.14:

$$[k_{local}] = \begin{bmatrix} k_{11} & 0 & 0 & -k_{11} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -k_{11} & 0 & 0 & k_{11} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Donde $k_{11} = (AE)/L$

Dado que la longitud y el material son los mismos, las matrices de rigidez local de las tres trabes son iguales. Sustituyendo los valores de E, A, L e I para las trabes se tiene:

$k_{11} = 24150.000000$

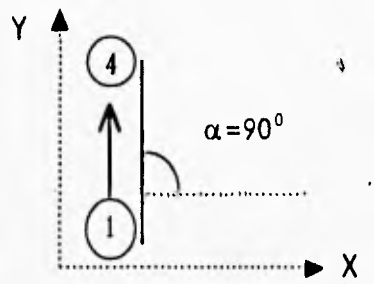
El siguiente paso es encontrar la matriz de rigidez en referencia global para cada barra. La ecuación que define la matriz de rigidez global de una barra es:

$$[K_{global}] = [T][k_{local}][T]^T$$

Donde:

$$[T] = \begin{bmatrix} R & 0 \\ 0 & R \end{bmatrix} \quad \text{siendo} \quad [R] = \begin{bmatrix} \cos\alpha & -\text{sen}\alpha & 0 \\ \text{sen}\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Para las columnas se tiene:



coseno 90=0
seno 90=1

La matriz [T] es igual a:

$$[T]=\begin{bmatrix} 0 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Las matrices de rigidez globales para las columnas 1, 2 y 3 son iguales, debido a que todas ellas tienen la misma inclinación con respecto al eje X_{global}, quedando finalmente de la forma:

$$[K^1]=[K^2]=[K^3]=\begin{bmatrix} 597.333 & 0 & -896 & -595.333 & 0 & -896 \\ 0 & 777000 & 0 & 0 & -777000 & 0 \\ -896 & 0 & 1792 & 896 & 0 & 896 \\ -597.333 & 0 & 896 & 597.333 & 0 & 896 \\ 0 & -777000 & 0 & 0. & 777000 & 0 \\ -896 & 0 & 896 & 896 & 0 & 1792 \end{bmatrix}$$

$$[K^4]=[K^5]=\begin{bmatrix} 376.163 & 0 & -658.285 & -376.163 & 0 & -658.285 \\ 0 & 666000 & 0 & 0 & -666000 & 0 \\ -658.285 & 0 & 1536 & 658.285 & 0 & 768 \\ -376.163 & 0 & 658.285 & 376.163 & 0 & 658.285 \\ 0 & -666000 & 0 & 0 & 666000 & 0 \\ -658.285 & 0 & 768 & 658.285 & 0 & 1536 \end{bmatrix}$$

Para las trabes la matriz de rigidez local es igual a la matriz de rigidez global ya que los ejes locales son paralelos a los de referencia global.

$$[K^6]=[K^7]=[K^8]=\begin{bmatrix} 24150 & 0 & 0 & -24150 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -24150 & 0 & 0 & 24150 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

El siguiente paso es hacer la correspondencia de las matrices globales de las barras de acuerdo a las coordenadas generalizadas de la estructura:

Barra 1:

$$[K^1]=\begin{array}{c} \begin{matrix} 0 & 0 & 0 & 1 & 2 & 3 \end{matrix} \\ \left[\begin{matrix} 597.333 & 0 & -896 & -595.333 & 0 & -896 \\ 0 & 777000 & 0 & 0 & -777000 & 0 \\ -896 & 0 & 1792 & 896 & 0 & 896 \\ -597.333 & 0 & 896 & 597.333 & 0 & 896 \\ 0 & -777000 & 0 & 0. & 777000 & 0 \\ -896 & 0 & 896 & 896 & 0 & 1792 \end{matrix}\right. \end{array} \begin{matrix} 0 \\ 0 \\ 0 \\ 1 \\ 2 \\ 3 \end{matrix}$$

Barra 2:

$$[K^2]=\begin{array}{c} \begin{matrix} 0 & 0 & 0 & 4 & 5 & 6 \end{matrix} \\ \left[\begin{matrix} 597.333 & 0 & -896 & -595.333 & 0 & -896 \\ 0 & 777000 & 0 & 0 & -777000 & 0 \\ -896 & 0 & 1792 & 896 & 0 & 896 \\ -597.333 & 0 & 896 & 597.333 & 0 & 896 \\ 0 & -777000 & 0 & 0. & 777000 & 0 \\ -896 & 0 & 896 & 896 & 0 & 1792 \end{matrix}\right. \end{array} \begin{matrix} 0 \\ 0 \\ 0 \\ 4 \\ 5 \\ 6 \end{matrix}$$

Barra 3:

$$[K^3] = \begin{array}{c} \begin{matrix} 0 & 0 & 0 & 7 & 8 & 9 \end{matrix} \\ \begin{bmatrix} 597.333 & 0 & -896 & -595.333 & 0 & -896 \\ 0 & 777000 & 0 & 0 & -777000 & 0 \\ -896 & 0 & 1792 & 896 & 0 & 896 \\ -597.333 & 0 & 896 & 597.333 & 0 & 896 \\ 0 & -777000 & 0 & 0. & 777000 & 0 \\ -896 & 0 & 896 & 896 & 0 & 1792 \end{bmatrix} \begin{matrix} 0 \\ 0 \\ 0 \\ 7 \\ 8 \\ 9 \end{matrix} \end{array}$$

Barra 4:

$$[K^4] = \begin{array}{c} \begin{matrix} 1 & 2 & 3 & 10 & 11 & 12 \end{matrix} \\ \begin{bmatrix} 376.163 & 0 & -658.285 & -376.163 & 0 & -658.285 \\ 0 & 666000 & 0 & 0 & -666000 & 0 \\ -658.285 & 0 & 1536 & 658.285 & 0 & 768 \\ -376.163 & 0 & 658.285 & 376.163 & 0 & 658.285 \\ 0 & -666000 & 0 & 0 & 666000 & 0 \\ -658.285 & 0 & 768 & 658.285 & 0 & 1536 \end{bmatrix} \begin{matrix} 1 \\ 2 \\ 3 \\ 10 \\ 11 \\ 12 \end{matrix} \end{array}$$

Barra 5:

$$[K^5] = \begin{array}{c} \begin{matrix} 4 & 5 & 6 & 13 & 14 & 15 \end{matrix} \\ \begin{bmatrix} 376.163 & 0 & -658.285 & -376.163 & 0 & -658.285 \\ 0 & 666000 & 0 & 0 & -666000 & 0 \\ -658.285 & 0 & 1536 & 658.285 & 0 & 768 \\ -376.163 & 0 & 658.285 & 376.163 & 0 & 658.285 \\ 0 & -666000 & 0 & 0 & 666000 & 0 \\ -658.285 & 0 & 768 & 658.285 & 0 & 1536 \end{bmatrix} \begin{matrix} 4 \\ 5 \\ 6 \\ 13 \\ 14 \\ 15 \end{matrix} \end{array}$$

Barra 6:

$$[K^6]=\begin{array}{c} \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 \end{matrix} \\ \begin{bmatrix} 24150 & 0 & 0 & -24150 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -24150 & 0 & 0 & 24150 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array} \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{matrix}$$

Barra 7:

$$[K^7]=\begin{array}{c} \begin{matrix} 4 & 5 & 6 & 7 & 8 & 9 \end{matrix} \\ \begin{bmatrix} 24150 & 0 & 0 & -24150 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -24150 & 0 & 0 & 24150 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array} \begin{matrix} 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{matrix}$$

Barra 8:

$$[K^8]=\begin{array}{c} \begin{matrix} 10 & 11 & 12 & 13 & 14 & 15 \end{matrix} \\ \begin{bmatrix} 24150 & 0 & 0 & -24150 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -24150 & 0 & 0 & 24150 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array} \begin{matrix} 10 \\ 11 \\ 12 \\ 13 \\ 14 \\ 15 \end{matrix}$$

El siguiente paso es ensamblar las matrices de rigidez globales de cada barra en la matriz de rigidez global del sistema estructural, quedando de la forma como se muestra en la siguiente página:

MATRIZ DE RIGIDEZ GLOBAL DEL EJEMPLO 2 - ANÁLISIS ELASTOPLÁSTICO CRÍTICO

$$[K_{general}] = \begin{bmatrix}
25123.49 & 0 & 237.71 & -24150 & 0 & 0 & 0 & 0 & 0 & -376.16 & 0 & -658.286 & 0 & 0 & 0 \\
0 & 1443000 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -666000 & 0 & 0 & 0 & 0 \\
237.71 & 0 & 3328 & 0 & 0 & 0 & 0 & 0 & 0 & 658.28 & 0 & 768 & 0 & 0 & 0 \\
-24150 & 0 & 0 & 49273.49 & 0 & 237.71 & -24150. & 0 & 0 & 0 & 0 & 0 & -376.16 & 0 & -658.28 \\
0 & 0 & 0 & 0 & 1443000 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -666000 & 0 \\
0 & 0 & 0 & 237.71 & 0 & 3328 & 0 & 0 & 0 & 0 & 0 & 0 & 658.286 & 0 & 768 \\
0 & 0 & 0 & -24150 & 0 & 0 & 24747.33 & 0 & 896 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 777000 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 896 & 0 & 1792 & 0 & 0 & 0 & 0 & 0 & 0 \\
-376.16 & 0 & 658 & 0 & 0 & 0 & 0 & 0 & 0 & 24526.16 & 0 & 658 & -24150 & 0 & 0 \\
0 & -666000 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 666000 & 0 & 0 & 0 & 0 \\
-658.28 & 0 & 768 & 0 & 0 & 0 & 0 & 0 & 0 & 658.28 & 0 & 1536 & 0 & 0 & 0 \\
0 & 0 & 0 & -376.16 & 0 & 658.28 & 0 & 0 & 0 & -24150 & 0 & 0 & 24526.16 & 0 & 658.28 \\
0 & 0 & 0 & 0 & -666000 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 666000 & 0 \\
0 & 0 & 0 & -658.28 & 0 & 768 & 0 & 0 & 0 & 0 & 0 & 0 & 658.28 & 0 & 1536
\end{bmatrix}$$

El último paso para encontrar los desplazamientos nodales es resolver el sistema de ecuaciones siguiente:

$$\{F\} = \left[K_{general}\right]\{D\}$$

Donde {F} es el vector de fuerzas de la estructura y está formado por:

$$\{F\} = \begin{bmatrix} 10 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 5 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

{D} es el vector de desplazamientos, que al resolver el sistema de ecuaciones queda determinado como:

Δ_1 = 5.33580E-02 m
Δ_2 = 0.00000E+00m
θ_3 = -3.15102E-02 rad
Δ_4 = 5.29921E-02 m
Δ_5 = 0.00000E+00m
θ_6 = -3.14280E-02 rad
Δ_7 = 5.26665E-02 m
Δ_8 = 0.00000E+00m
θ_9 = -2.63332E-02 rad
Δ_{10} = 1.89947E-01 m
Δ_{11} = 0.00000E+00m
θ_{12} = -4.27830E-02 rad
Δ_{13} = 1.89842E-01 m

Δ_{14} = 0.00000E+00 m

θ_{15} = -4.29360E-02 rad

A continuación se muestran los resultados obtenidos con el sistema CONFIA.

Una vez editada la estructura y dadas todas las características necesarias para el análisis, como son: materiales, cargas, resistencias y constantes en las barras. Se puede hacer el análisis mecánico seleccionando la opción analizar en el módulo Mecánico. Para la estructura analizada se tienen los siguientes resultados: deformada, desplazamientos nodales, fuerzas internas en las barras, e índices de confiabilidad beta, como se muestran en las figuras 4.8, 4.9, 4.10, 4.11, respectivamente.

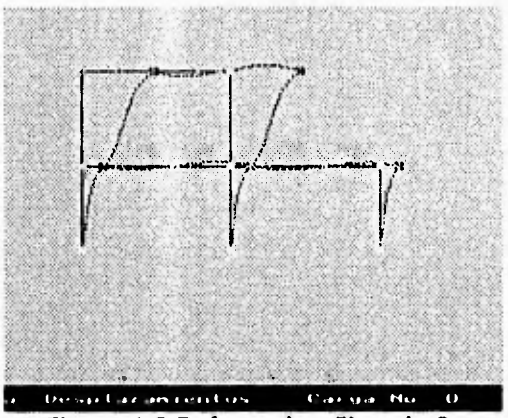


Figura 4.8 Deformada - Ejemplo 2.

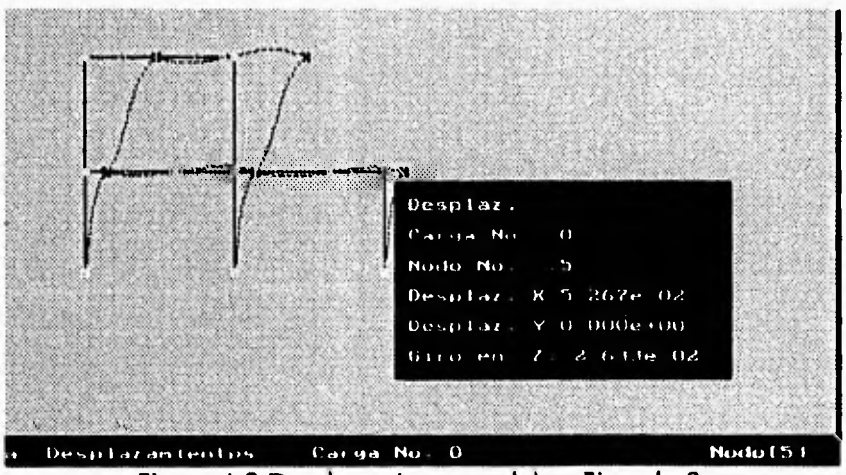


Figura 4.9 Desplazamientos nodales -Ejemplo 2.

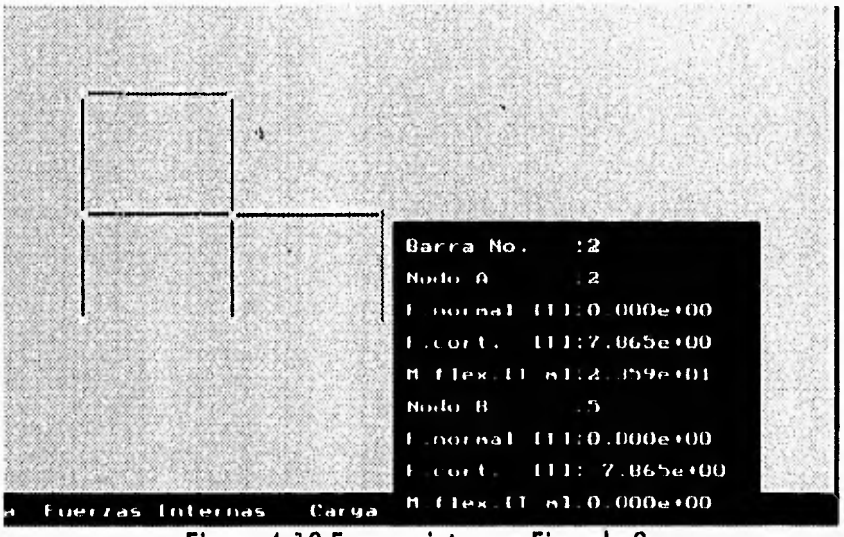


Figura 4.10 Fuerzas internas-Ejemplo 2.

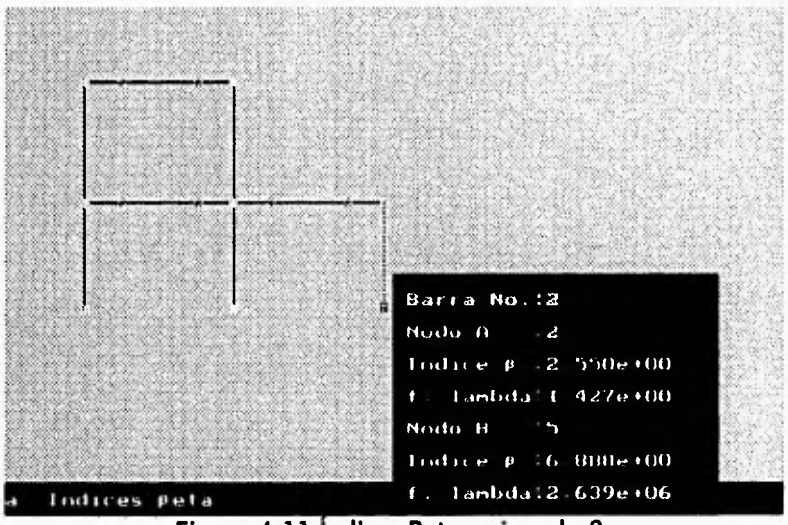


Figura 4.11 Índices Beta - ejemplo 2.

En el submenú Imprimir del módulo Archivo, se pueden mandar a imprimir los desplazamientos que tienen cada uno de los nodos, las fuerzas internas que actúan en los elementos y los índices de confiabilidad beta para cada barra. Estos archivos se muestran a continuación:

DESPLAZAMIENTOS NODALES

NODO	CARGA	DES.X	DES.Y	MOM. Z
0	0	0	0	0
	1	0	0	0
	2	0	0	0
	3	0	0	0
	4	0	0	0
	5	0	0	0
	6	0	0	0
	7	0	0	0
	8	0	0	0
1	0	0	0	0
	1	0	0	0
	2	0	0	0
	3	0	0	0
	4	0	0	0
	5	0	0	0
	6	0	0	0
	7	0	0	0
	8	0	0	0
2	0	0	0	0
	1	0	0	0
	2	0	0	0
	3	0	0	0
	4	0	0	0
	5	0	0	0
	6	0	0	0
	7	0	0	0
	8	0	0	0
3	0	5.33586e-02	0	-3.15106e-02
	1	2.25462e-02	0	-1.12309e-02
	2	3.08124e-02	0	-2.02797e-02
	3	-6.37916e-03	1.82754e-06	5.73328e-03
	4	-6.34934e-03	1.82754e-06	3.79442e-03
	5	-6.34934e-03	0	3.79442e-03
	6	-6.28693e-03	0	3.14928e-03
	7	-6.32328e-03	1.82754e-06	4.27273e-03
	8	-6.40424e-03	1.82754e-06	5.25432e-03
4	0	5.29928e-02	0	-3.14285e-02
	1	2.22740e-02	0	-1.11784e-02
	2	3.07188e-02	0	-2.02500e-02
	3	-6.33751e-03	-1.82754e-06	3.79442e-03
	4	-6.35247e-03	-1.82754e-06	5.72584e-03
	5	-6.35247e-03	1.82754e-06	5.72584e-03
	6	-6.32547e-03	1.82754e-06	3.15671e-03
	7	-6.36480e-03	-1.82754e-06	5.24052e-03
	8	-6.32419e-03	-1.82754e-06	4.27910e-03
5	0	5.26671e-02	0	-2.63336e-02
	1	2.21371e-02	0	-1.10685e-02
	2	3.05300e-02	0	-1.52650e-02
	3	-6.29856e-03	0	3.14928e-03
	4	-6.31343e-03	0	3.15671e-03
	5	-6.31343e-03	-1.82754e-06	3.15671e-03
	6	-6.40347e-03	-1.82754e-06	6.37138e-03

	7	-6.32569e-03	0	3.16284e-03
	8	-6.28533e-03	0	3.14266e-03
6	0	1.89949e-01	0	-4.27836e-02
	1	6.16249e-02	0	-1.11325e-02
	2	1.28324e-01	0	-3.16510e-02
	3	-2.30377e-02	1.82754e-06	4.27273e-03
	4	-2.30040e-02	1.82754e-06	5.24052e-03
	5	-2.30040e-02	0	5.24052e-03
	6	-1.73411e-02	0	3.16284e-03
	7	-3.59556e-02	3.95967e-06	1.42611e-02
	8	-3.59675e-02	3.95967e-06	1.00428e-02
7	0	1.89845e-01	0	-4.29366e-02
	1	6.16240e-02	0	-1.12751e-02
	2	1.28221e-01	0	-3.16615e-02
	3	-2.30244e-02	-1.82754e-06	5.25432e-03
	4	-2.30172e-02	-1.82754e-06	4.27910e-03
	5	-2.30172e-02	1.82754e-06	4.27910e-03
	6	-1.73412e-02	1.82754e-06	3.14266e-03
	7	-3.59120e-02	-3.95967e-06	1.00428e-02
	8	-3.60110e-02	-3.95967e-06	1.42813e-02

FUERZAS DE MIEMBRO DE ELEMENTOS TIPO VIGA

	NODO:I			NODO:J		
BARRA	FUERZA AXIAL	FUERZA CORTANTE	MOMENTO DE FLEXIÓN	FUERZA AXIAL	FUERZA CORTANTE	MOMENTO DE FLEXIÓN
0	0	3.639e+00	1.957e+01	0	-3.639e+00	-8.657e+00
0	0	3.404e+00	1.013e+01	0	-3.404e+00	7.556e-02
0	0	2.347e-01	9.437e+00	0	-2.347e-01	-8.733e+00
0	-1.420e+00	1.326e+00	-5.787e-01	1.420e+00	-1.326e+00	4.558e+00
0	-1.420e+00	-3.928e-01	-2.289e+00	1.420e+00	3.928e-01	1.110e+00
0	0	-3.928e-01	-2.289e+00	0	3.928e-01	1.110e+00
0	0	-9.336e-01	-2.811e+00	0	9.336e-01	1.041e-02
0	-1.420e+00	5.126e-02	-1.837e+00	1.420e+00	-5.126e-02	1.991e+00
0	-1.420e+00	8.824e-01	-1.030e+00	1.4200e+00	-8.824e-01	3.677e+00
1	0	3.494e+00	1.932e+01	0	-3.494e+00	-8.838e+00
1	0	3.289e+00	9.941e+00	0	-3.289e+00	-7.423e-02
1	0	2.053e-01	9.380e+00	0	-2.053e-01	-8.764e+00
1	1.420e+00	-3.858e-01	-2.278e+00	-1.420e+00	3.858e-01	1.121e+00
1	1.420e+00	1.335e+00	-5.614e-01	-1.420e+00	-1.335e+00	4.568e+00
1	-1.420e+00	1.335e+00	-5.614e-01	1.420e+00	-1.335e+00	4.568e+00
1	-1.420e+00	-9.499e-01	-2.839e+00	1.420e+00	9.499e-01	-1.079e-02
1	1.420e+00	8.935e-01	-1.007e+00	-1.420e+00	-8.935e-01	3.688e+00
1	1.420e+00	5.642e-02	-1.832e+00	-1.420e+00	-5.642e-02	2.001e+00
2	0	7.864e+00	2.359e+01	0	-7.864e+00	0
2	0	3.305e+00	9.917e+00	0	-3.305e+00	1.907e-06
2	0	4.559e+00	1.367e+01	0	-4.559e+00	3.337e-06
2	0	-9.405e-01	-2.821e+00	0	9.405e-01	-1.013e-06
2	0	-9.428e-01	-2.828e+00	0	9.428e-01	-1.788e-07
2	1.420e+00	-9.428e-01	-2.828e+00	-1.420e+00	9.428e-01	2.384e-07
2	1.420e+00	1.883e+00	-2.875e-02	-1.420e+00	-1.883e+00	5.680e+00

2	0	-9.446e-01	-2.833e+00	0	9.446e-01	1.788e-07
2	0	-9.386e-01	-2.815e+00	0	9.386e-01	-3.576e-07
3	0	2.473e+00	8.657e+00	0	-2.473e+00	-9.536e-06
3	0	-2.159e-02	-7.556e-02	0	2.159e-02	-4.768e-06
3	0	2.495e+00	8.733e+00	0	-2.495e+00	-9.536e-06
3	-4.306e-08	3.204e-01	1.121e+00	4.306e-08	-3.204e-01	1.1920e-06
3	-4.306e-08	-3.173e-01	-1.110e+00	4.306e-08	3.173e-01	2.861e-06
3	0	-3.173e-01	-1.110e+00	0	3.173e-01	2.861e-06
3	0	-2.975e-03	-1.041e-02	0	2.975e-03	1.192e-06
3	-1.420e+00	1.053e+00	-1.991e+00	1.420e+00	-1.053e+00	5.680e+00
3	-1.420e+00	-1.050e+00	-3.677e+00	1.420e+00	1.050e+00	1.907e-06
4	0	2.525e+00	8.83825e+00	0	-2.525e+00	-5.722e-06
4	0	2.120e-02	7.423e-02	0	-2.120e-02	-2.384e-06
4	0	2.504e+00	8.764e+00	0	-2.504e+00	-7.629e-06
4	4.306e-08	-3.203e-01	-1.121e+00	-4.306e-08	3.203e-01	1.430e-06
4	4.306e-08	3.174e-01	1.111e+00	-4.306e-08	-3.174e-01	1.430e-06
4	-4.306e-08	3.174e-01	1.111e+00	4.306e-08	-3.174e-01	1.430e-06
4	-4.306e-08	3.082e-03	1.079e-02	4.306e-08	-3.082e-03	1.311e-06
4	1.420e+00	-1.053e+00	-3.688e+00	-1.420e+00	1.053e+00	1.4305e-06
4	1.420e+00	1.050e+00	-2.001e+00	-1.420e+00	-1.050e+00	5.680e+00
5	8.834e+00	0	0	-8.834e+00	0	0
5	6.573e+00	0	0	-6.573e+00	0	0
5	2.260e+00	0	0	-2.260e+00	0	0
5	-1.006e+00	-1.420e+00	-5.680e+00	1.006e+00	1.420e+00	-6.083e-08
5	7.554e-02	-1.420e+00	6.083e-08	-7.554e-02	1.420e+00	-5.680e+00
5	7.554e-02	0	0	-7.554e-02	0	0
5	9.306e-01	0	0	-9.306e-01	0	0
5	1.002e+00	0	0	-1.002e+00	0	0
5	-1.933e+00	0	0	1.933e+00	0	0
6	7.864e+00	0	0	-7.864e+00	0	0
6	3.305e+00	0	0	-3.305e+00	0	0
6	4.559e+00	0	0	-4.559e+00	0	0
6	-9.405e-01	0	0	9.405e-01	0	0
6	-9.428e-01	0	0	9.428e-01	0	0
6	-9.428e-01	-1.420e+00	-5.680e+00	9.428e-01	1.420e+00	-6.083e-08
6	1.883e+00	-1.420e+00	6.083e-08	-1.883e+00	1.420e+00	-5.680e+00
6	-9.446e-01	0	0	9.446e-01	0	0
6	-9.386e-01	0	0	9.386e-01	0	0
7	2.526e+00	0	0	-2.526e+00	0	0
7	2.137e-02	0	0	-2.137e-02	0	0
7	2.504e+00	0	0	-2.504e+00	0	0
7	-3.203e-01	0	0	3.203e-01	0	0
7	3.173e-01	0	0	-3.173e-01	0	0
7	3.173e-01	0	0	-3.173e-01	0	0
7	3.005e-03	0	0	-3.005e-03	0	0
7	-1.053e+00	-1.420e+00	-5.680e+00	1.053e+00	1.420e+00	-6.083e-08
7	1.050e+00	-1.420e+00	6.083e-08	-1.050e+00	1.420e+00	-5.680e+00

SECCIONES MÁS ESFORZADAS

#	FACTOR LAMBDA	BETA
4	1.42730e+00	2.55048e+00
2	1.48023e+00	2.74027e+00
0	1.55069e+00	3.14949e+00
8	2.72656e+00	5.72843e+00
6	3.03761e+00	6.63293e+00
1	3.69367e+00	8.65005e+00
3	4.01189e+00	9.72731e+00
7	1.76161e+06	1.23957e+01
9	2.51658e+06	1.23957e+01
5	2.63860e+06	6.80780e+00

Al guardar los 3 mecanismos presentados, se puede calcular la probabilidad de falla total del sistema estructural, en el submenú Cálculo Pf del módulo de Confiabilidad. El resultado final de la probabilidad de falla del sistema estructural dados los mecanismos potenciales mostrados al inicio, se despliega en un menú como se muestra en la figura 4.12.

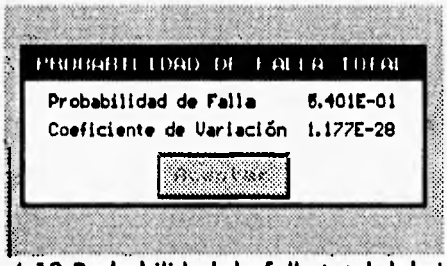
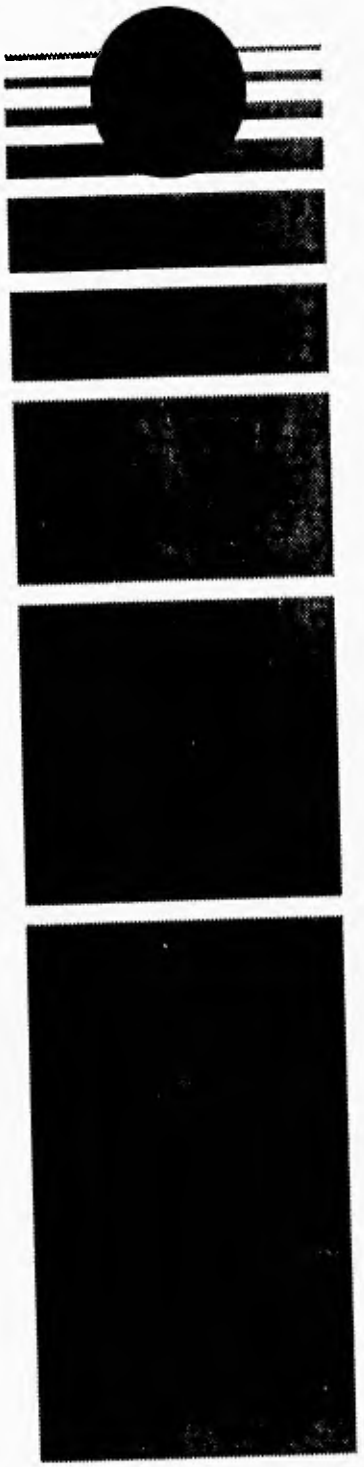


Figura 4.12 Probabilidad de falla total del sistema.

CONCLUSIONES



Actualmente en el departamento de Mecánica de la División de Estudios de Posgrado de la Facultad de Ingeniería se están desarrollando varios proyectos en el campo de la programación de sistemas para el análisis de sistemas estructurales, debido a la diversidad de métodos de estudio el camino por recorrer es todavía muy grande, pero ya se ha dado el primer paso para desarrollar software de buen nivel, que servirá para abrir las puertas para que otros compañeros continúen con el desarrollo del proyecto que se ha planteado inicialmente.

La tecnología orientada a objetos ha comenzado ha ser usada ampliamente en el desarrollo de diversos tipos de sistemas, en este trabajo específicamente para la creación del sistema CONFIA. Este sistema se ha visto muy beneficiado de las características de la tecnología orientada a objetos. El análisis y diseño se acerca más a la realidad y ha sido más fácil pasar de una etapa a otra. El maneja de menús, vectores y matrices, usados en CONFIA, ha sido más sencillo al considerarlos como objetos. Además se han implementado clases que pueden servir de base para futuros desarrollos como los cálculos matriciales y los menús, de esta forma los nuevos programadores no tendrán que comenzar desde cero.

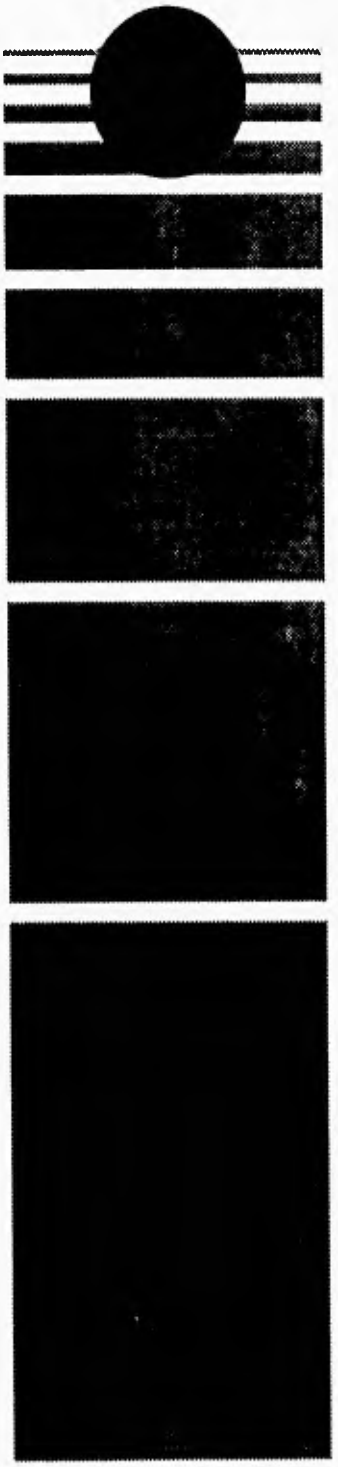
En un futuro próximo el sistema se ampliará de manera significativa, debido a que desafortunadamente en un proyecto de tesis no se pueden abarcar todas las posibilidades planteadas para desarrollar un software muy amplio y robusto tanto en variedad de opciones como en funcionalidad debido principalmente a la falta de tiempo y a los requerimientos que éste necesita; por esta razón las limitaciones encontradas en el sistema CONFIA serán superadas en un futuro cercano con la colaboración de otros tesistas e investigadores de la carrera de Ingeniería en Computación de la Facultad de Ingeniería principalmente.

Algunas de las mejoras que ya se tienen preparadas son: la migración de este sistema a otras dos plataformas de funcionamiento como lo son el ambiente Windows bajo el sistema operativo MS-DOS y el Open Windows que corre en estaciones de trabajo Sun para funcionar en un ambiente de red de tal forma que todos los alumnos e investigadores interesados en el tema tengan la herramienta a su disposición.

Otras de las funcionalidades que se le agregarán al sistema CONFIA, será la del análisis de estructuras tridimensionales, donde se agregarán diversos métodos de análisis de confiabilidad realizados por otros tesistas que ya se encuentran trabajando en ello. También se le implementará un módulo de análisis dinámico, con el objetivo de observar los fenómenos ocasionados en las estructuras por las acciones de los sismos, viento y oleaje de una forma más real que la encontrada en los programas actuales.

Como podemos darnos cuenta el camino por recorrer es todavía muy largo y con el tiempo irán surgiendo más necesidades de desarrollo que se podrán integrar al sistema sin ningún problema . para generar software de buen nivel, siguiendo las políticas de excelencia de la Facultad de Ingeniería de la Universidad Nacional Autónoma de México.

BIBLIOGRAFÍA



ANÁLISIS Y DISEÑO ORIENTADO A OBJETOS
James Martin-James J. Odell
ed. Prentice Hall Inc., 1994.

BORLAND C++ 3.1 OBJECT ORIENTED PROGRAMMING
Ted Faison
ed. Sams Pubishing, 1992.

DESIGNING OBJECT-ORIENTED SOFTWARE
Rebecca Wirfs-Brock, Brain Wilkerson, Lauren Wiener
ed. PTR Prentice Hall, Inc., 1990.

INGENIERÍA DE SOFTWARE, UN ENFOQUE PRÁCTICO
Roger S. Presssman
ed. McGrawHill, 1988.

LENGUAJE C, PROGRAMACIÓN AVANZADA
Herbert Schildt
ed. McGrawHill, 1988.

MASTERING BORLAND C++ 3.1
Tom Swan
ed. Sams, 1993.

OBJECT ORIENTED DESIGN WITH APPLICATIONS
Grady Booch
ed. Benjamin/Commings Publishing Company Inc., 1991.

OBJECT ORIENTED LANGUAGES SYSTEMS AND APPLICATIONS
Gordon Blair, John Gallagher
ed. Pitman Publishing, 1991.

OBJECT ORIENTED METHODS
Ian Graham
ed. Addison-Wesley Publishing Company, 1992.

OBJECT ORIENTED TECHNOLOGY: A MANAGER'S GUIDE
David Taylor, Ph.D.
ed. Servio Corporation, 1990.

PROGRAMACIÓN EN C++
Enrique Hernández Orallo
ed. Paraninfo, 1993.

SCIENTIFIC C++
Guido Buzzi-Ferraris
ed. Addison-Wesley Publishing, 1993.

TURBO C/C++ MANUAL DE REFERENCIA
Herbert Schildt
ed. Osborne/MacGraw-Hill, 1991.

TURBO C PARA IBM-PC Y COMPATIBLES
G. Leblanc
ed. Gustavo Gill. S. A, 1988.

DESARROLLO DE HERRAMIENTAS DE SOFTWARE PARA LA ENSEÑANZA DEL ANÁLISIS DE CONFIABILIDAD DE SISTEMAS ESTRUCTURALES
Reporte del proyecto no. CD702392-PAPIID-DGAPA.

MANUAL DE INSTALACIÓN DEL SISTEMA CONFIA

A.1 REQUERIMIENTOS DEL SISTEMA CONFIA

El sistema CONFIA es un programa diseñado para funcionar en computadoras personales IBM o compatibles, bajo el sistema operativo MS-DOS o cualquiera de sus variantes de acuerdo a la marca de computadora que se posea.

Habiéndose analizado el correcto funcionamiento del sistema CONFIA, se obtuvieron los requerimientos mínimos y óptimos para que este sistema trabaje de forma adecuada.

Los requerimientos mínimos del sistema CONFIA son:

- Computadora IBM-PC 80286 o compatible a 12mhz.
- Ratón compatible con Microsoft o IBM.
- Monitor VGA, con resolución de 640 X 480 pixeles monocromático.
- MS-DOS versión 5.0.
- Disco duro con 1Mb de espacio libre.
- 450Kb libres de memoria RAM.

Los requerimientos recomendados del sistema son:

- Computadora IBM-PC 80386DX o compatible a 40mhz o superior con coprocesador matemático.
- Ratón compatible con Microsoft o IBM.
- Monitor VGA, con resolución de 640 X 480 pixeles a colores.
- MS-DOS versión 6.2.
- Disco duro con 2.5Mb de espacio libre.
- 600Kb libres de memoria RAM.

CONFIA necesita de una computadora, con coprocesador matemático (opcional), debido al gran manejo de números, por lo mismo también se recomienda un equipo que cuente con un procesador de gran velocidad. El sistema hace uso del ratón durante toda su ejecución por lo tanto no se puede hacer uso de CONFIA si no se cuenta con un ratón. Además el sistema se debe instalar en una unidad de disco duro con una cantidad mínima de 2.5Mb ya que CONFIA hace uso de buffers de intercambio de memoria, y se crean archivos de resultados para que puedan ser vistos por el usuario.

A.2 INSTALACIÓN DEL SISTEMA CONFIA

Para instalar el sistema CONFIA en el disco duro se hace un directorio con el nombre deseado y se copian todos los archivos del disco flexible al disco duro de la siguiente forma:

- MD C:\CONFIA — se crea un subdirectorio para el sistema CONFIA.
- COPY A:*.* C:\CONFIA — se copian todos los archivos del disco flexible al subdirectorio CONFIA del disco duro.

- C:\COFIA\CONFIA se ejecuta el programa CONFIA.

Para el buen funcionamiento del sistema CONFIA su directorio debe contar con los siguientes archivos:

- CONFIA.EXE programa ejecutable para el análisis de confiabilidad.
- CONFBMP.EXE presentación del sistema CONFIA.
- CONFIA.CFG archivo de configuración general del sistema donde se tienen los límites del plano real, los valores del "snap" en los ejes X y Y, el tipo de estructura, tipo de análisis, tipo de interacción, que serán tomados por defecto al momento de cargar el sistema. El usuario puede activar los valores que desee tomar por defecto la próxima vez que entre a CONFIA con el submenú Configuración, opción Guardar-Configuración, del módulo de Edición del menú principal del programa.
- AYUDA.HLP archivo de ayuda, que es accesado cada vez que el usuario teclea F1, o acude al módulo de Ayuda. Este archivo presenta una ayuda instantánea general de todos los tópicos presentes en el sistema.
- AYUDA.IND archivo que presenta el índice de todos los tópicos de los que se puede obtener ayuda instantánea durante la operación del programa.
- ICONOS.BIT archivo que tiene todos los iconos que se presentan en la pantalla de trabajo del programa CONFIA.

CONFIA puede crear en el directorio por defecto algunos archivos que pueden ser de interés o no dependiendo de las necesidades que tenga cada usuario del sistema y del uso que se le pretenda dar. Los archivos de información generados por el programa son:

- RRESULT.DAT archivo de resultados del último análisis mecánico realizado antes de salir del programa, el usuario puede visualizar este archivo con cualquier editor de texto y verificar numéricamente los resultados del análisis de la estructura.
- RESULT.SIM archivo de resultados del último análisis de confiabilidad antes de dejar el programa. El usuario puede visualizar los resultados de estos análisis.

Los archivos temporales que genera el programa para intercambio de memoria y los cuales podrán ser eliminados al finalizar la sesión son:

- MENSAJE.TMP
- VERDIR.TMP
- AYUDA.TMP
- MENUPR.TMP
- MENU.TMP

A.3 ESPECIFICACIONES DE OPERACIÓN DEL PROGRAMA

El sistema CONFIA como cualquier paquete de software tiene ciertas limitaciones de operación debido a la capacidad de memoria que la computadora posee, debido a este problema se tienen límites de operación, como por ejemplo: el número de nodos y barras que se pueden

editar, el número de condiciones de carga que se pueden aplicar, el número de mecanismos que se pueden almacenar, etc.

Los límites máximos de edición de la estructura son:

- 150 nodos
- 150 barras
- 50 condiciones de carga se pueden aplicar
- 25 tipos de materiales se pueden definir
- 10 mecanismos como máximo se pueden almacenar

Los límites de visualización que el programa soporta son los siguientes:

- La pantalla de edición máxima es: 200 mts (-99.99 mts a +99.99 mts)
- La pantalla de edición mínima es: 0.1 mts.
- La escala mínima para los snaps es de 0.01 mts.
- La escala máxima de los snaps es de 10.0 mts.
- La escala máxima del mallador es de 120 puntos tanto horizontales como verticales, la fórmula para obtener esta relación es de la siguiente forma: No. de puntos = (límite máximo - límite mínimo) / escala del snapx o snapy, así por ejemplo si tenemos límites de -10 a 10 mts y la escala del snapx es de 0.05 mts. el número de puntos del mallador será (10-(-10)) /0.05 = 400 puntas por lo tanto el mallador na se podrá visualizar, habrá que cambiar ya sea los límites de la pantalla o la escala de los snaps.

Las unidades de medición que se aplican en el sistema CONFIA son las siguientes:

Longitud - metros (mts o m)
Fuerza - toneladas (Tn o T)
Ángulos - radianes (rad o r)

MANUAL DEL USUARIO DEL SISTEMA CONFIA

B.1 INTRODUCCIÓN

La presentación de este programa es del tipo interactivo, auxiliado con la utilización del ratón, así como con menús, iconos y ayudas para facilidad del utilizador. En la parte superior de la pantalla aparece el menú principal, en la parte inferior de la pantalla aparece una barra de estatus donde se indica alguna información complementaria, como son las coordenadas del ratón, el menú donde se encuentra el utilizador, en la parte derecha superior se establece un menú con iconos, y en la parte derecha inferior el valor de algunas variables generales como el tipo de análisis, tipo de interacción, el tipo de estructura, los límites de la pantalla, el snap. En la figura B.1 se presenta la imagen de presentación del programa.

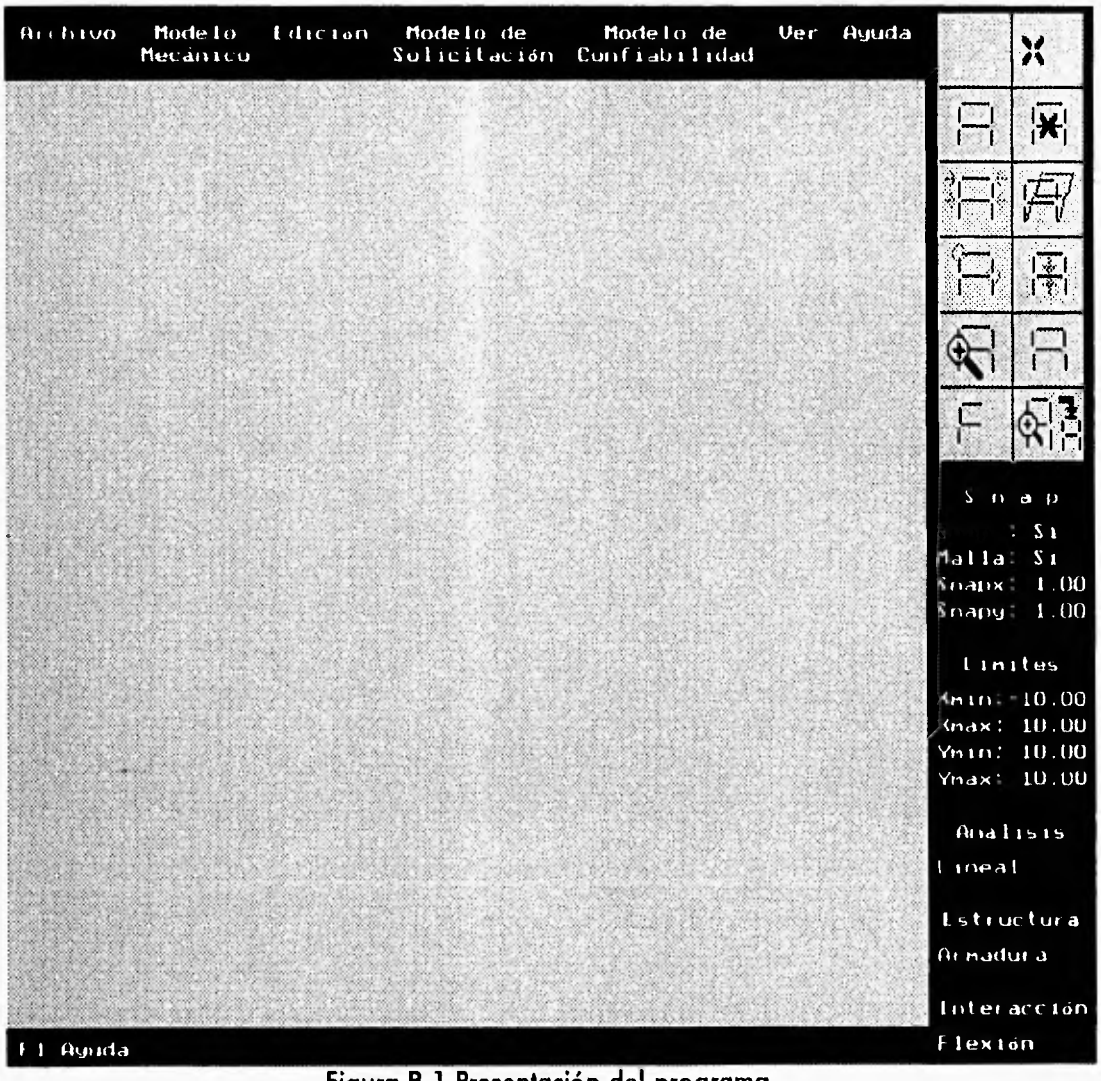


Figura B.1 Presentación del programa.

En el menú principal se tienen las opciones: Archivo, donde se efectúa el manejo de los archivos de datos de la estructura que se analizará, Modelo Mecánico, donde se establecen algunas variables generales como son el tipo de análisis, tipo de estructura, así como donde se ejecuta el análisis estructural, Edición, donde se crea o edita la estructura por analizar, Modelo de Solicitación, donde se calculan las fuerzas que actúan sobre la estructura, Modelo de Confiabilidad, donde se realiza el análisis de confiabilidad estructural, Ver, que presenta en forma visual los resultados del análisis mecánico, y Ayuda, donde se dan algunos comentarios sobre el método de solución del problema y sobre el manejo del programa.

B.2 MÓDULO DE ARCHIVO

El módulo de Archivo consta de 7 opciones como se muestra en la figura B.2, para seleccionar alguna de estas opciones sólo posicione el apuntador del ratón en la opción requerida y oprima el botón izquierdo.

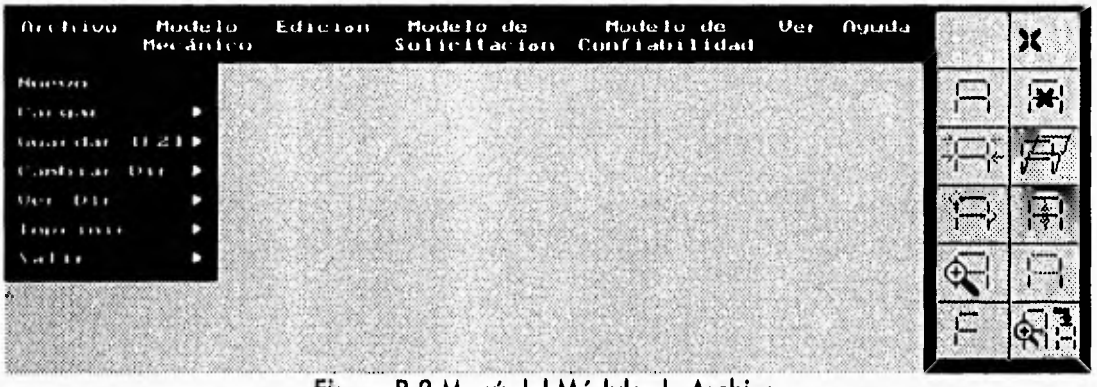


Figura B.2 Menú del Módulo de Archivo.

Nuevo

Esta opción inicializa toda la memoria del programa para comenzar una nueva estructura y asigna por defecto los valores de configuración así como el nombre de la estructura a editar (NUEVO.FIN). Se puede cambiar este nombre, indicando en Arch del submenú de la figura B.3, el nombre de archivo que se desea.

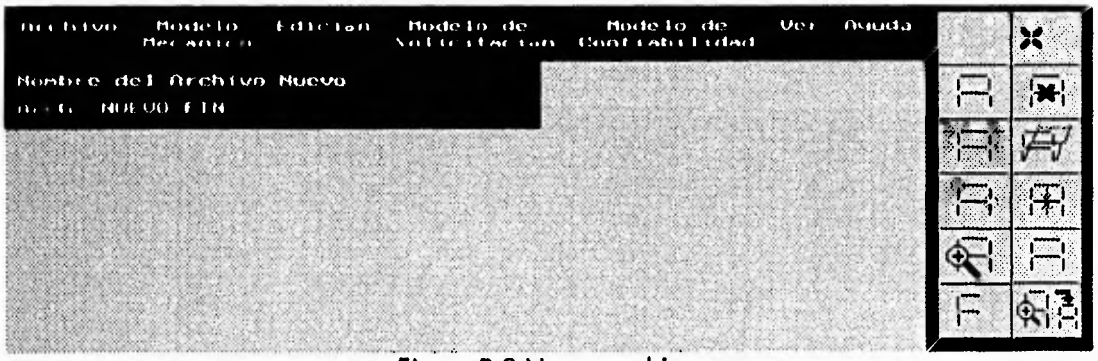


Figura B.3 Nuevo archivo.

Cargar

Esta opción permite cargar un archivo de una estructura ya existente del disco a la memoria del programa. Al seleccionar esta opción aparece un submenú como el de la figura B.4. La opción de **Dir Actual** indica el directorio y la unidad de disco en la que se encuentra actualmente y la opción de **Archivo** pide el nombre del archivo a cargar, se le puede dar el nombre del archivo sin trayectoria de directorio en la cual cargará el archivo del directorio actual que se indica en **Dir Actual**; si se desea cargar un archivo de otro directoria o unidad de disco se puede poner la trayectoria completa del archivo a cargar; por ejemplo se analizan dos casos comunes:

Dir Actual : C:\CONFIA
Archivo : D:\USUARIOS\ESTRUC.FIN
En este caso se encuentra en el directorio C:\CONFIA y se está cargando un archivo del directorio : D:\USUARIOS\ESTRUC.FIN.

Dir Actual : C:\CONFIA
Archivo : ESTRUC.FIN
En este caso se encuentra también en el directorio C:\CONFIA y se está cargando un archivo del directorio por defecto, es decir se está cargando el archivo C:\CONFIA\ESTRUC.FIN.

Guardar

Esta opción permite guardar un archivo de una estructura que ya se ha editado en el programa. Al seleccionar esta opción aparece un submenú como el de la figura B.5. La opción de **Dir Actual** indica el directorio y la unidad de disco en la que se encuentra actualmente y la opción de **Archivo** pide el nombre del archivo a guardar, se le puede dar el nombre del archivo sin trayectoria de directorio en la cual cargará el archivo del directorio actual que se indica en **Dir Actual**; si se desea guardar un archivo en otro directorio o unidad de disco se puede poner la trayectoria completa del archivo a guardar, el funcionamiento de estas opciones es análogo a la opción anterior **Cargar**. Se puede usar la tecla **F2** desde cualquier parte del programa para **Guardar** la estructura editada, pero antes de usar esta opción deberá tener un nombre el archivo a guardar, de lo contrario se grabará con el nombre "ANONIMO.FIN".

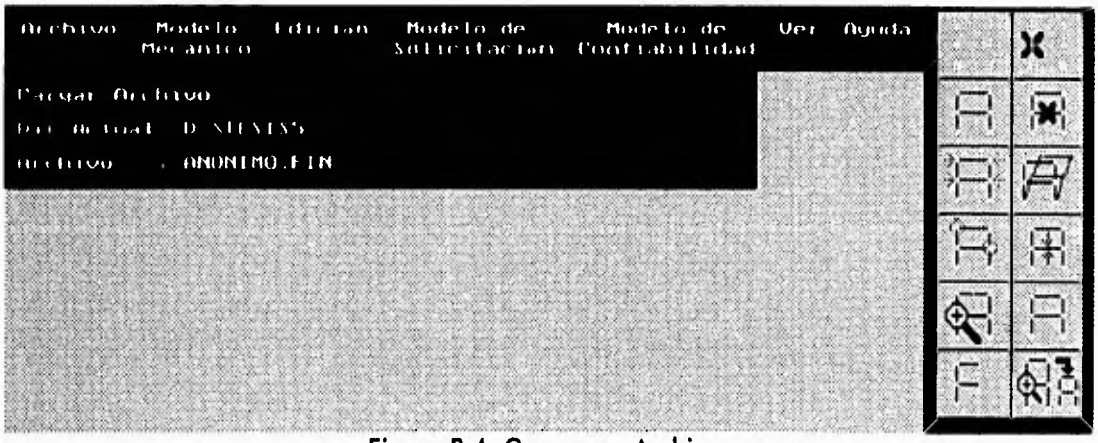


Figura B.4 Cargar un Archivo.

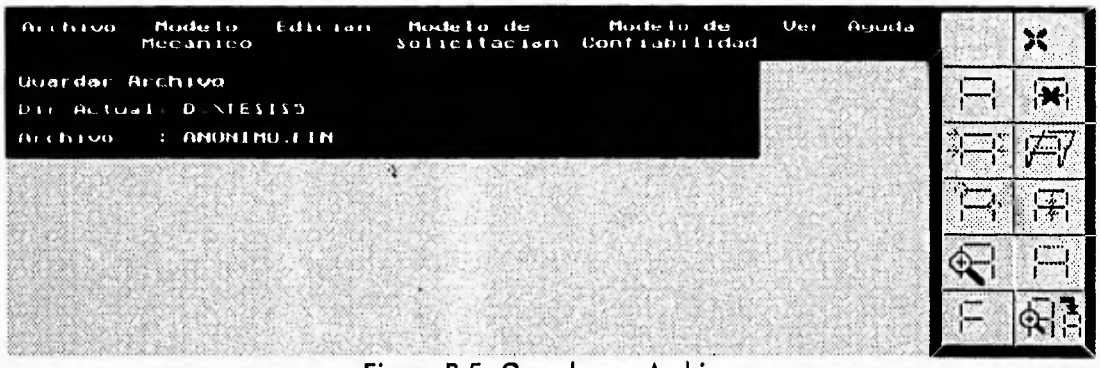


Figura B.5 Guardar un Archivo.

Cambiar Dir

Esta opción permite cambiar de directorio, así como de unidad de disco; al seleccionar esta opción aparece un submenú como el de la figura B.6, en la cual la opción Dir indica el directorio actual donde se está localizado sin especificar la unidad de disco, si se desea cambiar el directorio de trabajo se escoge esta opción y se teclea el nuevo directorio teniendo en cuenta que no se debe especificar la unidad de disco, para tal efecto hay que elegir la opción de Unidad: en la cual se indica la unidad de disco en la que se esta ubicado, si se quiere seleccionar una unidad diferente, se escoge esta opción y aparece una pequeña lista de unidades disponibles para cambiar. Se analiza un par de ejemplos:

Dir : \CONFIA
Unidad : C:
En este caso se encuentra en el directorio \CONFIA de la unidad: C: donde la trayectoria completa del directorio será: C:\CONFIA, se selecciona en Unidad la unidad de disco A: y en Dir se pone \, entonces aparecerá lo siguiente:
Dir : \
Unidad : A:

Lo que indica que estamos en el directorio raíz de la unidad A:. Si a continuación se selecciona la opción Unidad: y a continuación en la lista de unidades de disco disponibles seleccionamos la unidad C:, se regresará al directorio original que se tenía.

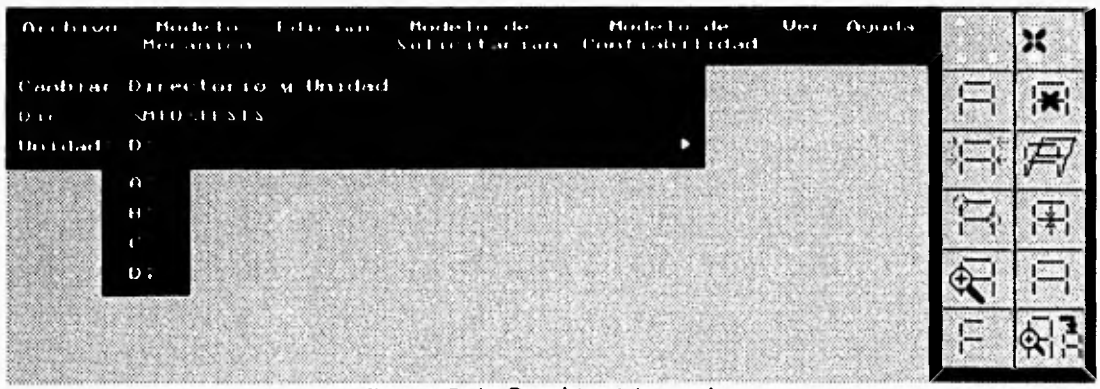


Figura B.6 Cambiar Directorio.

Verdir

Esta opción permite ver el directorio que se especifique en el submenú, figura B.7 para posteriormente seleccionar un archivo que será cargado en la memoria para su edición. Para realizar todo esto primero se selecciona del menú de Archivo, la opción de Verdir, a continuación aparece un submenú en el que se pide un directorio para visualizar los archivos que ahí se encuentran, en esta opción son válidas las combinaciones del comando de MS-DOS **dir**, por ejemplo:

Dir : *.* en este caso se listan todos los archivos del directorio actual.
Dir : *.FIN aquí se listan todos los archivos con extensión FIN del directorio actual.
Dir : C:\CONFIA*.FIN en este caso se mostrará todos los archivos que estén en el directorio C:\CONFIA y que tengan la extensión FIN.

Después de haber tecleado alguna opción valida aparecerá un recuadro en medio de la pantalla con la lista de todos los archivos que queremos visualizar, se podrá seleccionar algunos de estos archivos para editarlo posicionándose con el ratón en el archivo y oprimiendo el botón izquierdo para cargarlo en la memoria del programa, sino se quiere cargar ningún archivo de los que aparecen basta con oprimir el botón derecho del ratón para salir de esa opción.

Ahora si los archivos listados son más de 30 se podrá recorrer toda la lista de ellos oprimiendo el botón izquierdo del ratón en alguno de los recuadros que aparecen a la derecha del cuadro principal y así mismo seleccionar uno para su edición.

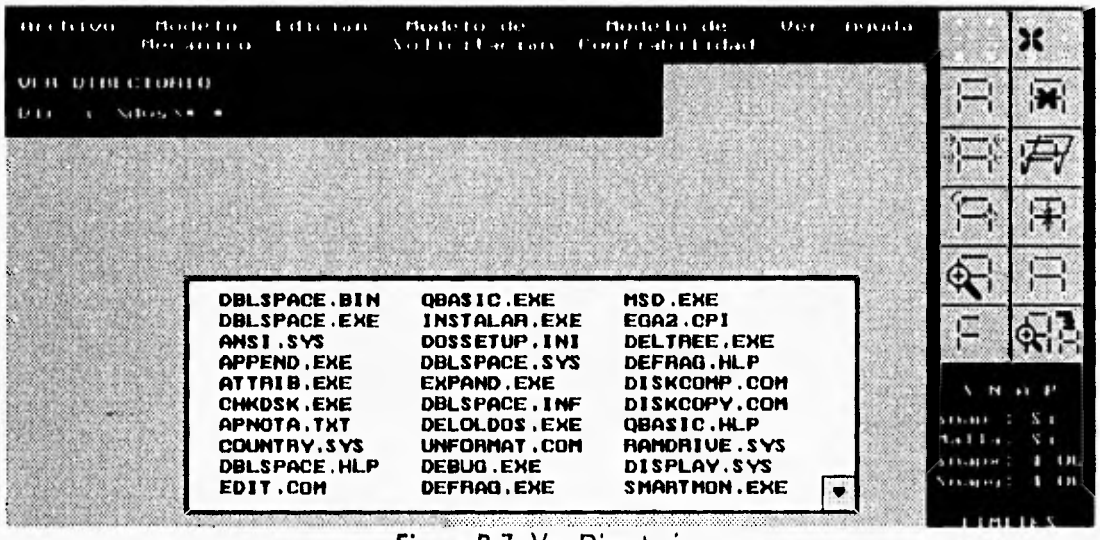


Figura B.7 Ver Directorio.

Imprimir

Esta opción permite realizar una impresión en papel de la estructura que se está analizando. Consta de un submenú de 3 opciones como se muestra en la figura B.8, estas opciones son:

Archivo: esta opción del submenú Imprimir permite mandar a la impresora toda la información relacionada con la estructura sin analizar.

Desplazamientos: esta opción permite imprimir los desplazamientos obtenidos del análisis de la estructura para cada nodo.

Fuerzas Internas: esta opción permite imprimir los resultados de las fuerzas internas como resultado del análisis de la estructura. Cabe mencionar que la impresora debe estar lista y en línea para recibir la información de lo contrario, se mandará un mensaje de error de impresión y se tiene que valver a realizar la operación.

Salir

Esta opción de módulo de Archivo permite salir de programa y consta de un submenú con dos opciones de salida como se muestra en la figura B.9 y que a continuación se explican.

La opción de Guardar y Salir permite salir del programa pero antes guarda la estructura en un archivo de disco con el nombre dado por defecto y toda la información de la configuración del programa en el archivo confia.cfg que posteriormente usará para cargar el programa en futuras sesiones. La opción de Salir permite salir del programa sin guardar la estructura, ni la configuración del programa.

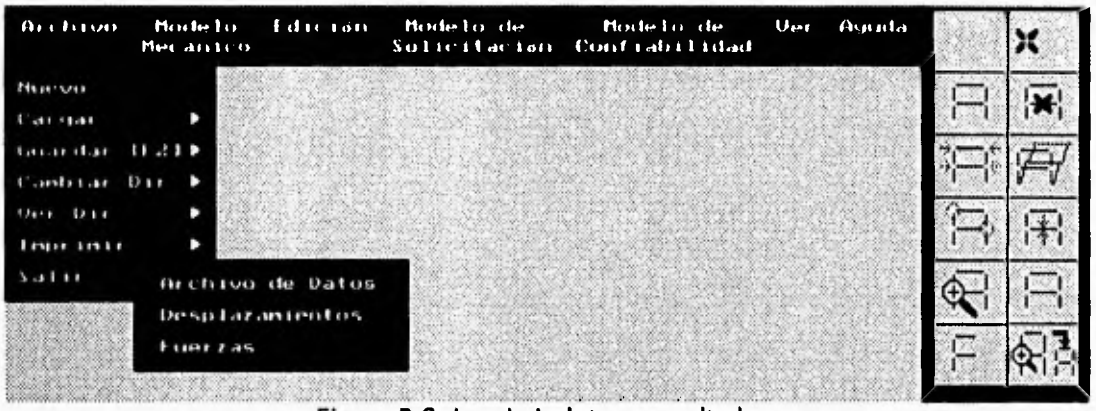


Figura B.8 Imprimir datos y resultados.

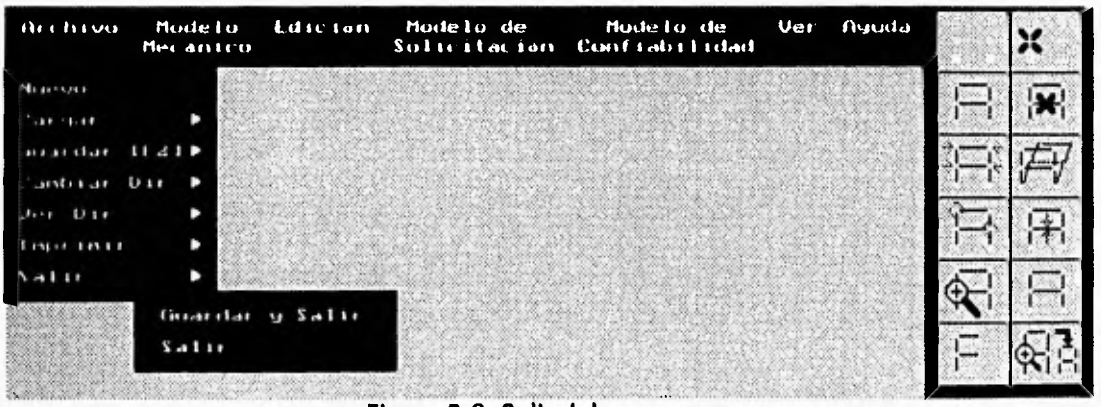


Figura B.9 Salir del programa.

B.3 MÓDULO DEL MODELO MECÁNICO

La ventana Modelo Mecánico consta de 5 opciones como se muestra en la figura B.10.

En el submenú Estructura se define el tipo de estructura que se quiere analizar, los tipos pueden ser: Armadura y Marco en el plano, como se muestra en la figura B.11; por defecto se asume Marco, como la estructura a editar.

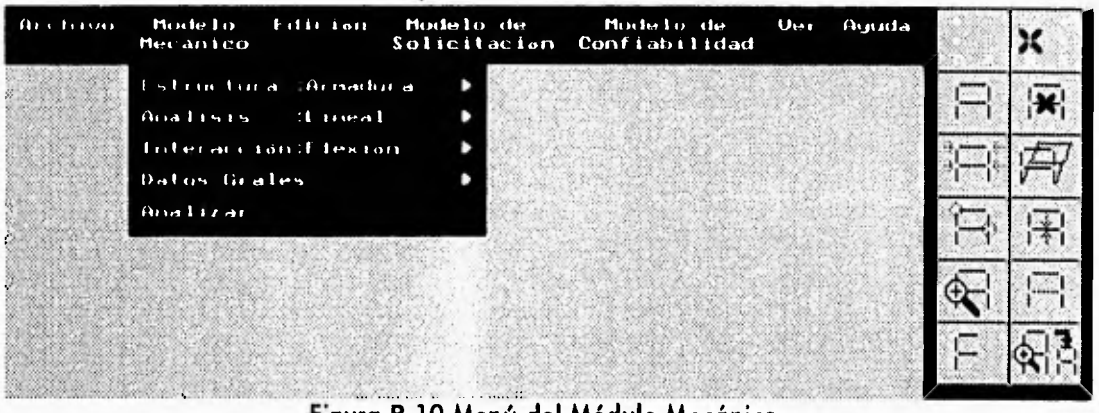


Figura B.10 Menú del Módulo Mecánico.

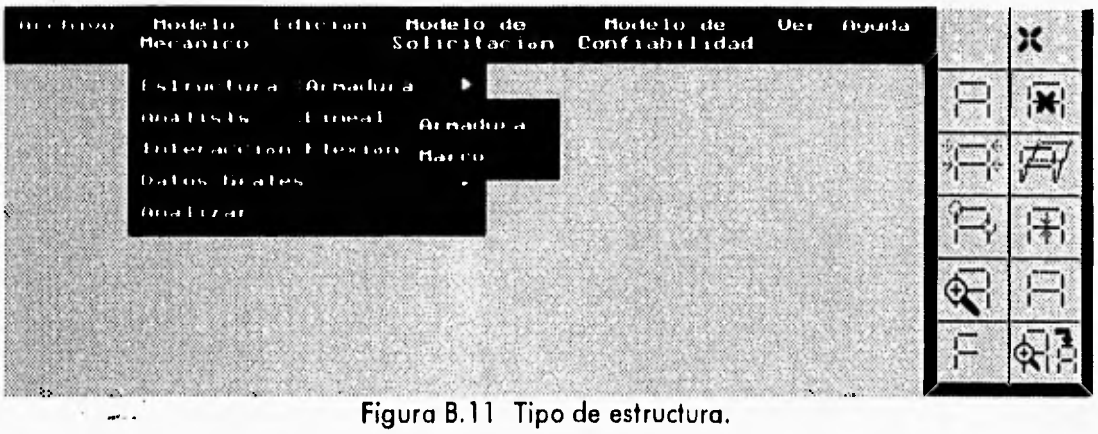


Figura B.11 Tipo de estructura.

En el submenú **Análisis** se define el tipo de comportamiento de los elementos estructurales y del sistema. Los tipos pueden ser: Lineal, Lineal Crítico, Elastoplástico y Elastoplástico Crítico, como se muestra en la figura B.12. Por defecto se considera un análisis lineal.

En el submenú **Interacción** se determina el tipo de interacción entre las fuerzas que se consideran en la fluencia del material. Los tipos de interacción pueden ser: a pura Flexión, que toma únicamente el efecto de la flexión en el elemento, F-M, que toma en cuenta los efectos combinados de carga axial y momento flexionante en el elemento, F-M-V, que toma en cuenta los efectos combinados de carga axial, momento flexionante y cortante en el elemento; y a

Tens-Compr, que toma en cuenta únicamente el efecto de la carga axial en tensión y comprensión en el elemento (caso de las armaduras). La figura B.13 muestra las opciones. Por defecto se asume a flexión.

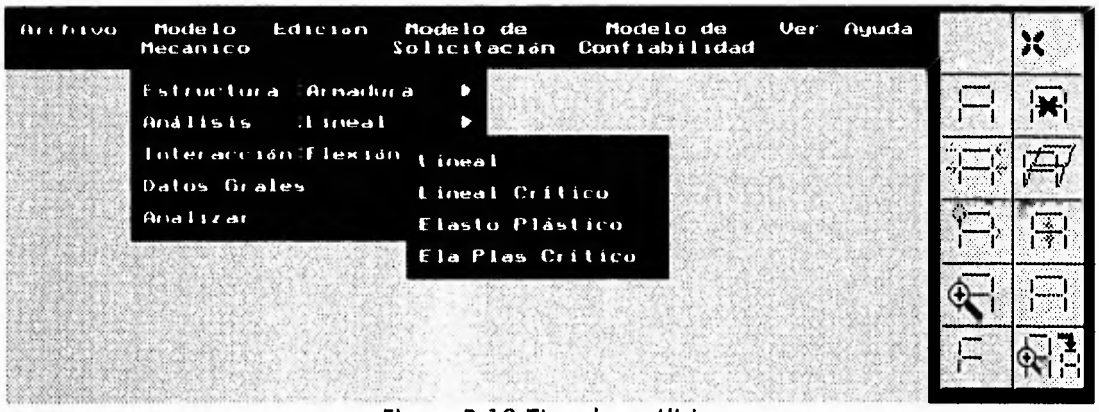


Figura B.12 Tipo de análisis.

En el submenú Datos Grales. de la figura B.14 es necesario determinar:

a) **Car Deter**, si existen cargas deterministas o no, figura B.15, por defecto es No.
b) **Tipo V.A.** , el tipo de variables aleatorias, puede ser Normal o Log-Normal, figura B.16, el valor por defecto es Normal
c) **No. S.M.E.**, el número de secciones más esforzadas por identificar.
d) **Fy**, el valor del esfuerzo de fluencia del material.

Toda la información anterior es necesaria para realizar los análisis críticos.

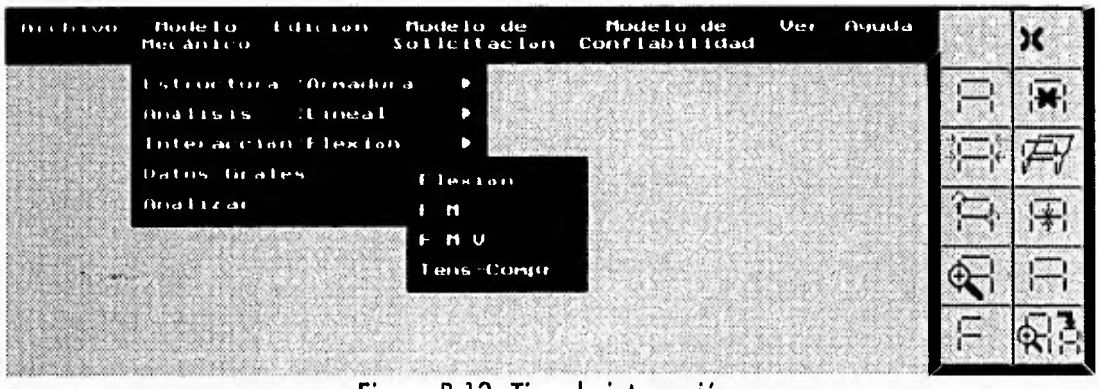


Figura B.13 Tipo de interacción.

Analizar
Una vez que se tiene una estructura, con todas las características necesarias, se puede desarrollar el análisis mecánico, seleccionando la opción Analizar del Modelo Mecánico.

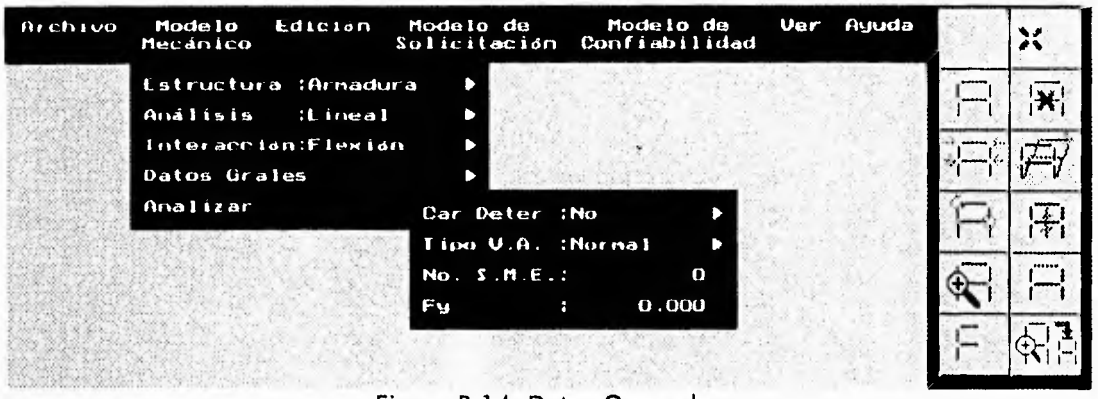


Figura B.14 Datos Generales.

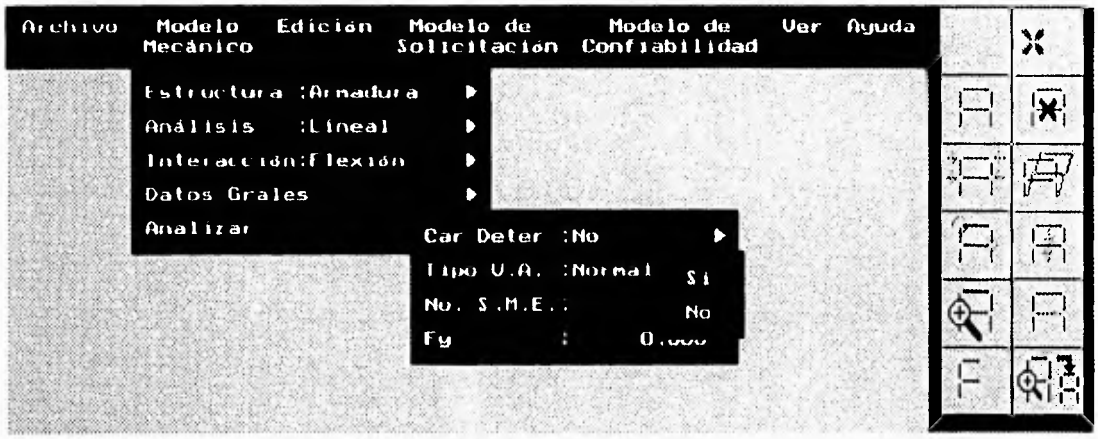


Figura B.15 Cargas Deterministas.

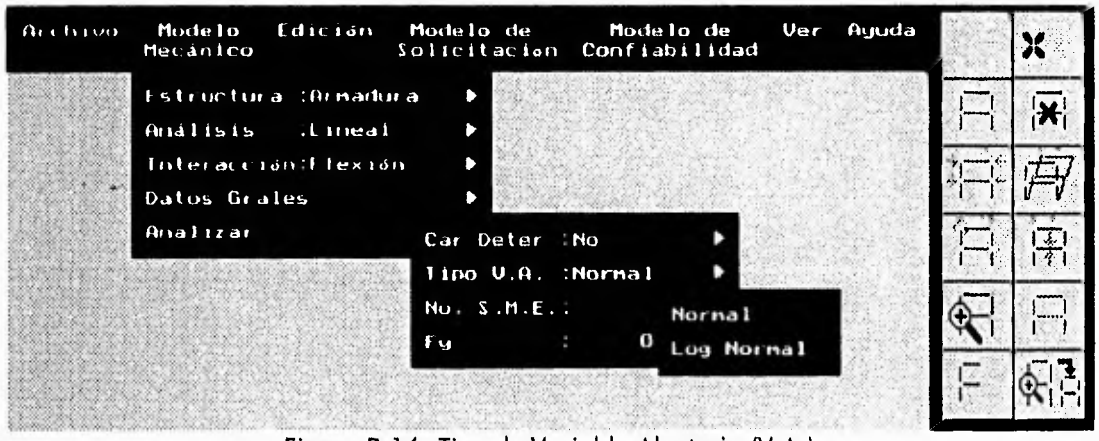


Figura B.16 Tipo de Variable Aleatoria (V.A.).

B.4 MÓDULO DE EDICIÓN

Este módulo tiene nueve opciones como se muestra en la figura B.17. Todas las opciones también pueden ser llamadas con los iconos de la parte derecha del menú principal, a excepción de Generar nodos.

Editar Nodos

En esta opción se pueden crear nodos o visualizar las características de los nodos ya existentes (coordenadas en X ,Y y grado de libertad), oprimiendo el botón izquierdo del ratón en este submenú, desaparece el menú principal y se tiene el cursor de edición de nodos; en la barra de estatus se visualizan las coordenadas del movimiento del cursor, si ya existe un nodo en tales coordenadas también aparece el número de nodo, si se desea dibujar un nuevo nodo o cambiar las características de un nodo ya existente sólo basta con oprimir el botón izquierdo del ratón en las coordenadas deseadas, automáticamente aparece un submenú al lado del nodo, como el que se muestra en la figura B.18, dicho submenú nos muestra el número del nodo, las coordenadas en el eje X y Y. Si se desea salir de Editar Nodos se debe oprimir el botón derecho del ratón.

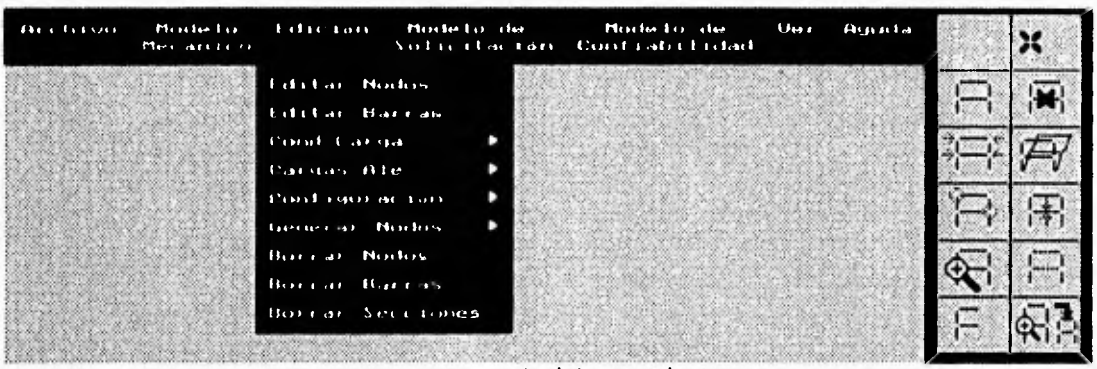


Figura B.17 Módulo de Edición.

En el submenú **dx-dy-dz,** de editar nodos, se pueden asignar los grados de libertad del nodo. Oprimiendo el botón izquierdo en dicho submenú aparecerá otro submenú como el de la figura B.19. Los tipos pueden ser:

'...'	NODO libre en Z, Y y X,	'..X'	NODO libre en Z y Y
'.Y.'	NODO libre en Z y X,	'.YX'	NODO libre en Z
'Z. '	NODO libre en Y y X,	'Z.X'	NODO libre en Y
'ZY.'	NODO libre en Z y Y,	'ZYX'	NODO totalmente empotrado

Si se desea cambiar las coordenadas del nodo, se debe oprimir el botón izquierdo del ratón en el submenú **Mover** de la figura B.18, automáticamente aparece un submenú, como el de la figura B.20 en el que se pueden cambiar las coordenadas. Únicamente si se oprime el submenú de **Mover** en este último menú, se moverá el nodo a las nuevas coordenadas solicitadas, de lo contrario seguirá en las mismas coordenadas.

Figura B.18 Editar Nodos.

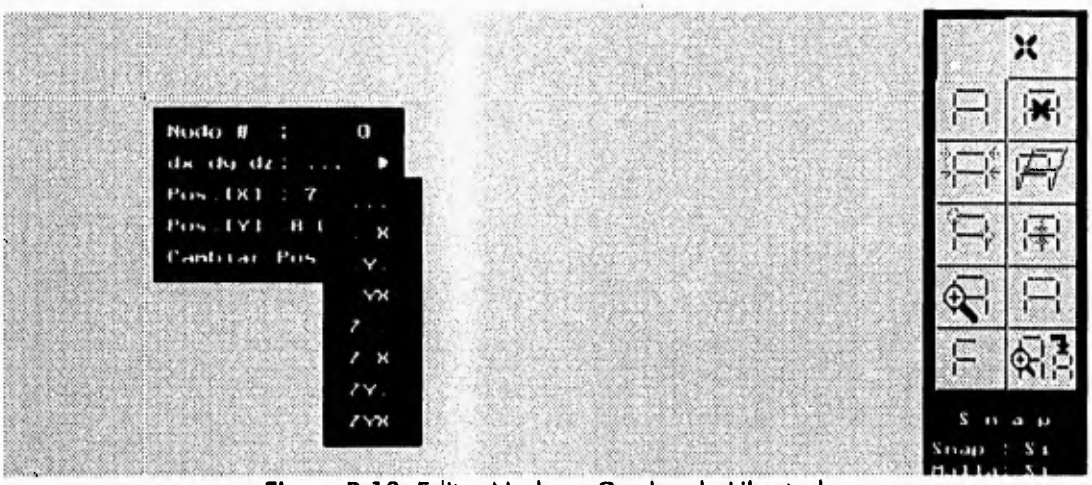
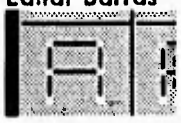


Figura B.19 Editar Nodos y Grados de Libertad.

Editar Barras



Oprimiendo el botón izquierdo en Editar Barras del menú de Edición (figura B.17) se pueden crear barras. Para indicar cuales son los nodos extremos de un barra se oprime el botón izquierdo del ratón en los 2 nodos deseados, automáticamente aparece un submenú al lado de la barra editada. Para análisis lineal o lineal crítico es semejante al de la figura B.21. Para análisis elastoplástico o elastoplástico crítico es parecido al de la figura B.22. En Editar Barras se proporciona: la Longitud del elemento de acuerdo con las coordenadas de sus nodos extremos, el Ángulo que forma la barra con respecto al eje horizontal, el Nodo A, el número del nodo en el extremo A y el Nodo B, el número del nodo en el extremo B. A cada nodo se le puede asignar su resistencia media Res Med y su coeficiente de variación C.V. (figura B.23). Si se quiere salir de Editar Barras, oprima el botón derecho del ratón.

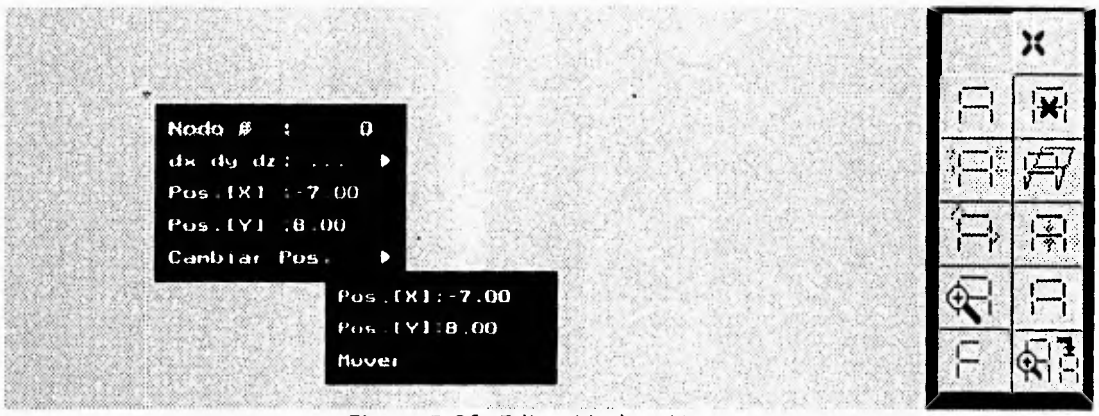


Figura B.20 Editar Nodos, Mover.

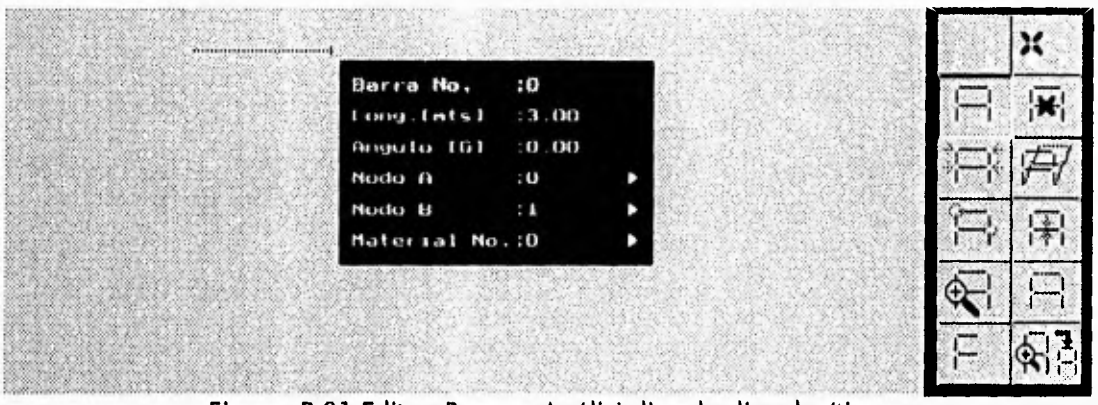


Figura B.21 Editar Barras, Análisis lineal o lineal crítico.

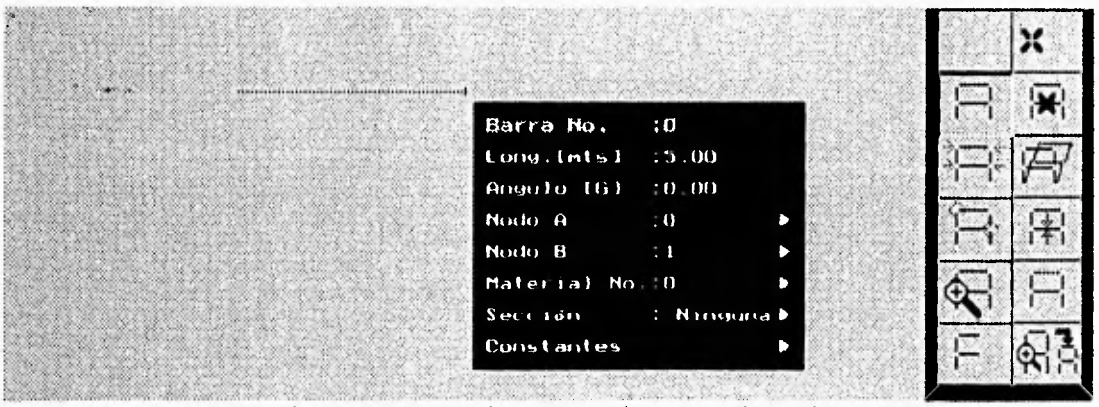


Figura B.22 Editar Barras, Análisis Elastoplástico o Elastoplástico Crítico.

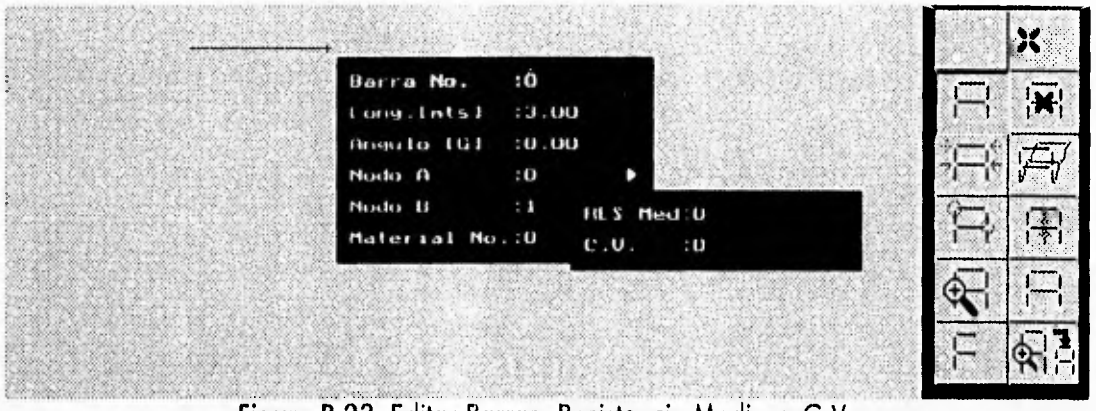


Figura B.23 Editar Barras, Resistencia Media y C.V.

En el submenú **Material**, es necesario proporcionar las propiedades geométricas de la barra: **E**- Módulo de elasticidad; **Área** - de la sección transversal; Iz - Momento de inercia, como se muestra en la figura B.24.

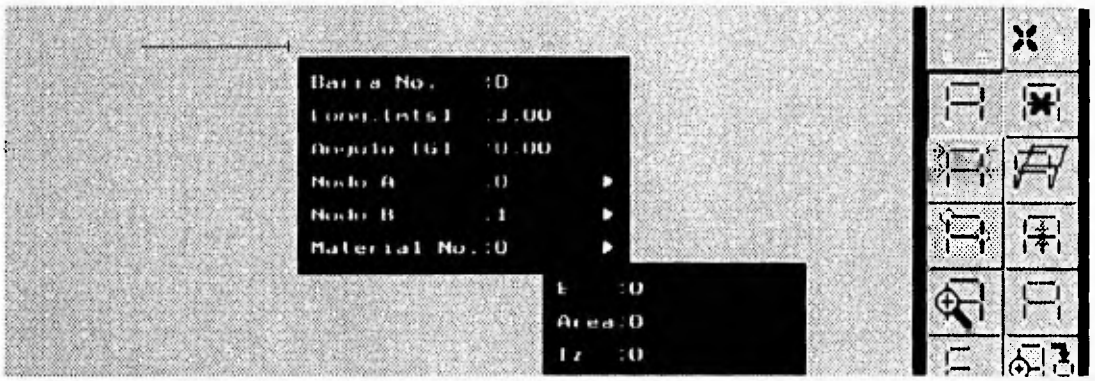


Figura B.24 Editar Barras, Material.

Para los análisis elastoplástico y elastoplástico crítico, en Sección, figura B.25, se proporcionan las condiciones de fluencia en los extremos de la barra: Ninguna = Elástico, Inicial = Plastificación en el extremo izquierdo, Final = Plastificación en el extremo derecho y Ambas = Plastificación en ambos extremos.

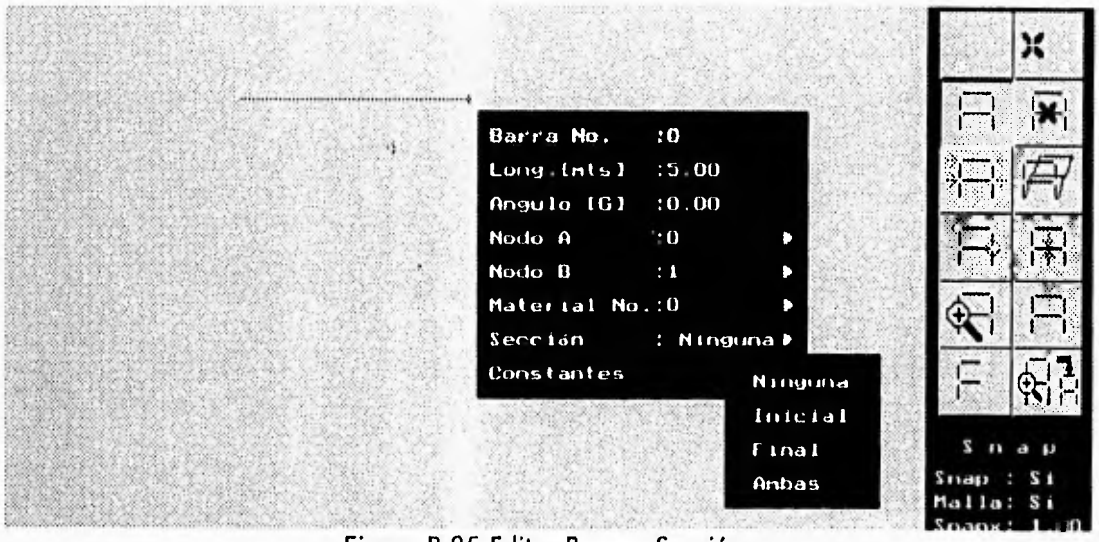


Figura B.25 Editar Barras, Sección.

En el submenú Constantes se define el estado límite plástico de la sección del elemento: C1, C2, C3 son los coeficientes de la función de estado límite de la sección extrema izquierda o del nodo inicial, C4, C5, C6 son los coeficientes de la función de estado límite de la sección extrema derecha o del nodo final y RP la resistencia límite de referencia.

Para interacción tipo flexión se muestra el submenú de la figura B.26, para interacción tipo F-M se muestra el submenú de la figura B.27 y para interacción tipo F-M-V se muestra el submenú de la figura B.28

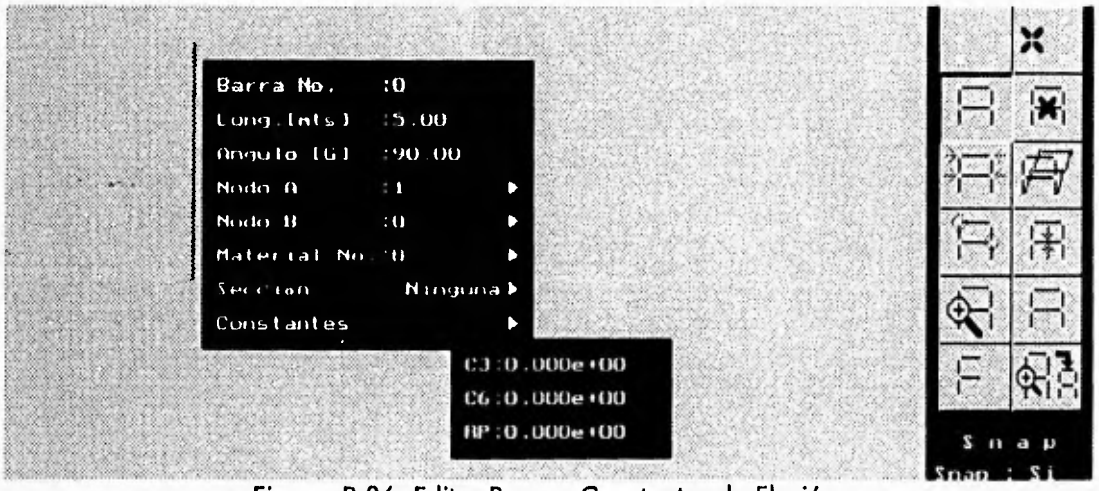


Figura B.26 Editar Barras, Constantes de Flexión.

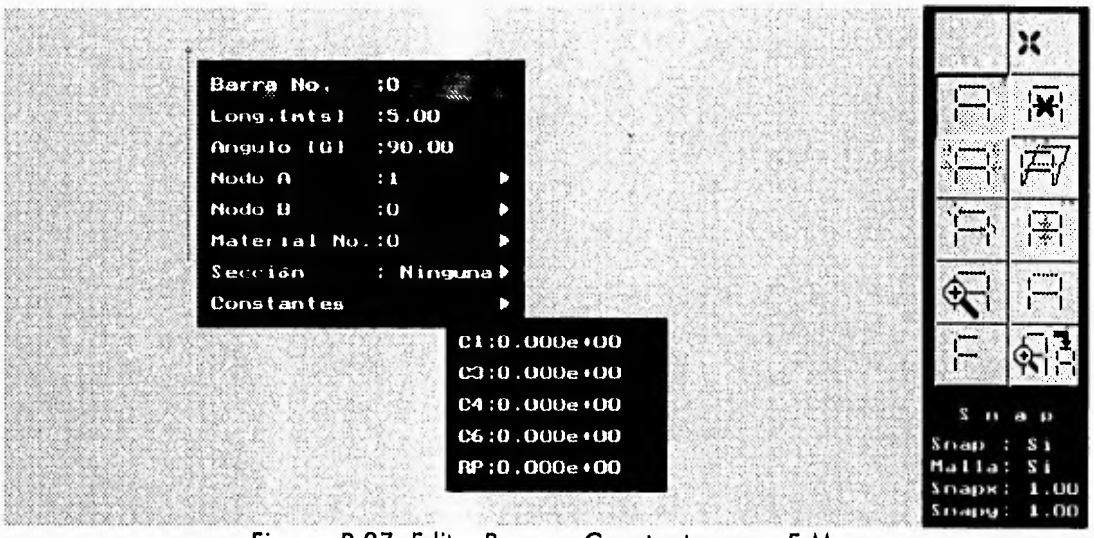


Figura B.27 Editar Barras, Constantes para F-M.

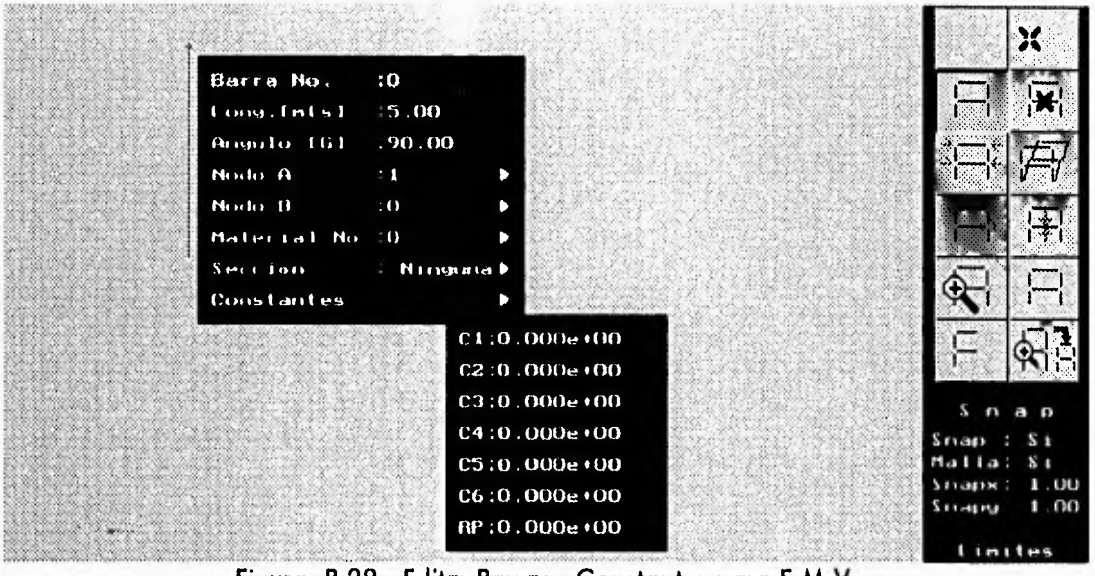
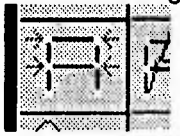


Figura B.28 Editar Barras, Constantes para F-M-V.

Cond. de Carga



En este submenú se indica: el Total de C. de Carga que se van a tener en la estructura, ver o editar una condición de carga en particular, Ver Cond. de Carga, y ver o editar todas las condiciones de carga que tenga la estructura de forma secuencial, Ver Todas las Cargas, como se muestra en la figura B.29.

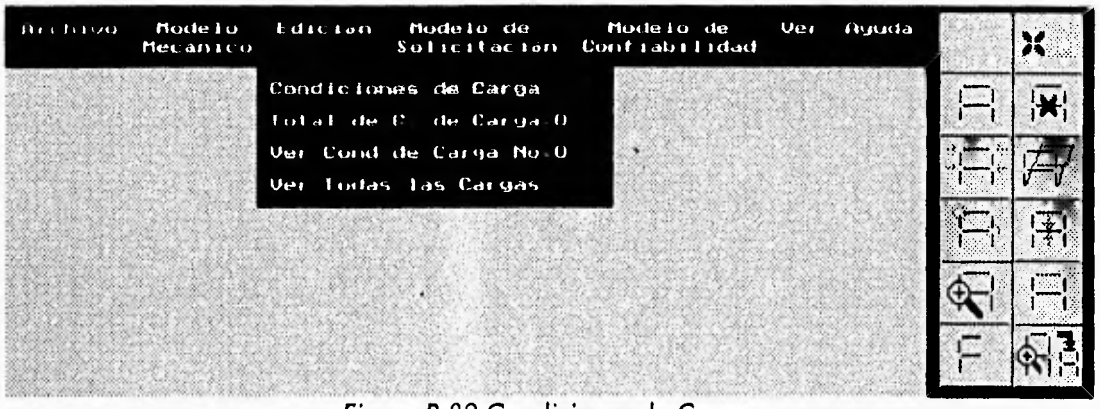


Figura B.29 Condiciones de Carga.

En Ver Cond. de Carga se especifica la condición de carga que se quiere editar, nuevamente se borra el menú principal para una mejor visualización de la estructura, se oprime el botón izquierdo en el nodo donde se quiere aplicar una carga y aparecerá el submenú de la figura B.30 donde se pide la fuerza en X: FX, fuerza en Y: FY y momento en Z: MZ; si el nodo ya tiene especificados valores para esa condición de carga aparecerán en el submenú. También se informa el número de nodo: Nodo No. y el número de condición de carga: C. Carga No.

Se hace la observación que para efectuar los análisis de confiabilidad de las estructuras, es necesario:
a) dar en la primera condición de carga, todas las cargas aleatorias que están aplicadas en el sistema estructural.
b) a partir de la segunda condición de carga dar una carga aleatoria por condición de carga y
c) en la última condición de carga dar o aplicar todas las cargas deterministas, si existen y fueron declaradas en los datos generales.

Por lo tanto el número de cond. de carga será igual a el número de cargas aleatorias más 2.
Si se quiere salir de Ver Cond. de Carga oprima el botón derecho del ratón.

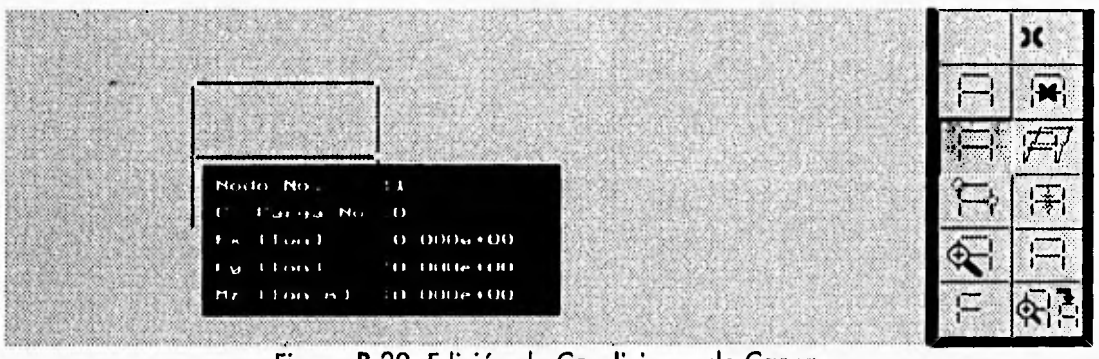


Figura B.30 Edición de Condiciones de Carga.

Cargas Ale.

En este submenú (figura B.31) se definen el número Total de Cargas Aleatorias que se tienen en la estructura, se puede ver o editar una carga aleatoria específica con la opción: Ver Carga Ale. No. y ver o editar todas las cargas aleatorias de la estructura de forma secuencial, con la opción Ver Todas las Cargas.

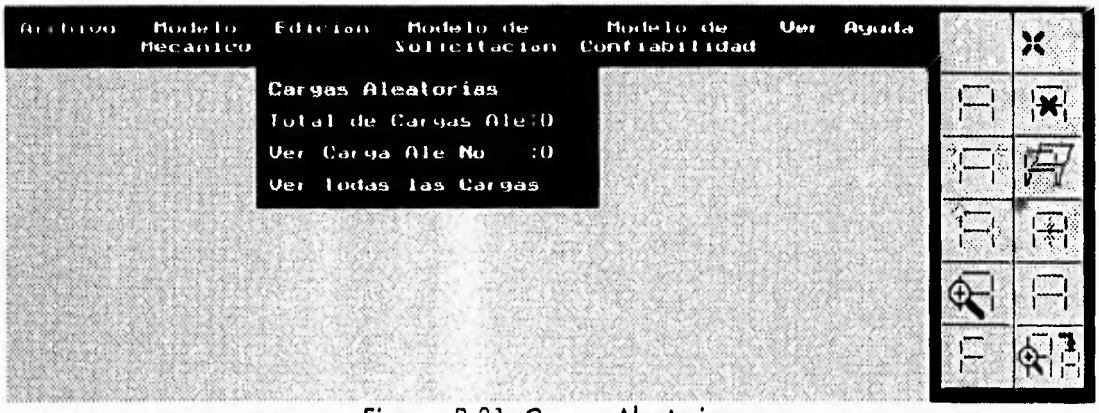


Figura B.31 Cargas Aleatorias.

En Ver Carga Ale. No. se especifica la carga aleatoria que se quiere editar, se oprime el botón izquierdo en el nodo donde desea aplicar la carga aleatoria y aparecerá un submenú donde se debe indicar la Media y el coeficiente de variación C.V. (figura B.32). En este submenú también aparece como información el número de nodo: Nodo No. y el número de carga aleatoria: C.A. No.

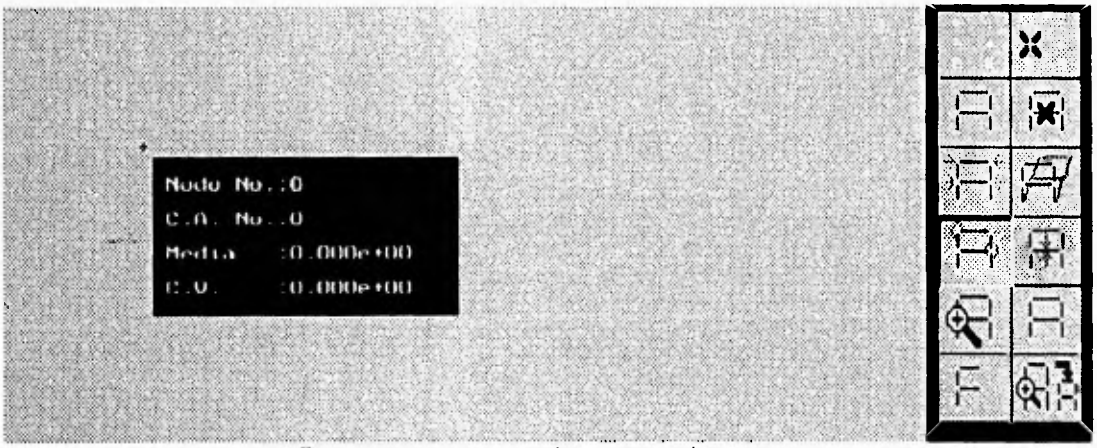


Figura B.32 Edición de Cargas Aleatorias.

Configuración

En Configuración aparece un submenú como el mostrado en la figura B.33, Las tres primeras opciones permiten hacer acercamientos, En Zoom Ventana se puede crear una ventana de

acercamiento. En Zoom Previo se dibuja la estructura de acuerdo a su estado anterior al de establecer un zoom de ventana. En Zoom Total se dibuja la figura con los límites máximos posibles en dirección "x" de -100 a 100 y en dirección "y" de -100 a 100. Abrir y Guardar Config cargan y guardan el archivo de configuración del programa respectivamente.

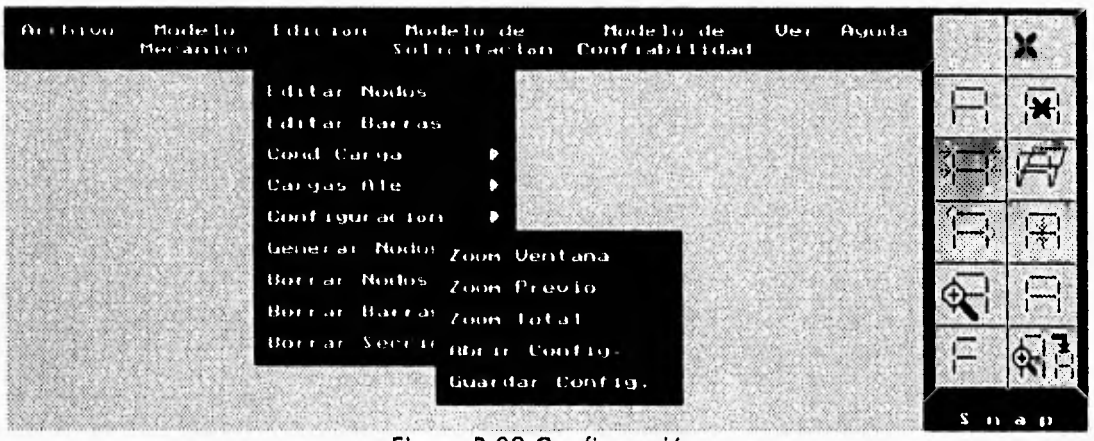
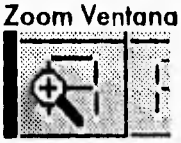


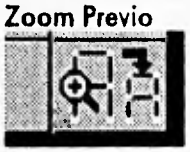
Figura B.33 Configuración.

Zoom Ventana



Para hacer un acercamiento de ventana oprima el botón izquierdo del ratón donde desee establecer el primer límite de la ventana y mantenga oprimido el botón izquierdo moviendo el ratón hasta establecer el otro límite de la ventana, Figura B.34, automáticamente se dibuja la estructura con los nuevos límites, figura B.35.

Zoom Previo



Con esta opción podemos regresar la imagen al estado anterior de establecer un acercamiento de ventana.

Generar Nodos

Para evitar el trabajo tedioso y lento de crear nodo por nodo, en el caso de ser demasiados nodos, con el submenú Generar Nodos (figura B.36) se pueden crear automáticamente varios nodos. La Dirección puede ser hacia arriba o hacia abajo sobre el eje Y, el Incremento o Decremento en Y, el total de #nodos a dibujar, #columnas a generar (pueden ser de 1 a 4 columnas) las Coordenadas del primer nodo de cada columna, y Generar es la opción generadora de nodos una vez que se han introducido todos los datos anteriores.

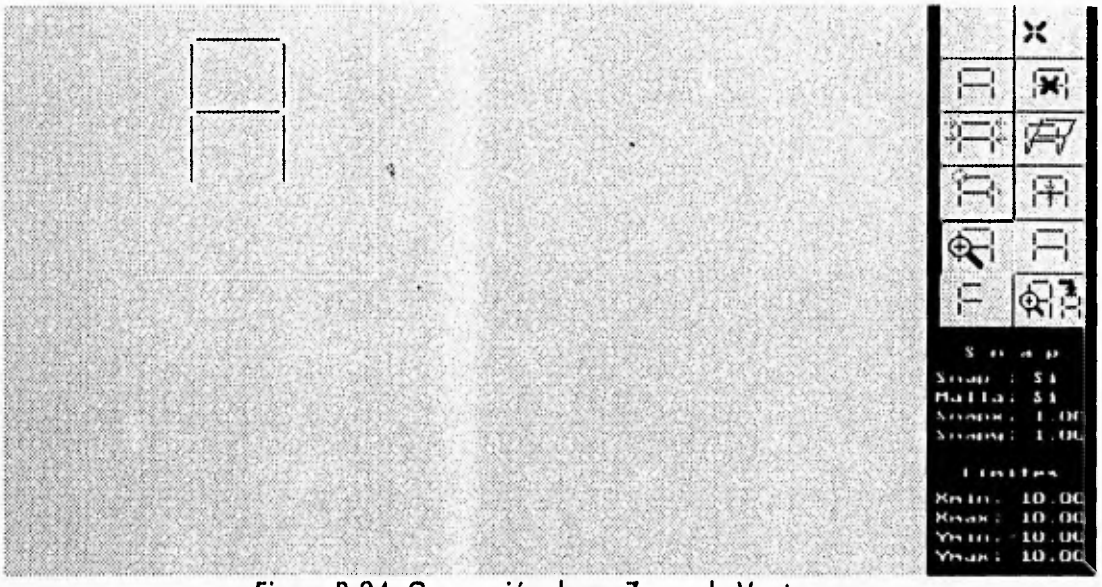


Figura B.34 Generación de un Zoom de Ventana.

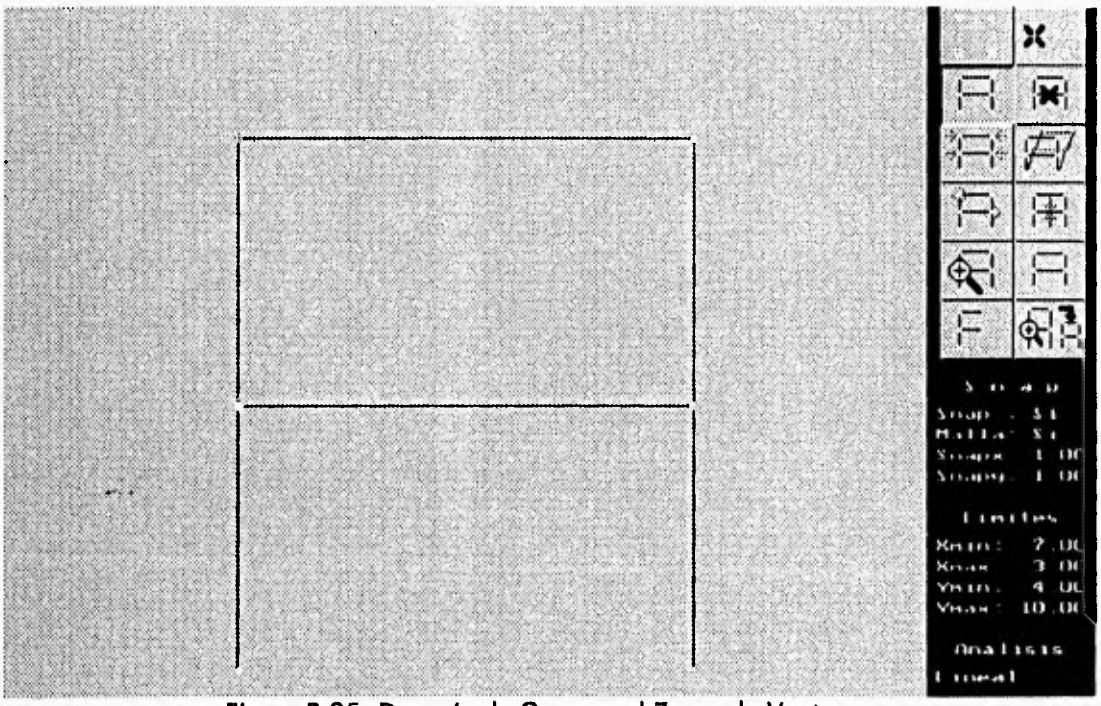


Figura B.35 Después de Generar el Zoom de Ventana.

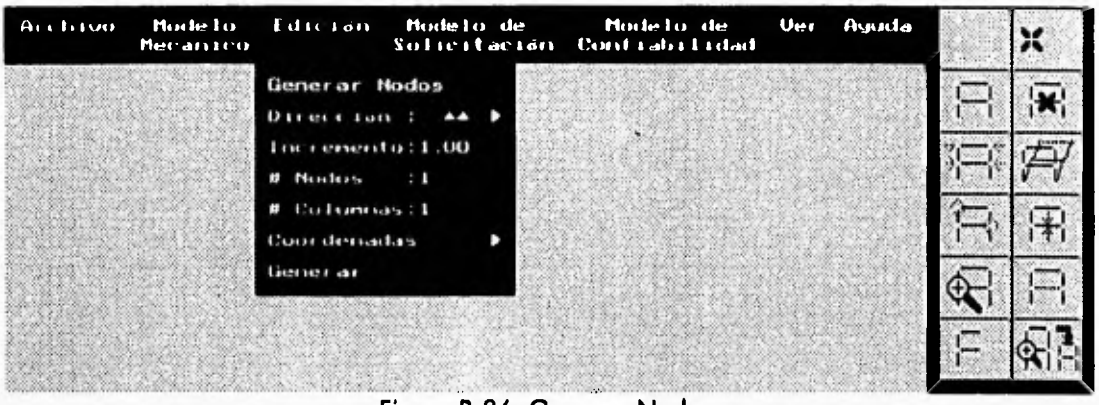
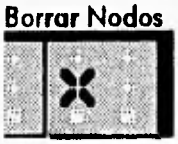


Figura B.36 Generar Nodos.

Borrar Nodos



En Borrar Nodos podemos borrar un nodo, cambiar sus coordenadas o cambiar su grado de libertad, se indica el nodo que se desea modificar oprimiendo el botón izquierdo del ratón en dicho nodo y aparece el submenú de la figura B.37. Si desea salir de Borrar Nodos oprima el botón derecho del ratón. Para poder borrar un nodo no debe haber barras que se conecten a dicho nodo.

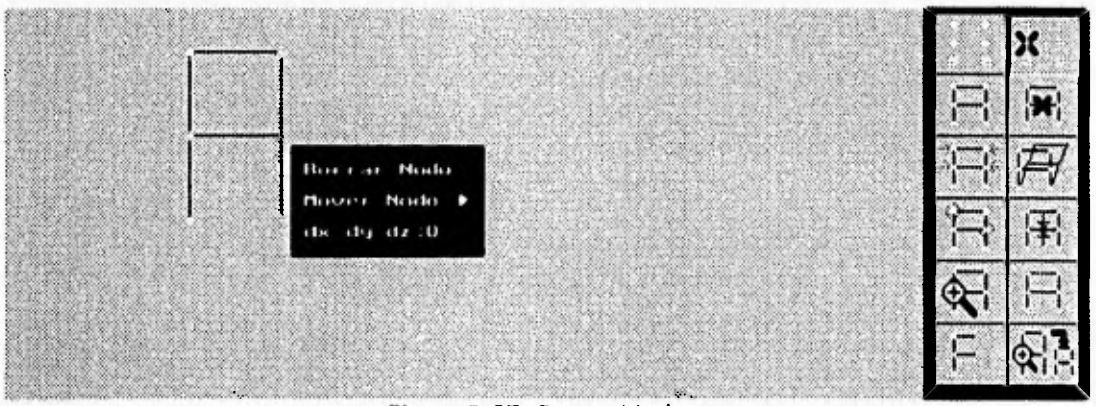


Figura B.37 Borrar Nodos.

Borrar Barras

En la opción de Borrar Barras, sólo indique la barra que desea eliminar oprimiendo el botón izquierdo del ratón en los nodos que conectan dicha barra y la barra desaparece.

Borrar Secciones

Elimina las plastificaciones de las barras, para análisis elastoplástico y elastoplástico crítico.

B.5 MÓDULO DE SOLICITACIÓN

El módulo de Solicitación consta de dos submenús, como se muestra en la figura B.38.

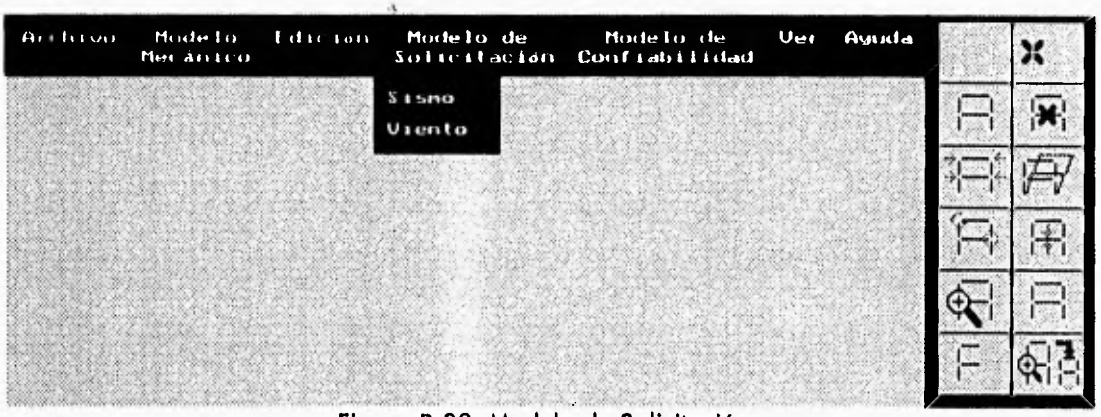


Figura B.38 Modelo de Solicitación.

El submenú Sismo (figura B.39), permite calcular las fuerzas sísmicas estáticas que actúan en la estructura de acuerdo con el reglamento del Distrito Federal RDF-87. En este submenú aparece el número de niveles de los pisos de la estructura: # Niveles y es necesario dar como datos, el tipo de suelo en que está cimentada la estructura: Tipo Terreno y el factor de comportamiento sísmico: **Q**.

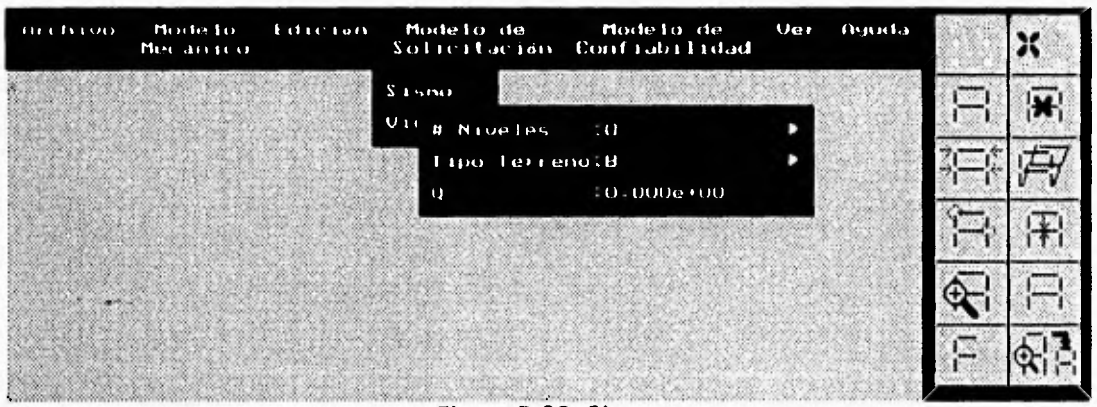


Figura B.39 Sismo

Al oprimir la opción **# niveles** con el botón izquierdo del ratón, aparece un submenú donde se solicita la altura y peso de cada nivel, figura B.40. Para cambiar de nivel se oprime el botón izquierdo del ratón.

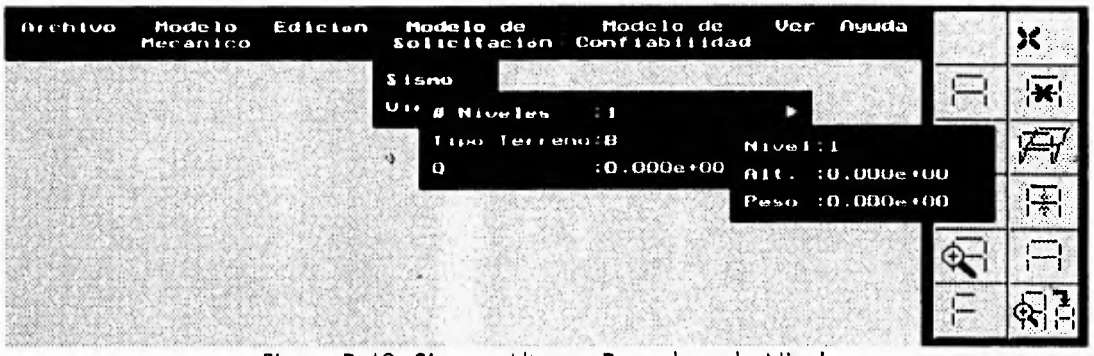


Figura B.40 Sismo, Altura y Peso de cada Nivel.

Tipo Terreno

El tipo de terreno a especificar puede ser Firme, de Transición o Compresible, como se muestra en la figura B.41.

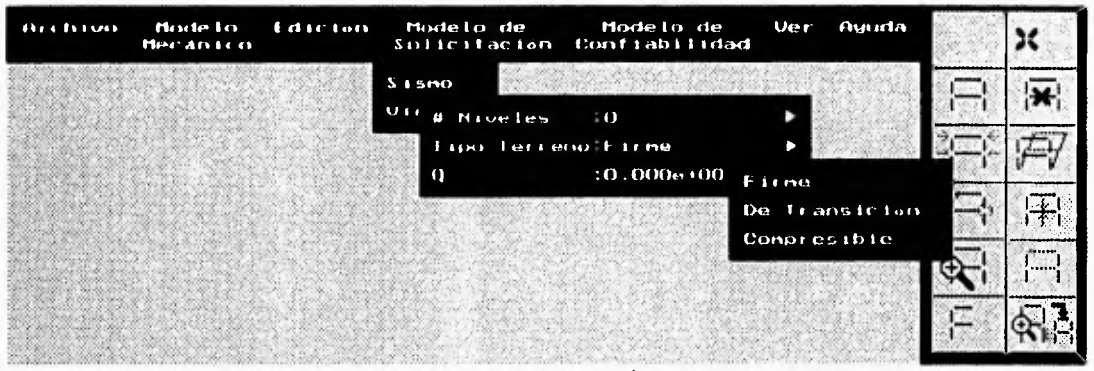


Figura B.41 Sismo, Tipo de Terreno.

El submenú Viento (figura B.42), permite calcular las fuerzas de viento estáticas que actúan en una estructura tipo edificio, de acuerdo con el reglamento del Distrito Federal RDF-87. En este submenú aparece el número de niveles: #Niv., el número de marcos: #Mar., el ancho total de la estructura: Anc T., y es necesario dar como dato el grupo de estructura: G. Est., puede ser A o B y Z. exp., zona por exposición y puede ser A, B o C.

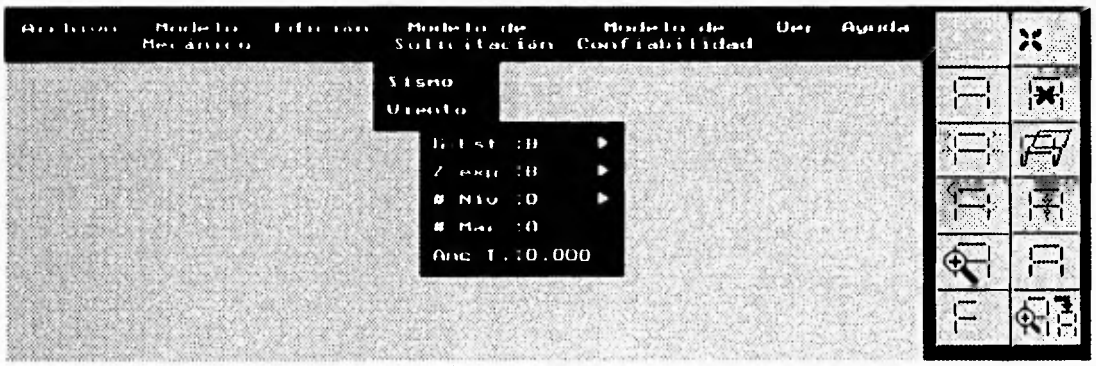


Figura B.42 Viento.

B.9 MÓDULO CONFIABILIDAD

Este módulo consta de nueve opciones como se puede observar en la figura B.43. En este módulo se dan los datos necesarios para realizar el análisis de confiabilidad estructural de la siguiente forma:

En **G. Mec. S.M.E.** se guarda el mecanismo actual, considerando la sección más esforzada de la estructura, para su posterior análisis. En **G. Mec. Secc.** se guarda el mecanismo actual, considerando la sección que se le asigna en esta opción, aquí hay que recordar que no se puede asignar una sección que esté plastificada, para el mecanismo que se quiere guardar. **P. Ri Rj** , aquí se especifica el coeficiente de correlación entre los extremos de un mismo elemento. **P Fi Fj**, aquí se especifica el coeficiente de correlación entre las fuerzas existentes en la estructura. En **# Mecanismos** se visualiza el número de mecanismos que están guardados. Hay que hacer notar que el máximo número de mecanismos que se pueden guardar es de 10.

En **Cálculo Pf**, se hace el cálculo de probabilidad de falla total del sistema estructural, de acuerdo a los mecanismos que estén almacenados. Para poder hacer este cálculo es necesario que existan por lo menos dos mecanismos guardados. **Ver Todos los Mecs**, permite visualizar cada uno de los mecanismos que están almacenados. Para cambiar de mecanismo se oprime el botón derecho del ratón. **Ver Mec. No.** visualiza sólo el mecanismo que se indica en esta opción. Oprimir el botón derecho del ratón si se desea salir de la visualización del mecanismo señalado. **Eliminar Mecs.** borra todos los mecanismos almacenados si se selecciona Si.

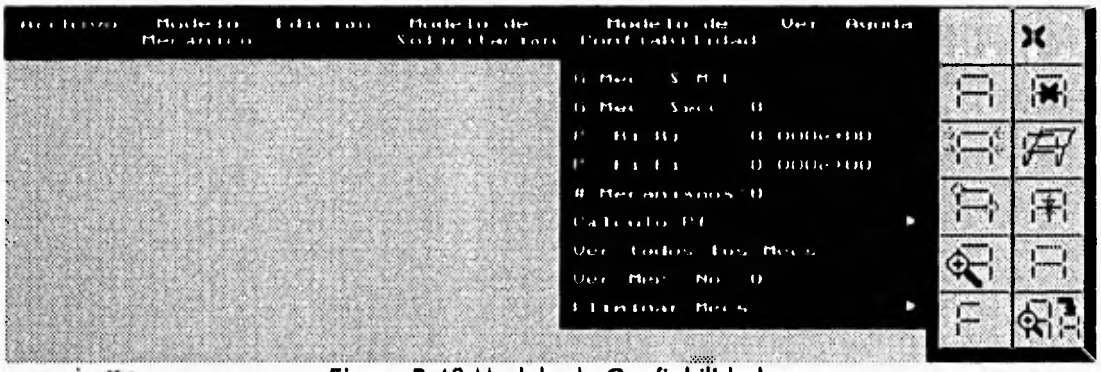


Figura B.43 Modelo de Confiabilidad.

B.8 MÓDULO VER

Una vez que se ha ejecutado el análisis mecánico se pueden ver gráficamente los resultados del análisis gracias al módulo Ver, los datos a visualizar son los índices betas, fuerzas internas y desplazamientos, como se muestra en la figura B.44. La última opción de este módulo permite redibujar sólo la figura, sin desplazamientos, fuerzas internas o índices betas.

Todas las opciones anteriores pueden ser también activadas con los iconos de la parte derecha del menú principal.

Figura B.44 Ver.

Índices Betas

Este submenú permite visualizar los índices de confiabilidad betas de las secciones extremas más esforzadas, para los análisis lineal crítico y elastoplástico crítico. Las barras donde existan betas diferentes de cero, se verán de color azul. Señale la barra, oprimiendo el botón izquierdo del ratón en los 2 nodos que componen la barra, de esta manera aparecerá el submenú de la figura B.45. Para salir de Índices Betas oprima botón derecho del ratón.

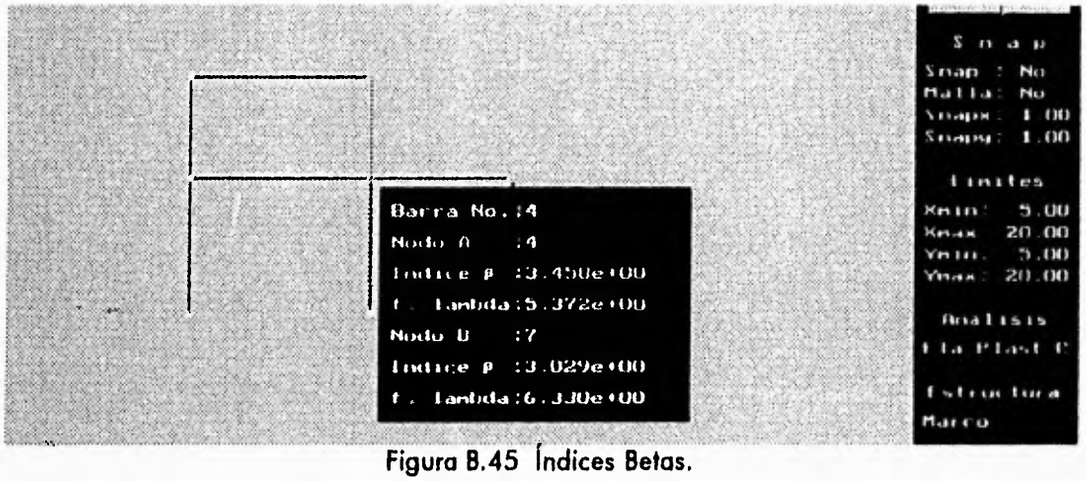
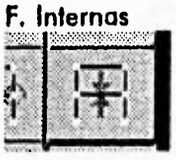


Figura B.45 Índices Betas.

F. Internas



El submenú Fuerzas Internas pide establecer una condición de carga específica donde se solicita visualizar las fuerzas internas, o tiene la opción de ver las fuerzas internas de todas las condiciones de carga, en forma secuencial, como se muestra en la figura B.46.

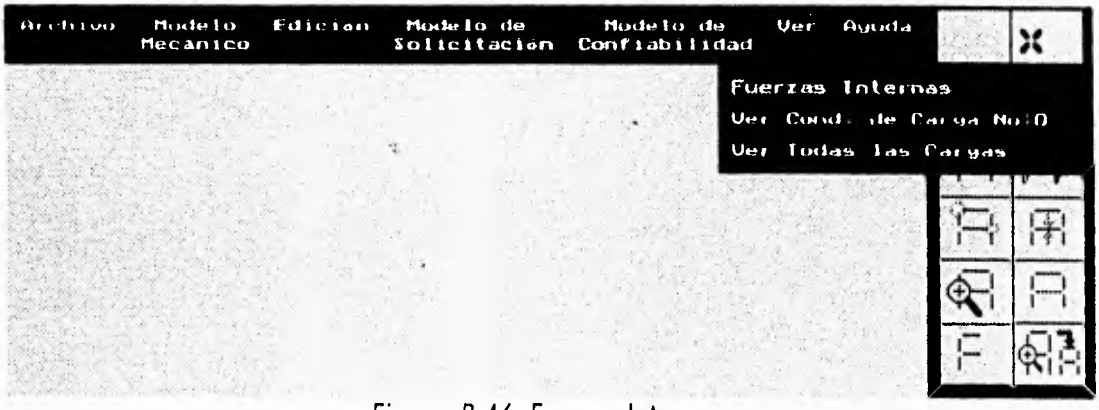


Figura B.46 Fuerzas Internas.

Una vez establecida la condición de carga, se permite visualizar las fuerzas internas o elementos mecánicos de los extremos de los elementos: F. normal, la fuerza axial, F. cort, la fuerza cortante y M. flex, el momento flexionante (figura B.47). Al igual que en índices betas es necesario elegir la barra de interés para mostrar los datos requeridos. Para salir de Fuerzas Internas oprima el botón derecho de ratón.

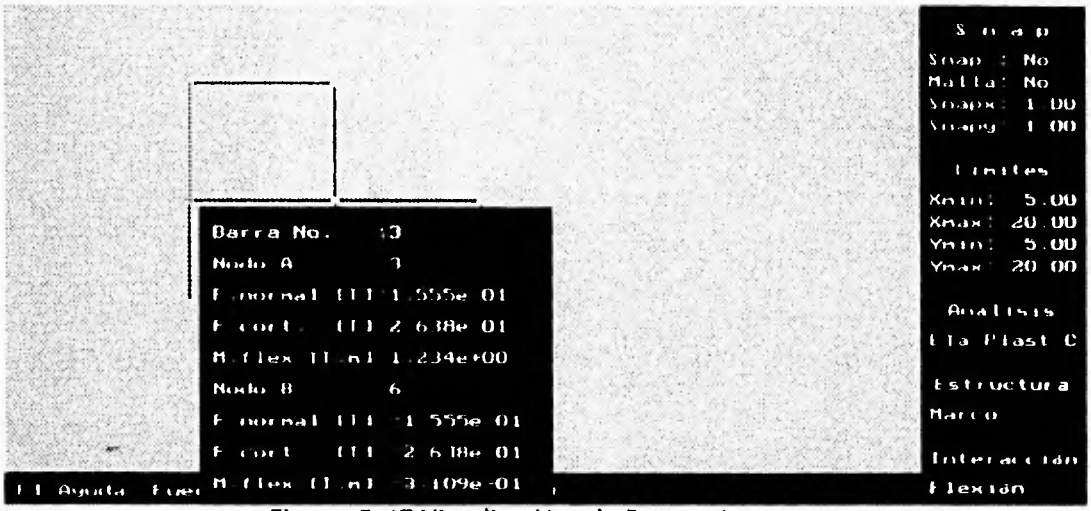
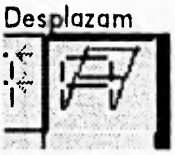


Figura B.47 Visualización de Fuerzas internas.

Desplazam



En este submenú (figura B.48), se establece una condición de carga específica en donde se desean ver los desplazamientos, o se puede indicar que se quieren ver todas las condiciones de carga, así mismo se soliciton un factor de deformación y un factor de desplazamiento.

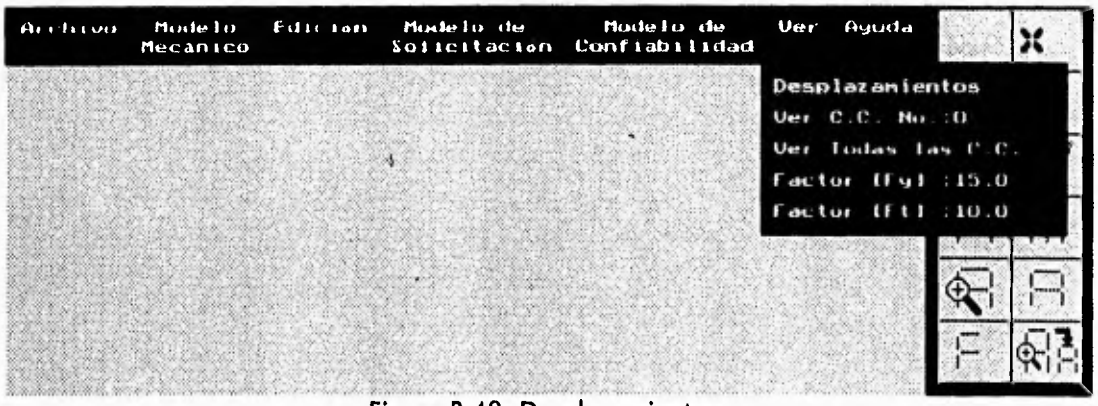


Figura B.48 Desplazamientos.

Una vez establecida la condición de carga, se permite visualizar la deformada de la estructura y los desplazamientos correspondientes a cada nodo en X, en Y, y el giro en Z, como se muestra en la figura B.49. El nodo se selecciona oprimiendo el botón izquierdo del ratón. Si se desea salir de Desplazamientos, se oprime el botón derecho de ratón.

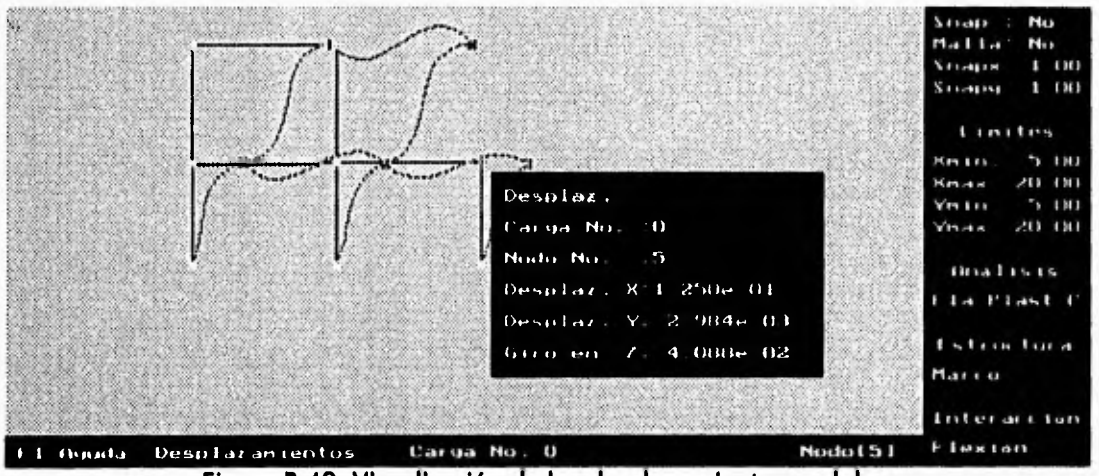


Figura B.49 Visualización de los desplazamientos nodales.

B.9 MÓDULO DE AYUDA

La ayuda puede ser activada en cualquier parte del programa, con la tecla F1, oprimiéndose se obtendrá la ayuda correspondiente de la sección del programa en la que se encuentre.

También se puede obtener ayuda en el Módulo de Ayuda el cual tiene 3 opciones como se muestra en la figura B.50.

Índice

Esta opción permite ver un índice de todos los módulos que componen el sistema, como se muestra en la figura B.51, si se desea tener ayuda de un módulo específico, posicione el cursor del ratón en el tópico de interés y oprima el botón izquierdo del ratón, de esta forma aparecerá la información. Si no quiere ver ningún tópico oprima el botón derecho del ratón para regresar al menú principal.

Buscar Tópico

En Buscar tópico, se escribe una palabra de la que se necesite ayuda, y si existe en el archivo de ayuda, se visualiza su información en una ventana; si no existe el tópico se visualiza un mensaje de error, indicando que no se ha encontrado la cadena correspondiente.

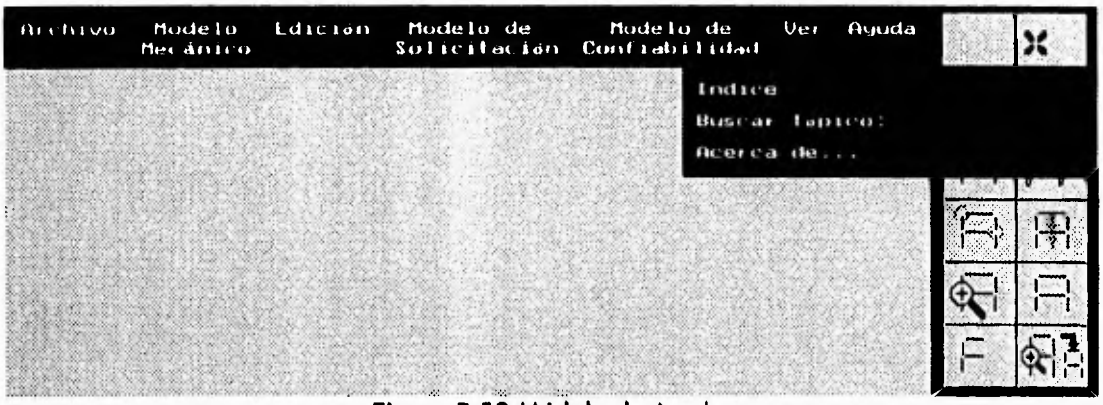


Figura B.50 Módulo de Ayuda.

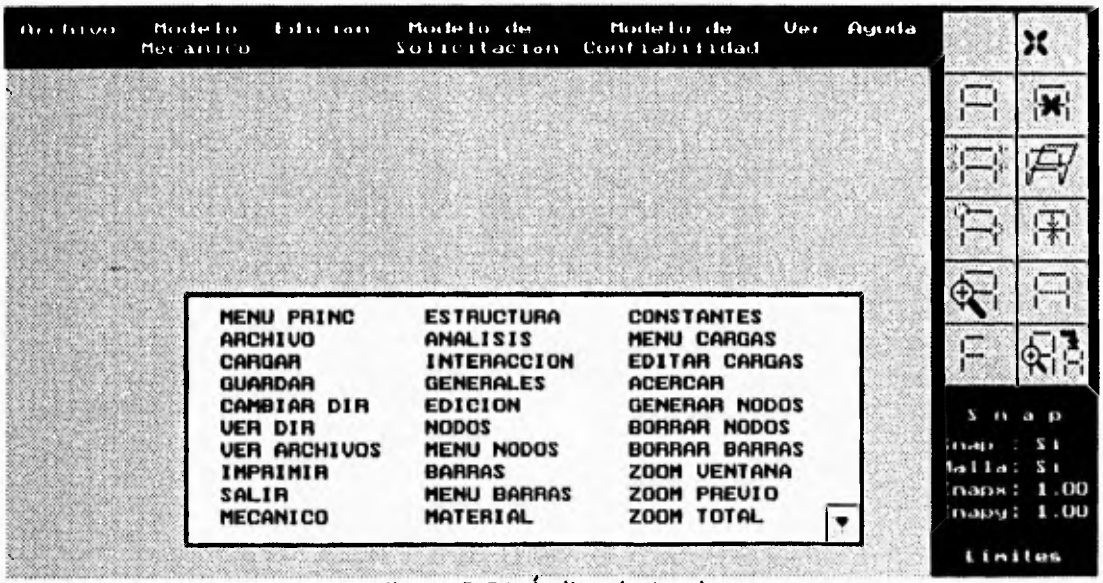


Figura B.51 Índice de Ayuda.

B.10 PARÁMETROS DE CONFIGURACIÓN

En la parte derecha inferior de la pantalla se muestra un menú donde se pueden activar: Snap, Límites y se puede seleccionar: Tipo de Análisis, Estructura e Interacción. Figura B.52.

Figura B.52.

S n a p

Si se desea cambiar las características del snap posicione el cursor del ratón en S n a p y oprima el botón izquierdo, aparecerá un submenú, como el mostrado a la figura B.53, donde se puede indicar si se quiere o no el Snap, si se quiere o no la Malla, para utilizar el mallador debe estar activado el snap; modificar el snap en coordenada X: Snapx y el snap en coordenada Y: Snapy. El rango permitido en estas dos últimas opciones es de 0.01 mts. a 10.00 mts.

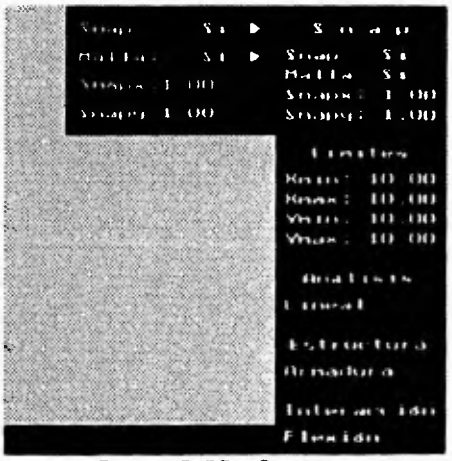


Figura B.53. S n a p.

Límites

Si se desean cambiar los límites del plano de edición, posicione el cursor del ratón en **Límites** y se oprime el botón izquierdo y en seguida aparecerá un submenú como el de la figura B.54, donde se pueden definir los límites: X y Y. Los límites mínimos deben de ser mayores a -100.00 mts. y menores a los valores máximos menos 0.01 mts. Y los límites X y Y máximos deben de ser menores a 100.00 mts. y mayores a los valores mínimos más 0.01 mts.

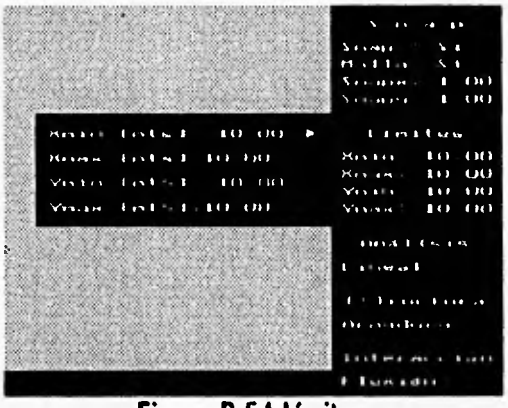


Figura B.54 Límites.

Las tres opciones finales: **Análisis, Estructura** e Interacción son las mismas opciones que ya se mostraron en el **Módulo Mecánico.**

BREVIARIO DE INGENIERÍA CIVIL

Armadura - es una serie de elementos (barras) rectos conectados entre sí mediante nodos de tal manera que forman una estructura que actúa como una viga de gran tamaño. Una armadura tiene formas triangulares en un solo plano de tal forma que las cargas externas sólo se aplican en los nodos, por lo que teóricamente, a causa de su geometría, los elementos de una armadura sólo están sometidos a fuerzas de tensión y comprensión exclusivamente.

Articulación plástica - cuando una estructura se le somete a cargas y ésta se deforma a causa de los momentos de flexión, existe un período en el que ya no puede absorber más fuerza y se plastifica, por lo tanto se dice que gira libremente y analíticamente esta situación o fenómeno se le sustituye en el punto plastificado por una articulación llamada articulación plástica.

Barras o elementos - son los componentes de que consta una estructura y que soportan las cargas o fuerzas externas aplicadas a la misma. Las barras pueden adquirir en determinado momento un nombre específica dependiendo del tipo de estructura que se está analizando por ejemplo véase: Vigas y Columnas.

Cargas - son fuerzas externas aplicadas al sistema estructural para simular un comportamiento dado y analizar los efectos que éstas causan. Existen diversos tipos de cargas por ejemplo véase: Cargas distribuidas, Cargas concentradas, etc.

Cargas concentradas - son las cargas que se concentran en un punto específico de la estructura, que por lo general se aplican en los nodos extremos a las barras para cuestiones de estudio.

Cargas distribuidas - son las cargas aplicadas a la estructura, que se distribuyen a lo largo de toda la barra en la que están aplicadas y que para cuestiones de estudio se consideran constantes sobre toda la superficie en que actúan.

Cargas muertas - son cargas de magnitud constante aplicadas a la estructura y que permanecen en un mismo lugar, constan del peso propio de la estructura y de otras cargas que están permanentemente unidas a ella. En un edificio con estructura de acero algunas de las cargas muertas son la estructura en sí, las paredes, los pisos, las tuberías y los accesorios que ésta posee.

Cargas por sismo - son fuerzas accidentales que producen máximos esfuerzos o deformaciones en un elemento de la estructura durante un sismo o fuerzas equivalentes aplicadas.

Cargas por viento - son las fuerzas máximas que el viento puede ejercer a una estructura en un intervalo de tiempo.

Cargas vivas - son cargas aplicadas a la estructura que pueden cambiar en magnitud y posición a través del tiempo, ejemplos de estas cargas serían: personas, camiones, lluvia, etc.

Confiabilidad estructural - al estudio de las estructuras que trata de la seguridad en condiciones de incertidumbre se le llama confiabilidad estructural y conlleva al empleo de la teoría de las probabilidades.

Elemento elástico - un elemento elástico es aquel que dentro de ciertos límites recupera sus dimensiones originales al cesar la fuerza que le produjo deformación; por ejemplo una varilla al aplicársele una fuerza se flexiona y al quitar la fuerza regresa a su forma original sin sufrir deformación permanente.

Elemento plástico - es aquel que al cesar la fuerza que produjo deformación no recupera su forma ni sus dimensiones originales. Lo anterior ocasiona una deformación permanente en dicho elemento. Un material prácticamente plástico sería la plastilina donde se aprecia que al cesar la fuerza que la deforma quedá así permanentemente.

Elemento elasto-plástico - los materiales usados en la construcción no son perfectamente elásticos ni perfectamente plásticos, es decir, que su comportamiento puede ser elasto-plástico, según sean las fuerzas que en ellos obren.

Estado límite - el estado límite de una estructura es cualquier etapa de su comportamiento a partir del cual presenta una respuesta inaceptable. Aquellos estados límites relacionados con la seguridad del sistema estructural se denominan estados límite de falla y corresponden a situaciones en las que la estructura sufre una falla total o parcial.

Estructura isostática - es aquella estructura que puede ser puesta en equilibrio por medio de las ecuaciones que proporciona la estática.

Estructura hiperestática - es aquella estructura que para encontrar su estado de equilibrio se tiene que hacer uso de ecuaciones adicionales a las que proporciona la estática.

Estructura estáticamente determinada - véase Estructura isostática.

Estructura estáticamente indeterminada - véase Estructura hiperestática.

Flexibilidad - la flexibilidad es lo inverso de la rigidez. Es decir, la flexibilidad da una medida de las cantidades de desplazamiento que estén asociadas con un conjunto dado de fuerzas. En su forma más simple, la flexibilidad es el desplazamiento que se produce en un punto por la aplicación de una carga unitaria en ese punto.

Fuerzas axiales - son fuerzas longitudinales que causan sólo tensión o comprensión en los elementos (barras) de la estructura. Las fuerzas axiales de tensión tienden a agrandar los elementos de la estructura. Las fuerzas axiales de compresión tienden a acortar los elementos de la estructura.

Fuerzas cortantes - son fuerzas perpendiculares a los elementos de la estructura y aplicadas en ellos.

Fuerzas externas - son las cargas o fuerzas aplicadas a la estructura. Véase Cargas.

Fuerzas de flexión - son fuerzas que se aplican en los elementos de la estructura y que tienden a flexionarlos o curvearlos, ejemplos de estas fuerzas son momentos aplicados en los extremos de las barras de la estructura, lo que provoca giros en estos elementos.

Fuerzas internas - cuando a una estructura se le aplican cargas o fuerzas externas, éstas se distribuyen de alguna forma sobre todos los elementos de la estructura, por lo tanto a la resistencia que ofrece cada una de estas barras se les llama fuerzas internas del elemento.

Fuerzas de tensión - véase Fuerzas axiales.

Fuerzas de compresión - véase Fuerzas axiales.

Función de estado - es una función que representa el comportamiento del sistema estructural, que distingue tres estados: estado de seguridad si la función es mayor que cero, estado de falla si la función es menor que cero y estado límite si la función es igual a cero.

Ley de Hooke - el enunciado de la ley de Hooke nos indica lo siguiente: dentro de ciertos límites, las deformaciones son proporcionales a los esfuerzos. Ésto quiere decir que si a un material se le somete a un esfuerzo determinado, éste sufrirá deformaciones que serán proporcionales a las fuerzas aplicadas dentro del límite elástico del material.

Limite elástico o límite proporcional - cuando las deformaciones ya no son proporcionales a los esfuerzos aplicados a un elemento como lo dice la ley de Hooke, se dice que se ha traspasado el límite elástico del material. En otras palabras es el esfuerzo más grande que se puede aplicar a un material sin que quede una deformación permanente, después de quitar la carga.

Límite de fluencia - es el punto a partir del cual el material sometido a un esfuerzo comienza a sufrir deformaciones sin aumentar la carga, es decir comienza a fluir.

Marco - estructura de forma rectangular que por el tipo de conexión que se tiene hace que los elementos estén sometidos a fuerzas de flexión, fuerzas cortantes y fuerzas axiales.

Matriz de rigidez - es un conjunto de información que representa las propiedades fundamentales de las características de un elemento de la estructura y que para efectos del análisis estructural se agrupan en matrices. Véase Rigidez.

Matriz de rigidez reducida - es un tipo especial de matriz de rigidez que surge en el análisis elastoplástico y que como característica principal es que cambian los grados de libertad de las secciones extremas de los elementos y producen una disminución de magnitud en los componentes de la matriz de rigidez.

Mecanismo de falla - cuando una estructura se deforma y se forman articulaciones plásticas, existen varios estados en los que las combinaciones de estas articulaciones producen inestabilidad y producen la falla en la estructura, a estos estados de inestabilidad se les llama mecanismos de falla.

Módulo de elasticidad - la palabra módulo indica coeficiente y en el caso de los materiales interesa conocer que tan elásticos pueden ser. A mayor elasticidad será mayor el módulo elástico o de elasticidad. Matemáticamente representa la relación esfuerzo-deformación y las

unidades son tonelada por metro cuadrado o kilogramo por centímetro cuadrado. En otras palabras es el esfuerzo que ocasiona una deformación unitaria y se designa con la letra E.

Nodo o Nudo- es el punto de unión entre 2 o más barras. Los nodos tienen diferentes tipos de unión que limitan los movimientos de éstos en la estructùra, véase también **Tipos de apoyo.**

Relación o coeficiente de POISSON - cuando se somete a un material a cargas axiales de tensión o compresión, no sólo se deforma en la dirección de las cargas sino también normal a ellas. Al estar bajo tensión, se reduce la sección transversal de un elemento y al estar bajo compresión se aumenta. La relación o coeficiente entre la deformación lateral unitaria y la deformada longitudinal unitaria se llama coeficiente de Poisson.

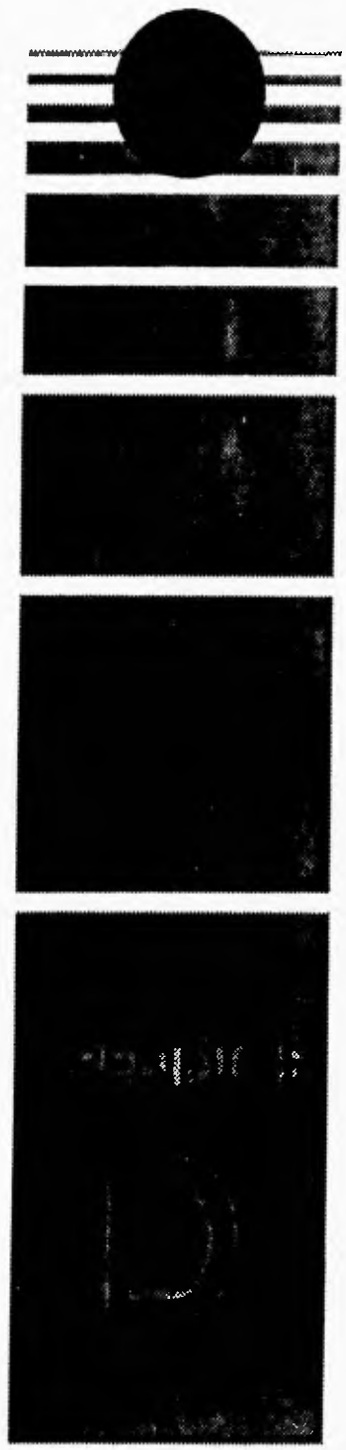
Rigidez - es la propiedad de los materiales que les permite soportar un gran esfuerzo, produciéndose deformaciones muy pequeñas. En relación con otros parámetros la rigidez da una medida de las fuerzas que están asociadas con un conjunto dado de desplazamientos. En su forma más simple, la rigidez es la fuerza que debe aplicarse en algún punto para producir un desplazamiento unitario en ese punto.

Sección crítica o más esforzada - es aquella sección de un elemento que por las condiciones de carga a las que se encuentra sometida está próxima a fallar.

Tipos de apoyo - el apoyo es un elemento que puede restringir todos o algunos de los desplazamientos de los elementos que soporta y es aplicado a los nodos de la estructura, existen tres tipos de apoyo: **Libre** - en este tipo de soporte el movimiento de los nodos no está restringido en ninguna dirección, de tal forma que al aplicar cargas los nodos con soporte libre podrán desplazarse en las dos direcciones (X y Y), así como sufrir desplazamientos angulares. **Articulado** - en este soporte el movimiento en el plano está restringido y sólo pueden aparecer desplazamientos angulares o de giro. **Empotrado** - este tipo de soporte no permite ningún tipo de desplazamiento ni giro.

Viga - son aquellos elementos estructurales generalmente usados en posición horizontal y sujetos a cargas de gravedad perpendiculares al eje longitudinal del elemento, o verticales en las que actúan fuerzas axiales y cortantes principalmente.

BREVIARIO DE INGENIERÍA EN COMPUTACIÓN



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

Clase derivada - [illegible] atributos de una o varias clases base o superclases. Véase Superclase.

Clase genérica - [illegible]

Constructor - es una operación usada en el lenguaje C++ que [illegible] un objeto y es usado para [illegible]

Destructor - es una operación usada en el lenguaje C++ que [illegible] todos sus componentes y limpiando la memoria.

Diseño - es el proceso de convertir los requerimientos de un sistema a una implementación que cumpla con las características deseadas.

DOO (Diseño orientado a objetos) - es un método que a partir del análisis orientado a objetos previamente realizado se profundiza y se particularizan detalles de tal forma que el paso a la implementación sea lo más automático y transparente posible [illegible]

Abstracción - es la habilidad de tomar algo como un todo, sin tener en cuenta los detalles. El propósito de la abstracción es la de entender algo sin tener una comprensión completa de la estructura interna. La programación orientada a objetos (POO) se basa en la abstracción tanto de datos como funcional.

Abstracción de datos - es la posibilidad de ocultar detalles específicos de la implementación de un tipo de datos. El lenguaje C++ permite definir tipos de datos por medio de clases, abstrayendo su implementación.

Abstracción funcional - permite observar una función, procedimiento o método, sin tener que conocer los algoritmos que usa. Si funciona correctamente no se necesita saber como está implementada, sino como funciona.

Análisis - es el proceso de extracción y codificación de los requerimientos de un sistema y el establecimiento de un modelo del problema.

AOO (Análisis orientado a objetos) - es un método que examina los requerimientos de un sistema desde una perspectiva de clases y objetos, así como la forma en que interactúan entre ellos para resolver un problema dado.

Clase - es la definición de un tipo de objetos que tienen características comunes. Una clase es una plantilla para crear nuevos objetos. Una clase puede contener tanto atributos como objetos. Existen importantes diferencias entre los términos **Clase** y **Objeto**. Mientras que un objeto es una entidad concreta que existe en el tiempo y el espacio, una clase representa sólo una abstracción del objeto. Por ejemplo podemos hablar de la clase mamíferos, la cual sólo representa las características generales de todos los mamíferos. Para identificar un mamífero específico de la clase, debemos hablar de este mamífero o de aquel mamífero.

Clase derivada - en nuestro contexto una clase derivada es aquella clase que hereda métodos y atributos de una o varias clases base o superclases. Véase **Superclase**.

Clase genérica - es aquella clase en que sus funciones o métodos son usados por todas las clases que conforman el sistema; generalmente estos métodos o funciones son públicos para tal efecto. Véase **Clase**, **Subclase**.

Constructor - es una operación usada en el lenguaje C++ que es llamada automáticamente al crearse un objeto y es usada para inicializar un objeto y/o asignarle espacio en memoria.

Destructor - es una operación usada en el lenguaje C++ que destruye un objeto eliminando todos sus componentes y limpiando la memoria.

Diseño - es el proceso de convertir los requerimientos de un sistema a una implementación que cumpla con las características deseadas.

DOO (Diseño orientado a objetos) - es un método que a partir del análisis orientado a objetos previamente realizado se profundiza y se particularizan detalles de tal forma que el paso a la implementación sea lo más automático y transparente posible, además se adecúa una notación

para representar los modelos lógico y físico del sistema bajo diseño. En general el DOO es una especificación mayor y más detallada del análisis orientado a objetos (AOO).

Encapsulación - la habilidad de poner todos los datos y funcionalidades en una única estructura de tal forma que esté aislada del resto del sistema. Las clases permiten al programador la libertad de encapsular variables y métodos en una única estructura.

Evento - es el cambio en el estado de un objeto mediante la realización de alguna operación.

Función - relación operacional entre varias entidades. Se suelen representar como una correspondencia entre sus parámetros y un resultado.

Herencia - es la capacidad de pasar propiedades de una clase a otra. La clase derivada tiene todas las propiedades de la clase base y se le pueden añadir nuevos atributos o métodos a la clase derivada.

Herencia múltiple - es la capacidad de heredar de varias clases los atributos y métodos que éstas poseen.

Ingeniería de software - es una disciplina que integra métodos, herramientas y procedimientos para el desarrollo de sistemas en una computadora.

Instancia - es la acción de crear un objeto a partir de una clase ya sea en tiempo de compilación o en tiempo de ejecución. Un objeto es una instancia de una clase.

Instanciación - es la acción de instanciar una clase. Dicho en otras palabras una instanciación es la creación de un objeto dentro del programa.

Jerarquía de clases - existe una jerarquía de clases cuando unas clases heredan de otras. Se puede pensar que una jerarquía de clases es como un árbol de familia, en el que las clases derivadas son los hijos de las clases bases (padres).

Lenguaje orientado a objetos - es un lenguaje que tiene la capacidad de poder definir clases, objetos y además permite la herencia y la modularidad.

Lenguaje procedural - es un lenguaje en el cual se le indica a la computadora como llevar a cabo una tarea en particular. En estos lenguajes el programador debe de saber como llevar a cabo la tarea que se le quiere encomendar a la computadora para podérsela transmitir al programa. Ejemplos de estos lenguajes son: PASCAL, C, C++.

Lenguaje no procedural - es un lenguaje en el cual se le indica que debe hacer y no como. En estos lenguajes se les dice que hacer y sólo ellos saben como llevar a cabo la tarea que se les ha encomendado. Ejemplos de estos lenguajes son: SQL, PROLOG.

Mallador o Grid - es un tipo de malla que si está activada aparece en forma de puntos simétricos en la pantalla y que sirve de ayuda para la edición de la estructura debido a que actúa como guía para los desplazamientos del ratón. Véase Snap.

Mensaje - es el mecanismo por medio del cual los objetos se comunican. Los métodos especifican que mensajes se deben de mandar entre los diversos objetos existentes. En otras palabras, un método es la especificación de los pasos mediante los cuales se realiza una operación. En los lenguajes orientados a objetos es simplemente el código que se ejecuta.

Método - es el proceso mediante el cual se define una operación sobre un objeto. Cuando un mensaje es enviado a un objeto, el método definido en ese objeto es ejecutado. Específicamente en C++ los métodos no son otra cosa que funciones miembros de las clases que son llamadas a su vez por otras funciones.

Modelo - abstracción de algo con el propósito de entenderlo y estudiarlo antes de construirlo.

Módulo - es un subconjunto de un sistema que desempeña una función específica y está formado por un grupo de clases y sus relaciones.

Objeto - es una entidad que tiene un estado y un conjunto definido de operaciones que actúan y modifican su comportamiento. El estado es representado por un conjunto de atributos del objeto. Las operaciones asociadas con el objeto dan servicio a otros objetos que piden estos servicios cuando se necesita realizar alguna operación.

Paradigma - es la forma de pensar o actuar sobre un aspecto concreto.

Polimorfismo - los objetos actúan en respuesta a los mensajes que reciben. El mismo mensaje puede originar acciones completamente diferentes al ser recibido por diferentes objetos, este fenómeno se conoce como polimorfismo.

Programación estructurada - metodología de programación en la que los sistemas se dividen en funciones que son usadas de manera secuencial.

POO (Programación orientada a objetos) - es un conjunto de métodos de implantación en los cuales los programas son organizados como una colección cooperativa de objetos y cada uno de éstos representa una instancia de alguna clase y tales clases son miembros de una jerarquía de clases unificada por medio de relaciones de herencia.

Snap - son los intervalos de desplazamiento del ratón y que sirve para la edición de la estructura dentro del programa, ya que su activación permite que el ratón se desplace a intervalos fijos que se indican en las escalas de cada uno de los ejes coordenados dentro del programa, lo que hace más fácil la edición de la estructura.

Subclase - en nuestro contexto una subclase es aquella clase que sólo es usada por una clase en particular y pasa desapercibida para el resto del sistema, debido a que el resto de las clases no hacen uso de ella.

Superclase (clase derivada) - una superclase o clase base es aquella clase de la que se derivan una o más clases, por medio de la herencia. Véase Clase, Clase derivada.