Universidad Nacional Autónoma de México

Facultad de Ingeniería

Análisis Estructural de Armaduras Planas Utilizando Microcomputadora

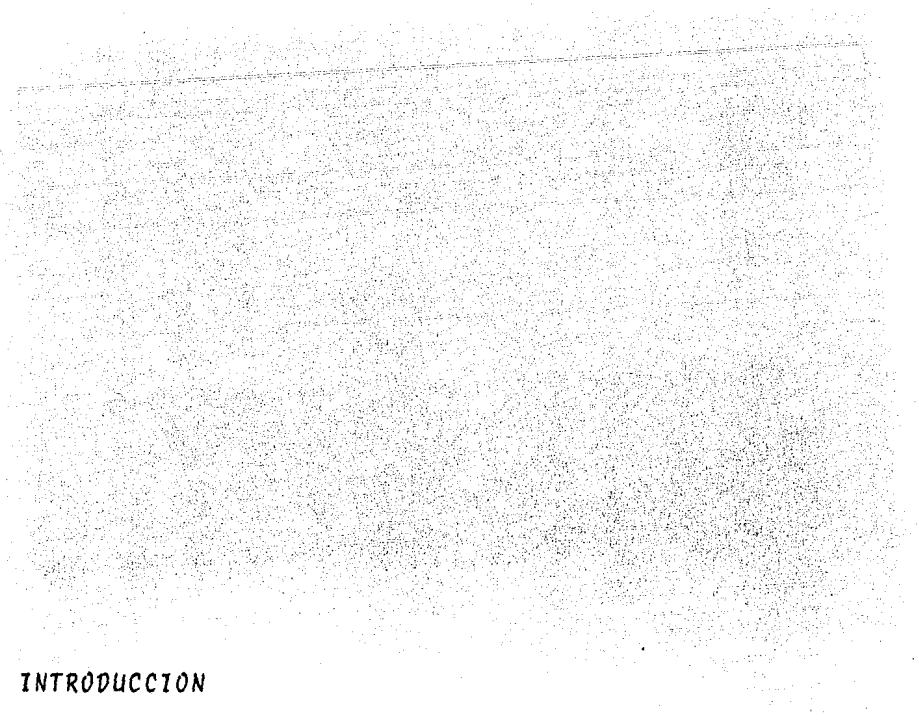
T E S I S

Que para obtener el título de:
INGENIERO CIVIL
p r e s e n t a :

FERNANDO MONROY MIRANDA

México, D. F. 1984

I N D I C E



INTRODUCCION

Como una gran parte de la Ingeniería Civil se refiere al cálculo, investigación, evaluación y otras actividades - donde se procesa información, la computadora se ha convertido en una herramienta fundamental para llevar a cabo dichas -- actividades.

En los últimos años las empresas de cómputo, han puesto en el mercado un tipo de computadoras llamadas microcomputadoras, las cuales son versiones pequeñas de las grandes computadoras.

Sucede en la industria de la computación que el área - de Software (programas de aplicación) ha sido descuidada o quizá este enfocada a otras aplicaciones de la computación, ya que gran parte del Software disponible en el mercado para las microcomputadoras es para aficionados (programas para juegos por ejemplo).

Sin embargo las microcomputadoras pueden ser aprovechadas para realizar bastantes actividades relacionadas con la Ingeniería Civil, y para algunas aplicaciones las microcomputadoras son tan capaces como los grandes sistemas.

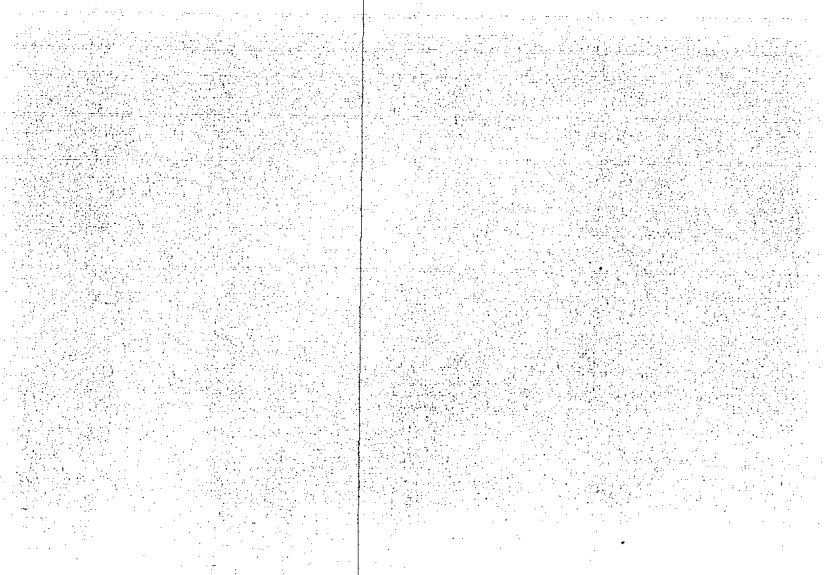
Por lo tanto se presenta en este trabajo una aplicación de las microcomputadoras a la solución de armaduras planas, debido a ello se tratan los temas relacionados con dicho problema.

En el primer capítulo se presenta una introducción a las computadoras y a la programación para ubicar dentro de la computación a las microcomputadoras así como el de conocer algunas de sus características y limitaciones de estas máquinas.

El segundo capítulo se refiere al lenguaje más común que las microcomputadoras utilizan para que el usuario pueda comunicarse con ellas, presentando en este capítulo una descripción de las características de dicho lenguaje.

El tercer capítulo contiene la teoría acerca de la solución que propone el análisis estructural para armaduras planas, utilizando un planteamiento matricial debido a que permite fácilmente la utilización de computadoras para su solución.

Una vez conocidos los elementos que intervienen en el problema, en el capítulo cuarto se presenta la solución de armaduras planas utilizando microcomputadora, este capítulo contiene un programa para ser empleado en estas máquinas, el cual fué utilizado para resolver los ejemplos que ahí se presentan.



1. INTRODUCCION A LAS COMPUTADORAS Y A LA PROGRAMACION

1.1 Generalidades sobre computadoras

El hombre primitivo efectuaba sus cuentas y algunas operaciones de aritmética simple con la ayuda de sus dedos, de piedras y de palos, relacionando estos objetos con otros tales como cabras, ovejas, ganado, etc. Su montón de piedras o palos constituyó la primera computadora digital primitiva.

El abaco, instrumento que fué inventado en China alrededor del año 2600 A. de C. representa el primer avance en computación digital. En 1642 Blas Pascal, un filósofo, religioso, científico, y matemático francés, desarrollo la primera máquina sumadora mecánica que era similar en principio a las máquinas actuales.

Alrededor de 1833, el matemático ingles Charles Babbage ideó y diseño en papel la primera computadora digital automática que llamó "Máquina Analítica". Tenía muchas de las características de las computadoras modernas, desgraciadamente, esta computadora automática nunca se construyó debido a dificultades técnicas y financieras.

No fué sino hasta 1949, que las computadoras, semejantes en principio a la que ideo Babbage, fueron finalmente construídas.

La primera de estas, conocida como Calculadora Automática de Secuencia Controlada, o Mark I fue ideada por Howard - Aiken en la Universidad de Harvard, y fué construída por la compañía International Business Machine (IBM) en 1944. La - Mark I era capaz de efectuar una secuencia fija de operaciones controladas por una cinta perforada. Constaba fundamentalmente de componentes mecánicos y electromecánicos y tenía una memoria capaz de almacenar 72 números de 24 dígitos cada uno.

La segunda guerra mundial dió un impetu definitivo al - desarrollo de las computadoras, y el Integrador y Calculador Electrónico Numérico, (ENIAC), fué construído bajo contrato con el ejercito de los Estados Unidos, lo completaron en - - 1946 los ingenieros de la Universidad de Pensylvania, y era similar a la Mark I excepto que su operación era predominantemente electrónica y por lo tanto efectuaba sus cálculos mucho más rápidamente, esta máquina fue la primera que utilizó tubos electrónicos al vacío para hacer sus cálculos, esta -- computadora ocupó todo el sótano de uno de los edificios de la Universidad (más de 150 m^2) y pesaba más de 30 toneladas, conteniendo más de 18 000 tubos electrónicos, la ENIAC podía completar en un día aquellos procesos que requerían 30 días en las computadoras electromecánicas.

Un avance importante en las computadoras se logró en -- 1945 cuando John Von Neumann del Instituto de Estudios Avanzados de Princeton y H.H. Goldstine, del Departamento de ordenanzas del ejército, propusieron almacenar en la memoria - de la computadora la secuencia de operaciones que había que efectuar, junto con los números sobre los que se debía ope-- rar.

Esta característica permitía que la computadora "Bifurcara", es decir, que siguiera cualquiera de dos secuencias - alternas de instrucciones dependiendo de alguna condición -- existente en el instante de efectuar la instrucción de bifurcación. También hacía mucho más fácil "formar ciclos", es - decir la ejecución repetitiva de porciones del programa almacenado internamente.

Si bien las ideas de máquinas inteligentes, robots y automatización industrial, estuvieron presentes en la primera mitad del siglo XX, las primeras computadoras son creadas para servir como instrumentos de cálculo en los Institutos de Investigación, organismos militares y estadísticos y departamentos de las grandes corporaciones industriales, aunque los resultados logrados eran importantes, el mercado para los -- grandes y costosos equipos de cálculo, era necesariamente res-

tringido, motivando a las industrias de cómputo a la busqueda de nuevas aplicaciones.

El uso de las computadoras en la automatización de procesos administrativos, que si bien requerían cálculos relativamente sencillos, hacían necesarias la ejecución de grandes volúmenes de datos y pronto se convirtió en la principal - - área de aplicación, esta nueva forma de aplicación conocida como "Proceso electrónico de datos", los requerimientos principales se encontraban en la entrada y salida de grandes volúmenes de información.

La utilización masiva y problemática de tarjetas perforadas como forma principal de almacenamiento y transferencia de información, orientó los esfuerzos industriales y de in-- vestigación al desarrollo de nuevos medios de almacenamiento cuyos resultados fueron entre otras, la aparación de la cinta magnética y el desarrollo de la impresoras de alta velocidad (del orden de 100 líneas por minuto), las cuales constituyeron los elementos principales del éxito logrado en el -- "proceso electrónico de datos" que abrió el mercado de las - computadoras a los sectores financiero, industrial y gobierno, todos ellos con enormes problemas de administración, los cuales en su totalidad fueron resueltos.

Un paso importante en el proceso de diversifiación y desarrollo de la computación fue la aparición de los llamados "sistemas de información" que tuvieron gran éxito durante la década de los setenta, entre los avances tecnológicos que -- hicieron factible este nuevo avance, podemos citar la introducción del disco magnético (por IBM en 1960), el desarrollo de multiprogramación y la capacidad de utilización de teletipos y posteriormente de terminales interactivas para dar lugar al "Tiempo compartido" introducido simultáneamente en -- forma comercial por Burroughs, Univac y General Electric. - Aunque las aplicaciones más conocidas de los sistemas de información se dieron en los bancos y en las compañías de aviación, su impacto en organismos gubernamentales, industrias y corporaciones comerciales fué también considerable.

1.2 Finalidad de las computadoras

Los mayores impactos de las computadoras se han sentido en el área de los cálculos científicos, de ingeniería y administrativos y en las actividades de procesamiento de datos.

Es importante darse cuenta desde un principio que la -- computadora es solamente una herramienta en manos del usua--

rio en contraste con las herramientas mecánicas que aumentan las capacidades del hombre en el procesamiento de la energía, la computadora es una herramienta lógica que aumenta su capacidad de procesar información.

Dado que una gran parte de la ingeniería civil se refiere a cálculo, evaluación y otras actividades de procesamiento de información, la computadora digital se ha convertido en parte integral de la educación, investigación y práctica en la ingeniería Civil.

La computadora es la última en una larga serie de herramientas tales como reglas de cálculo, sumadoras de escritorio, tablas, gráficas, monogramas, etc. que han tenido aplicación en la ingeniería civil, sin embargo las herramientas que la precedieron no cambiaron significativamente la manera de proceder de un ingeniero al desarrollar una tarea. Por otra parte, la computadora además de ser mucho más versátil que las otras herramientas, requiere un enfoque completamente diferente para la solución de problemas y para otras actividades del procesamiento de datos.

Requiere que antes de efectuar cualquier cálculo se tiene que especificar en su totalidad el proceso de solución del problema. Debe observarse que la computadora unicamente ejecuta una serie de instrucciones y no "resuelve problemas" por si sola, la computadora nos ofrece una ayuda muy valiosa proporcionando resultados cuantitativos para explorar diferentes alternativas.

Así pues deberemos de especificar a la computadora dicho proceso de solución para nuestro problema, en una serie de instrucciones ordenadas de manera lógica que a su vez la computadora entienda.

1.3 Clasificación de las Computadoras

Las computadoras se clasifican en digitales, analógicas e híbridas, las digitales implican que dentro de la computadora la información se representa por una serie de caracteres como sucede en una calculadora de escritorio o sumadora, donde los números se representan por dígitos, en las analógicas los números se representan por cantidades físicas de variación contínua.

En una computadora electrónica las operaciones de la má

quina son a base de circuitos electrónicos y no por sistemas mecánicos de engranes, con esto se obtiene una gran velocidad en las operaciones, por ejemplo el tiempo requerido para efectuar una suma en la computadora puede variar de menos de un microsegundo a un nanosegundo (billonesima de segundo).

La computadora analógica es una colección de dispositivos electrónicos que puden efectuar operaciones matemáticas básicas como suma, resta, multiplicación, división y generación de funciones, es muy útil en el estudio de sistemas de variación con el tiempo ya que la computadora simula fácilmente el comportamiento dinámico de cualquier sistema.

Las soluciones obtenidas de una computadora analógica se presentan usualmente en forma gráfica en un osciloscopio, el usuario tiene la oportunidad de cambiar los coeficientes en un potenciómetro y ajustar los valores de los parámetros, experimentando el comportamiento y observando el efecto del cambio de parámetros en una o más variables del sistema en estudio.

La computadora analógica tiene la característica de poder cambiar la escala de tiempo y hacer lentas las soluciones rápidas o acelerar las soluciones lentas, lo que nos lleva a un mejor análisis a un costo menor.

En algunos sistemas físicos es imposible o muy peligroso estudiar las condiciones críticas de operación del sistema, sin embargo, el modelo de dicho sistema en la computadora puede llevarse al límite de destrucción, proporcionando con esto la única manera de analizar con detalle el comportamiento del sistema en situaciones críticas. Las computadoras analógicas, tienen la desventaja de estar limitadas en el tamaño del problema que pueden resolver, de no tener dispositivo de memoria para almacenar soluciones y por lo tanto no pueden considerar decisiones lógicas con las soluciones obtenidas para decidir la secuencia de cálculo posterior.

Las personas que usan las computadoras digitales y analógicas se han especializado a tal grado que no hay comunicación entre ellas, aunque es de admitirse que algunos problemas son más adecuados para alguno de los tipos de computadora. Sin embargo, para algunos problemas de ingeniería se hizo necesario utilizar las técnicas de computación analógica y digital, dando como resultado la computadora híbrida.

La computadora híbrida es una combinación de computado-

ras analógica y digital, con un sistema adecuado de comunicación y operación, ya que la analógica opera básicamente en forma contínua y en paralelo, mientras que la digital opera en forma discretizada y secuencial. El problema de comunicación no es solo la conversión de analógica a digital y viceversa, sino también la solución de los complejos problemas de tiempo para asegurar que la transferencia de información entre los dos sistemas se realiza eficientemente.

1.4 Componentes de una Computadora

Las computadoras se fabrican en una gran variedad de tamaños, velocidades y capacidades e involucran conceptos internos de operación radicalmente diferentes, pero toda computadora está compuesta de una parte física llamada HARD WARE y otra lógica conocida como SOFT WARE , el Hardware son los equipos electrónicos, mecánicos y electromecánicos que forman físicamente la estructura de la computadora, esta parte se encarga de captar la información, de las operaciones aritméticas y lógicas, del almacenamiento de la información y de la impresión de resultados, en general esta compuesto de:

a) Dispositivos de entrada
b) Procesador central o CPU
c) Unidades de almacenamiento o memoria
d) Dispositivos de salida

El Software está formado por los programas escritos en un lenguaje apropiado a la estructura física de la máquina y gracias a estos es posible utilizarla, básicamente lo constituyen:

a) Sistema operativo
b) Compiladores
c) Intrinsecos
d) Interprete
e) Rutinas de utilería y paquetes de biblioteca

Desde el punto de vista del usuario todas las computadoras se pueden considerar formadas por 5 componentes funcionales que se muestran esquemáticamente en la siguiente figura:

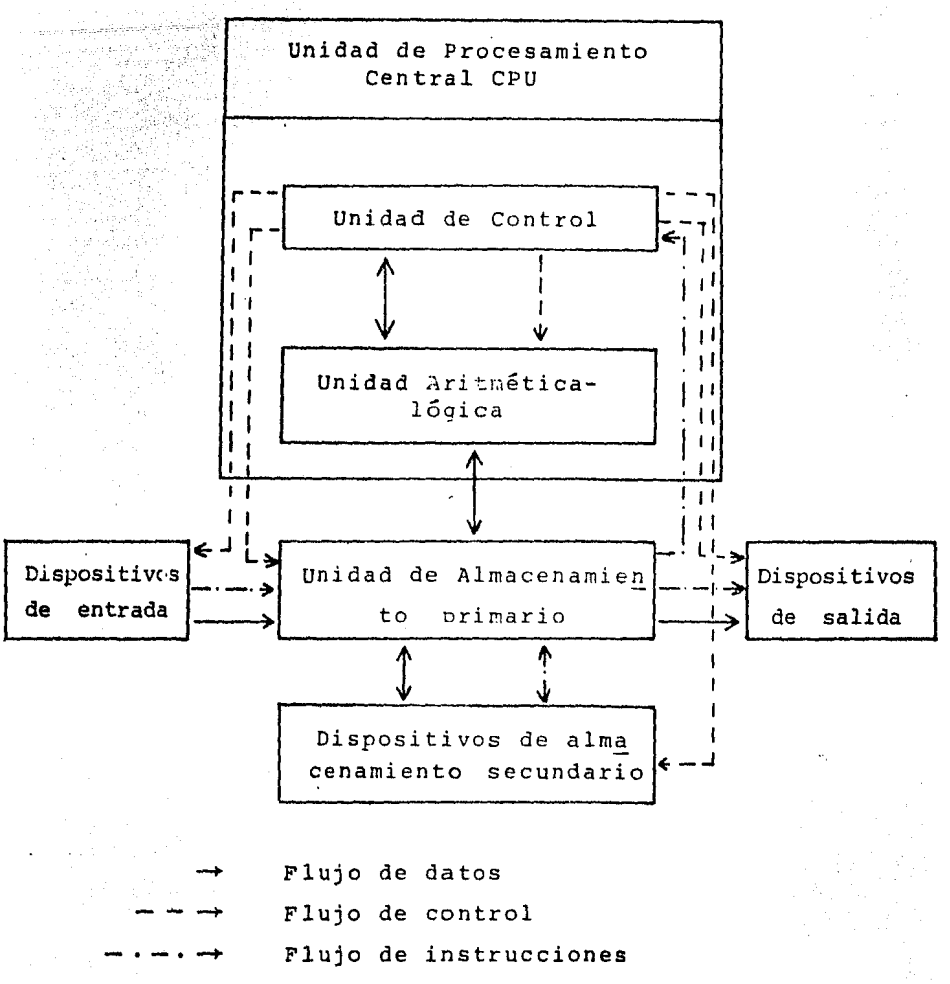


Fig. 1.4.1 COMPONENTES FUNCIONALES DE UNA COMPUTADORA

UNIDAD DE ENTRADA

Se usa esta unidad para introducir información a la computadora y puede consistir en lectoras de tarjetas perforadas, de cintas de papel o de cintas magnéticas, lectoras ópticas de caractéres, teletipo, tubos de rayos catódicos, etc. Las instrucciones, codificadas en un lenguaje accesible a la máquina, se transmiten mediante el dispositivo de entrada a

la unidad de memoria de la computadora, antes de efectuar -- los cálculos. Los datos pueden ser transmitidos por la unidad de entrada a la unidad de memoria antes de efectuar los cálculos o se pueden "leer" conforme se necesitan durante dichos cálculos.

MEMORIA O UNIDAD DE ALMACENAMIENTO

La unidad de memoria debe ser capaz de almacenar los -- números y otros caracteres que representan instrucciones y - datos. Aunque se han utilizado muchos dispositivos para almacenamiento predomina actualmente el uso de microcircuitos integrados en la memoria principal, debido a la gran rapidez con la que se puede localizar y transferir la información almacenada a otra componente.

La unidad de memoria recibe las instrucciones y los datos, y los almacena en localidades asignadas separadamente, cada una de estas localidades de almacenamiento se identifica mediante una dirección numérica que representa alguna localidad para la computadora, una vez almacenados los datos e instrucciones, se hace referencia a ellos mediante su dirección numérica, además de almacenar datos e instrucciones, la unidad de memoria se utiliza para almacenar resultados intermedios y finales que se van a utilizar después o que se van a transmitir al exterior o que se van a utilizar en alguna - comparación.

El tiempo que se requiere para que la computadora localice una instrucción y la transfiera a la unidad de control para su interpretación, o el tiempo que se requiere para localizar una unidad de datos en la memoría y transferirla a - la unidad aritmética para efectuar un cálculo se conoce como tiempo de acceso de la máquina en algunas computadoras modernas de alta velocidad tienen tiempos de acceso que se miden en nanosegundos.

UNIDAD ARITMETICA - LOGICA

La unidad aritmética lógica consiste en todos los circuitos electrónicos necesarios para efectuar las diferentes operaciones aritméticas, esta unidad es capaz de hacer operaciones como suma, resta, multiplicación, división, decidir - si un número es mayor que otro, si dos proposiciones son verdaderas simultáneamente, si por lo menos una de varias proposiciones es verdadera, etc., además suministra la lógica - de la computadora, esto último se logra generalmente efectuando pruebas sobre ciertas condiciones que existen en la -

máquina, para decidir cual de las sucesiones de instrucciones alternas se deben seguir.

UNIDAD DE CONTROL

La unidad de control de una computadora es el conjunto de circuitos electrónicos que se encarga de coordinar los demás elementos de la computadora. Así pues, es la unidad de control la que se encarga de leer la información de la memoria y enviársela en el orden adecuado a la unidad aritmética para que esta efectúe las operaciones deseadas. Asimismo cuando la unidad aritmética termina un cálculo, es la unidad de control la que se encarga de enviar el resultado a la posición que le corresponde en la computadora. Al ejecutar un programa, la unidad de control es la que se encarga de enviar todos los pulsos que sean necesarios para que dicho programa se ejecute línea por línea y se encarga también de hacer lo necesario cuando en el programa se encuentra una bifurcación que indica cual es la siguiente línea a ejecutarse, dependiendo del valor de un cierto número en la memoria.

En realidad, la diferencia entre la unidad aritmética lógica y unidad de control es un tanto arbitraria pues los circuitos electrónicos estan entremezclados, al conjunto de estas unidades se acostumbra llamarlo unidad central de proceso (CPU), dicha unidad es realmente el alma de la computadora.

MEMORIAS AUXILIARES

En algunas aplicaciones se requiere manejar cantidades muy grandes de información por lo que en dichas aplicaciones a menudo la capacidad de la memoria principal resulta insuficiente. Todas las computadoras modernas cuentan con equipo de memoria auxiliar, siendo los más usuales las cintas magnéticas, los discos, los tambores y otras.

Este almacenamiento complementa al almacenamiento principal de la computadora y generalmente guarda cantidades mayores de datos, estos dispositivos de almacenamiento auxiliar pueden guardar desde varios cientos de miles hasta varios cientos de millones de caracteres de datos.

UNIDAD DE SALIDA

Los resultados obtenidos como consecuencia de las operaciones aritméticas efectuadas por la computadora se deben --

comunicar al usuario, esto se logra mediante algún tipo de unidad de salida, dichas unidades de salida consisten usualmente en uno de los siguientes dispositivos: Perforadora de cinta de papel, unidad de cinta, perforadora de tarjetas, impresora en línea, graficadora, teletipo, etc.

Excepto las impresoras, los mecanismos de salida son esencialmente iguales a los de entrada, sólo que actúan en forma inversa, así por ejemplo para tarjetas perforadas, en lugar de leerlas, un mecanismo de salida las perfora, naturalmente se utilizan las mismas claves, de manera que una tarjeta que se obtiene de una perforadora puede ser leída subsecuentemente por una lectora de tarjetas sin ningún procesamiento adicional intermedio. La mayoría de las computadoras tienen conectados varios dispositivos de entrada y de salida para proporcionar mayores velocidades y una mayor flexibilidad en su uso.

El sistema operativo es un programa almacenado en memoria que se encarga de controlar la asignación del procesador y coordinar las funciones del Hardware, este programa reparte los recursos de la máquina en forma óptima.

Los compiladores son programas que generan un grupo de instrucciones en lenguaje de máquina (código que puede ejecutar la computadora) a partir de un programa escrito en algún lenguaje, por ejemplo BASIC, FORTRAN, PASCAL, etc., así se tienen compiladores BASIC, FORTRAN, PASCAL, etc., los cuales traducen un conjunto de instrucciones de un cierto lenguaje a uno igual pero en lenguaje de máquina, el cual es llamado programa objeto y puede ser ejecutado cuantas veces se desee.

Los intrínsecos son pequeños módulos de programas que pueden ser utilizados por diferentes usuarios, sin que ellos tengan que programarlos, por ejemplo, la raíz cuadrada, las funciones trigonométricas, etc.

El interprete son programas que traducen instrucciones en lenguaje de máquina, ejecutando cada instrucción traducida sin generar el programa objeto.

Las rutinas de utilería y paquetes de biblioteca son programas despecializados que simplifican o ejecutan procesos que comunmente se llevan a cabo.

1.5 Solución de un Problema por Computadora

Para que una computadora digital nos ayude en la solu--

ción de un problema, se debe, en la mayoría de los casos, -- realizar los siguientes pasos:

1. Especificación del problema. Es decir se debe de identificar perfectamente y en su totalidad el -- problema, sus limitaciones así como tener un buen conocimiento sobre el método general de solución para el problema, identificar todas las variables relacionadas con el problema, los datos necesarios y los resultados deseados.

2. Análisis. Es la formulación matemática detallada del problema, establecido un conjunto de acciones que determinen la secuencia de los pasos a seguir para resolver el problema a este conjunto de acciones también se le llama algorítmo, dicho algorítmo debe de contar con las siguientes características: El procedimiento para la solución de un problema - se debe de terminar en un número finito de pasos, todos los pasos deben estar definidos con precisión, por lo tanto no deben existir especificaciones cuya interpretación sea ambigua y de origen a elegir una decisión o a tomar un curso de acción no deseado, el desarrollo del algoritmo propuesto nos debe conducir a la solución del problema planteado, es decir que al ejecutar o realizar los pasos señalados el procedimiento nos conduzca al final del mismo a obtener el resultado buscado.

   Como la computadora digital es capaz de efectuar unicamente operaciones aritméticas, los problemas que no se puedan resolver en su forma usual me--- diante procedimientos aritméticos deben ser transformados a una forma consistente con dichos pro-- cedimientos aritméticos es decir que las funciones trigonométricas, integrales, derivadas, ecuaciones diferenciales y todas aquellas utilizadas en la solución del problema se deben expresar en términos de operaciones aritméticas adecuadas para la computadora.

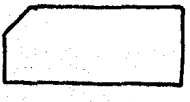
3. Programación. Consiste en establecer en detalle la sucesión o lógica de procesamiento del problema, (entendiéndose por lógica como una cadena -- de razonamientos), se considera dividida en dos partes, en la primera la sucesión de operaciones se presentan en forma gráfica en un diagrama de bloques o diagrama de flujo, en dicho diagrama - se contemplan los siguientes elementos:

- Inicio
- Especificación de los datos de entrada
- Operaciones a realizar con los datos o decisiones a tomar
- Especificación de salida (resultados)
- Fin

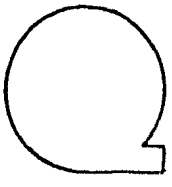
Hasta ahora no existen reglas o estándares que indiquen claramente la interpretación o uso que deba darse a todas -- las figuras geométricas que usualmente se utilizan para la - elaboración de un diagrama de flujo, sin embargo, las figu-- ras geométricas más comunes utilizadas, así como su interpre tación son las que a continuación se indican:

Esta figura en forma de óvalo se utiliza para indicar el -- inicio o fin de un procedimien to.

El rectángulo nos sirve para indicar cualquier operación que se tenga que realizar en el procedimiento.



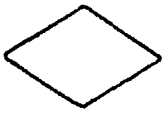
El rectángulo con un corte en el margen superior izquierdo indica una operación de entra da o salida a través de tarje tas perforadas.



El círculo se emplea para re- presentar una cinta magnética e indica entrada de datos al- macenados en ella o bien sali da de resultados que quedarán grabados en dicha cinta.

Este símbolo se utiliza para representar un disco magnético, e indica entrada de datos almacenados en el o bien salida -- que quedará grabada en dicho disco.

Un exágono se usa para indicar el inicio y el fin de un proceso iterativo.

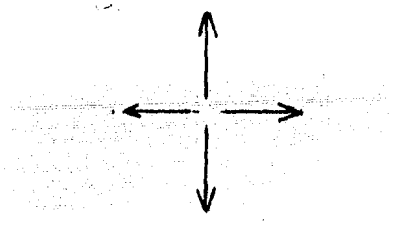


Un rombo se utiliza para indicar una decisión, es decir elegir una alternativa entre dos o más que se presentan, dependiendo de una cierta condición.

Este símbolo se utiliza para denotar que los resultados en una operación de salida aparecerán en hoja impresa.

Este pequeño círculo, llamado conector se utiliza para indicar cambios en la secuencia del procedimiento.

Este símbolo también llamado conector, se utiliza para indicar la continuación de un diagrama de hoja a hoja.



Las flechas se utilizan para - indicar hacia donde se dirige el flujo del proceso.

El diagrama de flujo permite dar una idea precisa de lo que se desea hacer, por ello es una importante herramienta - en la programación, ya que permite al programador planear la secuencia de operaciones en un programa antes de escribirlo, en un diagrama de flujo se debe tener cuidado de expresar en forma clara todas y cada una de las operaciones y/o decisiones a realizar, con el fin de que cualquier persona, aún aje na a la programación pueda interpretarlo, podemos señalar -- como ventajas al utilizar un diagrama de flujo, las siguientes:

- Permite planear en forma gráfica las operaciones y/o decisiones de un programa antes de escribirlo.
- Representa ayuda visual para observar las - correlaciones que se presentan entre operaciones o decisiones de un programa.
- Facilita la interpretación.
- Ayuda a la comunicación entre programadores
- Forma parte de la documentación de un pro-- grama.
- Es independiente del lenguaje de programa-- ción a emplear.

Como se mencionó antes, en muchas ocasiones se utiliza el término diagrama de bloque como sinónimo de diagrama de flujo, sin embargo, por lo general la acepción de diagrama - de bloque es utilizada para indicar operaciones o decisiones agrupándolas por objetivos.

Tendientes a eliminar algunas desventajas derivadas de - la utilización del diagrama de bloque y del de flujo, pero - principalmente con los grandes avances que se han tenido en la velocidad, capacidad y economía del Hardware de las com-- putadoras ha llevado a una creciente complejidad en las apli caciones automatizadas en las empresas, dada la gran impor-- tancia que en la mayoría de ellas tiene el proceso electróni co de datos ha llevado a los profesionales del área, a la -- búsqueda de técnicas formales para facilitar el desarrollo - de programas y sistemas, quizá, dentro de todas las técnicas que han surgido, destacan las siguientes:

1) Programación Estructurada
2) Pseudocodigo
3) Segmentación
4) Desarrollo Descendente
5) Hipo
6) Bibliotecas de Soporte

Una característica fundamental de todas las técnicas anteriores es, la sencillez de conceptos que involucran, de la cual deriva su gran aceptación y correspondiente éxito, estas, para su desarrollo, utilizan reglas del teorema de la estructura (programación estructurada) y son:

1. Usar solamente las siguientes estructuras lógicas (estructuras básicas).
   - Secuencia de una o más operaciones (SEQUENCE)
   - Bifurcación a una de dos operaciones y regreso (IF-THEN-ELSE)
   - Repetición de una operación mientras una condición sea verdadera (DO-WHILE)
2. Combinar las estructuras básicas
   - Sustituyendo una por otra
   - Anidando una dentro de otra

Algunas de las técnicas anteriores, principalmente la programación estructurada, posee varias diferencias las que a su vez pudieran también ser ventajas que esta técnica tiene con respecto al diagrama de flujo, y estas son:

- Nunca rompe su secuencia, es decir no hay bifurcaciones incondicionales (GO TO'S)
- Facilidad de lectura, de arriba hacia abajo
- Fácil de entender ya que solo contiene estructuras básicas.

Utilizando alguna técnica o algunas técnicas (inclusive combinándolas) de las antes mencionadas y con su práctica asociada (recomendaciones de dichas técnicas), se pueden lograr programas y sistemas en un mínimo de tiempo.

La segunda parte de la programación es la presentación de este diagrama en un lenguaje accesible a la máquina, es decir para que la computadora comprenda, interprete y ejecute una serie de instrucciones se le deben de transmitir estas en forma codificada, de acuerdo con la descripción dada

por el diagrama de flujo, por ello a esta parte se le denomina codificación.

4. Verificación. Es la prueba exhaustiva del programa para eliminar todos los errores que tenga, de manera que efectúe lo que se desea. Los resultados obtenidos con el programa que corresponden a datos de problemas ya resueltos, se comparan con los resultados que ya se tenían sobre dichos problemas.

5. Documentación. Consiste en colocar comentarios en diversos lugares del programa para identificar de manera general las partes o procesos que lo constituyen, sirviéndo de ayuda para hacer facilmente posteriores modificaciones que este requiera, así como en la elaboración de un instructivo de utilización del programa, especificando en este el objetivo y limitaciones del programa, el método general de solución empleado, el tipo, forma y secuencia de los datos de entrada así como la interpretación de los resultados, conteniendo además uno o varios ejemplos.

6. Producción. Consiste en la utilización masiva del programa, para esto se consideran datos de entrada del problema obteniéndose las soluciones correspondientes, en general se pueden introducir varios grupos de datos referentes a distintas condiciones del problema o problemas, obteniéndose las respuestas correspondientes para los distintos grupos de datos.

De lo antes expuesto se puede observar que es necesario un buen conocimiento del problema y de los campos de la física y de las matemáticas u otras disciplinas relacionadas con él. Es de mencionarse que la selección del método de análisis es muy importante, ya que de ello dependerá el número de operaciones y de decisiones que se deban de efectuar así como de los problemas que se tengan en la programación.

También se ha observado que tan importante como es aprender programación lo es evaluar, adaptar y utilizar los programas desarrollados por otros programadores.

1.6 Lenguajes de Programación

Como se vió anteriormente, un programa es una colección de instrucciones que una computadora puede entender y ejecutar, para comunicarle el programa a la computadora se utilizan las unidades de entrada-salida como lectoras de tarjetas, teletipos, etc.

Existen diversos niveles a los cuales se pueden escribir programas, los cuatro principales son:

a) A nivel de lenguaje de máquina
b) A nivel de ensamblador
c) A nivel de compilador
d) A nivel orientado a problemas en áreas específicas

Como los lenguajes naturales, los lenguajes de computadora son colecciones de símbolos y reglas que tienen significados específicos para la computadora y que sirven para establecer la comunicación entre el hombre y la máquina (o entre una máquina y otra).

Dependiendo del objetivo del lenguaje hay algunos que están orientados a la máquina y otros que estan orientados al hombre.

Un lenguaje de máquina es un conjunto de símbolos que la máquina puede entender con mucha facilidad, una instrucción en lenguaje de máquina es una sucesión de ceros y unos que en clave le informan a la computadora que hacer, dependiendo de la secuencia la máquina realiza las diferentes operaciones, el más pequeño error, un cero fuera de su lugar, facilmente puede inutilizar el trabajo a realizar, dadas las características este lenguaje es difícil de manipular.

Para solucionar este problema se han desarrollado lenguajes llamados ensambladores bastante cercanos a la operación de la computadora pero un poco más entendibles por el hombre, en la mayoría de los casos una instrucción en lenguaje ensamblador se traduce a una instrucción en lenguaje de máquina sin embargo, en lenguaje ensamblador, una instrucción en lugar de ser una sucesión de ceros y unos, es una línea de letras que el hombre puede entender con más facilidad, pues las letras que se usan son combinaciones mnemónicas de nombres de operaciones comunes como sumar, restar, multiplicar, etc. Así en algunos ensambladores se emplea la palabra ADD para efectuar una suma y SUB para una resta.

No obstante para muchos programas de aplicación, sigue siendo demasiado tedioso escribirlos en un ensamblador, para resolver esta dificultad se han desarrollado lenguajes aún más cercanos al hombre, los cuales son capaces de aceptar operaciones matemáticas y lógicas muy parecidas a los que usan los profesionales de diversas disciplinas (ingenieros, físicos, matemáticos, químicos, etc.) a estos lenguajes también se les llama super-lenguajes.

Entre los superlenguajes más comunes y que se generalizarón en la década de los 60, destacan el FORTRAN (FORmula TRANslation o Traducción de Fórmulas). el COBOL (COmmon -- Business Oriented Languaje o lenguaje orientado a los negocios) y el ALGOL (ALGOrithmic Languaje o lenguaje algorítmico), este último se popularizó fundamentalmente en Europa y en nuestro país solo se usa en unos cuantos equipos.

Posteriormente se generalizaron el BASIC, el PL/I, el APL, el Pascal y muchos otros, de estos últimos el PL/I y el APL se desarrollaron para equipos grandes, el BASIC tanto para equipos grandes y principalmente para pequeños (desde luego existen excepciones y nos encontramos con microcomputadoras que manejan FORTRAN y PASCAL).

En un esfuerzo adicional por tener lenguajes más cercanos al hombre, quien es, después de todo, al que la máquina debe servir, se han desarrollado lenguajes orientados a problemas específicos, en los cuales es posible con unas cuantas instrucciones ordenarle a una computadora que analice -- una complicada estructura de ingeniería civil o que simule la operación de un órgano humano.

En estos lenguajes la descripción de un problema se asemeja mucho a la terminología establecida en esa materia, de esta forma el problema se le describe a la computadora básicamente en los mismos términos en que se le describe a una persona conocedora de ese tema, son ejemplos de estos lenguajes: TEMPO, BASIS, OPTIM, COGO, SASII, STRES, ICES, etc. -- Los tres primeros se utilizan para el análisis matemático, estadístico y de optimización, el cuarto para problemas de topografía, el quinto para simulación y los dos últimos para problemas de ingeniería civil.

Con vertiginosa rapidez vemos aparecer más y más lenguajes de programación, unos totalmente nuevos, otros variantes de los ya conocidos, dialectos podríamos decir.

Ante esta situación, la actitud tanto de los especialistas en cómputo como los proveedores de equipo, han dejado mucho que desear. por un lado nos encontramos con que la inmensa mayoría de los programadores "profesionales" sólo conocen razonablemente bien un lenguaje, con lo que dejan de aprovechar las facilidades y ventajas que para cada aplicación particular pueden ofrecer otros lenguajes. Por otra parte, los proveedores de equipo en su afán mercadotécnico de diferenciar su producto, generan una variedad inútilmente grande de lenguajes y dialectos, que solo producen confusión y dependencia entre sus clientes.

Sustituir un equipo de cómputo por otro de otra marca, resulta muchas veces incosteable, por la necesidad de reprogramar todo lo ya hecho, en un lenguaje diferente, con lo que el cliente se puede considerar como cautivo del proveedor. A todos estos problemas debemos añadir el número creciente de personas y organismos que entran al cómputo cada año merced al abaratamiento del equipo, y que en su inexperiencia cometen serios errores en la selección de lenguajes.

Para evitar o reducir al mínimo dichos errores, debemos tener una idea de para qué sirve y como es cada uno de los lenguajes de programación de uso común. A continuación se describirán brevemente mediante un ejemplo, los lenguajes de programación que consideramos más importantes en nuestro medio.

Tratemos de resolver una ecuación cuadrática que tiene la forma general siguiente:

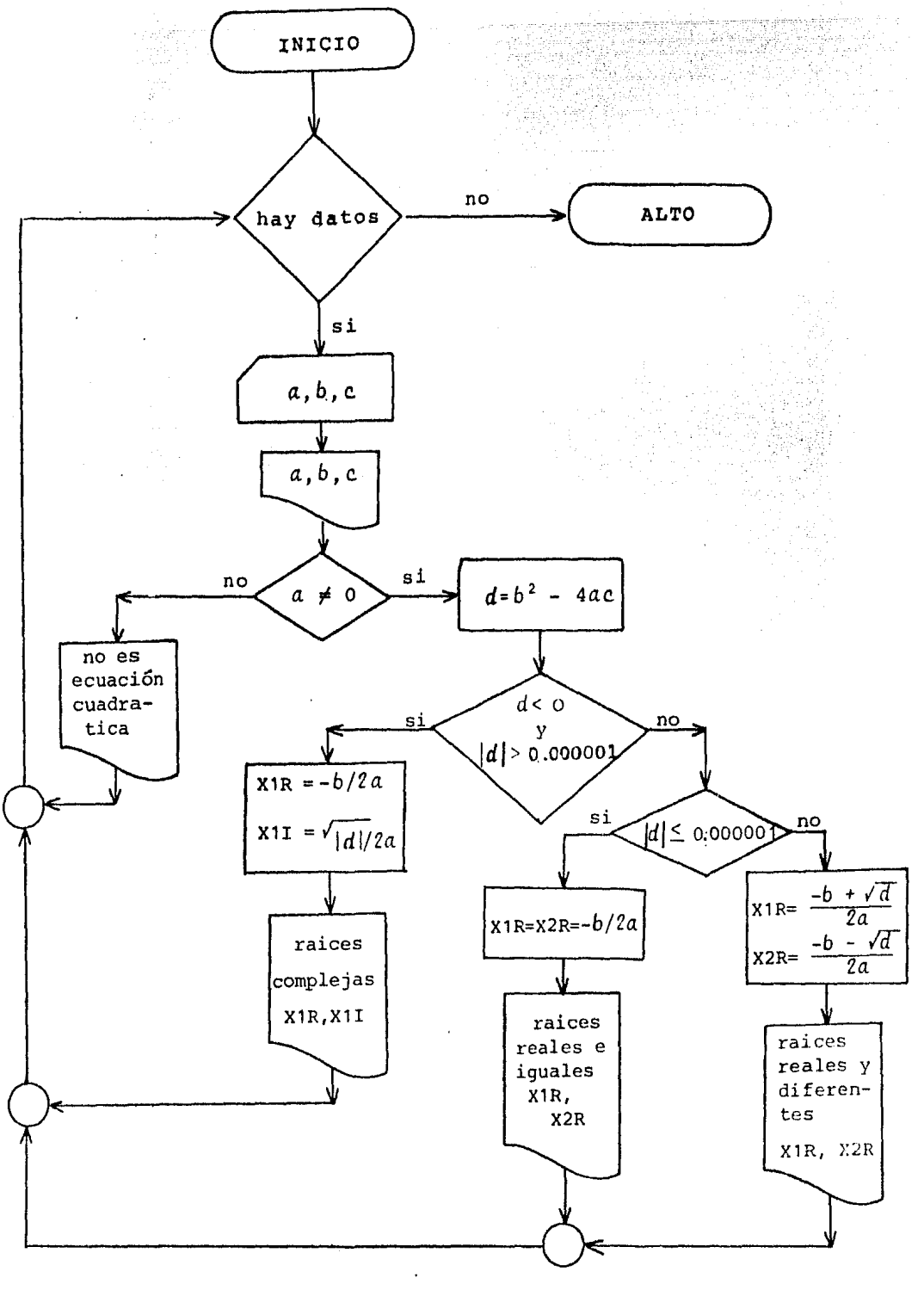
$$ax^2 + bx + c = 0$$

$$a \neq 0$$

recurriendo a la fórmula conocida que es:

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

el diagrama de flujo es el que a continuación se muestra:


INICIO
hay datos
no
ALTO
si
a, b, c
a, b, c
no
a ≠ 0
si
d=b² - 4ac
no es ecuación cuadratica
d < 0 y |d| > 0.000001
si
no
X1R = -b/2a
X1I = √|d|/2a
raices complejas X1R,X1I
si
|d| ≤ 0.000001
no
X1R=X2R=-b/2a
raices reales e iguales X1R, X2R
X1R= (-b + √d)/2a
X2R= (-b - √d)/2a
raices reales y diferentes X1R, X2R

FORTRAN

No obstante haber sido de los primeros superlenguajes, hoy en día sigue siendo sin duda alguna el lenguaje más usado en ciencias exactas e ingeniería.

Las líneas 3000 a 3020 son comentarios que se identifican con una C al principio y en este caso se usaron como --- identificación del programa, los comentarios se pueden usar en cualquier parte del programa.

La línea 3040 sirve para obtener los valores de las --- variables A,B y C. La línea 3050 en conjunción con la 3060 escriben los valores de dichas variables.

Otras instrucciones de escritura son las líneas 3030, - 3080, 3150, 3210, 3250 y 3260. Nótese que algunas de estas instrucciones llevan un número de etiqueta (1000 como en el- ejemplo) y dicho número hace referencia a una declaración -- FORMAT la cual sirve para especificar la forma y posición de datos de salida, también puede utilizarse con instrucciones de entrada de datos (READ). Nótese además que algunas ins-- trucciones de entrada (READ) tanto como de salida (WRITE) -- llevan en su estructura un número llamado número de designación de archivo, el cual previamente se ha asignado a un dispositivo físico para entrada o salida de información como -- puede ser una lectora de tarjetas perforadas, una impresora, etc.

La línea 3070 hace una comparación para determinar si - la variable A no es igual (NE) a cero, si esto es verdad la máquina pasará a ejecutar la instrucción etiquetada con el - número 10, otras comparaciones las encontramos en las líneas 3110 y 3120.

La línea 3100 nos ilustra la manera de indicar cálculos aritméticos, la computadora evaluará la expresión a la derecha del signo = y asignará ese valor a la variable que está a la izquierda de este signo, nótese que no hay un manejo -- algebráico de las ecuaciones, hay que darlas siempre despejadas, ya que de otra manera la máquina marcaría un error, - - otras instrucciones que indican cálculos son las que se en-- cuentran en las líneas 3130, 3140, 3190, 3200, 3230 y 3240.

FORTRAN posee un poderoso arsenal de funciones aritmétic cas de uso común, algunas de ellas como SQRT y ABS para obtener la raíz cuadrada y el valor absoluto de una expresión numérica respectivamente, se encuentran en las líneas 3110, -- 3140, 3230 y 3240.

```
B7800  FORTRAN COMPILATION MARK 5.2.177#3.2.1000*  TUESDAY, 11/30/82  11:
COMPILER/LOBETSSO ON P1.S
C*    SOLUCION DE ECUACIONES CUADRATICAS                                  0001 0000  C  0002:0000:3
C*    F. MONROY M.  AGOSTO DE 1982.                                       0001 0019  C  0003:0000:3
C*                                                                        0001 0022  C  0004:0000:3
                                                                                     *TAKE OF SEGMENT 002
      PRINT /,'** RESULTADOS DEL PROGRAMA EN FORTRAN **'                  0001 0030  C  0005:0000:3
                                                                                        FT4 IS 0003 LONG
5     READ(5,/,END=99) A,B,C                                              0001 0045  C  0006:0000:2
                                                                                        F14 IS 0003 LONG
      WRITE(6,1000) A,B,C                                                 0001 0053  C  0007:0017:1
1000  FORMAT(/,1X,"A = ",F10.4," B = ",F10.4," C = ",F10.4)               0001 0060  C  0008:0062:2
      IF(A.NE.0) GO TO 10                                                 0001 0070  C  0009:0020:2
      PRINT //,'NO ES ECUACION CUADRATICA'                                0001 0076  C  0010:0021:5
      GO TO 5                                                             0001 0103  C  0011:0021:2
10    D=B*B-4*A*C                                                         0001 0110  C  0012:0024:2
      IF(ABS(D).LE.0.0000001) GO TO 30                                    0001 0120  C  0013:0028:5
      IF(D) 20,30,40                                                      0001 0131  C  0014:0032:2
20    X1R=-B/(2*A)                                                        0001 0137  C  0015:0036:4
      X1I=SQRT(ABS(D))/(2*A)                                              0001 0150  C  0016:0052:2
      PRINT 1200,X1R,X1I,X1R,X1I                                          0001 0161  C  0017:0062:3
1200  FORMAT(/,1X,"LAS RAICES SON COMPLEJAS",//,1X,"X1 = ",F10.4,         0001 0173  C  0018:0062:2
     *    " + ",F10.4,"I",//,1X,"X2 = ",F10.4," - ",F10.4,"I")           0001 0176  C  0019:0062:2
      GO TO 5                                                             0001 0215  C  0020:0062:5
30    X1R=-B/(2*A)                                                        0001 0221  C  0021:0062:2
      X2R=X1R                                                             0001 0232  C  0022:0062:2
      PRINT //,'LAS RAICES SON REALES E IGUALES'                          0001 0240  C  0023:0062:5
      GO TO 50                                                            0001 0246  C  0024:0062:2
40    X1R=(-B+SQRT(D))/(2*A)                                              0001 0253  C  0025:0062:5
      X2R=(-B-SQRT(D))/(2*A)                                              0001 0264  C  0026:0062:2
      PRINT //,'LAS RAICES SON REALES Y DIFERENTES'                       0001 0272  C  0027:0062:1
50    PRINT 1500,X1R,X2R                                                  0001 0306  C  0028:0062:2
1500  FORMAT(/,1X,"X1 = ",F10.4,//,1X,"X2 = ",F10.4)                      0001 0313  C  0029:0062:2
      GO TO 5                                                             0001 0327  C  0030:0062:2
90    CALL EXIT                                                           0001 0336  C  0031:0062:5
      END                                                                 0001 0343  C  0032:0062:7
                                                                                     SEGMENT 002 IS 0079 LONG

                                                                                     SEGMENT 004 IS 0211 LONG
                                                                                     FORMAT SEGMENT IS 0032 LONG
                                                                                       START OF SEGMENT 00A
                                                                                     SEGMENT 00A IS 000C LONG

NO ERRORS DETECTED.  NUMBER OF LABDS = 32.
COMPILATION TIME = 7 SECONDS ELAPSED,  0.45 SECONDS PROCESSING.
02 STACK SIZE = 6 WORDS.  FILESIZE = 50 WORDS.  ESTIMATED CORE STORAGE REQUIREMENT = 262 WORDS.
TOTAL PROGRAM CODE = 149 WORDS.  ARRAY STORAGE = 0 WORDS.
NUMBER OF PROGRAM SEGMENTS = 10.  NUMBER OF DISK SEGMENTS = 23.
PROGRAM CODE FILE = (MZ81)CANDE/CODE1350 ON PACK.
COMPILED ON THE B7700 FOR THE B7700
```

BASIC

Posterior al lenguaje FORTRAN, el lenguaje BASIC, corrigió muchos de los problemas existentes en el primero, en especial los referentes al manejo de variables con contenido alfanumérico, simplificando además otras operaciones.

En el programa del ejemplo, las líneas 100 a 120 son comentarios (REMarks). Los valores de las variables A, B y C se obtienen en la línea 130 y en la 140 se imprimen estos valores sin la necesidad de un postulado FORMAT como en FORTRAN (en algunos sistemas y para algunas versiones del lenguaje FORTRAN, en las instrucciones de entrada/salida se puede omitir la especificación de entrada/salida y se les conoce comunmente como instrucciones de entrada/salida sin formato o con formato libre), como puede verse son instrucciones compactas y además bien sencillas.

Nótese que a diferencia de otros lenguajes, se emplean los números de las líneas del programa sin necesidad de establecer una segunda numeración para etiquetas, en el caso de que estas se requieran.

BASIC posee un número un poco reducido de funciones aritméticas, pero un número aceptable de funciones alfanúmericas, además en la mayoría de los sistemas grandes se dispone de instrucciones y funciones aritméticas para la manipulación compacta y completa de arreglos bidimensionales o matrices.

Tal vez algunos puntos en que FORTRAN es definitivamente más conveniente que BASIC, sea la facilidad para el manejo de subprogramas, es decir, pequeños (o a veces grandes) programas hechos con anterioridad incluídos o no en un programa y que pueden ejecutarse a petición de este, el inconveniente mayor del BASIC es fundamentalmente su poca flexibilidad para pasar argumentos o parámetros a los subprogramas cuando estos los requieren. Otro punto que suele resultar molesto en BASIC, es el hecho de que los nombres de variables usualmente solo constan de una letra y un dígito, o a lo sumo de una letra seguida de otra o de un dígito, lo que en programas grandes suele producir confusiones. Sin embargo en las nuevas versiones del BASIC dichos inconvenientes tienden a desaparecer.

Es importante hacer notar que el BASIC es, el lenguaje más fácil de aprender de entre todos los demás por lo que día a día gana popularidad.

```
>>RSSET LIST
NICO REM SOLUCION DE ECUACIONES CUADRATICAS


0110 REM F. MONROY M. AGOSTO DE 1982.
0120 REM
0125 PRINT "ME LAS RAICES SON "
0130 PRINT
0137 PRINT " ** RESULTADOS DEL PROGRAMA EN BASIC **"
0138 INPUT A,B,C
0145 PRINT
0146 PRINT " A = ";A;" B = ";B;" C = ";C
0147 PRINT
0150 IF A <> 0 THEN 190
0160 PRINT " NO ES ECUACION CUADRATICA"
0170 GO TO 150
0180 LET D=B*B-4*A*C
0185 IF ABS(D) <= .000001 THEN 280
0190 IF D > 0 THEN 350
0200 X2=SQR(ABS(D))/(2*A)
0205 X1=-B/(2*A)
0210 PRINT TAB(5);" COMPLEJAS"
0220 PRINT
0230 PRINT " X1 = ";X1;" + ";X2
0240 PRINT
0250 PRINT " X2 = ";X1;" - ";X2
0260 GOTO 150
0270 X1=X2=-B/(2*A)
0280 PRINT TAB(5);"REALES E IGUALES"
0290 PRINT
0300 PRINT " X1 = ";X1
0310 PRINT
0320 PRINT " X2 = ";X2
0340 GO TO 150
0350 X1=(-B+SQR(D))/(2*A)
0360 X2=(-B-SQR(D))/(2*A)
0370 PRINT TAB(5);"REALES Y DIFERENTES"
0380 GO TO 300
0390 END
```

```
PSEUDO-CODE IS 0093 LONG
       DATA IS 0025 LONG
SEGMENT(002) IS 00CA LONG
```

```
********** BASIC COMPILATION SUMMARY **********
PROGRAM FILE NAME:(M781)CAMDR/CODIGO ON PACK.
NUMBER OF ERRORS DETECTED = 0.
NUMBER OF SEGMENTS = 1.  TOTAL SEGMENT SIZE = 388 WORDS.  CORE ESTIMATE = 12726 WORDS.  STACK ESTIMATE = 36
PROGRAM SIZE = 37 CARDS.  258 SYNTACTIC ITEMS, 20 DISK SEGMENTS.
COMPILATION TIME = 0 SECONDS ELAPSED, 1 SECONDS PROCESSING.
**********
```

ALGOL, PASCAL y PL/I

Estos lenguajes pertenecen a la familia de los lenguajes "estructurados", siendo el ALGOL el decano de la familia.

Las líneas 1010 a 1030 del programa en ALGOL son comentarios los cuales se identifican con la palabra COMMENT y terminan con un punto y coma (;), en los lenguajes estructurados todas las instrucciones terminan con punto y coma, los comentarios en lenguaje PASCAL deben estar comprendidos entre los caracteres (* y *), líneas 1100 a 1300 del ejemplo en dicho lenguaje y para PL/I entre los caracteres /* y */, los comentarios pueden contener una o más líneas y ser utilizados en cualquier parte del programa.

En los lenguajes estructurados generalmente se observa la siguiente configuración o estructura en este orden:

1. Encabezado del programa
2. Sección de declaraciones
3. Cuerpo del programa
4. Fin del programa

El encabezado del programa generalmente consta de una declaración la cual indica el inicio del programa, por ejemplo, en el lenguaje ALGOL el programa se inicia con la declaración BEGIN y en los lenguajes PASCAL y PL/I generalmente la declaración de inicio del programa va seguida de una lista de parámetros del programa, así tenemos para el lenguaje PASCAL, el encabezado se forma por la siguiente declaración:

PROGRAM nombre del programa (<lista de parámetros del programa>);

y para el lenguaje PL/I tenemos:

Nombre del programa: PROCEDURE OPTIONS (<parámetro>);

A continuación del encabezado del programa, se encuentra la sección de declaraciones del programa en la cual se declarán los elementos que se utilizarán en el programa tales como; variables y su tipo, funciones y subprogramas definidos por el usuario, archivos de datos tanto internos como externos, etc.

Enseguida de las declaraciones, se encuentra el cuerpo del programa el cual esta formado por todos aquellos elementos propios del lenguaje utilizado como son: proposiciones de asignación, de entrada, de salida, de comparación,iterativas, etc.

Finalizando al programa se encuentra una declaración la cual especifica el fin físico del programa, esta es para los lenguajes ALGOL y PASCAL la declaración END. y para el lenguaje PL/I es la siguiente:

END nombre del programa;

Las instrucciones de entrada y salida de información en estos lenguajes tienen una estructura parecida a la de FORTRAN, es decir; instrucción propiamente dicha seguida de un identificador de archivo, luego viene la forma de presentación de información y por último la lista de variables cuyo contenido se quiere leer o imprimir, de todos los lenguajes estructurados posiblemente el que cuenta con un número mayor de atributos u opciones para instrucciones de entrada/salida, proporcionando con esto una mayor versatilidad en su uso sea el lenguaje ALGOL.

La característica principal de los lenguajes estructurados es la existencia de bloques de instrucciones, como se puede ver en las instrucciones de comparación del ejemplo, líneas 1220 y 1270 del programa en ALGOL y líneas 2900 a 3500 del programa en PASCAL.

Los bloques estan comprendidos, por ejemplo, entre una instrucción BEGIN y una instrucción END para el lenguaje ALGOL y PASCAL y entre una instrucción DO y una instrucción END para el lenguaje PL/I. Por ejemplo si el valor absoluto de la variable D es menor o igual a 0.000001 (líneas 1290 y 3800 en ALGOL y PASCAL respectivamente), se ejecutará el bloque completo que comprende las instrucciones entre las líneas 1300 y 1330 para el lenguaje ALGOL y las líneas 3850 a 4100 para el lenguaje PASCAL, en caso contrario el control pasará a la línea que está inmediatamente después de la instrucción de fin de bloque (END) correspondiente. Desde luego puede haber unos bloques dentro de otros como en el ejemplo. Esta estructura facilita el diseño de programas, sobre todo cuando estos son grandes, y posteriormente, también facilita el mantenimiento a que esta sujeto todo programa.

De estos 3 lenguajes estructurados posiblemente el que - mayor número de funciones aritméticas y matemáticas posea -- sea ALGOL sin embargo el lenguaje PL/I posee funciones para manipular renglones o columnas de arreglos bidimensionales, - en forma compacta, por otra parte el lenguaje PASCAL las tie ne pero para definir y manejar conjuntos tal y como los cono cemos y manejamos en Algebra.

Quizá una característica importante de estos lenguajes - sea la de poder manejar el contenido interno de una unidad - de almacenamiento (BIT) o de un conjunto de estos (Byte) den tro de una palabra de computadora, dentro de las funciones - de biblioteca que estos lenguajes ofrecen, poseen una buena parte para el manejo de variables con contenido alfanumérico, lo que permite el manejo de información de este tipo en una forma rápida y concisa.

El decidir que lenguaje de programación es conveniente - utilizar depende de algunos factores, tales como: lenguajes disponibles en el sistema de computo, características del -- problema a resolver, por ejemplo, si en nuestro problema se - involucran principalmente la manipulación y operaciones con - matrices el lenguaje más adecuado sería el BASIC, por el contrario si lo que se requiere son una gran variedad de funciones matemáticas y subprogramas el lenguaje más apropiado sería el FORTRAN o el ALGOL.

```
BURROUGHS B7900 ALGOL COMPILER, VERSION 32.C?R.129,  THURSDAY, 11/25/82, 12:5? PM.

                 ( ? / ? ? ) C A N D E / C O D E 1 ? 6 ?   O N   P A C K
                 = = = = = = = = = = = = = = = = = = = = = = = = = = = =

BEGIN
COMMENT                                                                       1
        SOLUCION DE ECUACIONES CUADRATICAS
        F. MONROY M.           AGOSTO DE 1982.
;
DEFINE ENDIF= #,ENDDO= #;
FILE  ENT(KIND=REMOTE),SAL(KIND=REMOTE);

FORMAT FO1(/,X1,"LAS RAICES SON COMPLEJAS",//,X1,"X1 = ",F10.4,
           " + ",F10.4,//,X1,"X2 = ",F10.4," - ",F10.4),
       FO2(/,X1,"X1 = ",F10.4,//,X1,"X2 = ",F10.4);
REAL A,B,C,
     D,X1R,X1I,X2R;
WRITE(SAL,</,X1,"** RESULTADOS DEL PROGRAMA EN ALGOL **">);
WHILE NOT READ(ENT,/,A,B,C) DO
    BEGIN
    WRITE(SAL,</,X1,"A = ",F10.4," B = ",F10.4," C = ",F10.4>,A,B,C);  2
    IF A EQL 0 THEN
        WRITE(SAL,</,X1,"NO ES ECUACION CUADRATICA">)
    ELSE
        BEGIN
        D:=B*B-4*A*C;                                                         3
        IF ABS(D) GTR 0.000001 AND D LSS 0 THEN
            BEGIN
            X1R:=-B/(2*A);                                                    4
            X1I:=SQRT(ABS(D))/(2*A);
            WRITE(SAL,FO1,X1R,X1I,X1R,X1I);
            END
        ELSE                                                                  5
            BEGIN
            IF ABS(D) LEQ 0.000001 THEN                                       4
                BEGIN
                X1R:=X2R:=-B/(2*A);                                           5
                WRITE(SAL,</,X1,"LAS RAICES SON REALES E IGUALES">);
                END
            ELSE                                                              5
                BEGIN
                X1R:=(-B+SQRT(D))/(2*A);                                      5
                X2R:=(-B-SQRT(D))/(2*A);
                WRITE(SAL,</,X1,"LAS RAICES SON REALES Y DIFERENTES">);
                END
            ENDIF;                                                            5
            WRITE(SAL,FO2,X1R,X2R);
            END
        ;                                                                     4
        END
    ENDIF;                                                                    3
    END
ENDDO;                                                                        2
END.
**************************************************************************************
NUMBER OF ERRORS DETECTED = 0.

NUMBER OF SEGMENTS = 6.  TOTAL SEGMENT SIZE = 21? WORDS.  CORE ESTIMATE = 1031 WORDS.  STACK ESTIMATE = 16
PROGRAM SIZE = 49 CARDS, 558 SYNTACTIC ITEMS, 21 DISK SEGMENTS.
PROGRAM FILE NAME: (??????)CANDE/CODE1?? ON PACK. B7?00 CODE GENERATED.
COMPILATION TIME = ?.?22 SECONDS ELAPSED: ?.601 SECONDS PROCESSING; 0.331 SECONDS I/O.
**************************************************************************************
```

```
                                    ROYLO PASCAL COMPILER X1.14C.009

(02,0000) PROGRAM CUADRATICA(INPUT,OUTPUT);                                                     00001000
(02,0000) (*      SOLUCION DE ECUACIONES CUADRATICAS                          *)                00001100
(02,0000) (*      F. MORALES M.              AGOSTO DE 1982                   *)                00001200
(02,0000) (*                                                                  *)                00001300
(02,0000) VAR A,B,C,
(02,0037)     D,X1R,X11,X2R : REAL;                                                             00001400
(02,003E) BEGIN                                                                                 00001500
02:0000:0 WRITELN(' ** RESULTADOS DEL PROGRAMA EN PASCAL **');                                  00001600
02:0000:2 WHILE NOT EOF(INPUT) DO                                                               00001700
02:0000:4     BEGIN                                                                             00001800
02:0000:4     READLN(A,B,C);                                                                    00001900
02:0014:0     WRITELN;WRITELN(' A = ',A:10,' B = ',B:10,' C = ',C:10);                          00002000
02:0017:0     WRITELN;                                                                          00002100
02:0017:1     IF A = 0 THEN                                                                     00002200
02:0020:2         WRITELN(' NO ES ECUACION CUADRATICA')                                         00002300
02:0030:3     ELSE                                                                              00002400
02:0030:3         BEGIN                                                                         00002500
02:0034:0             WRITE(' LAS RAICES SON ');                                                00002600
02:0041:0             D:=B*B-4*A*C;                                                             00002700
02:0048:2             IF (ABS(D) > 0.000001) AND (D < 0) THEN                                   00002800
02:0065:3                 BEGIN                                                                 00002900
02:0065:3                 WRITE('COMPLEJAS');WRITELN;WRITELN;                                   00003000
02:0065:5                 X1R:=-B/(2*A);                                                        00003100
02:0065:5                 X11:=SQRT(-D)/(2*A);                                                  00003200
02:0065:5                 WRITELN(' X1 = ',X1R:10,' + ',X11:10,'I');WRITELN;                    00003300
02:0082:3                 WRITELN(' X2 = ',X1R:10,' - ',X11:10,'I')                             00003400
02:0082:3                 END                                                                   00003500
02:0074:2             ELSE                                                                      00003600
02:0074:5                 BEGIN                                                                 00003700
02:0074:5                 IF ABS(D) <= 0.000001 THEN                                            00003800
02:007A:5                     BEGIN                                                             00003900
02:007D:6                     WRITE('REALES E IGUALES');                                        00004000
02:0092:4                     X1R:=-B/(2*A);X2R:=X1R;                                           00004100
02:0098:5                     END                                                               00004200
02:0098:5                 ELSE                                                                  00004300
02:00EA:5                     BEGIN                                                             00004400
02:00EA:5                     WRITE('REALES Y DIFERENTES');                                     00004500
02:00F8:6                     X1R:=(-B+SQRT(D))/(2*A);                                          00004600
02:0118:1                     X2R:=(-B-SQRT(D))/(2*A);                                          00004700
02:0138:2                     END                                                               00004800
02:0191:2                 (*ENDIF*);                                                            00004900
02:0191:2                 WRITELN;WRITELN;WRITELN(' X1 = ',X1R:10);WRITELN;                     00005000
02:01A1:4                 WRITELN(' X2 = ',X2R:10);                                             00005100
02:01AC:3                 (*ENDIF*)                                                             00005200
02:01AC:3                 END                                                                   00005300
02:01AC:3             (*ENDIF*)                                                                 00005400
02:01AC:3             END                                                                       00005500
02:01AC:3         (*ENDIF*)                                                                     00005600
02:01AC:3     END                                                                               00005700
02:01AC:3 (*ENDWHILE*)                                                                          00005800
02:01AC:3 END.(* PROGRAMA CUADRATICA *)                                                         00005900

*****************************************************************************************************
NUMBER OF ERRORS DETECTED =    0.    2, TOTAL SEGMENT SIZE =   267 WORDS, CORE ESTIMATE =  5452 WORDS.
NUMBER OF PROGRAM SEGMENTS =   2, TOTAL STRING POOL =   36 WORDS, TOTAL DATA AREA = 6094 WORDS.
STACK SIZES: 01 =   12 WORDS, 02 =   17 WORDS, STRING POOL =   36 WORDS, TOTAL DATA AREA = 6094 WORDS.
PROGRAM SIZE =   51 WORDS,   39 STATEMENTS,   267 SYNTACTIC ITEMS,   18 DISK CONSTANT RECORDS.
COMPILATION TIMES =    0.8 CPU SECONDS,    0.7 I/O SECONDS,   12.4 ELAPSED SECONDS.
*****************************************************************************************************
```

CUADRO COMPARATIVO DE ALGUNOS LENGUAJES DE PROGRAMACION

	Uso Principal	Ventajas	Desventajas
FORTRAN	Ciencias e Ingeniería	Muy difundido, buen manejo de funciones de usuario	Ciertas limitaciones en el manejo de ar-- chivos e información alfanumérica
BASIC	Ciencias e Ingeniería	Fácil de aprender, disponible en micro computadoras de bajo costo.	Ciertas limitaciones en el manejo de ar-- chivos, en los nom-- bres de las variables y en la definición de subprogramas de usuario
PASCAL, ALGOL y PL/I	Uso general, ciencias, inge niería y admi- nistración	Tienen las ventajas de FORTRAN y algunas de BASIC, permiten el manejo de todo tipo de información y ade- más son estructurados	A excepción de PASCAL han tenido poca difu- sión, además de que - uno de ellos, ALGOL, solo se encuentra en unos cuantos sistemas

1.7 Generaciones de Computadoras

Dependiendo de los componentes electrónicos básicos utilizados por las computadoras podemos decir que han existido 4 generaciones de estas, para darnos una idea de las diferencias entre la primera y cuarta generación de computadoras, como la primera generación, utilizó tubos al vacío como componentes electrónicos básicos, esto dió como resultado que el costo, volúmen, consumo de fuerza, calentamiento, retardo de lógica y la cantidad de fallas en el sistema fueran muy elevadas comparadas con los sistemas de semiconductores utilizados hoy en día.

La evolución de la computación es sorprendente y contempla otros aspectos adicionales a los de la electrónica y componentes utilizados; aspectos como el tipo de dispositivos periféricos y el software, aunque es difícil encasillar en un esquema de cuatro etapas por el desarrollo casi contínuo de los sistemas de computación, a algunas inexactitudes en los años asociados a la aparición de cada etapa y a la existencia de algunas computadoras de transición adelantadas a su época y a la tecnología existente como fué la B-5500 de Burroughs (entre otros equipos).

Tomando en cuenta lo anterior se presenta una breve descripción de los aspectos que más destacan de cada una de las generaciones.

a) *Computadoras de la primera generación:*

Entrada al mercado:	1950 aproximadamente
Aplicación principal:	Instrumentos de cálculo
Tecnología utilizada:	Tubos de vacío Memoria de cilindro magnético
Unidades periféricas:	Lectoras y perforadoras de tarjetas y cinta de papel, equipo unitario, etc.
Sistema operativo:	No existía
Lenguajes de programación:	Lenguaje de máquina, ensambladores primitivos
Alfabeto:	Numérico

Administración:	Trivial, no se requería
Aspectos cuantitativos:	M. Central 1,000 a 8,000 palabras
	Proceso 10^4 ops/seg.
	Precio 100,000 a 2,5 millones US.
Modelos típicos:	IBM-650 Bendix-G15, - - Univac SS90, Bull-PT, IBM-709.

b) *Computadoras de la segunda generación:*

Entrada al mercado:	1960 aproximadamente
Aplicaciones principales:	Proceso de datos Instrumento de cálculo
Tecnología utilizada:	Transistores y ferritas
Unidades periféricas:	Lectoras y perforadoras de tarjetas, impresoras y cintas magnéticas.
Sistema operativo:	Rudimentario, controla periféricos, inicia y - termina tareas.
Lenguajes de programación:	Ensambladores y primeros compiladores (FORTRAN, - ALGOL).
Alfabéto:	Números y letras, algu-- nos caracteres especia-- les.
Facilidades adicionales:	Existencia de bibliote— cas.
Administración:	Primitiva, planeación de producción con procesos masivos.
Aspectos cuantitativos:	MC 8,000 a 32,000 pala-- bras.
	Procesadores 10^5 ops/seg.
	Precio 10^5 a 10^6 US.
Modelos típicos:	CDC-160, IBM-7090, IBM-1401, Burroughs-5500, - REA-305, Bendix-G20, -- CDC-3600.

c) *Computadoras en la tercera generación:*

Entrada al mercado:	Aproximadamente entre -- 1968 y 1970.
Aplicaciones principales:	Sistemas de Información
Tecnología utilizada:	Circuitos integrados (LSI), memoria de películas magnéticas.
Unidades periféricas:	Cintas y discos magnéticos terminales de video y teletipos.
Arquitectura:	Multiprogramación, multiproceso, sistemas de interrupción, optimización de código.
Lenguajes y facilidades de programación:	Lenguajes de alto nivel -- COBOL, PL, Bases de Datos (DMS).
Alfabeto:	Números, letras y caracteres especiales.
Sistema operativo:	Manejo de archivos, multiproceso, memoria dinámica, memoria virtual, etc.
Facilidades adicionales:	Edición y prueba interactiva de programas.
Administración:	Compleja y especializada.
Aspectos cuantitativos:	MC 64 a 256 K palabras. Procesador 10^6 ops/seg. Memoria secundaria 10^9 caracteres. Rango de precios 5×10^4 a 10^8 US.
Modelos típicos:	IBM-360, Burroughs-6700, PDP 10, PDP 11, Univac-1106, Cyber 170.

d) *Computadoras de la cuarta generación:*

Entrada al mercado:	Entre 1977 y 1981.
Aplicaciones principales:	Sistemas de comunicación, sistemas de información - para negocios pequeños, -

	uso personal.
Tecnologías utilizadas:	Micro-electrónica VLSI, - memorias-Mos (metal oxide sylicates).
Unidades periféricas:	Terminales inteligentes, discos y cintas magnéticas, equipos de graficación, lectores ópticos y digitalizadores.
Arquitectura:	Proceso distribuido, uso de microprocesadores.
Lenguajes y facilidades de programación:	Bases de datos distribuidas, lenguajes interactivos, descriptivos y gráficos.
Alfabeto:	Irrestricto, mayúsculas y minúsculas, símbolos matemáticos, alfabeto Arabe, Japonés, etc.
Sistema operativo:	Proceso sin interrupción, comunicación entre máquinas, rutinas de recuperación, etc.
Facilidades adicionales:	Metaprocesadores, correo electrónico, manejadores de texto.
Administración:	Muy simple para equipos personales. Muy complejo para redes de proceso distribuido.
Aspectos cuantitativos:	Memoria Central 64 K a 10^7 caracteres. Procesadores 10^7 ops/seg. Memoria secundaria 10^{10} - caracteres. Rango de precios 10^3 a -- 10^8 US.
Modelos típicos Grandes:	IBM-4330, Univac 1100, -- Burroughs B-6900, 7900.
Medianos:	Prime 550 MP 3100 VACS.
Pequeños:	Apple, TRS80

Las ventajas surgidas debido a los adelantos de la cuarta generación de computadoras son las siguientes:

a) Se autogobierna y maneja sin detenerse pasando de un trabajo a otro.

b) El operador interviene sólo cuando hay problemas.

c) La máquina se autodiagnostica, indica cual es la anomalía y se autocorrige.

d) Todas las operaciones del sistema se realizan simultaneamente.

e) Han evolucionado notablemente los lenguajes de comunicación con la máquina.

f) Gran autocontrol y autoverificación.

g) Multiprogramación y multiproceso

h) La velocidad entrada-proceso-salida se ha incrementado notablemente.

i) Gran desarrollo de accesorios para las computadoras.

Con el desarrollo de la micro-electrónica de la nueva tecnología de las comunicaciones y del reconocimiento de for

mas, logrados por la cuarta generación de computadoras, el camino para los robots y la automatización industrial a escala masiva ha quedado abierto.

Finalmente, la gran capacidad de almacenamiento y recuperación de información lograda con las computadoras actuales, no compite con la capacidad del hombre que a través de sus sentidos (especialmente el visual), está captando constantemente imagenes que implican volumenes impresionantes de información, que son de inmediato sintetizados en forma óptima para su almacenamiento, lo anterior ligado a la estructura y funcionamiento esencialmente distinto al de las computadoras actuales, permiten concluir el desarrollo futuro de otras formas de cómputo.

1.8 Uso de Microcomputadoras

Una microcomputadora es una versión miniaturizada y relativamente poco costosa de la computadora digital que en muchos casos no requiere de instalaciones especiales ni de grandes áreas para colocarlas y es capaz de ayudarnos a resolver problemas de razonable complejidad.

Podemos decir que una microcomputadora posee tres características: económicamente accesible para un individuo común, operada por una sola persona y fácil de usar para alguien no especializado. Puede pensarse que algunas calculadoras programables cumplen con las características anteriores, pero no es así, una diferencia consiste en que éstas sólo pueden manejar caracteres numéricos, pero no alfabéticos como las microcomputadoras, últimamente han salido al mercado calculadoras programables que aceptan también caracteres alfabéticos, así como microcomputadoras de tamaño semejante a las calculadoras programables, por lo que en el futuro muchas diferencias desaparecerán, en la actualidad puede considerarse que la unica diferencia es la capacidad de memoria siendo mucho más pequeña en las calculadoras programables.

Se puede decir que la característica más sobresaliente de las microcomputadoras es la comunicación tan directa que existe entre la máquina y el usuario, siendo este el más beneficiado, ya que el hecho de que para algunas aplicaciones se tenga que esperar sólo unos segundos o escasos minutos no es comparable con el tiempo que es necesario esperar operando con un gran sistema.

Las microcomputadoras como las conocemos en la actualidad, aparecieron en California, E.U. en el año de 1972 cuan-

do la compañía Intel introdujo su microprocesador 8008, y para el año de 1975 la compañía Mits introdujo la Altair 8800 basada en el microprocesador de Intel, que uno mismo podía ensamblar.

El mercado de las microcomputadoras realmente se creó, cuando las compañías Comodore International, Apple Computer y Tandy Corporation esta última conocida comunmente como Radio Shack introdujeron respectivamente la Pet 2002, Apple II y TRS-80, estas a diferencia de la Altair, no se requiere ser un experto para operarlas y sus capacidades son mucho mayores.

Como se vió anteriormente una computadora requiere de dispositivos para que se le comunique la información que deseamos que procese y las instrucciones que le indican como hacerlo o lo que es lo mismo, el programa, además se necesitan otros dispositivos que nos den a conocer los resultados de dicho proceso, dispositivos que llamamos de entrada/salida. El dispositivo de entrada que comunmente utilizan las microcomputadoras es un teclado muy semejante al de una máquina de escribir con el que se le alimenta directamente la información.

Cuando la cantidad de información que se le ha de suministrar es muy grande, al hacerlo a la velocidad con que teclea una persona puede tomar varias horas, durante las cuales la microcomputadora estaría ocupada recibiendo la información, para evitar esto, por lo general esta información previamente se almacena, ya sea en cintas magnéticas (como los cassettes musicales comunes) o en discos o diskettes, también similares a los discos musicales.

El medio más utilizado en las microcomputadoras era el cassette, este se podía conectar a una grabadora común, actualmente se emplea más el diskette o floppy disk, mucho más eficiente y del cual es posible recuperar información grabada, por ejemplo a la mitad del disco, sin tener que recorrerlo desde el principio.

Una vez grabada toda la información en la cinta o disco, puede alimentarse a la microcomputadora en cuestión de segundos, asimismo en estos dispositivos es posible grabar los resultados que entrega la microcomputadora durante el proceso de la información, convirtiéndose entonces en dispositivos de salida. Otro tipo de dispositivo mediante el cual la computadora proporciona resultados es la impresora (parecida a una máquina de escribir), que funciona a gran velocidad - -

(5000 caracteres por segundo aproximadamente) y emplea un rollo de papel contínuo, la impresora más usada en las microcomputadoras es la termoimpresora, que en tamaño es más pequeña que una máquina de escribir portátil.

Aún con todo, el dispositivo de salida más usado en las microcomputadoras, ya sea utilizándolo solo o en combinación con alguno de los mencionados con anterioridad, es el tubo de rayos catódicos (TRC), es decir la conocida pantalla de TV (en algunas microcomputadoras puede conectárseles un televisor común), este dispositivo, además de ofrecer los resultados en forma visual, tiene la ventaja de mostrar los datos desde que se teclean, esto permite su verificación y observar su estado en cualquier etapa del proceso, sirviendo como monitor de todo el proceso de cómputo. Otro dispositivo, también utilizado en las microcomputadoras es el graficador en el cual pueden obtenerse gráficas en color.

Generalmente los parámetros utilizados para comparar los diversos dispositivos son la velocidad con que entregan la información (en caracteres o líneas de caracteres por segundo) y su capacidad de almacenamiento de datos (en miles de bytes o kilobytes).

Lo más común es que tanto los datos como las instrucciones del programa se almacenen en la memoria central en forma de hileras de números binarios (unos y ceros), que representan cada uno de los caracteres alfabéticos o numéricos de los datos o del programa. Electrónicamente, los números binarios corresponden a un conjunto de interruptores, unos prendidos (representando un uno) y otros apagados representando un cero) en una secuencia determinada. El número de bits binarios empleado para representar un carácter alfabético (una sola letra) o un número (un solo número) de nuestro sistema decimal, es de ocho, a este conjunto de ocho bits se le denomina byte o "palabra de computadora".

Las memorias de las microcomputadoras consisten en pequeños microcircuitos integrados o chips de alrededor de 2 o 3 centímetros cuadrados en cuyo interior se encuentran cientos de dispositivos semiconductores. Existen básicamente dos tipos de memoria, la llamada ROM cuyos contenidos se leen pero no se borran, y la RAM cuyos contenidos pueden leerse y también borrarse para volver a ser grabados, una configuración típica de una microcomputadora es de entre 12 y 16 Kb (Kilobytes o miles de bytes) en ROM, y hasta 128 Kb en RAM.

La unidad de control y proceso utilizada, por la mayoría

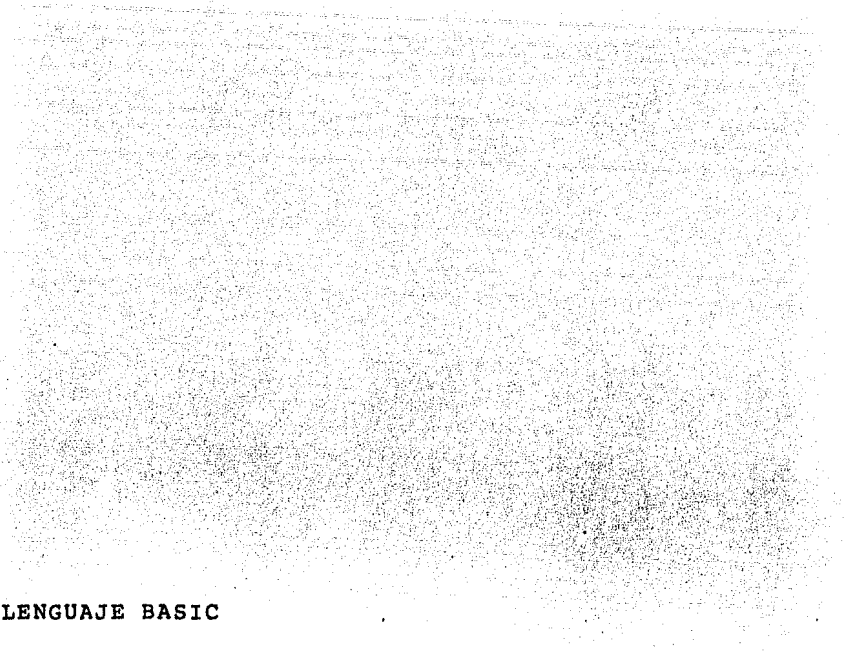
de las microcomputadoras es el microprocesador, que generalmente se encuentra contenido en un solo microcircuito e incluye también a la unidad aritmética-lógica.

Como siempre ha sucedido en la industria de la computación, el área de software (paquetes de aplicación) es la más descuidada por los proveedores de microcomputadoras, sus productos de software generalmente están atrasados uno o más años con respecto a sus productos de hard-ware (las máquinas y accesorios), gran parte del software disponible en microcomputadoras es para aficionados (programas para juegos, por ejemplo).

El lenguaje generalmente utilizado por las microcomputadoras es el BASIC o Extended BASIC, aunque algunos proveedores tienen ya disponibles el lenguaje PASCAL y el lenguaje FORTRAN.

Para algunas aplicaciones, las microcomputadoras son tan capaces como los grandes sistemas y en algunos casos con ventajas adicionales, tales como excelentes pantallas y como ya se mencionó una extraordinaria interacción con el usuario.

Si bien es cierto que tienen algunas deficiencias en cuanto a la calidad y la reducida variedad de paquetes de programas para aplicaciones específicas, así como una velocidad y capacidad relativamente limitadas, pero por otro lado, el precio de los elementos y de las microcomputadoras en si "sigue" disminuyendo, mientras que sus capacidades van aumentando rápidamente.



2. EL LENGUAJE BASIC

2.1 Elementos del Lenguaje BASIC

Los elementos básicos de este lenguaje lo constituyen las constantes, variables, arreglos, expresiones y proposiciones. Un programa consiste en un grupo de proposiciones las cuales están formadas por expresiones y operadores, que siguen reglas previamente establecidas semejantes a las reglas gramaticales, a continuación se presentan los elementos básicos del lenguaje BASIC.

NUMERACION DE RENGLONES

Cada renglón de un programa en lenguaje BASIC se debe numerar, dicho número se debe de colocar al principio de cada renglón e identifica a este renglón, ningún número de renglón debe repetirse en un mismo programa, además estos números de proposición no deben de tener signo.

Antes de proceder a la ejecución de un programa la computadora reordena las proposiciones o instrucciones de acuerdo a los números de renglón correspondiente en orden ascendente, por esta razón pueden introducirse las proposiciones o instrucciones en cualquier orden, pudiendo modificar o añadir proposiciones en cualquier posición del programa, es por ello que se recomienda no numerar los renglones en forma con

secutiva ya que para introducir un nuevo renglón intercalado, tendríamos que renumerar las proposiciones restantes del programa.

CONSTANTES

Una constante es un número cualquiera que no cambia de valor al utilizarse de una ejecución a la siguiente ejecución del programa, estas constantes pueden ser positivas o negativas una constante puede no llevar punto decimal a menos de -- que sea necesario, el signo más es opcional para números positivos, la longitud y magnitud de una constante dependen del - tipo de sistema de cómputo, en la generalidad de los sistemas la longitud máxima es de 9 dígitos y la magnitud puede ser -- cero o estar comprendida en el intervalo entre 10^{-75} y 10^{75} en las constantes no se permite el uso de comas.

Para números muy grandes o muy pequeños, se puede utilizar la nomenclatura exponencial, que consiste en un número - seguido de la letra E y de una constante sin punto decimal - positiva o negativa de uno o dos dígitos numéricos que indican la potencia por la que se multiplica el número que procede a la letra E, son ejemplos de constantes:

Válidas	*No válidas*
0	100,05 (no se permite coma)
+15	14 3589467678, (muy grande)
-24	0,35874721075345 (muy pequeño)
-175.26	+- 785.39 (no se permiten 2 signos)
+4988.739	E-3 (falta la base)
-1.5 E-14	3,785 E-34 (no se permite coma)

VARIABLES

Las variables indican cualquier cantidad a la que se puede hacer referencia y puede tener distintos valores durante la ejecución de un programa, esta variable se indica por - - cualquier letra del alfabeto, seguida por un solo dígito, entre 0 y 9 y por lo tanto podemos tener un total de 286 nombres para identificar variables, los valores y rangos de variación de las variables son los mismos que los de las constantes, todas las variables se inicializan automáticamente - con valor cero, sin embargo pueden existir excepciones según el tipo de computadora, son ejemplos de variables:

Válidas	*No válidas*
X	5B (no empieza con letra)
A1	BC (dos letras)
YO	B33 (dos números)
Y5	T* (el segundo carácter no numérico)
T4	.5 (el primer carácter no es letra)

Existen variables alfanuméricas, que permiten el manejo de datos alfanuméricos, como nombres, direcciones y cualquier otro tipo de información estas variables se indican por una sola letra del alfabeto seguida por le símbolo de pesos, en algunos sistemas después de la letra puede ir un dígito numérico, son ejemplos de variables alfanuméricas:

Válidas	*No válidas*
T$	T$$ (repetición del signo $)
A$	AA$ (repetición de letra)
A5$	5$ (no empieza con letra)
O$	A.$ (no se permite el punto)

EXPRESIONES

Una expresión se define como una constante, una variable, una función o cualquier combinación de estos elementos, separados por operadores o paréntesis, que cumplen con ciertas reglas para formar expresiones, los símbolos de los operadores aritméticos se presentan a continuación.

Símbolo	*Operación*
+	Adición
-	Sustracción
*	Multiplicación
/	División
↑ ó **	Exponenciación

En una expresión no se permite que dos operadores se encuentren uno junto al otro, considerándose el caso de exponenciación (**) como un solo símbolo, se permite el uso de paréntesis para establecer agrupamientos de la misma manera

que en la notación matemática ordinaria, cuando el orden de las operaciones no se indica completamente con el uso de paréntesis, el orden de ejecución es el siguiente: en primer lugar se efectúan las exponenciaciones, después las multiplicaciones y divisiones, y finalmente las sumas y restas, sin embargo en el caso de que el nivel de operaciones sea el mismo, como por ejemplo sumas y restas, o multiplicaciones y divisiones se procede de izquierda a derecha en el orden de ejecución, por lo que respecta al uso de paréntesis, el orden de ejecución es del paréntesis más interior hacia el paréntesis más exterior.

En la operación de exponenciación existe la regla de que una expresión cualquiera se puede elevar a un exponente real, positivo o negativo, pero no se permite que una cantidad negativa se eleve a un exponente real, ya que en general el resultado es complejo, son ejemplos de expresiones:

Expresión Matemática	*Expresión en BASIC*
$b-2$	B-2
ab	A*B
$d/(-c)$	D/(-C)
$-a^3$	-A↑3
$(a+b)c$	(A+B)*C
mb/aj	(M*B)/(A*J)
c^{e+5x}	C**(E+5*X)

FUNCIONES PROPORCIONADAS

Todos los sistemas tienen disponible una serie de funciones matemáticas de uso frecuente, y a las cuales se puede hacer referencia fácilmente, los nombres de las funciones que el sistema BASIC ofrece consisten de tres letras, seguidas de un argumento encerrado dentro de paréntesis, dicho argumento puede ser una constante, variable o expresión, a continuación se presentan algunas funciones en BASIC.

Función Matemática	*Nombre*	*Ejemplo*
Seno de un ángulo (radianes)	SIN	SIN (T)
Coseno de un ángulo (radianes)	COS	COS (A+B)
Tangente de un ángulo (radiantes)	TAN	TAN (X1/Y2)
Angulo tangente (radiantes)	ATN	ATN (3.14-X)
Exponencial	EXP	EXP (5*X-Y)
Raíz Cuadrada	SQR	SQR (X*X+R*Y)
Valor Absoluto	ABS	ABS (R)
Logaritmo Natural	LOG	LOG (5.2-X)

Algunas otras funciones son:

INT que calcula el máximo entero que no excede el valor del argumento, por ejemplo:

INT(5.8) = 5 e INT(-2.7) = -3

SGN evalúa el signo del argumento, teniéndose:

SGN(X) =-1 para X<O
SGN(X) = 0 para X=O
SGN(X) =+1 para X>O

RND genera números pseudoaleatorios

PROPOSICIONES

Las proposiciones o instrucciones que se usan en el lenguaje BASIC pueden ser de los tipos siguientes: aritmética, de control, entrada y/o salida, de especificación y de sub-programa.

PROPOSICION ARITMETICA LET.

La proposición LET es una proposición aritmética que se utiliza para especificar las operaciones de cálculo que hay que efectuar y es de la forma:

LET R=E

En donde R representa una variable y E indica una expresión aritmética, en estas proposiciones el signo de igualdad significa reemplazo o sustitución y no debe interpretarse como el signo igual usado en la notación matemática ordinaria.

Por lo tanto se efectúan todas las operaciones indicadas en la expresión E y el resultado final se sustituye o almacena en la variable R, por ejemplo:

Proposicion Aritmética		*Significado*
50	LET C= B/7.5	El cociente de B sobre 7.5 es almacenado en C.
75	LET D=A+B+C	La suma de los valores A,B y C es guardada en D.
90	LET I+I-1	Al valor de I se le resta - uno y el resultado de la -- resta es el nuevo valor de I.
115	LET Y=5 * COS(X)	El resultado del producto - de 5 por el COS(X) se guarda en Y.

En la mayoría de los sistemas se puede omitir la palabra LET y las proposiciones aritméticas tienen la forma R=E, en los ejemplos anteriores el número que aparece al principio - de cada proposición es el número de renglón.

PROPOSICION REM

Es una proposición de documentación, que permite la in-- serción de comentarios dentro de un programa, tiene la forma general:

REM comentario

En donde el comentario no tiene ningún efecto en la ejecución del programa, sirviendo para efectos de identifica--- ción de parte del programa o para cualquier uso que se desee, por ejemplo:

```
 125 REM    "SOLUCION DE ARMADURAS PLANAS"
 130 REM    ------ OBTENCION DE LOS DESPLAZAMIENTOS
5000 REM    --VERIFICACION-----DEL EQUILIBRIO
```

Los comentarios pueden intercalarse en cualquier parte - del programa y no hay limitación en cuanto al número de co-- mentarios que pueden utilizarse en un programa. En algunos sistemas se puede insertar un comentario después de una proposición, lo anterior se logra poniendo después de la terminación de la proposición un apostrofe (') y en seguida el comentario, por ejemplo:

```
900  I=J*3-N1  'INDICE QUE DEFINE AL RENGLON
950  LET J= I* 2-1  'INDICE QUE DEFINE LA COLUMNA
```

2.2 Proposiciones de entrada/salida

Las proposiciones de entrada/salida controlan la transmisión de información entre la computadora y las unidades de entrada/salida. Esta información se transmite en grupos de datos o archivos comprendidos de uno o varios registros, cada uno de los cuales esta formado por elementos, son ejemplos de registros: renglones impresos, tarjetas o cinta de papel perforadas, o las imágenes de esta información en disco o cinta magnética, como proposiciones de entrada/salida se tienen las siguientes:

PROPOSICIONES READ Y DATA

Las proposiciones READ y DATA asignan los valores numéricos de las variables, que se utilizan en el programa y no puede usarse una proposición sin la otra, tienen la forma general.

```
READ    Lista de variables
DATA    Lista de constantes
```

En donde la lista de variables en la proposición READ indica los nombres de las variables de los datos de entrada, separados por comas y a las que se especificarán en ese orden, los valores indicados en la lista de constantes de la proposición DATA. La computadora antes de ejecutar un programa, organiza todas las proposiciones DATA, de acuerdo a su número de renglón en un solo bloque de datos, por ejemplo, son equivalentes:

```
155  DATA  5,3          175  DATA  5,3,7,6,9
160  DATA  7,6,9
```

El bloque de datos proporciona las constantes en el orden correspondiente y según las variables que indican las proposiciones READ durante la ejecución del programa. Las proposiciones DATA pueden encontrarse en cualquier parte del programa, prestándose cuidado de que se encuentren en el orden correcto. Por lo general las proposiciones DATA se agrupan inmediatamente antes de la proposición END, de manera que se localicen en un solo lugar y con la ventaja de reducir

a un mínimo de renumeración de renglones cuando se necesitan añadir datos aun programa por medio de proposicones DATA. -- Por ejemplo son equivalentes.

```
15  READ  A,B,X,Y          15  READ  A,B
          :                 20  READ  X
          :                 25  READ  Y
          :                       :
50  DATA  7                       :
55  DATA  3,5,4             60  DATA  7,3,5,4
```

Lo anterior causa la asignación a las variables A,B,X,Y, los valores 7,3,5,4, respectivamente.

PROPOSICION INPUT

Esta proposición permite la entrada de datos mientras se está corriendo el programa, la proposición tiene la forma general:

INPUT Lista de variables

en donde la lista de variables indica los nombres de las variables cuyos valores se introducen por medio del teclado de la terminal, por ejemplo al ejecutarse.

```
15  INPUT  X1, Y, Z
```

la computadora imprime el signo de interrogación, y espera - a que el usuario introduzca los valores numéricos respecti-- vos de las variables X1, Y, Z, separando cada uno de estos - números por una coma a continuación se oprime la tecla - - - RETURN y la computadora prosigue con el resto del programa. Es evidente, que esta proposición no es ventajosa para el caso en que se desea introducir una gran cantidad de datos, -- debido a la dificultad de verificar posibles errores al in-- troducirlos y por el tiempo excesivo de máquina que puede requerirse.

PROPOSICION PRINT

Esta proposición permite la comunicación entre la máquina y el usuario, mediante la impresión de información, y tiene la forma general:

PRINT Lista de elementos

en donde la lista de elementos incluye valores de variables y expresiones, resultados de cálculos numéricos, impresión de - mensajes o títulos, saltar renglones en blanco para el espa— ciamiento vertical de la información, así como combinaciones de los elementos anteriores, por ejemplo:

```
40  PRINT    X1, X2,  5*X1*X2, SQR(X1/X2)
```

produce la impresión de los valores de las variables X1,X2 y de los valores de las expresiones $5 \times X1 \times X2$ y $(X1 \div X2)^{1/2}$ - respectivamente en un mismo renglón.

```
80  PRINT   "VALOR X", "VALOR Y"
```

produce la impresión en un solo renglón, de la información -- comprendida entre comillas. Cuando los elementos se separan por comas, como en los dos ejemplos anteriores, la impresión se efectúa automáticamente, dividiendo el renglón en cinco - campos o zonas, procediendo de izquierda a derecha, cada zo- na tiene asignadas 15 columnas o caracteres, para un total - de 75 caracteres por renglón, por ejemplo, la proposición:

```
40  PRINT   "EL CUBO DE ";X;"ES"; X↑3
```

produce para X = 2, la impresión

```
EL CUBO DE 2 ES 8
```

la proposición sin lista de elementos,

```
10  PRINT
```

produce el salto de un renglón en blanco, es común utilizar las proposiciones PRINT e INPUT para identificar la informa- ción de entrada, por ejemplo:

```
20  PRINT  "LA CONSTANTE ES IGUAL A"
25  INPUT  C1
```

produce la impresión de

```
LA CONSTANTE ES IGUAL A
?
```

en donde el valor de la constante C1 se indicará a continuación del signo de interrogación, algunos sistemas aceptan -- la impresión de información en la proposición INPUT, por - - ejemplo:

```
70  INPUT  "LA CONSTANTE ES IGUAL A";C1
```

causaría la impresión de:

```
LA CONSTANTE ES IGUAL A ?
```

La notación E se usa también para la impresión de valores numéricos muy grandes o muy pequeños y su rango de aplicación depende del sistema de cómputo.

FUNCION DE IMPRESION TAB

Esta función nos permite especificar la posición de impresión en la cual se iniciará la transferencia de datos, tiene la forma

```
TAB  (expresión)
```

en donde la expresión se calcula y es utilizada solo la parte entera de dicha expresión para definir la posición en la que se inicia la impresión, esta función se utiliza con la - instrucción PRINT por ejemplo:

```
70  PRINT    X1; TAB(15); X2; TAB(30+J); X3
```

produce que una vez impreso el valor de X1, la impresión se continúe en la columna 15 para la variable X2, en la columna 35 sí J vale 5 para la variable X3. En algunos sistemas se permite el uso de comas después de la función TAB, la especificación de la función TAB se ignora cuando el argumento es menor que la posición de impresión en la que se encuentra.

PROPOSICION PRINT USING

Con esta proposición se puede obtener la impresión de -- información con un formato especificado, que se indica con - una proposición de imagen o máscara. Estas proposiciones -- tienen la forma:

```
PRINT USING  n, Lista
n:  máscara o imagen
```

en donde *n* es el número de renglón de la proposición asocia da de máscara o imagen

Lista es la identificación de las constantes, números, variables (numéricas y alfanuméricas) y funcio-- nes, separados por comas, cuya impresión se desea.

mascara o imagen Define el formato de impresión mediante el uso - de caracteres de control y constantes de impre-- sión.

La proposición de imagen con número de renglón *n* puede - encontrarse en cualquier parte del programa, no siendo necesario que se encuentre junto a la proposición PRINT USING - correspondiente. Por ejemplo, las proposiciones

```
 30  LET  A=5
 40  PRINT  USING  100,  A,A*A,A↑3
100  ' EL CUADRADO DE ### ES ###.## Y EL
     CUBO ES ##.##↑↑↑↑
```

Producirán la impresión siguiente

```
EL CUADRADO DE 5 ES 25.00 Y EL CUBO ES 1.25 E+02
```

El caracter de control para especificaciones numéricas - es el símbolo # y se utiliza en las conversiones enteras, de cimales y exponenciales. Para números enteros se usan uno o más caracteres # sin punto decimal y el valor del número se trunca, imprimiéndose solo la parte entera. En la especificación real o decimal se utilizan uno o más caracteres # con un punto decimal en cualquier posición, aunque por lo menos un caracter # debe preceder al punto decimal, en esta especi ficación el último dígito del formato se redondea.

En el caso exponencial se cumple la especificación anterior y se añaden cuatro caracteres ↑ que se utilizan para la letra E, el signo y una constante de uno o dos dígitos, que indica la potencia de 10 por la que se multiplica el número que precede a la letra E. Todas las especificaciones de va lores numéricos se justifican a la derecha y en todos los - casos se reserva el primer caracter para el signo del número. En algunos sistemas se amplian los campos a la derecha cuando el número correspondiente es demasiado grande y en -- general, cuando se excede la capacidad de un campo especifica do, se imprimen asteriscos en los campos cuya capacidad se - ha excedido.

La información alfanumérica se puede manipular utilizando campos limitados por comillas, que son los caracteres de control para especificaciones alfanuméricas. La longitud del campo comprende los espacios y/o caracteres dentro de las comillas, incluyendo también los espacios que usan las comillas, por lo que la información que se sustituye tendrá dos o más caracteres. Los campos alfanuméricos se justifican a la izquierda con espacios en blanco para las columnas del campo que no se usen y truncando los caracteres que exceden la longitud del campo.

Por ejemplo, las proposiciones

```
10  READ N$
20  PRINT  USING 30; N$
30  :      "NOMBRE
40  DATA  HERNANDEZ
```

producen la lectura de la variable N$, que es el nombre HERNANDEZ, cuya impresión se pide en el renglon 20, con el formato que indica el renglon 30 que permite un máximo de 8 caracteres (6 para NOMBRE y 2 para comillas). La información dentro de comillas se sustituye por la variable alfanumérica HERNANDEZ, que tiene 9 caracteres, por lo que se pierde el último, y la impresión que proporciona la máquina es:

```
HERNANDE
```

sería

```
GOMEZ
```

Si el renglon 40 fuera

```
40  DATA  GOMEZ
```

Algunos sistemas tienen el caracter de control ", que se sustituye por el primer caracter de la información alfanumérica, independientemente del número de caracteres que tenga la variable alfanumérica correspondiente. Por ejemplo, si en los casos antes considerados, se tuvieran las proposiciones:

```
20  PRINT  USING 30, N$, "MB"
30  : "NOMBRE"    '
```

se producirían las impresiones

```
HERNANDE    M
GOMEZ       M
```

En el caso de que la proposición PRINT USING contenga menos elementos que las especificaciones de conversión de la proposición de imagen correspondiente, las especificaciones que sobran se ignoran y si contiene más elementos que las especificaciones indicadas, estas se reutilizan en nuevos renglones hasta completar la impresión de los elementos indicados en la lista de la proposición PRINT USING.

PROPOSICION RESTORE

Esta proposición permite en un programa, la lectura de un mismo grupo de datos, tantas veces como sea necesario. La forma de esta proposición es

```
RESTORE
```

y cada vez que se usa ocasiona que el indicador del bloque de datos se reincialice al primer elemento de la primera proposición de datos, es decir a la primera proposición DATA del programa. Por ejemplo, las proposiciones:

```
 50  READ  A,B,C
      "
      "
      "
 70  RESTORE
 90  READ  X,Y,Z
      "
      "
      "
110  DATA  5,3
120  DATA  13.25
```

produce dos veces en el programa, la lectura del mismo grupo de datos, el que aparece una sola vez en las proposiciones de los renglones 110 y 120.

2.3 Proposiciones de Control

Las proposiciones de control se utilizan para especificar el orden de ejecución de las proposiciones de un progra-

ma. En un programa las instrucciones se ejecutan en orden progresivo indicado por los números de renglón o de proposición, sin embargo se puede alterar el orden de ejecución de las proposiciones utilizando proposiciones de control, que en general transfieren el control a una parte específica dentro del programa, el sistema BASIC proporciona las siguientes proposiciones de control.

PROPOSICION GO TO

Esta proposición tiene la forma:

GO TO n

en donde n es el número de renglón de una proposición existente en cualquier parte del programa. Al encontrarse la proposición anterior, la siguiente proposición que se ejecuta es la que tiene el número de renglón n. Por ejemplo:

```
 .        .        .
100      GO TO    140
110       .
120       .
130       .
140       .
```

al llegar el control a la instrucción 100, la proposición que se ejecutará después de esta, será la 140, no ejecutándose las instrucciones de los renglones 110, 120 y 130.

PROPOSICION ON-GO TO

Con esta proposición se especifica también la proposición que se ejecuta a continuación, tiene la forma general.

ON a GØ TO $n_1, n_2, \ldots\ldots, n_m$

en donde a representa una expresión numérica y $n_1, n_2, \ldots\ldots, n_m$ son número de renglones a los que se puede hacer la transferencia, los cuales se encuentran en alguna parte del programa. Si el valor de a es 1, se hace la transferencia al renglón n_1 y si es m se hace la transferencia al renglón n_m, por lo tanto, lo que se considera es la parte entera de la expresión a, que debe estar entre los valores 1 a m, si -

no se encuentra en este interválo se imprime un mensaje indicando el error, en esta proposición se pueden incluir tantos números de renglón como espacio disponible se tenga en dicho renglón. Por ejemplo:

```
100  ON  B*B-4*A*C   GO  TO  50,110,170
```

transferirá el control a los números de renglón 50, 110 ó -- 170 dependiendo de si B*B-4*A*C tiene, respectivamente, los valores 1,2 ó 3 si el valor fuera 3.9, la parte entera es 3 y el control se transfiere al renglón con el número 170.

PROPOSICION IF-THEN

Con esta proposición se puede alterar el orden de ejecución, dependiendo de si cierta relación entre expresiones, - es verdadera o falsa, la forma de esta proposición es:

IF *a* operador de relación *b* THEN *n*

en donde *a* y *b* son expresiones válidas relacionadas por un operador de relación, si la relación entre las expresiones - es verdadera, el control se transfiere a la proposición con número de renglón *n* y si es falsa se hace transferencia al - renglón que sigue inmediatamente a esta proposición en el -- programa. Los operadores de relación son seis y para representarlos se usan los siguientes símbolos.

Símbolo	*Significado*	*Ejemplo*
=	igual que	A = B
<	menor que	A < x
>	mayor que	0.1>B*B-4*A*C
<=	menor o igual a	5*A <= 3-2*B
>=	mayor o igual a	5>=5*COS(X)
<>	no igual a	A<>A+B-C

por ejemplo, en

```
25   IF   X+Y <= 3   THEN 85
30   .
35   .
```

si X+Y suman 3 o menos, la relación es verdadera y el con--- trol se transfiere al renglón número 85, sin embargo, si X+Y suma más de 3, la relación es falsa y el control se transfie

re a la proposición con número de renglón 30, que es la inmediatamente a continuación de la proposición IF-THEN.

PROPOSICION STOP

Esta proposición tiene la forma.

STOP

y el resultado de la ejecución de dicha instrucción es la -- terminación de la ejecución del programa, por ejemplo:

490 STOP

termina la ejecución del programa en la proposición de número de renglón 490.

PROPOSICION END

Tiene la forma general siguiente:

END

e indica el final de un programa, siendo físicamente la última proposición de todo programa, por lo tanto esta proposición debe de tener el número de renglón mayor de todas las - proposiciones en un programa.

PROPOSICION CHAIN

Esta proposición permite parar la ejecución del programa que esté corriendo, iniciando la compilación y ejecución de otro programa que se indica en esta proposición, tiene la - forma general.

CHAIN nombre de programa, *n*

en donde nombre de programa identifica al nuevo programa cuya ejecución se inicia, pudiéndose aplicar a programas de -- biblioteca, la letra *n* especifica el número de renglón del nuevo programa, en el cual se inicia la ejecución, es opcional y si no aparece, la ejecución del programa se inicia en la primera proposición ejecutable de dicho programa, los nombres de programas se forman con un máximo de seis caracteres alfanuméricos, requiriéndose en la mayoría de los sistemas - que el primer caracter del nombre del programa sea una letra,

por ejemplo:

```
20  CHAIN  RAICES
90  CHAIN  IMPRIM, 500
```

los nombres de los programas, cuya ejecución se inicia son - RAICES e IMPRIM, respectivamente, iniciándose la ejecución - en el renglón 500 del segundo programa.

2.4 Instrucción Iterativa

Esta instrucción permite indicar la ejecución repetida de una serie de instrucciones, las cuales se denominan rango o ciclo de la iteración, pudiéndose cambiar el valor de cier-- tas variables en cada iteración. Forman a esta instrucción dos proposiciones, FOR y NEXT, de estas la proposición FOR - es la primera del rango o ciclo y la proposición NEXT la - - última, la forma general de estas proposiciones es:

```
FOR   R = a  TO  b  STEP  c
    .
    .
    .
NEXT  R
```

en donde R representa una variable, generalmente llamada - - índice, que debe ser la misma para ambas proposiciones FOR y NEXT y *a*,*b*,c indican cualquier expresión válida en BASIC. - La primera vez que se ejecutan las instrucciones, la varia-- ble R tiene el valor *a*, y para cada repetición el índice R se recalcula incrementándole al valor de *a* acumulativamente el valor de c, hasta obtener el valor de la expresión *b* el - incremento c puede ser negativo o positivo; si es positivo - las iteraciones se ejecutan hasta el valor máximo del índice que no exceda (menor o igual) al valor de la expresión *b* y si el incremento c es negativo las iteraciones se ejecutan has- ta que el valor último del índice sea mayor o igual al de la expresión *b*. Después de ejecutarse el ciclo o iteración por última vez, el control se transfiere a la proposición inme-- diatamente a continuación de la proposición NEXT. En el ca- so en que el incremento c sea igual a +1, se puede omitir la parte STEP c, escribiéndose la proposición FOR como sigue:

```
FOR  R = a   TO  b
```

En el caso en que el valor inicial de la expresión *a* sea mayor que el valor final de la expresión *b* (o menor si el -- incremento c es negativo), el ciclo no se ejecuta y el con-- trol se transfiere inmediatamente a la proposición que sigue a la proposición NEXT, son ejemplos:

```
 50 FOR N=1  TO  8  STEP  3      El ciclo se repite
     .                           para N=1,4,7
     .
 90 NEXT N
 60 FOR  A=10  TO 3  STEP - 2    El ciclo se repite
     .                           para A=10,8,6,4
     .
100 NEXT A
150 FOR   B1=0  TO  4            El ciclo se repite
     .                           para B1 = 0,1,2,3,4
     .
170 NEXT  B1
315 FOR  J=SQR(7+X)  TO  Y*Z1  STEP  (X-Y)**2
     .
     .
385 NEXT J
```

En algunos sistemas, estos ciclos o iteraciones se ejecutan evaluando la expresión *a*, que se asigna a la variable o índice, recorriéndose el ciclo hasta la proposición NEXT; - en este instante se evalúa la expresión c, que se añade a la expresión *a* para el nuevo valor del índice, comparándose con el valor que se calcula de la expresión *b*. Si el valor del índice no excede al valor de *b*, se repite el ciclo con el -- nuevo valor del índice, recalculándose las expresiones *b* y c en cada iteración. Por lo tanto, la expresión *a* se calcula una sola vez, pero las expresiones *b* y c se evalúan en todas y cada una de las repeticiones del ciclo.

Al utilizar las proposiciones FOR y NEXT, se debe tener en cuenta que el valor del índice no debe cambiarse con otra proposición dentro del rango o ciclo. Sin embargo, la forma de ejecución de las iteraciones en algunos sistemas permite que las expresiones *a*,*b* y c contengan valores que pueden cambiar durante la ejecución de las iteraciones en el ciclo o - rango. Existen sistemas en los que las expresiones *a*,*b* y c - se calculan una sola vez al principio de la primera ejecu--- ción, no recalculándose en las ejecuciones posteriores de -- los ciclos.

Dentro del rango de proposiciones FOR-NEXT, pueden existir otras proposiciones FOR-NEXT, consituyendo una nidificación o anidación de proposiciones FOR-NEXT, con la condición de que un rango interior se encuentre totalmente dentro de un rango exterior. Se permite la transferencia fuera del rango de las proposiciones FOR-NEXT en cualquier instante. Sin --

embargo, lo que la mayoría de los sistemas no permite es la - transferencia de control fuera de un rango y el regreso poste rior del control dentro de este rango, especialmente cuando - se cambian los valores de R, *a*, *b* y *c*. Un caso especial lo - constituyen los subprogramas, permitiéndose el uso y regreso de un subprograma al rango de proposiciones FOR-NEXT, en una nidificación de proposiciones FOR-NEXT. Es de mencionarse -- que no se permite el uso de la misma variable R (índice) en - distintos rangos de una nidificación de proposiciones - - - - FOR-NEXT.

2.5 Manejo de Arreglos

Comunmente pueden presentarse o nosotros podemos formar — conjuntos de valores ordenados de cierta manera, comunmente - designamos como arreglos a estos conjuntos. En algunos casos puede ser adecuado manipular a cada uno de ellos como un todo, dichos arreglos estan constituídos por elementos, a los cua-- les también podemos hacer referencia a cada uno de ellos por separado.

VARIABLES CON SUBINDICE

Una variable con subíndice se forma en BASIC con el nombre de un arreglo, seguido de paréntesis, en los que se comprenden uno o dos subíndices separados por una coma. El nombre del arreglo, que es una variable, se forma de la misma manera que esta, es decir con el nombre de la variable con subíndice se hace referencia a un arreglo formado por un conjunto ordenado de cantidades, donde el o los subíndices indican una cierta posición que un elemento contenido en dicho arre-- glo ocupa dentro del mismo. El número de subíndices representa la dimensión del arreglo, los arreglos de una dimensión se denominan arreglos líneales o vectores y los de dos dimensiones se designan matríces, los subíndices pueden ser constan-- tes, variables o expresiones.

Los elementos de un arreglo se numeran en forma consecutiva, empezando en general con el número 1. Existen sistemas - que no aceptan los subíndices cero o negativos, sin embargo, hay sistemas que sí aceptan el subíndice cero.

Se permite que el nombre de un arreglo se use también como nombre de una variable sin subíndice, en un mismo programa. Por lo tanto C y C(I) representan elementos diferentes.

PROPOSICION DE ESPECIFICACION DIM

La proposición DIM es una proposición de especificación - que proporciona información a la máquina con respecto a la - naturaleza y características de determinados arreglos, en la mayoría de los sistemas debe preceder a la primera proposi-- ción ejecutable del programa, esta proposición tiene la for-- ma general:

DIM $a(K_1)$, $b(K_2)$, ... $c(K_3)$

en donde a, b, c son nombres de arreglos y K_1, K_2, K_3 se - - componen de una o dos constantes enteras sin signo, separadas por coma, y representando cada una el valor máximo reservado para cada subíndice del arreglo.

Cuando un arreglo no aparece indicado en la proposición - DIM la máquina reserva automáticamente 10 localidades para — cada subíndice, por lo tanto, si ningún subíndice en los - - arreglos de un programa excede el valor de 10, puede omitirse la proposición DIM. La ventaja de especificar un subíndice - menor de 10, es reservar menos espacio de memoria que el que automáticamente se reservaría de 10 localidades por subíndi— ce, si no se hace ninguna indicación. Este ahorro de espa-- cio puede utilizarse para aumentar la capacidad de un progra ma, aumentando por ejemplo el número de instrucciones. Se - permite el uso de arreglos alfanuméricos, pero únicamente si son de una dimensión.

En un programa pueden aparecer varias proposiciones DIM, aunque por lo general la especificación de un arreglo solo - se hace una vez, sin embargo, algunos sistemas permiten rede finir el tamaño de los arreglos. Se tendría por ejemplo:

```
10  DIM   A(20,20)
      .
      .
60  LET   N= . . . . . .
70  DIM   A(2*N, N+5)
```

En los arreglos bidimensionales o matrices, el primer -- subíndice define el número de renglones y el segundo define el número de columnas del arreglo.

2.6 Subprogramas

Para evitar la programación repetida de varios cálculos, el programador puede escribir sus propios subprogramas los - cuales permiten escribir una sola vez las proposiciones del cálculo y permiten hacer referencia a dicho cálculo en cualquier parte del programa, sin necesidad de repetir en cada - ocasión todas las proposiciones del cálculo.

Los subprogramas se dividen en funciones de proposición - y en subrutinas, los subprogramas de proposición o funciones definidas por el usuario pueden constar de uno o varios renglones y calculan un solo valor para el programa principal y forman parte integral del programa en el que aparecen, el valor retornado puede ser numérico o alfanumérico dependiendo de la naturaleza de las variables involucradas en la definición de la función.

PROPOSICION DEF

Con esta proposición se define al subprograma de función de proposición y tiene la forma:

DEF nombre(*a*) = *b*

en donde nombre especifica el nombre de la función de proposición; *a* indica un argumento y *b* representa una expresión - válida en BASIC, limitada en general a no exceder el renglón correspondiente a esta proposición.

Los nombres de la función de proposición constan de 3 letras, siendo las dos primeras letras FN en todos los casos, por lo tanto, existen 26 posibilidades de nombres para las - funciones de proposición: FNA, FNB, ..., FNZ. El argumento consiste en general de una o más variables, a las cuales se les denomina variables mudas y deben de ir separadas por comas. La expresión a la derecha del signo igual puede in- - - cluir variables adicionales a las indicadas en el argumento y también otras funciones, incluyendo funciones definidas -- por otras proposiciones DEF. A continuación se muestran varias definiciones típicas de funciones de un renglón.

```
10 DEF FNA(X) = X**3 + 2*X**2 -3*X+4
20 DEF FND(X,Y)= SQR(X*X+Y*Y)
30 DEF FNT(A,B,C)= (-B+SQR(ABS(B*B-4*A*C)))/(2*A)
```

El nombre de una función alfanumérica debe de consistir - de tres letras seguidas del signo dólar, de nuevo las dos primeras deben ser FN y se pueden definir hasta 26 funciones alfanuméricas en un programa (FNA$, FNB$, ..., FNZ$); una función numerica y una alfanumérica que tenga las mismas letras (por ejemplo, FNP y FNP$) son entidades diferentes y por lo - tanto pueden aparecer en un mismo programa.

Una proposición DEF puede aparecer en cualquier lugar del programa, sin embargo, es una buena práctica en programación agrupar todas las definiciones de funciones y colocarlas cerca del comienzo o al final del programa, lo cual contribuye - a una estructuración ordenada y legible del programa.

PROPOSICION FNEND

Existen muchos cálculos que no se pueden llevar a cabo -- utilizando un solo renglón, o pueden existir casos en donde - el valor de una función depende de ciertas condiciones que - deben de cumplir sus parámetros, para los cálculos de este - tipo es especialmente apropiado el esquema de funciones de - varios renglones.

Una función de varios renglones, como las funciones de - uno solo, pueden tener cualquier número de argumentos mudos de entrada, también retornan un solo valor y la primera proposición debe ser una proposición DEF; sin embargo, en contraposición con las funciones de un renglón, la definición - de la función no se incluye en la proposición DEF; y la última proposición, debe ser una proposición FNEND, que consiste simplemente en un número de proposición seguido de la palabra clave FN END.

Entre las proposiciones DEF y FN END puede haber cualquier número de proposiciones que definen la función, una de las cuales debe asignar un valor al nombre de la función (lo cual se hace normalmente por medio de la proposición LET), - en la que el nombre de la función aparece a la izquierda del signo igual; con estas funciones se utiliza la misma convención de nombres utilizada con las funciones de un renglón.

A las funciones de varios renglones se les aplican las - mismas reglas gramaticales que a las funciones de un renglón; adicionalmente, no se puede transferir el control entre una proposición dentro de la función y un punto exterior a la -- función. A continuación se muestra la estructura de varias funciones de varios renglones:

```
200   DEF   FNA (X,Y,Z)
      ...
250   LET   FNA = .....
260   FN END
```

```
 20   DEF   FNM(A,B)
 30   LET   FNM=A
 40   IF   A <= B   THEN 60
 50   LET   FNM=B
 60   FN END
```

Debe mencionarse que la presencia de la proposición DEF - sólo sirve para definir una función; para evaluar la función es necesario hacer referencia al nombre de la función en alguna otra parte del programa, en la misma forma que se hace - con una función proporcionada por el sistema, esto se hace -- especificado el nombre de la función en una proposición en - BASIC, seguido de un conjunto apropiado de argumentos encerrados entre paréntesis y separados por coma (cuando estos existan) en la definición de la función.

Cuando se evalúa una función los valores de los argumentos los especifíca la referencia a la función, y no la definición de la función, por esta razón los argumentos que aparecen en la definición de una función (proposición DEF), se llaman argumentos mudos; no es necesario que los nombres delos argumentos en la referencia de la función sean los mismos de los de la definición, sin embargo, el número de argumentos debe ser el mismo, y los argumentos deben ser del mismo tipo (numéricos o alfanuméricos). A continuación se muestra una parte de un programa en BASIC que contiene referencias a una función definida por el usuario:

```
 10   DEF   FNA(X)= X*X-4*X+5
 20   DEF   FNE(X)= SQR(X+4)
 30   DEF   FNF(Y)= Y**2+A*Y+B
       .
       .
 50   W=FNA(Y) + Z - 5.28
       .
       .
 90   IF   FNA(C) >= C1 THEN 140
       .
       .
120   PRINT   TAB(10); FNA(Z)
       .
       .
200   LET Z = FNE(FNF(X))
```

En algunos sistemas se acepta que después de la definición de la función de varios renglones, colocar una lista de variables locales separadas por comas las cuales solo serán usadas por la función, lo anterior puede ser útil en algunos casos, como por ejemplo, en el siguiente programa que calcula el factorial de un número:

```
                                                                                  C           0000 0000 0000 0000
                                                                                  C           0000 0000 0000 0000
                                                                                                 SEGMENT 00000
                                                                                              DATA IS 0000 LONG
                                                                                              DATA IS 0000 LONG
[illegible] PRINT [illegible]; "MAST WRITE REC FACTORIAL R. J & "$%                C           0000 0000 0012
0010 FOR I=1 TO M                                                                 C           0000 0000 0014
0120     XMFAC(I)                                                                 C           0000 0001 0016
0130     PRINT                                                                    C           0000 0001 0018
0140     PRINT TAB(5); "N= FACTORIAL OF "$$$"=$"$"X                               C           0000 0001 0020
0150 NEXT I                                                                       C           0000 0002 0022
0160 STOP                                                                         C           0000 0002 0024
0170 REM ***** FUNCTION FACTORIAL *****                                           C           0000 0002 0026
0180 DEF FNF(N) I,X                                                               C           0000 0002 0028
0190 IF N<>1 THEN 210                                                             C           0000 0002 0030
0200 FNF=1                                                                        C           0000 0002 0032
0205 GO TO 240                                                                    C           0000 0002 0034
0210 X=1                                                                          C           0000 0002 0036
0220 FOR I=1 TO N                                                                 C           0000 0002 0038
0230     X=X*I                                                                    C           0000 0003 0040
0240 NEXT I                                                                       C           0000 0003 0042
0250 FNF=X                                                                        C           0000 0003 0044
0260 FNEND                                                                        C           0000 0003 0046
0270 END                                                                          C           0000 0003 0048
                                                                                              DATA IS 0000 LONG
                                                                                  STORAGE(0000) IS 0002 LONG
```

********** BASIC COMPILATION SUMMARY **********

PROGRAM FILE NAME=[illegible]

NUMBER OF ERRORS DETECTED = 0

NUMBER OF [illegible] = 1. TABLE SPACE = [illegible] WORDS. CORE ESTIMATE = [illegible] WORDS. STACK ESTIMATE = 48

PROGRAM SIZE = 22 CARDS, 152 SYNTACTIC ITEMS, 13 DISK SEGMENTS.

COMPILATION TIME = [illegible] SECONDS, [illegible] ELAPSED PROCESSING.

****

En el ejemplo anterior las variables I e X se han declarado como variables locales en la definición de la función (línea 70), y su uso dentro de la misma no tiene ningún efecto sobre las variables globales I e X usadas en el programa (Líneas 10 a 40), así el programa anterior produce los siguientes resultados:

```
CETENCIC? DEL FACTORIAL DE 0 A  15
EL FACTCFIAL DE  0 ES 1
EL FACTCFIAL DE  1 ES 1
EL FACTCFIAL DE  2 ES 2
EL FACTCFIAL DE  3 ES 6
EL FACTCFIAL DE  4 ES 24
EL FACTCFIAL DE  5 ES 120
EL FACTCFIAL DE  6 ES 720
EL FACTCFIAL DE  7 ES 5040
EL FACTCFIAL DE  8 ES 40320
EL FACTCFIAL DE  9 ES 362880
EL FACTCFIAL DE  10 ES 3628800
EL FACTCFIAL DE  11 ES 39916800
EL FACTCFIAL DE  12 ES 479001600
EL FACTCFIAL DE  13 ES 6227020800
EL FACTCFIAL DE  14 ES 87178291200
EL FACTCFIAL DE  15 ES 1.3076744+12
```

Algunas veces es más conveniente estructurar una secuencia de proposiciones como una subrutina que como una función; las subrutinas son similares a las funciones de varios renglones en el sentido de que, se pueden referenciar desde otros lugares del programa, pero a diferencia de aquellas no se les da un nombre y se pueden utilizar para determinar el valor de más de una variable; es más, no utiliza argumentos, en esta forma una subrutina puede intercambiar información con el resto del programa en forma muy general (lo cual para algunas aplicaciones puede ser ventajoso, pero para otras no).

PROPOSICION GOSUB

Con esta proposición se hace transferencia del programa a una subrutina y tiene la forma:

GOSUB *n*

en donde *n* es el número de renglón donde empiezan las proposiciones de la subrutina a la cual se hace la transferencia de control. Una subrutina no necesita comenzar con una proposición especial, puede comenzar con cualquier proposición en -- BASIC y puede contener cualquier número de estas.

PROPOSICION RETURN

Esta proposición tiene la forma:

```
RETURN
```

y especifica la terminación de la ejecución de una subrutina; esta proposición hace que se transfiera el control de regreso a la siguiente proposición de referencia a la subrutina - (GOSUB), se permite la existencia de varias proposiciones -- RETURN especificadas en la subrutina. A continuación se -- muestra una parte de un programa en BASIC, que hace referencia a una subrutina.

```
100  INPUT  A,B,C,N
     .
     .
150  GOSUB  300
160  .
     .
200  STOP
300  REM **** INICIO DE LA SUBRUTINA ****
310  LET C1=(A+B+C)/3
320  LET C2=SQR (A*A+B*B+C*C)
330  LET C3=SQR (A*B*C)
340  ON N 350, 380, 410
350  LET X + .....
     .
370  RETURN
380  LET Y= .....
     .
400  RETURN
410  LET Z+ .....
     .
430  RETURN
440  END
```

El cual ocasiona la transferencia de control al renglón-300 al ejecutarse la proposición GOSUB, indicada en el renglón 150, una vez ejecutadas las proposiciones del subprograma de subrutina (líneas 300 a 430), el control regresaría a la proposición con número de renglón 160, lo anterior permi-

te hacer múltiple referencia a una misma subrutina mediante el uso de varias proposiciones GOSUB en distintas partes del programa principal, como por ejemplo:

```
      .
      .
 50   GOSUB   2000
 60   .
      .
150   GOSUB   2000
160   .
      .
400   GOSUB   2000
410   .
      .
```

En donde el control regreso al renglón 60 después de la primera ejecución de la subrutina y a los renglones 160 y 410 después de la segunda y tercera ejecución de la subrutina respectivamente. Observese que si la proposición inmediatamente antes del inicio de la subrutina es una proposición STOP, como se muestra en el renglón 200 del ejemplo anterior, se asegura que el acceso a la subrutina es unicamente por medio de proposiciones GOSUB.

Un programa puede tener varias subrutinas y se permite que una subrutina haga referencia a otra de ellas, formándose una nidificación de subrutinas. Las subrutinas anidadas deben mantener un orden jerárquico estricto; ésto es, sí una subrutina A referencia a una subrutina B, entonces la subrutina B no puede referenciar a la subrutina A; por otra parte, si se puede referenciar a la subrutina B desde cualquier punto del programa al igual que a la subrutina A.

2.7 Características Especiales

A continuación se presentan algunas características especiales del lenguaje BASIC, se presentan proposiciones aplicables al manejo de las variables alfanuméricas así como algunas funciones de biblioteca para dichas variables, la notación simplificada para la ejecución de operaciones algebráicas matriciales (algebra matricial), manejo de archivos de datos externos y algunas proposiciones implantadas en algunos sistemas y que todavía no son de uso generalizado, en la mayoría de los sistemas.

VARIABLES ALFANUMERICAS

Estas variables permiten el procesamiento de datos alfanuméricos y en algunos sistemas se permite, para la identificación de una variable alfanumérica, después de una letra del alfabeto si se quiere puede ir un dígito numérico y por último el signo de pesos (el cual sí es necesario), en la mayoría de los sistemas la longitud máxima del contenido de una variable alfanumérica es de 15 caracteres o dígitos incluyendo espacios en blanco, sin embargo en algunos sistemas la longitud de una variable de este tipo puede ser hasta de 255 caracteres.

Para definir a las variables alfanuméricas en un programa se pueden utilizar las proposiciones LET, READ, INPUT, por ejemplo:

```
 50  LET  R$ = "VALOR  FINAL"
100  READ A$, B$, C$
      .
      .
150  DATA  VALOR,"  ", INICIALIZACION
      .
250  INPUT T$, O$, Z1$
300  STOP
```

En la ejecución del programa anterior, al imprimirse en la pantalla el signo ?, el usuario introducirá los datos alfanuméricos correspondientes a las variables T$, O$, Z1$, los espacios en blanco se deben de tomar en cuenta, así en el renglón 50 del ejemplo anterior la variable R$ contiene 11 caracteres, para tener en cuenta espacios en blanco, estos se encierran entre comillas, por ejemplo una alternativa para la variable R$ es:

```
 50  READ R$
      .
      .
200  DATA "VALOR  FINAL"
      .
```

En la mayoría de los sistemas, también se permite la forma:

N LET <variable alfanumérica> = <expresión alfanumérica>

En donde N es un número de línea, la palabra LET puede - no aparecer, y a la izquierda del signo igual deberá aparecer, una variable alfanumérica, y a la derecha del mismo una expresión alfanumérica, la cual puede ser una constante, -- una variable o una función proporcionada por el sistema o - definida por el usuario, todas ellas alfanuméricas y en algunos sistemas se permite que la expresión alfanumérica sea una combinación de constantes, variables o funciones alfanuméricas separadas por un operador de concatenación o unión, el cual en algunos sistemas es el signo más, por ejemplo:

```
100 C$ = "C" +B$ + SPACE(10) + FNA$ (X,T$)
```

En donde la expresión esta formada por la constante - - "C", la variable B$, la función de biblioteca SPACE y la -- función A$ definida por el usuario, todas ellas alfanuméricas.

Se pueden utilizar variables alfanuméricas con subíndice, pero en la mayoría de los sistemas únicamente para arreglos de una dimensión, si el arreglo no se indica en la proposición de dimensión DIM, automáticamente se le reservan - 10 localidades de memoría, por lo tanto, los arreglos con - más de 10 elementos deben especificarse en la proposición - DIM, por ejemplo, la proposición:

```
10  DIM  A$(5), B(15), M1$(25), C(15,20)
```

reserva 5 localidades de memoria para el arreglo A$ y 25 para el arreglo M1$ ambos alfanuméricos, además reserva 15 y 300 localidades para los arreglos numéricos B y C respectivamente.

En todas las proposiciones, exceptuando la proposición de asignación, las variables alfanuméricas y las numéricas - pueden aparecer en una misma lista, separadas por una coma - o por un punto y coma, el uso de este último después de una variable alfanumérica en una lista de impresión, produce que la impresión de la variable siguiente se inicie inmediata-- mente a continuación de la variable alfanumérica que precede al punto y coma, como se vió anteriormente se pueden manejar variables alfanuméricas mediante la proposición PRINT USING.

FUNCIONES PARA VARIABLES ALFANUMERICAS

En general este tipo de funciones para la manipulación de variables alfanuméricas se encuentran en sistemas gran-

des, aunque hoy en día encontramos microcomputadoras que poseen, aunque en reducido número, funciones para la manipulación de dichas variables, a continuación se mencionan algunas de estas funciones.

FUNCION LEN

La función LEN regresa el número de caracteres contenidos en una variable alfanumérica y se referencia de la siguiente manera:

```
LEN (A$)
```

donde A$ es cualquier variable alfanumérica, por ejemplo:

```
100  T$ = "UNAM"
110  T = LEN (T$)
120  PRINT "LA LONGITUD DE :";T$;" ES ";T
130  END
```

Durante la ejecución del programa anterior se produce la siguiente salida:

```
LA LONGITUD DE: UNAM ES 4
```

FUNCION EXT$

La función EXT$ extrae un cierto número de caracteres de una variable alfanumérica y puede referenciarse de la siguiente manera:

```
EXT$ (A$,I,F)
```

donde I es la posición del caracter dentro de la variable A$ desde donde se iniciará la extracción y F es la posición del caracter dentro de la misma variable en donde terminará la extracción, I y F pueden ser constantes numéricas, variables aritméticas o expresiones aritméticas A$ puede ser una constante, variable o expresión alfanumérica, por ejemplo:

```
110  T$ = "    UNAM"
120  U$ = EXT$ (T$,4,7)
130  PRINT U$
140  END
```

Después de la ejecución el valor de la variable U$ será "UNAM".

FUNCION LEFT

La función LEFT regresa un cierto número de caracteres que se encuentran más a la izquierda dentro de una variable alfanumérica, puede ser referenciada de la siguiente manera:

```
LEFT(A$,N)
```

donde N es el número de caracteres que serán extraídos estando más la izquierda dentro de la variable alfanumérica A$.

FUNCION RIGHT

Esta función retorna un cierto número de caracteres que se encuentran más a la derecha dentro de una variable alfanumérica, se referencia de la siguiente manera:

```
RIGHT (A$,N)
```

N es el número de caracteres que se encuentran más a la derecha dentro de la variable A$ de la cual serán extraídos.

FUNCION STR$

La función STR$ regresa un conjunto de caracteres que corresponden a un valor específico, se referencia de la siguiente manera:

```
STR$(N)
```

N es un valor el cual será regresado como alfanumérico, por ejemplo:

```
100 Q$ = STR$(500)
400 END
```

Después de la ejecución del programa anterior el contenido de la variable Q$ será "500".

FUNCION VAL

La función VAL regresa una constante numérica correspondiente a una variable, constante o expresión alfanumérica, por ejemplo:

```
100 A$ ="123."
110 B$ ="375E+21"
120 N = VAL (A$+B$)
900 END
```

Después de la ejecución del programa anterior, N tendrá el valor 1.23375 E+23.

FUNCION SPACE

Esta función regresa un número específico de espacios en blanco, es útil en la insersión de estos dentro de variables alfanuméricas, el ejemplo siguiente:

```
110  F$ = "FACULTAD"
120  D$ = "DE"
130  I$ = "INGENIERIA"
140  F1$ = F$ +SPACE(2) + D$ + SPACE(2) + I$
150  PRINT
160  END
```

produce la siguiente salida:

```
FACULTAD  DE  INGENIERIA
```

Además de las mencionadas existen otras funciones para la manipulación de información alfanumérica, por otra parte algunos sistemas en sí, poseen funciones disponibles en BASIC, las cuales proporcionan; la fecha, hora del día tanto en forma numérica como alfanumérica, así como el tiempo total de procesador que el programa ha utilizado hasta el momento en que ha sido invocada dicha función, estas funciones pueden ser útiles para algunas aplicaciones.

MANIPULACION DE VECTORES Y MATRICES

Algunos sistemas contienen un grupo de proposiciones especiales en BASIC, conocidas como proposiciones matriciales las cuales permiten llevar a cabo las operaciones más comunes con matrices, siendo esta una característica distintiva, poderosa y útil, del lenguaje BASIC.

Vector y matriz son términos matemáticos para referirse a listas y tablas respectivamente; así un vector es un arreglo unidimensional y una matriz un arreglo bidimensional; como un vector es una clase especial de matriz, la mayoría de las reglas generales que se aplican a las matrices son válidas también para los vectores.

Los vectores y las matrices pueden nombrarse con una cualquiera de las letras del alfabeto, seguidas si se desea de un dígito numérico y su tamaño o dimensión puede especificarse en una proposición DIM (ver proposición de especificación DIM, inciso 2.5), por ejemplo:

```
10  DIM A(3,5), B(15,15), C(50)
```

Reserva 15 localidades de memoria para la matriz A, 225 para la matriz B y 50 para el vector C, para las matrices el primer índice se refiere al número de renglones y el segundo al número de columnas de éstas, por lo que la matriz A tiene 3 renglones y 5 columnas y B es una matriz cuadrada de 15 renglones y 15 columnas.

Las dimensiones de las matrices pueden redefinirse posteriormente en el programa, cuidando de no exceder el espacio reservado para cada matriz en la proposición de dimensión DIM, aunque en sistemas grandes se permite que dicho espacio ya reservado sea excedido.

Las proposiciones que permiten redefinir las dimensiones de una matriz y que sirven también para inicializar valores de una matriz, son las cinco proposiciones siguientes.

Para la lectura de matrices puede usarse:

```
20  MAT  READ  S(2,4)
```

La instrucción anterior produce la lectura de la matriz S de 2 renglones y 4 columnas, la lectura de datos se hace por renglones, o sea que primero se leen los elementos S(1,1) S(1,2), S(1,3), S(1,4), y después los elementos S(2,1), - - S(2,2), S(2,3), S(2,4), la instrucción siguiente:

```
40 MAT READ B, P(5,2),A,G(2*N,M)
```

Produce la lectura de la matriz B (de 15 renglones y 15 columnas), como se definío en la proposición DIM de número de renglón 10, enseguida se leerá la matriz P (de 5 renglones y 2 columnas), para luego leerse la matriz A (de 3 renglones y 5 columnas) como también se definio en la proposición DIM de número de renglón 10, para por último leerse la matriz G (que sí N y M valen 3 y 4 respectivamente, tendrá 6 renglones y 4 columnas, en general N y M pueden ser cualquier expresión aritmética válida en BASIC).

Los valores correspondientes a las matrices que serán - leídas mediante la proposición MAT READ deben estar contenidas en una o varias proposiciones DATA.

Para la lectura de matrices, cuando los datos van a ser introducidos por una terminal, se tiene la proposición siguiente:

```
N MAT  INPUT  M1,M2, ...., M5
```

donde N es un número de línea y M1,M2, ...,M5 son nombres - de matrices seguidos de manera opcional de su dimensión, por ejemplo:

```
250 MAT  INPUT  B,A(5,3), C(a,b)
```

La línea 250 produce la lectura de la matriz B dimensionada con anterioridad, enseguida se produce la lectura de la matriz A (solo serán leídos los 5 renglones conteniendo cada uno de ellos 3 elementos) y por último se leerá la matriz C de a renglones y b columnas donde, en general, a y b son cualquier expresión numérica válida en BASIC.

Para inicializar a cero todos los elementos de una matriz, se tiene:

```
30  MAT A= ZER(3,3)
```

que iguala a cero todos los elementos de una matriz cuadrada de orden 3, o también:

```
40  MAT A = ZER (N,M)
50  MAT B = ZER
```

La proposición 40 iguala a cero todos los elementos de la matriz A, la cual fué además redimensionada de N renglones y M columnas, así como la instrucción 50 iguala a cero todos los elementos de la matriz B, cuya dimensión pudo haber sido definida en una proposición previa.

De manera semejante, para que todos los elementos de una matriz se igualen al número 1, se usa la forma:

```
50 MAT B= CON(N,M)
```

igualándose a 1 todos los elementos de la matriz B, redefiniéndose a N renglones y M columnas;

```
60 MAT A=CON
```

que iguala a 1 todos los elementos de la matriz A cuya dimensión pudo haber sido definida previamente.

Para inicializar una matriz con la matriz unitaria o identidad en donde los elementos sobre la diagonal principal son iguales al número 1 siendo nulos todos los elementos restantes de la matriz, se tiene la proposición siguiente:

```
70 MAT B=IDN
```

o bien la forma general, que es:

```
80 MAT B= IDN(a,b)
```

en donde a y b son cualquier expresión válida en BASIC, pudiendo ser variables o constantes numéricas, pero por definición de matriz identidad, se requiere que las matrices sean cuadradas, sin embargo a y b pueden no ser iguales.

A continuación se presentan otras proposiciones para el manejo de matrices, pero estas a diferencia de las anteriores no redimensionan a la o a las matrices que se incluyen en dichas proposiciones.

Para la impresión de matrices, se dispone de la siguiente proposición:

```
N MAT PRINT A1<, ó ;> A2 <, ó ;> .... A5
```

donde N es como siempre un número de línea y A1, A2,.. A5, son nombres de matrices si estos estan separados por coma los elementos de la matriz que le precede a la coma, serán impresos en zonas predefinidas y si es punto y coma éstos, serán impresos en forma comprimida es decir dejando un número mínimo de espacios en blanco entre cada elemento, en ambos casos la impresión se realiza por renglones, y en cualquier caso la impresión de un nuevo renglón se realiza en una nueva línea.

En la mayoría de los sistemas que poseen funciones para el manejo de matrices, disponen de funciones para realizar algunas operaciones matriciales tales como; suma, resta, multiplicación, multiplicación por un escalar, inversión y transposición de matrices, obedeciendo las matrices involucradas en dichas funciones a las mismas leyes del algebra matricial. Por ejemplo para la asignación, suma, resta y multiplicación de matrices, podemos utilizar:

```
100  MAT  A=B
110  MAT  C=A+B
120  MAT  D=C-B
130  MAT  E=A*B
140  MAT  B2= B*B
```

En donde A,B,C,D,E y B2 son matrices, así pues la línea 120 efectúa la resta de las matrices C y B, la cual es almacenada en la matriz D y la línea 140 efectúa el producto de la matriz B por sí misma y lo almacena en la matriz B2, en general, en una operación matricial solo se permite la aparición de un solo operador (+, -, ó *) a la derecha del signo igual y que actúe directamente sobre la matriz, y también no se permite que la misma matriz aparezca en la parte izquierda como en la derecha del signo igual, sin embargo algunos sistemas sí lo permiten, pero en todos los casos deben de ser conformables para la operación a realizar con ellas.

La multiplicación escalar puede ser ejecutada utilizando la forma siguiente:

```
N  MAT  A1= (E)*A2
```

donde N es un número de línea, A1 y A2 son matrices y E es - una expresión aritmética válida en BASIC, por ejemplo:

```
200 MAT  Z= (3.1416-SIN(Z(1,4)))*A
```

Las matrices pueden ser invertidas o transpuestas usando las siguientes proposiciones:

```
N  MAT  A1 = INV(A2)
N  MAT  A1 = TRN(A2)
```

donde N es un número de línea, A1 y A2 son matrices, la matriz A2 debe ser no singular para que exista su inversa y poder ser utilizada en la función INV.

FUNCION DET

La función DET regresa el valor del determinante de la - última matriz invertida utilizando la función INV, esta función no necesita argumentos y puede ser utilizada en una - - expresión aritmética o referida directamente por la proposición PRINT, por ejemplo:

```
150  MAT  K1 = INV(K)
160  D1 = DET
170  MAT K2 = (1/DET)*A
180  PRINT  "EL DETERMINANTE DE LA MATRIZ K1 ES:"; DET
```

Como ejemplo de aplicación, a continuación se presenta - un programa en BASIC, el cual resuelve armaduras planas utilizando matrices en forma tradicional, es decir formando y - manipulando en forma convencional todas y cada una de las matrices que intervienen en la solución del problema, sin aprovechar las ventajas que puede tenerse de ciertas características que presenta la matriz $[K]$, las cuales se mencionan en los capítulos siguientes. El tema se trata con mayor amplitud en los capítulos III y IV del presente trabajo, por ahora solo se limita a la ejemplificación de la manipulación de matrices en el lenguaje BASIC; las ecuaciones matriciales de que se dispone son:

$$[e] = [a][d] \qquad (3.3.2) \quad \text{Continuidad}$$

$$[p] = [k][e] \qquad (3.3.3) \quad \text{Ley de Hooke}$$

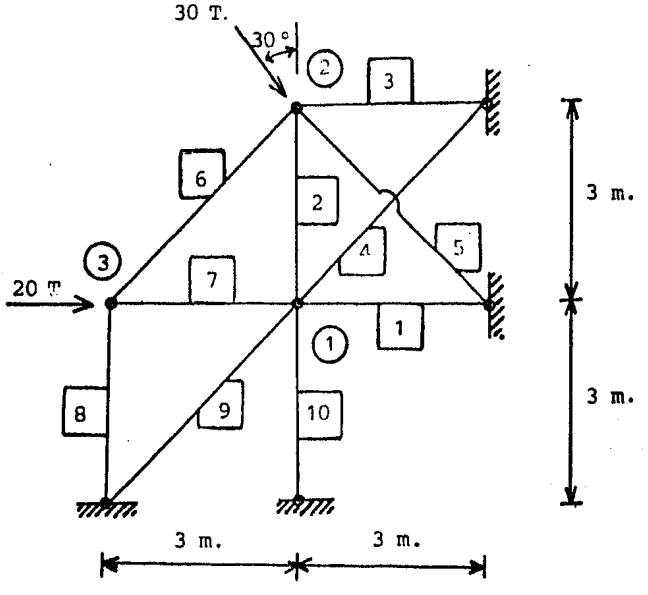
$$[F] = [a]^T[p] \qquad (3.3.4) \quad \text{Equilibrio}$$

en donde los datos son las matrices $[a]$, $[k]$ y $[F]$, las incognitas son las matrices $[e]$, $[d]$ y $[p]$, utilizando la forma de solución siguiente:

$$[d] = [K]^{-1}[F] \qquad (3.4.3)$$

donde: $$[K] = [a]^T [k][a] \qquad (3.4.2)$$

El significado de las variables involucradas en el programa puede verse en los resultados que el mismo proporciona, el cual se utilizó para la solución de la siguiente armadura (la cual también es resuelta con otros programas, -- los cuales se explican en los capítulos III y IV del presente trabajo):



Para la utilización de este programa es necesario formar las matrices $[a]$, $[k]$ y $[F]$ las cuales se consideran como datos del programa, y a partir de ellas se obtienen las incognitas que como ya se mencionó son las matrices $[e]$, $[d]$ y $[p]$.

BURROUGHS B78.H BASIC COMPILER VERSION 3.2.127 TUESDAY, 08/12/83, 02:03 PM.
BASIC PROGRAM NAME : ARBASIC ON PDC.

```
1100 $SET LIST                                                          C       030:0013:1
1102 $SET SINGLE                                                        C       030:0023:1
1104 REM SOLUCION DE UNA ARMADURA UTILIZANDO METODOS MATRICIALES        C       030:0033:1
                                                                                SEGMENT 00002
                                                                        DATA IS 0008 LONG
                                                                        DATA IS 0038 LONG
1120 REM  AUTOR : FERNANDO MONROY MIRANDA  FACULTAD  DE  INGENIERIA     C       002:0000:1
1130 REM          U. N. A. M.        MAYO DE 1981                       C       002:0030:1
1140 REM                                                                C       002:0030:1
1145 FILES BMII                                                         C       002:0030:1
1150 DIM A(50,50),T(50,50),K(50,50),K3(50,56),F(50,50),I(50,50),D(50,50)C       002:0032:1
1160 DIM M1(50,5.),E(50,5.),E(5 ,5 ),F(5 ,5 ),F1(50,53)                 C       002:0011:2
1180 REM LECTURA DEL NUMERO DE NUDOS, PARRAS Y CONDICIONES DE CARGA     C       002:001C:1
1182 READ # 1, N,B,C,T                                                  C       002:001C:1
1184 N2=0                                                               C       002:0027:1
1186 N$=""                                                              C       002:0027:5
1188 IF T < 2 THEN 1130                                                 C       002:002A:2
1190 IF T > 3 THEN 1130                                                 C       002:002B:4
1192 N2=1                                                               C       002:0020:0
1194 N$=" PLANAS "                                                      C       002:002D:4
1196 IF T = 2 THEN 1130                                                 C       002:0030:4
1198 N$=" ESPACIALES "                                                  C       002:0032:0
1110 PRINT                                                              C       002:0035:1
1111 PRINT TAB(1);"ANALISIS DE ARMADURAS";N$;"UTILIZANDO ";             C       002:0039:5
1112 PRINT "METODOS MATRICIALES"                                        C       002:0044:5
1114 PRINT                                                              C       002:0049:3
1116 PRINT TAB(1.);"NUMERO DE NUDOS ";TAB(42);N                         C       002:004E:1
1118 PRINT                                                              C       002:0058:1
1120 PRINT TAB(1.);"NUMERO DE BARRAS ";TAB(42);B                        C       002:005C:5
1122 PRINT                                                              C       002:0066:5
1124 PRINT TAB(10);"NUMERO DE CONDICIONES DE CARGA  ";C                 C       002:006B:3
1126 PRINT                                                              C       002:0073:5
1128 PRINT TAB(1.); "TIPO DE ARMADURA";TAB(42);T;" *"                   C       002:0078:5
1130 PRINT                                                              C       002:0084:3
1132 PRINT TAB(10); "* SI TIPO = 2, SE TRATA DE UNA ARMADURA PLANA;"    C       002:0089:1
1134 PRINT                                                              C       002:008F:5
1136 PRINT TAB(1.); "  SI TIPO = 3, SE TRATA DE UNA ARMADURA ESPACIAL."  C       002:0094:3
1138 IF N2 = 1 THEN 1146                                                C       002:0098:1
```

```
1140 PRINT                                                    C    0032:0090:2
1142 PRINT "**********  F I N  D E  L A  **********"          C    0032:00A1:0
1144 STOP                                                     C    0032:00A6:1
1146 REM LECTURA DE LA MATRIZ DE CONTINUIDAD                  C    0032:00A6:4
1148 LET N2 = T * N                                           C    0032:00A6:4
1150 MAT A = ZER(E,N2)                                        C    0032:00A8:9
1160 MAT READ # 1, A                                          C    0032:00AA:5
1165 LET M$ = "LA MATRIZ"                                     C    0032:00B0:5
1170 PRINT                                                    C    0032:00B4:3
1180 PRINT TAB(1);M$;" DAT DE CONTINUIDAD ES :"               C    0032:00B9:1
1190 PRINT                                                    C    0032:00C2:0
1200 MAT PRINT A                                              C    0032:00C6:4
1210 REM CALCULO DE LA MATRIZ DE EQUILIBRIO = DAT TRANSPUESTA C    0032:00CB:3
1220 MAT T = ZER(N2,B)                                        C    0032:00CB:3
1230 MAT T = TRN(A)                                           C    0032:00CE:2
1250 PRINT TAB(1);M$;" DT DE EQUILIBRIO ES :"                 C    0032:00D0:0
1260 PRINT                                                    C    0032:00D8:5
1270 MAT PRINT T                                              C    0032:00DD:3
1280 REM LECTURA DE LAS RIGIDECES DE LAS BARRAS               C    0032:00E2:2
1290 MAT K1 = ZER(P,B)                                        C    0032:00E2:2
1300 FOR I = 1 TO B                                           C    0032:00E5:1
1310     READ # 1, K1(I,I)                                    C    0032:00E8:1
1320 NEXT I                                                   C    0032:00F3:0
1340 PRINT TAB(1);M$;" DK1 DE RIGIDECES DE LAS BARRAS ES :"   C    0032:00F3:5
1350 PRINT                                                    C    0032:00FC:4
1360 MAT PRINT K1                                             C    0032:0101:2
1370 REM OBTENCION DE LA MATRIZ DE RIGIDECES DE LA ESTRUCTURA C    0032:0106:1
1380 MAT K = ZER(N2,B)                                        C    0032:010A:1
1390 MAT K = T * K1                                           C    0032:0109:0
1400 MAT K3 = ZER(N2,N2)                                      C    0032:0108:1
1410 MAT K3 = K * A                                           C    0032:013E:0
1420 PRINT TAB(1);M$;" DK3 DE RIGIDECES DE LA ESTRUCTURA ES :"C    0032:0110:1
1440 PRINT                                                    C    0032:0119:0
1450 MAT PRINT K3                                             C    0032:0110:4
1460 REM LECTURA DE LA MATRIZ DE FUERZAS EN LOS NUDOS         C    0032:0122:3
1470 MAT F = ZER(N2,C)                                        C    0032:0122:1
1480 MAT READ # 1, F                                          C    0032:0125:2
1500 PRINT TAB(1);M$;" DFT DE FUERZAS EN LOS NUDOS ES :"      C    0032:012C:2
1510 PRINT                                                    C    0032:0135:1
1520 MAT PRINT F                                              C    0032:0139:5
1530 REM OBTENCION DE LOS DESPLAZAMIENTOS DE LOS NUDOS        C    0032:013E:4
1540 MAT I = ZER(N2,N2)                                       C    0032:013E:4
```

```
155  MAT I = INV(V3)                                                    - 83
157  PRINT TAB(1-);M$;" [I] INVERSA DE ";M$;" DE RIGIDECES DE LA ";
158  PRINT " ESTRUCTURA ES :"
159  PRINT
160  MAT PRINT I
161  MAT D = ZER(N2,C)
162  MAT D = I * F
164  PRINT TAB(1-);M$;" [D] DE DESPLAZAMIENTOS DE LOS NUDOS ES :"
165  PRINT
166  MAT PRINT D
167  REM OBTENCION DE LAS DEFORMACIONES DE LAS BARRAS
168  MAT E = ZER(B,C)
169  MAT E = A * D
171  PRINT TAB(1-);M$;" [E] DE DEFORMACIONES DE LAS BARRAS ES :"
172  PRINT
173  MAT PRINT E
174  REM OBTENCION DE LAS FUERZAS EN LAS BARRAS
175  MAT P = ZER(B,C)
176  MAT P = V1 * C
178  PRINT TAB(11);M$;" [P] DE FUERZAS EN LAS BARRAS ES :"
179  PRINT
181  MAT PRINT P
181  REM COMPROBACION EN LA EC. DE EQUILIBRIO
182  MAT F1 = ZER(N2,C)
183  MAT F1 = T * P
185  PRINT TAB(11);M$;" [F1] DE FUERZAS CALCULADAS EN LOS NUDOS ES :"
186  PRINT
187  MAT PRINT F1
2,2  END
```

```
C          0J2:0149:3
C          0J2:0145:4
C          0J2:0150:4
C          0J2:0155:5
C          0J2:015A:3
C          0J2:015F:2
C          0J2:0162:1
C          0J2:0164:2
C          0J2:0160:1
C          0J2:0171:5
C          0J2:0176:4
C          0J2:0176:4
C          0J2:0179:5
C          0J2:017B:4
C          0J2:0184:3
C          0J2:0189:1
C          0J2:018E:0
C          0J2:018E:0
C          0J2:0198:5
C          0J2:0193:0
C          0J2:0198:5
C          0J2:01A0:3
C          0J2:01A5:2
C          0J2:01A5:2
C          0J2:01AB:1
C          0J2:01AA:2
C          0J2:01B3:1
C          0J2:0187:5
C          0J2:018C:4
PSEUDO-CODE IS 0037 LONG
       DATA IS 0070 LONG
       DATA IS 0008 LONG
SEGMENT(0J2) IS 0270 LONG
```

```
********** BASIC COMPILATION SUMMARY **********
PROGRAM FILE NAME:OBJECT/TESIS/ARRASIC ON DISK.
NUMBER OF ERRORS DETECTED = 0.
NUMBER OF SEGMENTS = 1.  TOTAL SEGMENT SIZE = 1476 WORDS.  CORE ESTIMATE = 10895 WORDS.  STACK ESTIMATE = 93
PROGRAM SIZE = 106 CARDS, 772 SYNTACTIC ITEMS, 43 DISK SEGMENTS.
COMPILATION TIME = 5 SECONDS ELAPSED, 3 SECONDS PROCESSING.
**********
```

ANALISIS DE ARMADURAS PLANAS UTILIZANDO METODOS MATRICIALES

NUMERO DE NUDOS 5

NUMERO DE BARRAS 10

NUMERO DE CONDICIONES DE CARGA 1

TIPO DE ARMADURA 2 *

* SI TIPO = 2, SE TRATA DE UNA ARMADURA PLANA;
SI TIPO = 3, SE TRATA DE UNA ARMADURA ESPACIAL.

LA MATRIZ [A] DE CONTINUIDAD ES :

-1	0	1	0	0	
0	-1	0	0	0	
-0.7071	-0.7071	0	0	0	
1	0	0.7071	-0.7071	-0.7071	
0	1	0.7071	0	0	
0.7071	-0.7071	0	0	0	

LA MATRIZ [T] DE EQUILIBRIO ES :

-1	0	-0.7071	0	0	1	0
0.7071	0	-0.7071	0	0	0	0
0.7071	-1	0	-0.7071	0.7071	0	0
0	0	0	0.7071	0.7071	0	0
0	0	0	0	-0.7071	-1	0
0	0	0	0	-0.7071	0	1

LA MATRIZ [K1] DE RIGIDECES DE LAS BARRAS ES :

175	0	0	0	0	0	0
0	175	0	0	0	0	0
0	0	175	0	0	0	0
0	0	0	123.744	0	0	0
0	0	0	0	123.744	0	0
0	0	0	0	0	123.744	0
0	0	0	0	0	0	175
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	175

LA MATRIZ [K3] DE RIGIDECES DE LA ESTRUCTURA ES :

473.74162659	123.74162659	-175	0	-61.871813295	-61.871813295
123.74162659	473.74162659	0	-175	-61.871813295	-61.871813295
-175	0	298.74162659	-61.871813295	298.74162659	61.871813295
0	-175	-61.871813295	298.74162659	61.871813295	298.74162659

LA MATRIZ [F] DE FUERZAS EN LOS NUDOS ES :

15
-25.98
26

LA MATRIZ [I] INVERSA DE LA MATRIZ DE RIGIDECES DE LA ESTRUCTURA ES :

.[illegible]	-9.[illegible]-4	4.[illegible]-4	-9.[illegible]-5	[illegible]2733554919	-6.25815[illegible]-4
-9.[illegible]-4	[illegible]	4.[illegible]-5	[illegible]	-3.[illegible]-4	5.76376533-4
4.36521768-4	2.[illegible]-5	[illegible]	2.[illegible]-4	[illegible]216668.4	7.8[illegible]146272-4
-9.[illegible]153-5	[illegible]	4.[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	-3.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
-2.25815491-4	5.[illegible]	9.[illegible]	[illegible]	[illegible]	[illegible]

LA MATRIZ [D] DE DESPLAZAMIENTOS DE LOS NUDOS ES :

.[illegible]
-.[illegible]
.[illegible]
-.[illegible]
.[illegible]
-.[illegible]

LA MATRIZ [E] DE DEFORMACIONES DE LAS BARRAS ES :

-.[illegible]
-.[illegible]
.[illegible]
-.[illegible]
.[illegible]
-.[illegible]
.[illegible]
-.[illegible]
.[illegible]
-.[illegible]

LA MATRIZ [P] DE FUERZAS EN LAS BARRAS ES :

-11.16289287
-8.[illegible]
-12.[illegible]
-.[illegible]
-14.[illegible]
-10.[illegible]
-12.46283[illegible]
-7.[illegible]
[illegible]
-8.[illegible]

LA MATRIZ [F1] DE FUERZAS CALCULADAS EN LOS NUDOS ES :

2.2737368E-1

15
-25.98
26
-5.82[illegible]7661-11

```
  100      3,10,1,2,
  200      -1,0,0,0,0,0,
  300      0,-1,0,1,0,0,
  400      0,0,-1,0,0,1,
  500      -.7071,-.7071,0,0,0,0,
  600      0,0,-.7071,.7071,0,0,
  700      0,0,.7071,.7071,-.7071,-.7071,
  800      1,0,0,0,-1,0,
  900      0,0,0,0,0,1,
 1000      .7071,.7071,0,0,1,0,
 1100      0,1,0,0,0,0,
 1200      175,175,175,123.744,123.744,123.744,175,175,
 1300      123.744,175,
 1400      0,0,15,-25.98,20,1,
```

DESARROLLOS DE PROPOSICIONES

Los desarrollos efectuados en las proposiciones, le proporcionan una mayor flexibilidad al lenguaje BASIC, estos desarrollos son:

En las proposiciones aritméticas se permite el uso múltiple del signo de igualdad relacionando cualquier número de variables, permitiéndose el uso de solo una expresión a la derecha del último signo igual, por ejemplo:

```
15 A = F = G = H = (SQRT(X)+Y)*Z
25 B$ = K$ = R$ = " PRIMER VALOR "
```

La proposición de control IF-THEN puede también ser cambiada por la proposición IF-THEN GO TØ ó bien por la proposición IF-GO TO, por ejemplo son válidas las siguientes proposiciones:

```
35 IF X+4 < Z↑2  THEN 90
   .
40 IF A=B  THEN  GOTO  110
   .
60 IF ABS(X) <> X GO TO  120
```

Principalmente disponibles en las microcomputadoras se pueden colocar en un mismo renglón dos o más proposiciones en BASIC separándolas entre sí por dos puntos, por ejemplo son válidas:

```
30  PRINT  TAB(10);"RESULTADOS": N = N + 1 : X = O
```

Se pueden colocar todas las proposiciones que puedan ser admitidas en un renglón lógico, el cual puede contener hasta 254 caracteres incluyendo su número de renglón y espacios en blanco, puede ser expresado en varios renglones físicos y de ellos solo el primero deberá llevar número de renglón, por ejemplo:

```
100  FOR I=1 TO N: A(I,J) = A(I,J)/N
     : X(I) = A(I,J)/X(I) : PRINT X(I) :
     X(I) = ABS(X(I)) : A(I,J) + A(I,J) + X(I):
     NEXT I
```

Disponible también en el BASIC de las microcomputadoras tenemos la proposición IF-THEN-ELSE la cual tiene la forma siguiente:

IF *a* operador de relación *b* THEN *n1* ELSE *n2*

Todos los elementos que intervienen en esta proposición deberán estar contenidos en un solo renglón lógico, en donde *a* y *b* tienen el mismo significado de la proposición - -- IF-THEN las cuales estan relacionadas por un operador de relación y sí la relación es verdadera ejecuta la instrucción o instrucciones *n1* que se encuentran entre las palabras - - THEN y ELSE, para luego ejecutar la siguiente instrucción - con número de proposición, y si es falsa ejecuta la o las -- instrucciones *n2* que están después de la palabra ELSE y hasta el siguiente número de renglón, para luego ejecutar la - instrucción que este número de renglón posea (recuérdese -- que toda la proposición IF-THEN-ELSE debe estar contenida - en un solo renglón lógico), por ejemplo:

```
 90 .
100 IF  A = B  THEN : A = A+1 : B = B - 1 : PRINT  A, B :
    X = X + A + B    ELSE  GOSUB  1000 : B = A = 0 :
    X = X + 1
110  .
```

También los operadores de relación que intervienen en - la construcción de proposiciones IF-THEN o sus equivalentes pueden ser cambiadas por los siguientes:

Operador de relacion	*También se puede utilizar*	*Significado*
<	LT	menor que
< = , = <	LE	menor o igual que
=	EQ	igual que
> = , = >	GE	mayor o igual que
>	GT	mayor que
>< , <>	NE	no igual a

El número de dígitos que se desean en la impresión de resultados numéricos, puede especificarse por medio de la expresión SET DIGITS, que tiene la forma:

SET DIGITS(*a*)

en donde *a* es una expresión numérica, en donde su valor se calcula conservándose solo la parte entera, que representa el número de dígitos que aparecen en la impresión de resultados, el valor entero calculado debe estar comprendido entre los valores 1 y 11, recurriéndose a la notación exponencial en caso de exceder el número de dígitos especificados, unicamente los resultados impresos posteriores a la ejecución de la proposición SET DIGITS se ven afectados por esta proposición.

MANEJO DE ARCHIVOS

En el lenguaje utilizado en el ámbito de la computación, un archivo (FILE) es un conjunto ordenado de información de cualquier tipo; así un archivo puede contener una secuencia de proposiciones en BASIC (o en cualquier otro lenguaje), o puede contener valores tanto numéricos como alfanuméricos; el primero es un programa en un determinado lenguaje y al segundo se le denomina un archivo de datos.

Los archivos de datos proporcionan una forma conveniente de almacenar conjuntos de datos, principalmente cuando el número de datos es bastante grande, ya que si estos se introdujeran durante la ejecución de un programa el hacerlo a la velocidad con que una persona teclea puede tomar bastante tiempo, durante el cual la computadora estaría ocupada recibiendo la información, además puede ser que algún dato no sea el que se quería introducir con lo cual puede inutilizar el trabajo ya realizado, para evitar estos problemas es conveniente almacenar previamente los datos en un archivo, ya que durante esta operación estos pueden ser verificados. De esta manera el tiempo de lectura de datos cuando estos estan almacenados en un archivo, no es comparable con el requerido cuando los datos se proporcionan por una persona durante la ejecución del programa.

Algunos sistemas poseen en lenguaje BASIC proposiciones para el manejo de archivos, las cuales en la mayoría de los casos son similares a las proposiciones matriciales, aunque existen diferencias en la forma de las proposiciones del BASIC para el manejo de archivos entre las distintas versiones del BASIC o entre un sistema y otro.

3. ANALISIS ESTRUCTURAL DE ARMADURAS EN DOS DIMENSIONES

3.1 Objetivos del Análisis Estructural

En este capítulo se estudiará el análisis de armaduras planas, es decir armaduras en dos dimensiones, las cuales se idealizan como un sistema de barras contenidas en un plano e interconectadas en juntas o nudos (Fig. 3.1.1) para la realización del análisis se ha supuesto que la estructura tiene un comportamiento elástico y lineal.

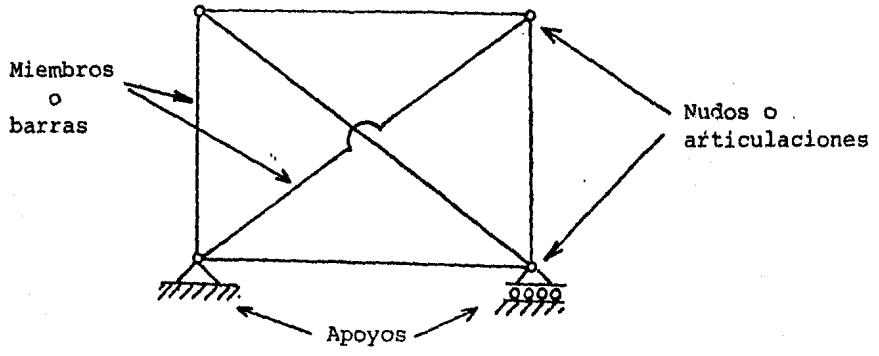


Fig. 3.1.1 Elementos que constituyen a una armadura

Se consideran como datos de un problema típico para el - Análisis Estructural de una estructura, los siguientes:

a) Geometría de la estructura. Es decir se conocen las dimensiones de los ejes y de las secciones (lo que - supone un prediseño).

b) Propiedades mecánicas de los materiales. Como lo es el módulo de elásticidad E, relación de Poissón ν , etc.

c) Solicitaciones. Se estudiará solo las solicitaciones estáticas (solicitaciones aplicadas gradualmente para no producir vibraciones).

Cuando una estructura esta bajo la acción de solicitaciones, los miembros de ella sufren deformaciones (pequeños -- cambios en su forma), y como consecuencia, puntos dentro de la estructura se desplazarán hacia nuevas posiciones, en general, todos los puntos de la estructura, excepto puntos de apoyo inmóviles, sufrirán dichos desplazamientos, estos últimos causados por los efectos acumulativos de las deformaciones de todos los elementos.

Un desplazamiento es generalmente una traslación, rotación o ambas de algún punto de una estructura, una traslación se refiere a la distancia recorrida en línea recta por un punto de una estructura, y una rotación significa el ángulo de giro de la tangente a la curva elástica en un punto.

El cálculo de estos desplazamientos es una parte esencial del análisis estructural y para ello, es necesario comprender las deformaciones, las cuales producen estos desplazamientos.

Consideremos un segmento de longitud arbitraria cortado de un miembro de una estructura, por simplicidad se considera que la barra tiene una sección circular (Fig. 3.1.2). - En cualquier sección transversal, por ejemplo en el extremo derecho de la barra, existirán componentes de las fuerzas - internas, desarrolladas por la barra debido a la acción de solicitaciones externas que en el caso general están formadas por tres fuerzas y tres pares.

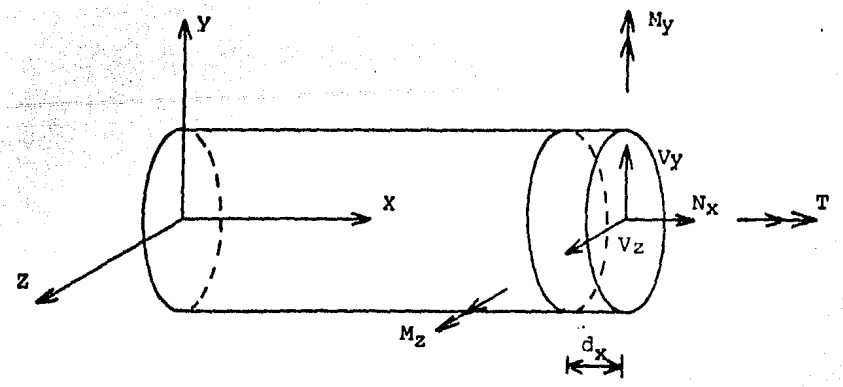


Fig. 3.1.2 Componentes de las fuerzas que actúan en una sección

Las fuerzas son la fuerza axial N_x y las fuerzas cortantes V_y y V_z ; los pares son el par torsionante T, y los pares flexionantes M_y y M_z, las deformaciones de la barra -- pueden analizarse tomando separadamente cada componente de las fuerzas y determinando su efecto sobre un elemento de - la barra, el elemento se obtiene aislando una porción de la barra entre dos secciones transversales separándolas una -- pequeña distancia dx (Fig. 3.1.2).

El efecto de la fuerza axial N_x sobre el elemento se -- muestra en la (figura 3.1.3*a*), suponiendo que la fuerza actúa en el centroide del área de la sección tranversal, se en--- cuentra que el elemento se deforma uniformemente, y las deformaciones significantes del elemento son deformaciones -- paralelas a la dirección de la aplicación de la fuerza axial.

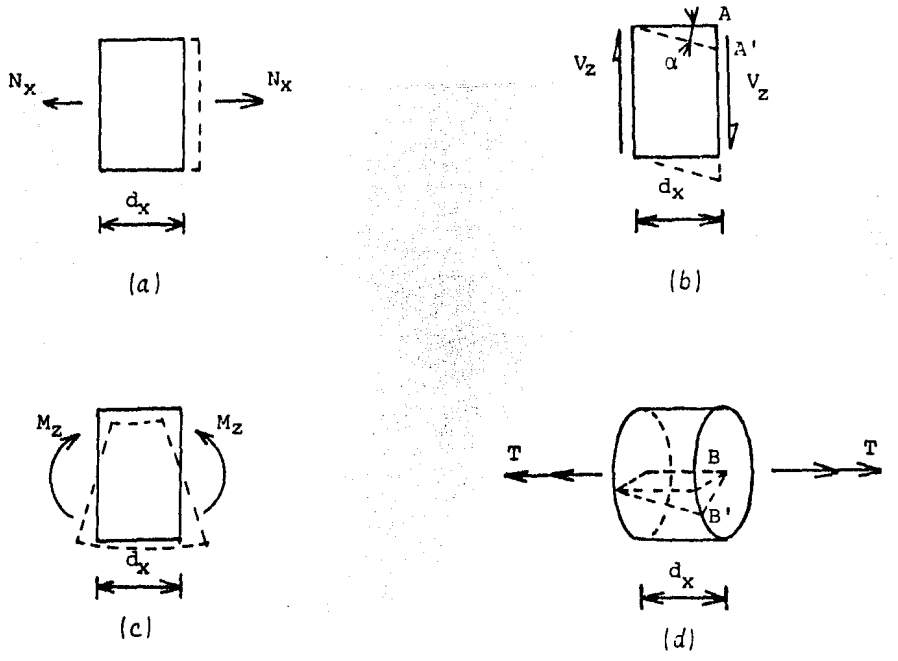


Fig. 3.1.3 Tipos de deformación: (*a*) Axial, (*b*) De cortante, (c) de Flexión y (*d*) de Torsión

En el caso de una fuerza cortante, una sección plana se desplaza lateralmente con respecto a la otra (Fig. 3.1.3*b*), causando la distorsión del ángulo α con esto el punto A pasa a una posición A', un momento flexionante causa una rotación relativa de las dos secciones transversales provocando que ya no queden paralelas una de otra (Fig. 3.1.3c), las deformaciones resultantes en el elemento son en la dirección longitudinal del elemento, y consisten en un acortamiento en el lado de compresión y un alargamiento en el lado de tensión, además ambos provocan un desplazamiento en la dirección perpendicular al eje de la flexión, por último el par de torsión T causa una rotación relativa de las dos secciones transversales alrededor del eje de torsión (en este caso el eje x) y, por ejemplo (Fig. 3.1.3*d*), el punto B se desplaza a la posición B'.

Las deformaciones mostradas en las figuras 3.1.3 *a*,*b*,c y *d* se llaman deformaciones axial, de cortante, de flexión y de torsión respectivamente, su evaluación depende de la forma de la sección transversal de la barra y de las propiedades mecánicas del material del cual está hecha la barra.

El desplazamiento total de un punto cualquiera de una -

estructura será la suma de los desplazamientos producidos por las deformaciones debidas a la acción de las fuerzas actuantes sobre dicha estructura, es decir se supone válido el principio de superposición de causas y efectos.

En una estructura particular, no todos los tipos de deformaciones serán significantes en el cálculo de los desplazamientos, por ejemplo, en vigas generalmente las únicas deformaciones importantes son las debidas a la flexión, y es usual no considerar las deformaciones por cortante y más aún las deformaciones axiales, por supuesto hay situaciones en las que se requiere que las barras soporten grandes cargas axiales y bajo dichas circunstancias la deformación axial debe incluirse en el análisis.

En general para poder omitir un cierto tipo de deformación será necesario que los resultados obtenidos considerando dicha deformación no difieran significativamente de los obtenidos al no considerarla.

En una armadura plana las acciones que van a estar actuando sobre esta pueden ser fuerzas actuando en el plano de la armadura, estas pueden ser fuerzas concentradas aplicadas en los nudos, así como cargas que actúan en los propios miembros, para propósitos de análisis, estas últimas cargas pueden reemplazarse por cargas estáticamente equivalentes que actúan en las articulaciones.

Para armaduras planas el análisis se puede efectuar de dos maneras. Si los nudos de la armadura se idealizan como articulaciones incapaces de desarrollar fuerzas de fricción permitiendo que los extremos de las barras que concurren al nudo giren libre e independientemente unos de otros y si todas las cargas actúan unicamente en los nudos, entonces el análisis involucra unicamente deformaciones axiales de los miembros, así pues, el análisis de la armadura dará como resultado fuerzas axiales de tensión o de compresión que actúan en los miembros de la armadura, si existen cargas que actúan sobre los miembros estas (como ya se mencionó) pueden reemplazarse por cargas estáticamente equivalentes que actúan en los nudos, proporcionando el análsis de nuevo, fuerzas axiales en los miembros, además existirán momentos flexionantes y fuerzas cortantes en aquellos miembros que tienen cargas actuando directamente sobre ellos.

Si los nudos de una estructura tipo armadura son realmente rígidos o así van a ser considerados en el análisis, entonces los miembros sufren (aunque todas las cargas actúen

en los nudos) los efectos de todas las acciones (flexión, cortante y axial) debiéndose, en general, considerar las deformaciones producto de estas, en tal caso la estructura puede analizarse como un marco plano, en este caso el análisis dará como resultado (entre otras cosas), los valores de los elementos mecánicos actuantes (fuerza axial, fuerza cortante y momento flexionante) en las barras que constituyen a la estructura.

Así pues, considerando a los nudos como articulaciones y que las cargas a las que va a estar sujeta una armadura, serán fuerzas concentradas aplicadas en las articulaciones, entonces, los objetivos que se persiguen al realizar el análisis estructural de una armadura plana (los cuales se consideran como incognitas hasta antes de la realización de dicho análisis), son los siguientes:

a) Desplazamientos lineales de los nudos de la armadura.

b) Deformaciones lineales de las barras que forman a la armadura.

c) Esfuerzo axial, generalmente en lugar del esfuerzo axial se obtiene la fuerza axial actuante en cualquier sección de las barras que constituyen a la armadura.

3.2 Principios fundamentales del Análisis Estructural

Para la determinación de los desplazamientos lineales de los nudos, deformaciones lineales de las barras y de la fuerza axial actuante en las barras, estos, como ya se mencionó son los objetivos del análisis estructural, se cuenta con los siguientes principios aplicables a nuestro problema y son:

a) Principio de continuidad o de compatibilidad geométrica.

Este principio supone que los desplazamientos en todos los puntos de la estructura son funciones contínuas de posición, en donde, además se supone que las deformaciones que sufren las barras de una estructura son pequeñas, y por lo tanto se pueden obtener relaciones entre las deformaciones lineales de las barras (alargamiento o acortamiento) y los desplazamientos lineales de los nudos.

Las ecuaciones de compatibilidad deben de, por ejemplo, ser satisfechas en todos los puntos de apoyo en donde es necesario que los desplazamientos de la estructura sean consistentes con las condiciones de apoyo, por ejemplo en un apoyo fijo no puede haber desplazamientos lineales de los extremos de las barras que concurren a dicho apoyo, las ecuaciones de compatibilidad también deben de satisfacerse en todos los puntos del interior de la estructura, usualmente, son las condiciones de compatibilidad en los nudos de la estructura las que son de interés, por ejemplo en una conexión entre varios miembros las deformaciones lineales de las barras deben de ser consistentes con el desplazamiento lineal del nudo existente en dicha conexión.

b) Equilibrio.

Recordando que uno de los objetivos de cualquier análisis estructural es el de determinar las fuerzas internas (y de ellas las reacciones en los apoyos), una solución correcta para cualquiera de estas cantidades debe satisfacer todas las condiciones de equilibrio estático, no sólo para toda la estructura, sino también para cualquier parte de ella tomada como un cuerpo libre.

Cuando todas las fuerzas que actúan sobre un cuerpo libre, estando aquellas y este en un plano (como en el caso de una armadura plana), y todos los momentos actuantes son alrededor del eje perpendicular al del plano donde actúan dichas fuerzas, se dispone de tres ecuaciones escalares de equilibrio estático, por ejemplo, si el plano es el formado por los ejes x e y, siendo el z perpendicular a los dos primeros las ecuaciones de equilibrio son:

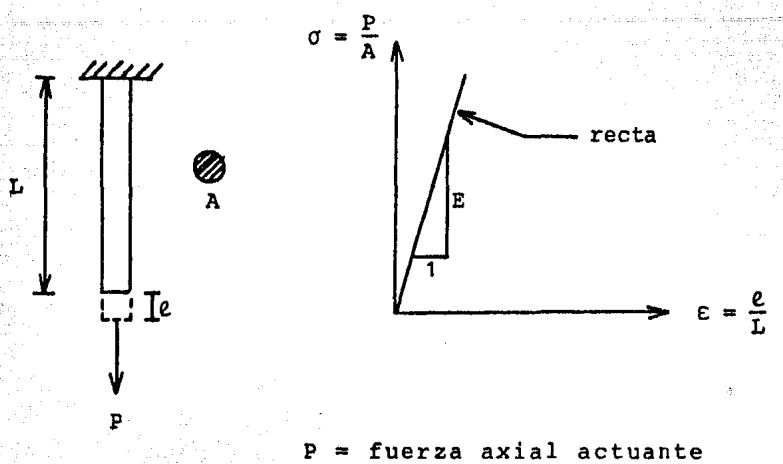
$$\Sigma F_x = 0 \; ; \qquad \Sigma F_y = 0 \; ; \qquad \Sigma M_z = 0$$

Generalmente en el caso de las armaduras planas es condición necesaria y suficiente que el equilibrio se cumpla, aislando como cuerpo libre a cualquier nudo, en este caso solo se verificarán las dos primeras ecuaciones de equilibrio, es decir deberá cumplirse que: $\Sigma F_x = 0$; $\Sigma F_y = 0$.

c) Ley de Hooke.

Se considerará que el material del cual estan hechos todas las barras que forman a la estructura tienen un comportamiento elástico y lineal el cual puede ser representado por la Ley de Hooke (Fig. 3.2.1), la cual puede ser expresada -

$$\sigma = E\,\varepsilon \qquad (3.2.1)$$



P = fuerza axial actuante
e = deformación longitudinal
A = área de la sección transversal
L = longitud de la barra

Fig. 3.2.1 Ley de Hooke

La pendiente de la recta obtenida de la gráfica esfuerzo axial (σ) y deformación unitaria (ε) es conocida como el módulo de elasticidad del material o módulo de Yung (E), el cual es una característica de cada material.

Sustituyendo en la ecuación 3.2.1 a σ y ε por P/A y e/L respectivamente, tenemos:

$$\frac{P}{A} = E\,\frac{e}{L}$$

$$P = \frac{EA}{L}\,e$$

$$P = k\,e \qquad (3.2.2)$$

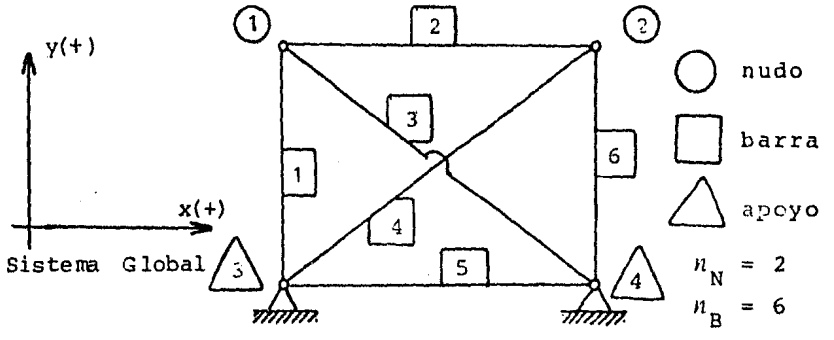
$$k = \frac{EA}{L}$$

En la ecuación 3.2.2, P es la fuerza axial actuante en la barra, e su deformación longitudinal y k es llamada su rigidez axial y es la relación de su fuerza axial entre su deformación longitudinal.

3.3 Aplicación de los principios a armaduras en dos dimensiones

Recordando que una armadura es una estructura formada por barras que solo trabajarán a fuerza axial, los nudos son articulaciones y las fuerzas externas se aplican solo en los nudos; el desplazamiento lineal de un nudo lo podemos considerar formado por los desplazamientos lineales en dos direcciones perpendiculares entre sí, así como también las fuerzas externas pueden tener dos componentes según las direcciones perpendiculares.

Al sistema de ejes ortogonales utilizados tanto para referenciar los desplazamientos lineales de los nudos, así como a las componentes de las fuerzas externas aplicadas a los nudos, se le llama sistema global de referencia (Fig. 3.3.1), el cual también, en algunos casos, se utiliza para referenciar la posición de los nudos que forman parte de la armadura.



n_N = Número de nudos libres

n_B = Número de barras

Fig. 3.3.1 Elementos que forman a una armadura plana

En el caso de que un apoyo no sea completo es decir que tenga alguna posibilidad de desplazamiento, se sustituira este apoyo por un nudo libre y una barra de rigidez infinita que impida el desplazamiento del nudo en la dirección que inicialmente estaba restringido.

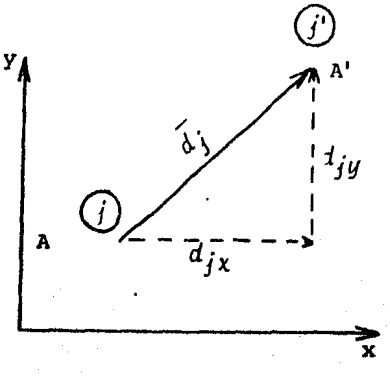
En la solución de armaduras planas por computadora, el enfoque matricial facilita la utilización de estas máquinas,

por lo tanto se plantearán todos los elementos y relaciones que intervengan en dicha solución en forma matricial, así tendremos los siguientes vectores y matrices estructurales:

a) Desplazamientos lineales de los nudos.

El desplazamiento lineal de un nudo cualquiera, j por ejemplo tendrá dos componentes los cuales serán; d_{jx} y d_{jy} - así los desplazamientos lineales de los nudos 1,2,3,4,..., - n_N, los podemos representar en forma matricial de la siguiente manera:

$$[d] = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ \cdot \\ \cdot \\ \cdot \\ d_{nN} \end{bmatrix} = \begin{bmatrix} d_{1x} \\ d_{1y} \\ --- \\ d_{2x} \\ d_{2y} \\ --- \\ d_{3x} \\ d_{3y} \\ --- \\ \cdot \\ \cdot \\ \cdot \\ d_{nNx} \\ d_{nNy} \end{bmatrix}$$



En donde:

$[d]$ = vector o matriz de desplazamientos

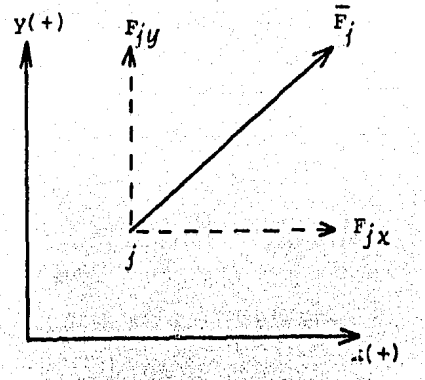
Orden de $[d] = 2_{nN} \times n_c$

n_c = número de condiciones de carga

b) Fuerzas externas concentradas aplicadas en los nudos.

Una fuerza externa aplicada a un nudo cualquiera, j por ejemplo, tendrá dos componentes las cuales seran F_{xj} y F_{yj}, así tendremos en forma matricial:

$$[F] = \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ . \\ . \\ . \\ . \\ F_{nN} \end{bmatrix} = \begin{bmatrix} F_{1x} \\ F_{1y} \\ --- \\ F_{2x} \\ F_{2y} \\ --- \\ F_{3x} \\ F_{3y} \\ --- \\ . \\ . \\ . \\ . \\ F_{nNx} \\ F_{nNy} \end{bmatrix}$$



En donde:

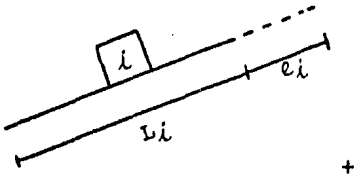
$[F]$ = vector o matriz de fuerzas externas

Orden de $[F] = 2_{nN} \times n_C$

c) Deformaciones lineales de las barras.

Estas serán los acortamientos o alargamientos de las --- barras, en forma matricial tenemos:

$$[e] = \begin{bmatrix} e_1 \\ e_2 \\ . \\ . \\ . \\ . \\ e_{nB} \end{bmatrix}$$



+ alargamiento

- acortamiento

En donde:

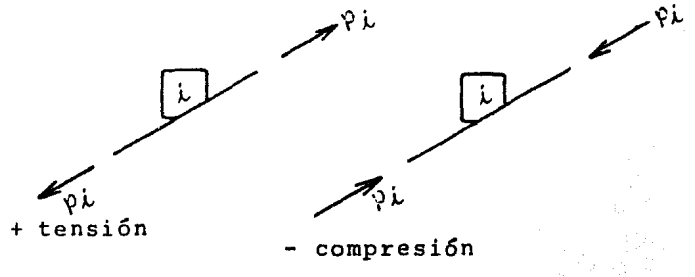
$[e]$ = vector o matriz de deformaciones lineales de las barras

Orden de $[e] = n_B \times n_C$

d) Fuerzas internas en las barras.

Estas serán las fuerzas axiales actuantes en las barras, para todas las que forman a una armadura plana tendremos:

$$[p] = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ \cdot \\ \cdot \\ p_{nB} \end{bmatrix}$$

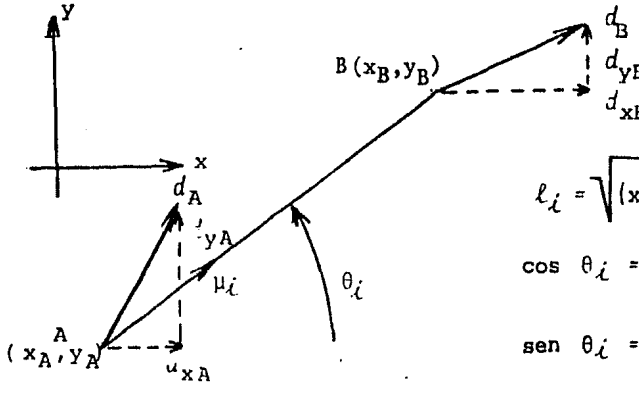


$[p]$ = vector o matriz de fuerzas axiales en las barras

Orden de $[p] = n_B \times n_c$

e) Matriz de continuidad.

La matriz de continuidad relaciona los desplazamientos lineales de los nudos con las deformaciones lineales de las barras y para obtenerla aplicaremos el principio de continuidad, primero obtengamos el valor e_i (alargamiento o acortamiento de una barra i) en función de los desplazamientos de sus nudos extremos A y B (puede verse que solo depende de los desplazamientos de dichos extremos), para ello, sea $\bar{\mu}_i$ un vector unitario paralelo a la barra i, consideremos que las barras tienen orientación de A a B.



$$\ell_i = \sqrt{(x_B - x_A)^2 + (y_B - y_A)^2}$$

$$\cos\ \theta_i = \frac{(x_B - x_A)}{\ell_i}$$

$$\text{sen}\ \theta_i = \frac{(y_B - y_A)}{\ell_i}$$

El alargamiento o acortamiento de la barra i será igual a:

$$e_i = \text{Proy}_{\bar{\mu}i}\ \bar{d}_B - \text{Proy}_{\bar{\mu}i}\ \bar{d}_A$$

pero:

$$\bar{d}_B = \begin{bmatrix} d_{xB} \\ d_{yB} \end{bmatrix}, \quad \bar{d}_A = \begin{bmatrix} d_{xA} \\ d_{yA} \end{bmatrix}, \quad \bar{\mu}_i = \begin{bmatrix} \cos\theta_i \\ \text{sen}\,\theta_i \end{bmatrix}$$

por lo tanto:

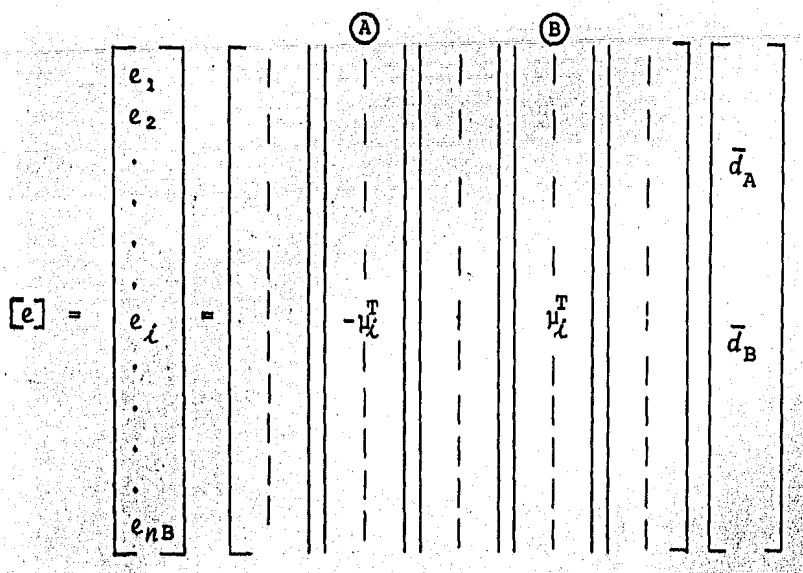
$$\text{Proy}_{\bar{\mu}i}\ \bar{d}_B = \bar{d}_B \cdot \bar{\mu}_i = \left[\bar{\mu}_i\right]^T \left[d_B\right]$$

$$\text{Proy}_{\bar{\mu}i}\ \bar{d}_A = \bar{d}_A \cdot \bar{\mu}_i \quad \left[\bar{\mu}_i\right]^T \left[d_A\right]$$

finalmente se obtiene:

$$e_i = \left[\bar{\mu}_i\right]^T \left[d_B\right] - \left[\bar{\mu}_i\right]^T \left[d_A\right] = - \left[\bar{\mu}_i\right]^T \left[d_A\right] + \left[\bar{\mu}_i\right]^T \left[d_B\right] \qquad (3.3.1)$$

Con la ayuda de la expresión 3.3.1 obtendremos la regla para obtener el renglón i de la matriz $[a]$ de continuidad - que corresponde a los coeficientes de la deformación de la barra i, es decir e_i .



$$[e] = [a][d] \qquad 3.3.2$$

en donde:

$[e]$ = vector o matriz de deformaciones lineales de las barras,

$[d]$ = vector o matriz de desplazamientos lineales de los nudos.

$[a]$ = matriz de continuidad.

Orden de $[a]$ = n_B X $2n_N$

A la relación 3.3.2 se le conoce como ecuación de continuidad. Cada columna doble de $[a]$ corresponde a un nudo libre, es decir a un nudo que puede desplazarse tanto en la dirección x como en la dirección y, nótese que todos los demás elementos del renglón i son cero, esto físicamente corresponde a que, la deformación lineal e_i de la barra i no depende de los desplazamientos lineales de los nudos que no sean sus extremos, de lo anterior se observa que la matriz de continuidad $[a]$ es porosa y lo es más para armaduras con un gran número de nudos y de barras.

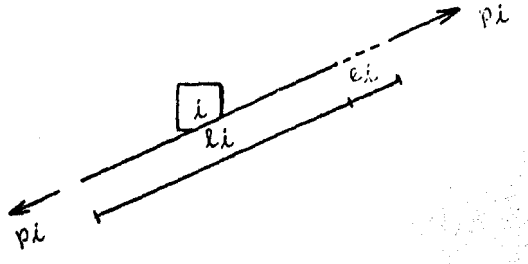
Cuando un extremo de cualquier barra (A ó B) es un apoyo completo (apoyo fijo), no es necesario hacerlo intervenir en la relación anterior ya que los desplazamientos de dicho apoyo, tanto en la dirección x como en la dirección y son nulos.

f) Matriz de rigideces de las barras.

Esta matriz relaciona las deformaciones lineales de las barras e_i con las fuerzas internas de las mismas y se obtiene aplicando la Ley de Hooke, considerando que la barra tiene una seccion transversal constante a lo largo de toda su longitud, tenemos:

$$p_i = \left(\frac{EA}{L}\right)_i e_i$$

$$p_i = k_i e_i$$



en forma matricial

$$\begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_i \\ \vdots \\ p_{nB} \end{bmatrix} = \begin{bmatrix} k_{11} & 0 & 0 & \cdots & & & \\ 0 & k_{22} & \cdots & & & & \\ & & & k_{ii} & & & \\ & & & & & & k_{nB\,nB} \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_i \\ \vdots \\ e_{nB} \end{bmatrix}$$

O bien

$$[p] = [k]\,[e] \qquad 3.3.3$$

donde:

$k_{ii} = \frac{E_i A_i}{l_i}$ = rigidez axial de la barra i

$[p]$ = vectór o matriz de fuerzas internas de las barras

$[e]$ = vector de deformaciones lineales de las barras

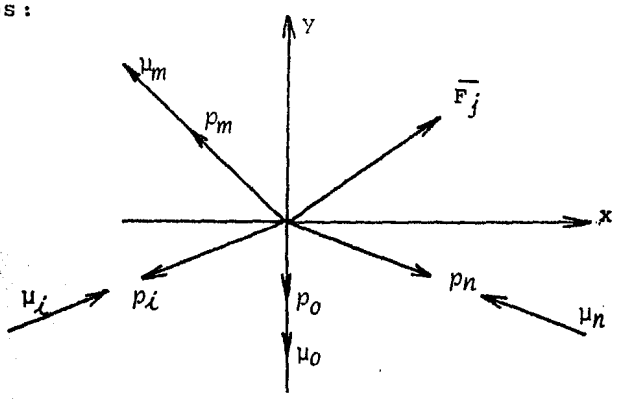
$[k]$ = matriz de rigideces de las barras

Orden de $[k] = n_B \times n_B$

Observese que la matriz $[k]$ es cuadrada, diagonal y por lo tanto bastante porosa.

g) Matriz de equilibrio.

La matriz de equilibrio relaciona las fuerzas externas aplicadas en los nudos con las fuerzas internas de las barras, y se obtiene aplicando la condición de equilibrio a todos -- los nudos de la armadura, para un nudo cualquiera j por ejemplo, tendremos:



$$\Sigma F_x = 0; \quad F_{xj} - p_i \cos\theta_i - p_m \cos\theta_m + p_n \cos\theta_n + p_o \cos\theta_o = 0$$

$$F_{xj} = + p_i \cos\theta_i + p_m \cos\theta_m - p_n \cos\theta_n - p_o \cos\theta_o$$

$$\Sigma F_y = 0; \quad F_{yj} - p_i \operatorname{sen}\theta_i + p_m \operatorname{sen}\theta_m - p_n \operatorname{sen}\theta_n - p_o \operatorname{sen}\theta_o = 0$$

$$F_{yj} = +p_i \operatorname{sen}\theta_i - p_m \operatorname{sen}\theta_m + p_n \operatorname{sen}\theta_n + p_o \operatorname{sen}\theta_o$$

o bien:

$$[F_j] = +[\mu_i]\, p_i - [\mu_m]\, p_m + [\mu_n]\, p_n - [\mu_o]\, p_o \qquad 3.3.4$$

De la expresión anterior obtendremos la regla para formar por renglones dobles a la matriz de equilibrio, cada columna corresponde a una barra, si el vector $\overline{\mu}_i$ de esa barra, sale del nudo, le corresponde $-[\mu_i]$; si entra al nudo le corresponde $+[\mu_i]$, así tendremos:

$$\begin{bmatrix} \begin{bmatrix} F_{x1} \\ F_{y1} \end{bmatrix} \\ \vdots \\ [F_j] \\ \vdots \\ [F_{nN}] \end{bmatrix} = \begin{bmatrix} | & | & | & | \\ +[\mu_i] & -[\mu_m] & +[\mu_n] & -[\mu_o] \\ | & | & | & | \end{bmatrix} \begin{bmatrix} p_1 \\ \vdots \\ p_i \\ p_m \\ p_n \\ p_o \\ \vdots \\ p_{nB} \end{bmatrix}$$

A las barras que no inciden en el nudo j le corresponde un vector nulo, puede verse que esta regla coincide con la regla para formar a la matriz $[a]$, si se forma toda la matriz de equilibrio puede verse que los renglones de esta matriz son las columnas de la matriz de continuidad, por lo que la matriz de equilibrio es la matriz de continuidad, pero transpuesta, por lo tanto también se puede obtener la matriz de equilibrio transponiendo a la matriz de continuidad, si esta ya es conocida o viceversa, así tenemos:

$$[F] = [a^T]\ [p] \qquad 3.3.5$$

Donde:

$[F]$ = vector o matriz de fuerzas externas aplicadas en los nudos.

$[p]$ = vector o matriz de fuerzas internas en las barras.

$[a^T]$ = matriz de equilibirio

Orden de $[a^T]$ = $2n_B \times n_B$

3.4 Planteamiento del método de las rigideces para la solución de armaduras.

Básicamente se tienen 3 ecuaciones matriciales derivadas como se vió anteriormente de los 3 principios del análisis estructural, las cuales son:

Principio de Continuidad:	$[e] = [a][d]$	3.3.2
Ley de Hooke:	$[p] = [k][e]$	3.3.3
Equilibrio:	$[F] = [a^T][p]$	3.3.5

En las ecuaciones anteriores los datos de que se dispone son:

$[a]$ = matriz de continuidad
$[k]$ = matriz de rigidez de las barras
$[F]$ = matriz de fuerzas externas aplicadas en los nudos
$[a^T]$ = matriz de equilibrio

Y las incógnitas son:

$[e]$ = matriz de deformación lineal de las barras
$[d]$ = matriz de desplazamientos lineales de los nudos
$[p]$ = matriz de fuerzas internas en las barras

Puede verse que se tienen 3 matrices incógnitas y se dispone de 3 ecuaciones matriciales, por lo tanto si existe solución del problema desde el punto de vista algebráico, si se sustituye la ecuación 3.3.2 en la 3.3.3 se tiene:

$$[p] = [k][a][d]$$

Y sustituyendo esta ecuación en la 3.3.5 se llega a:

$$[F] = [a^T][k][a][d] = [k][d] \qquad 3.4.1$$

En donde:

$$[K] = [a^T][k][a] \qquad 3.4.2$$

A la matriz $[K]$ se le denomina matriz de rigidez de la estructura, la cual puede verse que es cuadrada y no singular si proviene de una estructura estable, por lo tanto su inversa existe, así de la ecuación 3.4.1 tenemos:

$$[d] = [K^{-1}][F] \qquad 3.4.3$$

De la ecuación anterior se puede obtener $[d]$ ya que son conocidas $[K^{-1}]$ y $[F]$, una vez conocida $[d]$ y sustituyendo en 3.3.2 se obtiene $[e]$, para finalmente obtener $[p]$ sustituyendo $[e]$ en 3.3.3.

Al planteamiento anterior para la solución de armaduras se le conoce como método de las rigideces o de los desplazamientos ya que en dicha solución intervienen las matrices de rigidez de las barras y de rigidez de la estructura - - - - ($[k]$ y $[K]$), y en dicho método básicamente se tiene como -- incógnita la matriz de desplazamientos lineales de los nudos ($[d]$), ya que como se vió anteriormente, una vez conocida -- $[d]$ pueden obtenerse $[e]$ y $[p]$.

3.5 Ejemplos.

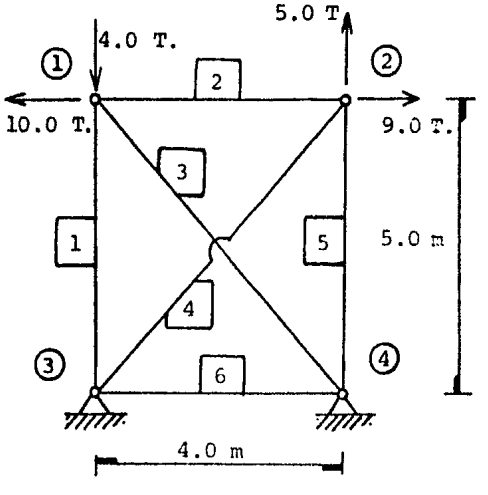
Se presentan a continuación varios ejemplos, aplicando la forma de solución dada en el inciso anterior, para ello - se desarrolló un programa de computadora en lenguaje FORTRAN, el cual escencialmente consta de un programa principal y de una serie de subrutinas para la manipulación de matrices, -- las cuales realizan las operaciones necesarias para la solución del problema, este programa requiere para resolver una

armadura plana, que se le proporcione el número de nudos, de barras y de apoyos que constituyen a la armadura plana, así como el número de condiciones de carga a que va a estar sujeta, también requiere de las coordenadas de todos los nudos - incluyendo los apoyos, además se le debe de proporcionar para todas y cada una de las barras el área, el módulo de elasticidad así como de su posición y orientación dentro de la armadura, las cuales son dadas por el nudo "inicio" y el nudo "fin" de la barra, por último se le proporciona la matriz de fuerzas.

Con los datos anteriores el programa forma, los vectores unitarios de cada barra y a partir de ellos forma la matriz de continuidad, enseguida forma la matriz de rigidez de las barras de aquí en adelante se sigue el método de solución descrito en el inciso anterior, formando y manipulando (el -- programa), todas y cada una de las matrices involucradas en la solución del problema.

El listado del programa no se presenta ya que no es el adecuado para ser utilizado en una microcomputadora, dada la gran cantidad de memoria requerida para su ejecución, detalles que se tratan más ampliamente en el capítulo IV del presente trabajo.

Ejemplo 3.5.1 Resolver la siguiente armadura:



$A_i = 1.0$ cm

$E_i = 1 \frac{Ton}{cm^2}$

$n_c = 1$

NOTA: Observese que la barra 6 por estar entre dos apoyos fijos no se deforma por lo tanto no -- resiste fuerza axial pudiéndose con esto, no considerar en el problema.

SOLUCION DE UNA ARMADURA PLANA

NUMERO DE NUDOS 4

NUMERO DE BARRAS 6

NUMERO DE CONDICIONES DE CARGA 1

NUDOS, COORDENADAS Y TIPO

NUDO	X	Y	TIPO
1	[illegible]0	500.00	"L" = LIBRE
2	[illegible]01	[illegible].00	"L" = LIBRE
3	0.00	0.00	"A" = APOYO
4	400.00	0.00	"A" = APOYO

EL VECTOR ORDEN BARRAS ES :

1	2	3	4	5	6
1.[illegible]E+0[illegible]	2.[illegible]E+0[illegible]	3.00E+00	4.[illegible]E+00	5.[illegible]E+0[illegible]	6.[illegible]E+00

ORIENTACION DE LAS BARRAS

BARRA	NUDO INICIAL	NUDO TERMINAL
1	3	1
2	1	2
3	1	4
4	3	2
5	4	2
6	3	4

EL VECTOR AREAS BARRAS ES :

1	2	3	4	5	6
1.[illegible]E+0[illegible]	1.[illegible]E+0[illegible]	1.[illegible]E+0[illegible]	1.[illegible]E+00	1.[illegible]E+0[illegible]	1.[illegible]E+00

EL VECTOR MOD. ELASTIC ES :

1	2	3	4	5	6
1.[illegible]E+0[illegible]	1.[illegible]E+0[illegible]	1.00000E+0[illegible]	1.00000E+0[illegible]	1.00000E+0[illegible]	1.00000E+00

MATRIZ F

	1
1	-1.[illegible]E+01
2	-4.[illegible]E+00
3	9.[illegible]E+00
4	5.[illegible]E+0[illegible]

MATRIZ U

	1	2
1	0.	1.0000E+00
2	1.0000E+00	0.
3	6.2470E-01	-7.8087E-01
4	6.2470E-01	7.8087E-01
5	0.	1.0000E+00
6	1.0000E+00	0.

EL VECTOR RIG. BARRAS ES :

1	2	3	4	5	6
2.0000E-03	2.5000E-03	1.5617E-03	1.5617E-03	2.0000E-03	2.5000E-03

MATRIZ A

	1	2	3	4
1	0.	1.0000E+00	0.	0.
2	-1.0000E+00	0.	1.0000E+00	0.
3	-6.2470E-01	7.8087E-01	0.	0.
4	0.	0.	6.2470E-01	7.8087E-01
5	0.	0.	0.	1.0000E+00
6	0.	0.	0.	0.

MATRIZ ATKB

	1	2	3	4	5	6
1	0.	-2.5000E-03	-9.7561E-04	0.	0.	0.
2	2.0000E-03	0.	1.2195E-03	0.	0.	0.
3	0.	2.5000E-03	0.	9.7561E-04	0.	0.
4	0.	0.	0.	1.2195E-03	2.0000E-03	0.

MATRIZ KR

	1	2	3	4
1	3.1095E-03	-7.6182E-04	-2.5000E-03	0.
2	-7.6182E-04	2.9523E-03	0.	0.
3	-2.5000E-03	0.	3.1095E-03	7.6182E-04
4	0.	0.	7.6182E-04	2.9523E-03

MATRIX O

	1
1	-5.7547642540E+03
2	-2.5198782270E+03
3	-2.2922803540E+03
4	2.2851217370E+03

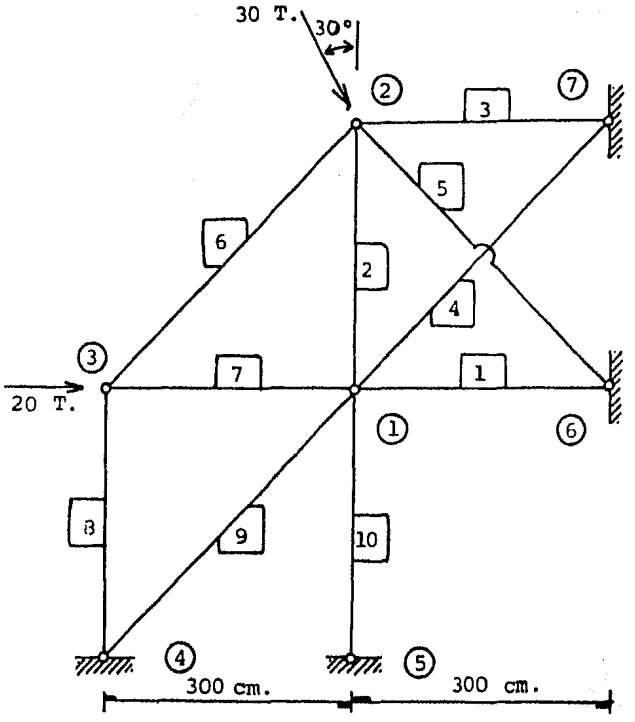
MATRIX E

	1
1	-2.8598782270E+03
2	5.4624779910E+03
3	1.3774005040E+03
4	5.5240036420E+02
5	2.2851217370E+3
6	0.

MATRIX P

	1
1	-5.6798840D00
2	0.6962840D00
3	2.4151110D00
4	5.5034E-01
5	4.5720E+00
6	0.

Ejemplo 3.5.2 Resolver la siguiente armadura



$E = 2.1 \times 10^6$ Kg/cm^2

$= 2.1 \times 10^3$ Ton/cm^2

$A_i = 25$ cm^2

$n_c = 1$

```
*REP
*
*RUNNING 6211

        SOLUCION DE UNA ARMADURA PLANA

        NUMERO DE NUDOS    7

        NUMERO DE BARRAS  10

        NUMERO DE CONDI-
        CIONES DE CARGA    1

        NUDOS, COORDENADAS Y TIPO

        NUDO       X         Y      TIPO

          1      300.00    300.00  'L' = LIBRE

          2      300.00    600.00  'L' = LIBRE

          3        0.00    300.00  'L' = LIBRE

          4        0.00      0.00  'A' = APOYO

          5      300.00      0.00  'A' = APOYO

          6      600.00    300.00  'A' = APOYO

          7      600.00    600.00  'A' = APOYO

                    EL VECTOR ORDEN BARRAS ES :

   1         2         3         4         5         6         7         8         9        10

 1.0E+00   2.0E+00   3.0E+00   4.0E+00   5.0E+00   6.0E+00   7.0E+00   8.0E+00   9.0E+00   1.0E+01

        ORIENTACION DE LAS BARRAS

        BARRA     NUDO          NUDO
                  INICIAL       TERMINAL

          1         1             6

          2         1             2

          3         2             7

          4         1             7

          5         2             6

          6         3             2

          7         3             1

          8         4             3

          9         4             1

         10         5             1
```

EL VECTOR AREAS BARRAS ES :

1	2	3	4	5	6	7	8	9	10
2.500E+01	2.500E+01	2.500E+01	2.500E+01	2.500E+01	2.500E+01	2.500E+01	2.500E+01	2.500E+01	2.500E+01

EL VECTOR MOD. ELASTIC ES :

1	2	3	4	5	6	7	8	9	10
2.100E+03	2.100E+03	2.100E+03	2.100E+03	2.100E+03	2.100E+03	2.100E+03	2.100E+03	2.100E+03	2.100E+03

MATRIZ F

	1
1	0.
2	0.
3	1.500E+01
4	-2.598E+01
5	2.000E+01
6	0.

MATRIZ U

	1	2
1	1.000E+00	0.
2	0.	1.000E+00
3	1.000E+00	0.
4	7.071E-01	7.071E-01
5	7.071E-01	-7.071E-01
6	7.071E-01	7.071E-01
7	1.000E+00	0.
8	0.	1.000E+00
9	7.071E-01	7.071E-01
10	0.	1.000E+00

EL VECTOR RIG. BARRAS ES :

1	2	3	4	5	6	7	8	9	10

1.750E+02 1.750E+02 1.750E+02 1.237E+02 1.237E+02 1.237E+02 1.750E+02 1.750E+02 1.237E+02 1.750E+02

MATRIZ A

	1	2	3	4	5	6
1	-1.000E+00	0.	0.	0.	0.	0.
2	0.	-1.000E+00	0.	1.000E+00	0.	0.
3	0.	0.	-1.000E+00	0.	0.	0.
4	-7.071E-01	-7.071E-01	0.	0.	0.	0.
5	0.	0.	-7.071E-01	7.071E-01	0.	0.
6	0.	0.	7.071E-01	7.071E-01	-7.071E-01	-7.071E-01
7	1.000E+00	0.	0.	0.	-1.000E+00	0.
8	0.	0.	0.	0.	0.	1.000E+00
9	7.071E-01	7.071E-01	0.	0.	0.	0.
10	0.	1.000E+00	0.	0.	0.	0.

MATRIZ ATKB

	1	2	3	4	5	6	7	8	9	10
1	-1.750E+02	0.	0.	-8.750E+01	0.	0.	1.750E+02	0.	8.750E+01	0.
2	0.	-1.750E+02	0.	-8.750E+01	0.	0.	0.	0.	8.750E+01	1.750E+02
3	0.	0.	-1.750E+02	0.	-8.750E+01	8.750E+01	0.	0.	0.	0.
4	0.	1.750E+02	0.	0.	8.750E+01	8.750E+01	0.	0.	0.	0.
5	0.	0.	0.	0.	0.	-8.750E+01	-1.750E+02	0.	0.	0.
6	0.	0.	0.	0.	0.	-8.750E+01	0.	1.750E+02	0.	0.

MATRIZ KR

	1	2	3	4	5	6
1	4.737E+02	1.237E+02	0.	0.	-1.750E+02	0.
2	1.237E+02	4.737E+02	0.	-1.750E+02	0.	0.
3	0.	0.	2.987E+02	0.	-6.187E+01	-6.187E+01
4	0.	-1.750E+02	0.	2.987E+02	-6.187E+01	-6.187E+01

5	-1.750E+02	0.	-6.187E+01	6.187E+01	2.369E+02	6.187E+01
6	0.	0.	-6.187E+01	-6.187E+01	6.187E+01	2.369E+02

MATRIZ D

	1
1	6.378759E-02
2	-5.328224E-02
3	6.925040E-02
4	-9.913606E-02
5	1.350036E-01
6	-4.306971E-02

MATRIZ E

	1
1	-6.378759E-02
2	-4.585383E-02
3	-6.925040E-02
4	-7.428409E-03
5	-1.190672E-01
6	-8.613943E-02
7	-7.121600E-02
8	-4.306971E-02
9	7.428409E-03
10	-5.328224E-02

MATRIZ P

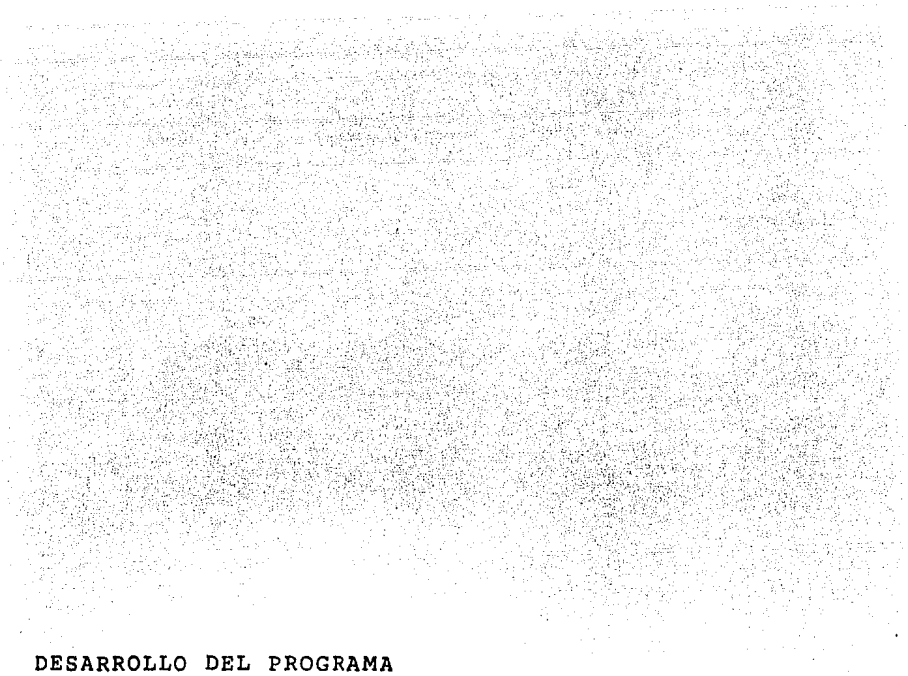
	1
1	-1.116E+01
2	-8.024E+00

```
  3  -1.212E+01
  4  -9.192E-01
  5  -1.473E+01
  6  -1.066E+01
  7  -1.246E+01
  8  -7.537E+00
  9   9.192E-01
 10  -9.324E+00

                    MATRIZ FCALC.

          1
  1   6.548E-11
  2   0.
  3   1.500E+01
  4  -2.598E+01
  5   2.000E+01
  6   0.
*!


#QUEUED
#QUEUED
#QUEUED
#QUEUED
#QUEUED
#QUEUED
#ET=4132.0 PT=1.3 IO=0.2
#
#
#
#
#
#
```



4. DESARROLLO DEL PROGRAMA

4.1 Consideraciones Generales.

En el desarrollo de un programa de computadora se deben de tener en cuenta ciertos aspectos, los cuales en general se refieren al equipo para el cual se va a desarrollar el programa y al algoritmo de solución utilizado, aunque pudiera decirse que este algoritmo debe de tener en cuenta al equipo en el cual se va a utilizar, dicho algoritmo deberá considerar tanto las ventajas como las limitaciones que ofrezca tanto la versión disponible en el equipo del lenguaje en que se vaya a codificar el programa como del equipo en sí.

El programa se desarrollará para poder ser utilizado en una microcomputadora, debido a que el lenguaje BASIC ha tenido gran difusión en estos equipos, se codificará en este lenguaje, si bien en la actualidad el desarrollo de las microcomputadoras se realiza a pasos agigantados, hoy en día (como se mencionó en el capítulo I del presente trabajo), se tienen ciertas limitaciones en cuanto a la cantidad de información que estas máquinas pueden procesar debido a la memoria limitada que poseen.

Por lo tanto el programa por realizar deberá tener en cuenta este aspecto; esto puede lograrse durante la etapa de

codificación definiendo o utilizando nuevas variables solo cuando realmente sea necesario, además de, reutilizar en - partes posteriores del programa los nombres de las varia-- bles ya definidas con anterioridad, esto puede hacerse para aquellas variables cuyos valores no se requieran posteriormente.

En algunos casos el hecho de no definir nuevas varia-- bles puede conducir a repetir en más de una ocasión una operación o grupo de operaciones para obtener por ejemplo, el valor de ciertas variables involucradas en la solución del problema; lo cual trae como consecuencia un aumento de tiempo requerido por el programa para resolver dicho problema.

Sin embargo en una microcomputadora es aceptable el - - aumento de tiempo de ejecución siempre y cuando ello nos -- permita obtener un ahorro de memoria (ya que como se ha mencionado con anterioridad, la memoria que las microcomputadoras poseen es reducida en comparación con la que poseen los grandes sistemas, la cual puede ser insuficiente para la solución de un determinado problema), memoria que a su vez el programa puede disponer al momento de su ejecución aumentando con ello el tamaño del problema que se pueda resolver.

También puede lograrse un ahorro de memoria si a la -- presentación que se le de a los resultados de salida (formatos) no sea muy elaborada, pero sí, corta, explícita y sencilla.

En las microcomputadoras podemos obtener un ahorro considerable de memoria, siempre y cuando utilicemos la totalidad de un renglón lógico, que físicamente en algunas de - - ellas puede contener hasta 254 caracteres, ya que si estos son utilizados o no la máquina de cualquier manera utiliza la memoria necesaria para almacenar la totalidad de caracteres (incluyendo espacios en blanco) que constituyan al renglón lógico correspondiente. Como se mencionó en el capítulo II del presente trabajo, se pueden incluír varias proposiciones en BASIC en un mismo renglón lógico separándolas - entre sí por dos puntos (:), sin embargo lo anterior puede traer como consecuencia que el programa en sí, sea poco - - agradable a la vista y aparente estar desordenado.

Por lo que respecta al algoritmo por utilizar, si este fuera el expuesto en el inciso 3.4 del capítulo III del presente trabajo, es decir si se formarán y manipularan todas las matrices involucradas en la solución del problema tal y como ahí se expone presenta desventajas (con respecto a un algoritmo que posteriormente se presenta) para poder ser --

utilizando en una microcomputadora, ya que como se vió, las matrices de continuidad, equilibrio y la de rigidez de las barras son bastante porosas y el orden de las matrices involucradas en la solución del problema suele ser grande, por lo anterior se necesita bastante memoria (en comparación con la que requiere el algoritmo que se presenta en el siguiente inciso) para almacenar a estas matrices.

4.2 Ensamble de matrices $[K]$ para armaduras planas

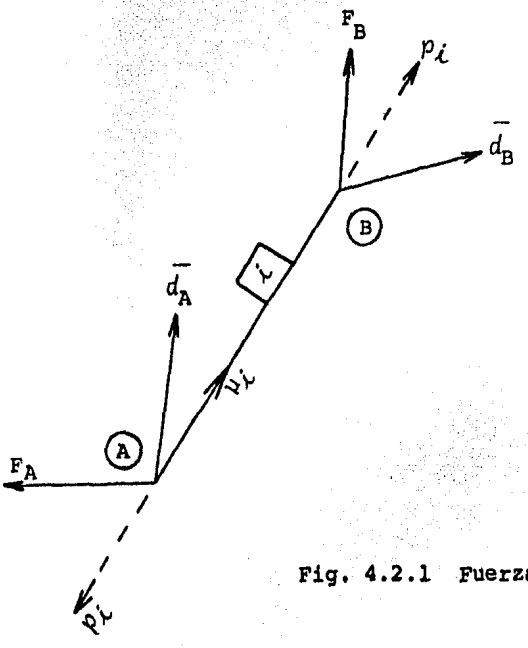
Recordando que los objetivos del análisis estructural de armaduras planas es la determinación de los vectores $\{d\}$, $\{e\}$ y $\{p\}$ lo cual como ya se mencionó con anterioridad se logra resolviendo el sistema formado por:

$$\{F\} = [K]\{d\} \qquad 3.4.1$$

Observese que la fuerza axial de una barra depende de su deformación axial la que a su vez solo depende de los desplazamientos lineales de sus extremos, por lo tanto prácticamente la única matriz que es necesario obtener es la matriz $[K]$ que solo se utiliza para obtener $\{d\}$.

Una manera de obtener $[K]$ sería como se presentó en el capítulo III, es decir efectuando el producto $[a]^T[k][a]$ y para ello sería necesario obtener a las matrices $[a]$ y $[k]$, sin embargo con el método que a continuación se presenta se puede obtener a $[K]$ sin necesidad de formar a las matrices $[a]$ y $[k]$ con lo cual se obtiene un ahorro considerable de memoria de las computadoras necesaria para el almacenamiento de estas matrices.

Considerese una barra que forma parte de una armadura (figura 4.2.1) cuyos extremos A y B en general pueden desplazarse encontrándose orientada de A a B.



$\{\mu_i\}$ = vector unitario asociado a la barra i

$\{d_A\}, \{d_B\}$ = desplazamientos lineales de los extremos A y B de la barra i respectivamente.

p_i = fuerza axial de la barra i

Fig. 4.2.1 Fuerzas actuando sobre la barra i

La fuerza axial actuante en la barra i es:

$$p_i = \left(\frac{EA}{L}\right)_i e_i = k_i e_i \qquad 3.2.2$$

donde:

$$e_i = \{\mu_i\}^T \{d_B\} - \{\mu_i\}^T \{d_A\} \qquad 3.3.1$$

La fuerza axial p_i se puede expresar vectorialmente como:

$$\{F_B\} = \{\mu_i\}\, p_i \qquad 4.2.1$$

Sustituyendo 3.3.1 y 3.2.2 en 4.2.1 tenemos

$$\{F_B\} = \{\mu_i\}\, k_i\, e_i = \{\mu_i\}\, k_i \left[\{\mu_i\}^T \{d_B\} - \{\mu_i\}^T \{d_A\}\right]$$

$$\{F_B\} = k_i \{\mu_i\}\{\mu_i\}^T\{d_B\} - k_i \{\mu_i\}\{\mu_i\}^T\{d_A\} \qquad 4.2.2$$

Igualmente puede verse que:

$$\{F_A\} = -\{\mu_i\}\, p_i$$

$$\{F_A\} = -k_i \{\mu_i\}\{\mu_i\}^T\{d_B\} + k_i \{\mu_i\}\{\mu_i\}^T\{d_A\} \qquad 4.2.3$$

A las ecuaciones 4.2.2 y 4.2.3 podemos acoplarlas y expresarlas como:

$$\begin{Bmatrix}\{F_A\}\\ \{F_B\}\end{Bmatrix} = \begin{bmatrix}+\{\mu_i\}\{\mu_i\}^T & -\{\mu_i\}\{\mu_i\}^T\\ -\{\mu_i\}\{\mu_i\}^T & +\{\mu_i\}\{\mu_i\}^T\end{bmatrix} \times k_i \begin{Bmatrix}\{d_A\}\\ \{d_B\}\end{Bmatrix} \qquad 4.2.4$$

A la matriz que interviene en la ecuación anterior se le denomina matriz de rigidez acoplada de la barra y se puede observar que relaciona vectorial y directamente a los desplazamientos lineales de los extremos de la barra con las fuerzas en dichos extremos, la ecuación 4.2.4 también se puede escribir como:

$$\begin{Bmatrix}\{F_A\}\\ \{F_B\}\end{Bmatrix} = \begin{bmatrix}[k_{AA}] & [k_{AB}]\\ [k_{BA}] & [k_{BB}]\end{bmatrix} \begin{Bmatrix}\{d_A\}\\ \{d_B\}\end{Bmatrix} \qquad 4.2.5$$

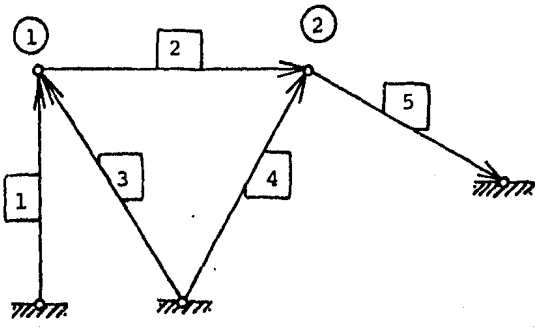
donde:

$$[k_{AA}] = [k_{BB}] = +\{\mu_i\}\{\mu_i\}^T \times k_i$$

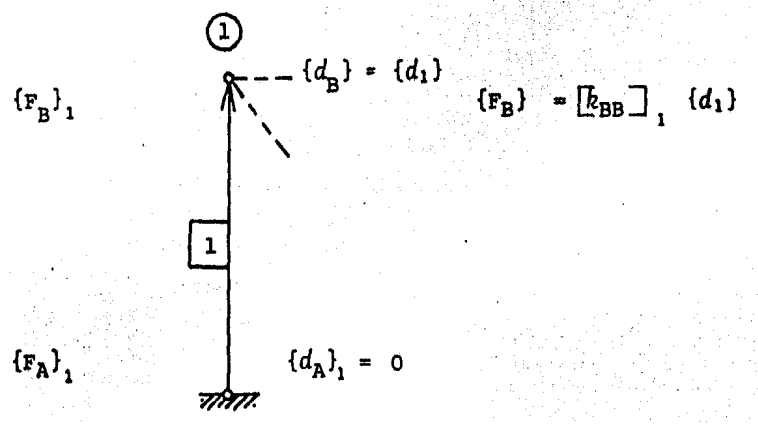
$$[k_{AB}] = [k_{BA}] = -[k_{AA}] = -\{\mu_i\}\{\mu_i\}^T \times k_i$$

Con ayuda de la ecuación 4.2.5 podemos obtener la matriz de rigidez de la armadura, para ello consideremos el siguiente ejemplo:

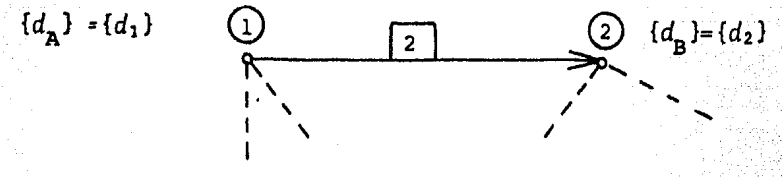
Ejemplo 4.2.1 . Obtengase la matriz de rigidez [K] de la siguiente armadura, considerese como dato la matriz acoplada de cada barra.



Aplicando la ecuación 4.2.5 a la barra [1] tenemos



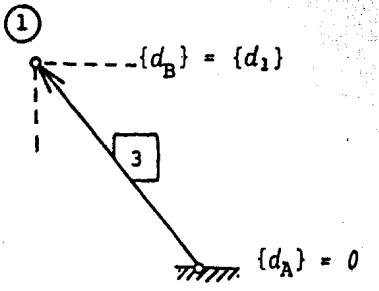
Para la barra 2.



$$\{F_A\}_2 = [k_{AA}]_2 \{d_1\} + [k_{AB}]_2 \{d_2\}$$

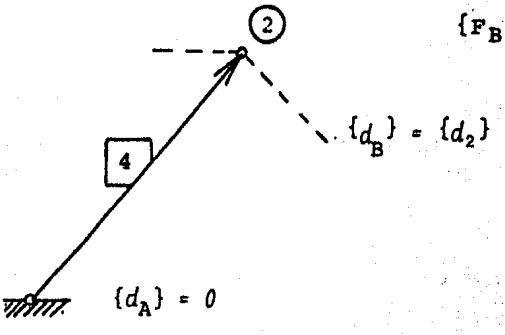
$$\{F_B\}_2 = [k_{BA}]_2 \{d_1\} + [k_{BB}]_2 \{d_2\}$$

Para la barra 3.



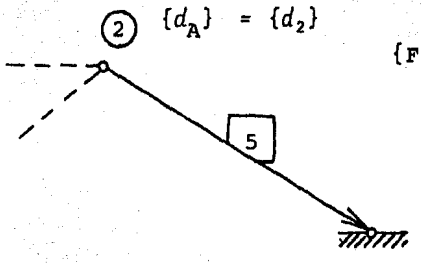
$$\{F_B\}_3 = [k_{BB}]_3 \{d_1\}$$

Para la barra 4.



$$\{F_B\}_4 = [k_{BB}]_4 \{d_2\}$$

Para la barra 5.



Por equilibrio de los nudos:

$$\{F_1\} = \{F_B\}_1 + \{F_A\}_2 + \{F_B\}_3$$

$$\{F_2\} = \{F_B\}_2 + \{F_B\}_4 + \{F_A\}_5$$

Sustituyendo a $\{F_A\}$ y $\{F_B\}$ por sus valores correspondientes tenemos;

$$\{F_1\} = [k_{BB}]_1 \{d_1\} + [k_{AA}]_2 \{d_1\} \quad [k_{AB}]_2 \{d_2\} \quad [k_{BB}]_3 \{d_1\}$$

$$\{F_1\} = \left[[k_{BB}]_1 + [k_{AA}]_2 + [k_{BB}]_3\right] \{d_1\} + [k_{AB}]_2 \{d_2\}$$

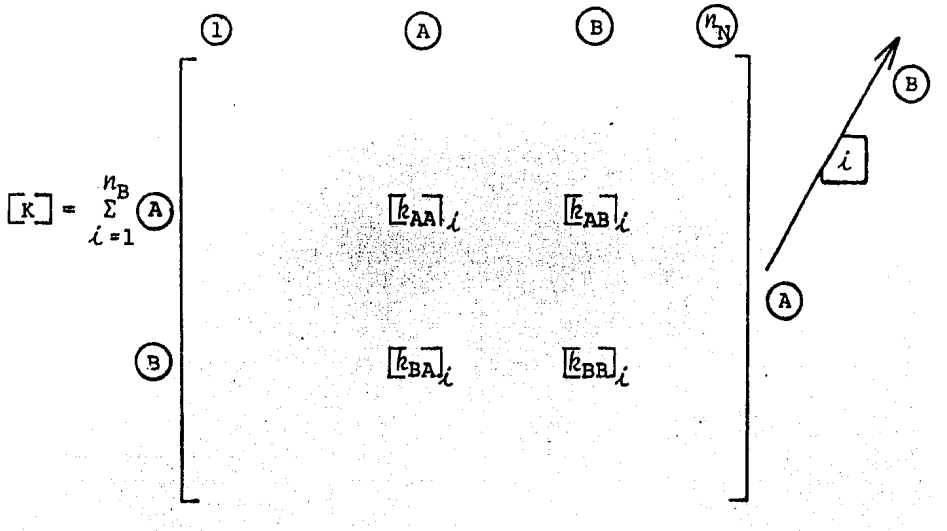
$$\{F_2\} = [k_{BA}]_2 \{d_1\} + [k_{BB}]_2 \{d_2\} + [k_{BB}]_4 \{d_2\} + [k_{AA}]_5 \{d_2\}$$

$$\{F_2\} = [k_{BA}]_2 \{d_1\} + \left[[k_{BB}]_2 + [k_{BB}]_4 + [k_{AA}]_5\right] \{d_2\}$$

y finalmente tenemos:

$$\begin{Bmatrix} \{F_1\} \\ \hline \{F_2\} \end{Bmatrix} = \left[\begin{array}{c|c} [k_{BB}]_1 + [k_{AA}]_2 + [k_{BB}]_3 & \\ \hline [k_{BA}]_2 & [k_{BB}]_2 + [k_{BB}]_4 + [k_{AA}]_5 \end{array}\right] \times \begin{Bmatrix} \{d_1\} \\ \hline \{d_2\} \end{Bmatrix}$$

A la forma como se encuentra expresada la matriz de rigidez de la armadura, se le conoce como expresión topológica, la cual es solo la descripción de la disposición de los elementos que constituyen a la armadura, observese del ejemplo anterior que puede obtenerse una regla para la obtención --- topológica de la matriz de rigidez, conocida como regla de - la suma o ensamble, la cual es:



donde:

$[K]$ = matriz de rigidez de la estructura

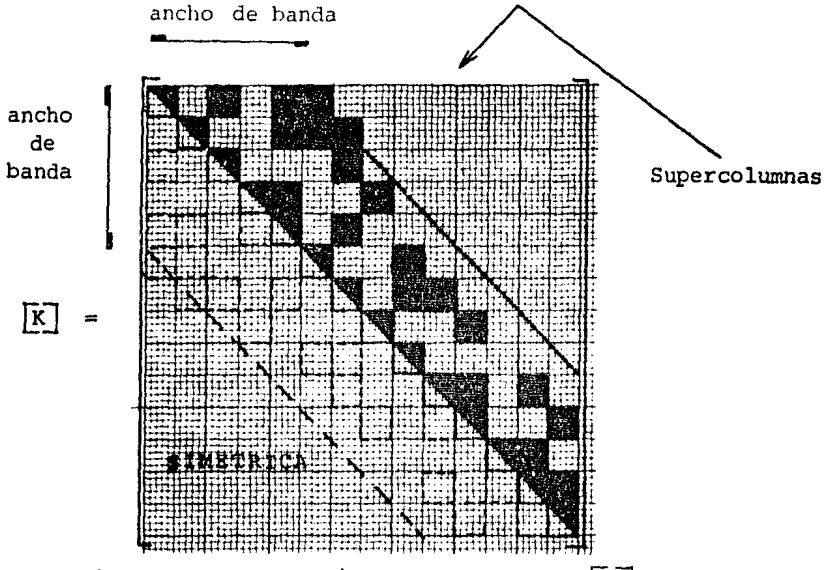
$[k_{AA}]$, $[k_{AB}]$, ... = matrices de rigidez de la barra i

n_B = número de barras

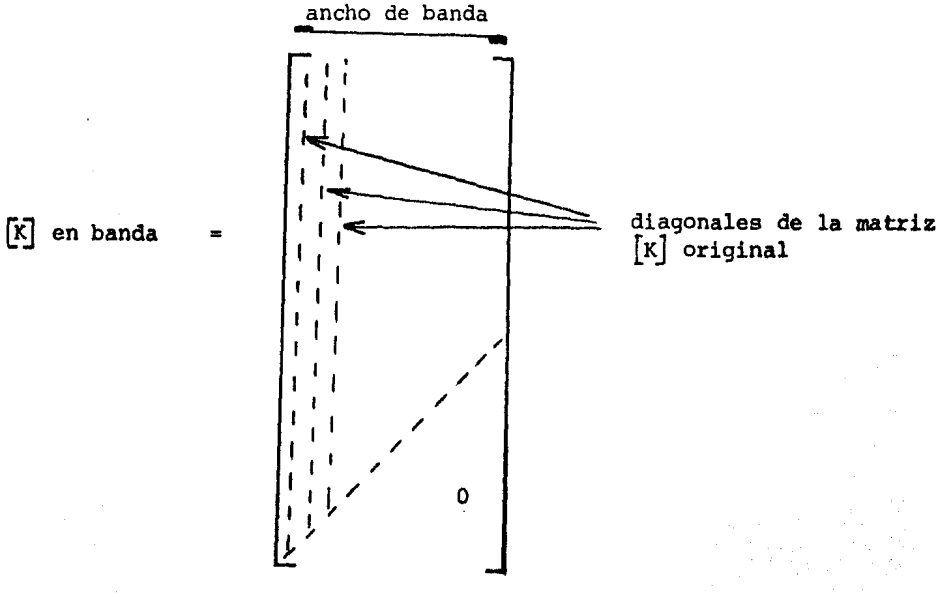
n_N = número de nudos

Obsérvese que la matriz $[K]$ de rigidez es simétrica y - además de que si se enumeran adecuadamente los nudos de la armadura existe en la matriz de rigidez una zona a ambos - extremos de la diagonal principal que se denomina ancho -- de banda y todos los elementos de la matriz $[K]$ que se encuentran fuera de esa zona son nulos, el tamaño de la banda depende de como se enumeren los nudos, como puede verse en - la forma topológica de la matriz K , siendo este ancho de banda igual a la máxima diferencia de los nudos extremos de las barras que forman a la estructura más uno.

Por lo tanto si se enumeran los nudos de manera que esa diferencia sea mínima se obtendrá un ancho de banda menor - que el número total de supercolumnas de la matriz [K] (número de nudos libres de la estructura), simbólicamente tene-- mos:



Aprovechando la simetría de la matriz [K] , podemos lograr un ahorro de memoria si el ancho de la banda es menor que el número de supercolumnas de [K] y si solo almacena-- mos a la banda de dicha matriz, almacenando, en las columnas de un arreglo las diagonales de [K] quedando de la siguiente manera:



El número de columnas de $[K]$ en banda será igual al ancho de banda multiplicado por el número de grados de libertad de un nudo libre de la estructura, que en el caso de armaduras planas es de dos.

4.3 Forma de Solución.

Podemos resumir que para la solución de nuestro problema será necesario seguir los siguientes pasos:

1.- Utilizando la regla de la suma obtener la matriz de rigidez $[K]$ de la estructura.

2.- Resolver el sistema de ecuaciones formado por:

$$\{F\} = [K]\{d\} \qquad 3.4.1$$

obteniendo con ello el vector de desplazamientos $\{d\}$.

3.- Una vez conocidos los dezplazamientos de los extremos A y B de una barra, obtener la deformación axial de la misma utilizando la ecuación:

$$e_i = \{U_i\}^T\{d_B\} - \{U_i\}^T\{d_A\} \qquad 3.3.1$$

4.- Obtener la fuerza axial actuando en cada una de las barras, por medio de la ecuación:

$$p_i = k_i\, e_i \qquad 3.3.2$$

Un punto importante dentro de la solución de estructuras por computadora es la solución de sistemas de ecuaciones lineales, desde el punto de vista matricial esto se logra invirtiendo a la matriz de coeficientes del sistema y luego premultiplicarla por el vector de terminos independientes, sin embargo la inversión de matrices por computadora requiere de un número considerable de operaciones, lo cual trae como consecuencia un aumento en el tiempo requerido para resolver el problema, por lo anterior esta manera de resolver el sistema de ecuaciones se considera ineficiente.

Existen otros métodos iterativos para resolver sistemas de ecuaciones lineales sin embargo algunos de ellos tienen

el problema de que su convergencia sea lenta necesitándose un gran número de operaciones para obtener la solución además de que no es exacta ya que depende del número de iteraciones realizadas.

Un método de solución rápido es el llamado método de Cholesky, el cual puede ser aplicado a nuestro problema, sus ventajas con respecto a los otros métodos de solución son: la solución obtenida es exacta, se realiza un número menor de operaciones aritméticas para obtener la solución en comparación con el requerido por otros métodos. Además en este método se puede considerar el ancho de banda y solo trabajar con la matriz en banda para obtener la solución del sistema.

4.4 Metodo de Cholesky

Si una matriz $[K]$ cuadrada es simétrica, de orden nxn, se puede descomponer en:

$$[K] = [L][L]^T \qquad 4.4.1$$

Donde $[L]$ es una matriz triangular inferior con elementos no nulos en la diagonal principal y por debajo de ella y nulos todos los que estan por encima de dicha diagonal, $[L]^T$ es la transpuesta de $[L]$ siendo esta triangular superior, ambas matrices de orden nxn.

Sustituyendo 4.4.1. en 3.4.1 tenemos:

$$\{F\} = [L][L]^T \{d\} \qquad 4.4.2$$

Hagamos:

$$\{Z\} = [L]^T \{d\} \qquad 4.4.3$$

Sustituyendo 4.4.3 en 4.4.2 se tiene:

$$\{F\} = [L]\{Z\} \qquad 4.4.4$$

Observese en la ecuación 4.4.4, que como $[L]$ es una matriz triangular inferior, los valores del vector auxiliar $\{Z\}$ se pueden obtener despejándolos de dicha ecuación empezando por la primera y de arriba hacia abajo y sustituyendo

cada valor obtenido en las ecuaciones siguientes, es decir - se recurre a los valores previamente obtenidos para calcu-- lar el nuevo elemento; una vez obtenido el vector $\{Z\}$ se -- puede proceder a la obtención del vector $\{d\}$ utilizando para ello la ecuación 4.4.3, observese que puede obtenerse el último elemento del vector $\{d\}$ para después obtener el pe-- núltimo y así sucesivamente hasta el primero, también se re curre a los valores obtenidos con anterioridad para obtener el nuevo elemento del vector $\{d\}$.

Por lo tanto para la obtención de los vectores $\{Z\}$ y -- $\{d\}$ es necesario conocer a las matrices $[L]$ y $[L]^T$ las cuales pueden ser obtenidas de la siguiente manera:

De la ecuación

$$[K] = [L][L]^T \qquad 4.4.1$$

Expresándola en forma explícita tenemos:

$$\begin{bmatrix} K_{11} & \cdot & \cdot & \cdot & \cdot \\ K_{21} & K_{22} & \cdot & \cdot & \cdot \\ K_{31} & K_{32} & K_{33} & \cdot & \cdot \\ K_{41} & K_{42} & K_{43} & K_{44} & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \end{bmatrix} = \begin{bmatrix} \ell_{11} & 0 & 0 & 0 & \cdot \\ \ell_{21} & \ell_{22} & 0 & 0 & \cdot \\ \ell_{31} & \ell_{32} & \ell_{33} & 0 & \cdot \\ \ell_{41} & \ell_{42} & \ell_{43} & \ell_{44} & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \end{bmatrix} \begin{bmatrix} \ell_{11} & \ell_{12} & \ell_{13} & \ell_{14} & \cdot \\ 0 & \ell_{22} & \ell_{23} & \ell_{24} & \cdot \\ 0 & 0 & \ell_{33} & \ell_{34} & \cdot \\ 0 & 0 & 0 & \ell_{44} & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \end{bmatrix}$$

Simétrica — Triangular Inferior — Triangular Superior

Desarrollando la multiplicación tenemos:

Para la primera columna:

$$K_{11} = \ell_{11} \cdot \ell_{11} \quad \therefore \quad \ell_{11} = \sqrt{K_{11}}$$
$$K_{21} = \ell_{21} \cdot \ell_{11} \quad \therefore \quad \ell_{21} = K_{21}/\ell_{11}$$
$$K_{31} = \ell_{31} \cdot \ell_{11} \quad \therefore \quad \ell_{31} = K_{31}/\ell_{11}$$
$$K_{41} = \ell_{41} \cdot \ell_{11} \quad \therefore \quad \ell_{41} = K_{41}/\ell_{11}$$

Para la segunda columna:

$$K_{22} = \ell_{21} \cdot \ell_{12} + \ell_{22} \cdot \ell_{22} \quad \therefore \quad \ell_{22} = \sqrt{K_{22} - \ell_{21}\,\ell_{12}}$$

$$K_{32} = \ell_{31} \cdot \ell_{12} + \ell_{32} \cdot \ell_{22} \quad \therefore \quad \ell_{32} = \frac{K_{32} - \ell_{31}\,\ell_{12}}{\ell_{22}}$$

$$K_{42} = \ell_{41} \cdot \ell_{12} \quad \ell_{42} \cdot \ell_{22} \quad \therefore \quad \ell_{42} = \frac{K_{42} - \ell_{41}\,\ell_{12}}{\ell_{22}}.$$

Para la tercer columna:

$$K_{33} = \ell_{31} \cdot \ell_{13} + \ell_{32} \cdot \ell_{23} + \ell_{33} \cdot \ell_{33}$$

$$\therefore \quad \ell_{33} = \sqrt{K_{33} - \ell_{31} \cdot \ell_{13} - \ell_{32}\ell_{23}}$$

$$K_{43} = \ell_{41} \cdot \ell_{13} + \ell_{42} \cdot \ell_{23} + \ell_{43} \cdot \ell_{33}$$

$$\therefore \quad \ell_{43} = \frac{K_{43} - \ell_{41} \cdot \ell_{13} - \ell_{42} \cdot \ell_{23}}{\ell_{33}}$$

Para la cuarta columna:

$$K_{44} = \ell_{41} \cdot \ell_{14} + \ell_{42} \cdot \ell_{24} + \ell_{43} \cdot \ell_{34} + \ell_{44} \cdot \ell_{44}$$

$$\therefore \quad \ell_{44} = \sqrt{K_{44} - \ell_{41} \cdot \ell_{14} - \ell_{42} \cdot \ell_{24} - \ell_{43}\,\ell_{34}}$$

Continuando el desarrollo anterior podemos llegar a las siguientes expresiones de recurrencia.

$$\ell_{ii} = \sqrt{K_{ii} - \sum_{k=1}^{i-1} \ell_{ik}\,\ell_{ki}} = \sqrt{K_{ii} - \sum_{k=1}^{i-1} \ell_{ik}^{2}}$$

$$\ell_{ij} = \frac{K_{ij} - \sum_{k=1}^{j-1} \ell_{ik} \cdot \ell_{kj}}{\ell_{jj}} \qquad \text{para} \quad i > j$$

Las expresiones anteriores nos permiten obtener a las - matrices $[L]$ y $[L]^T$.

Expresando ahora en forma explicita a la ecuación 4.4.4 se tiene:

$$\begin{Bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \\ . \\ . \\ . \\ F_n \end{Bmatrix} = \begin{bmatrix} \ell_{11} & 0 & 0 & 0 & . & . & . \\ \ell_{21} & \ell_{22} & 0 & 0 & . & . & . \\ \ell_{31} & \ell_{32} & \ell_{33} & 0 & . & . & . \\ \ell_{41} & \ell_{42} & \ell_{43} & \ell_{44} & . & . & . \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \end{bmatrix} \begin{Bmatrix} Z_1 \\ Z_2 \\ Z_3 \\ Z_4 \\ . \\ . \\ . \\ Z_n \end{Bmatrix}$$

Efectúando la operación;

$$F_1 = \ell_{11} \cdot Z_{11} \quad \therefore \quad Z_1 = F_1/\ell_{11}$$

$$F_2 = \ell_{21} \cdot Z_1 + \ell_{22} \cdot Z_2 \quad \therefore \quad Z_2 = \frac{F_2 - \ell_{21} \cdot Z_1}{\ell_{22}}$$

$$F_3 = \ell_{31} \cdot Z_1 + \ell_{32} \cdot Z_2 + \ell_{33} \cdot Z_3 \quad \therefore$$

$$Z_3 = \frac{F_3 - \ell_{31} \cdot Z_1 - \ell_{32} \cdot Z_2}{\ell_{33}}$$

$$F_4 = \ell_{41} \cdot Z_1 + \ell_{42} \cdot Z_2 + \ell_{43} \cdot Z_3 + \ell_{44} \cdot Z_4$$

$$\therefore \quad Z_4 = \frac{F_4 - \ell_{41} \cdot Z_1 - \ell_{42} \cdot Z_2 - \ell_{43} \cdot Z_3}{\ell_{44}}$$

Continuando el desarrollo anterior se puede obtener la correspondiente fórmula de recurrencia.

$$z_i = \frac{F_i - \sum_{j=1}^{i-1} \ell_{ij} \cdot z_j}{\ell_{ii}} \qquad \text{para } i > 1$$

$$z_i = \frac{F_i}{\ell_{ii}} \qquad \text{para } i = 1$$

Por último expresando a la ecuación 4.4.3 en forma explícita se tiene:

$$\begin{Bmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \\ . \\ z_n \end{Bmatrix} = \begin{bmatrix} \ell_{11} & \ell_{12} & \ell_{13} & \ell_{14} & . & . \\ & \ell_{22} & \ell_{23} & \ell_{24} & . & . \\ & & \ell_{33} & \ell_{34} & . & . \\ & & & \ell_{44} & . & . \\ & & & & . & . \\ & & & & & . \end{bmatrix} \begin{Bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ . \\ d_n \end{Bmatrix}$$

Desarrollando la multiplicación se puede obtener la fórmula de recurrencia para la obtención de los elementos del vector $\{d\}$, siendo:

$$d_i = \frac{z_i - \sum_{j=i+1}^{n} \ell_{ij} \cdot d_i}{\ell_{ii}} \qquad \text{para } i < n$$

$$d_i = z_i / \ell_{ii} \qquad \text{para } i = n$$

La cual como ya se mencionó debe de ser aplicada de abajo hacia arriba.

El método de Cholesky también puede ser aplicado a matrices simétricas en banda y se observa que las matrices $[L]$ y $[L]^T$ también resultan en banda, además estas matrices pueden ser almacenadas en la matriz $[K]$ y los vectores $\{z\}$ y $\{d\}$ en $\{F\}$, lo anterior es posible debido a la forma en que se procede a la solución del sistema; con ello no se incrementa la memoria necesaria para almacenar a estas matrices, resultando adecuado para ser utilizado en un progra

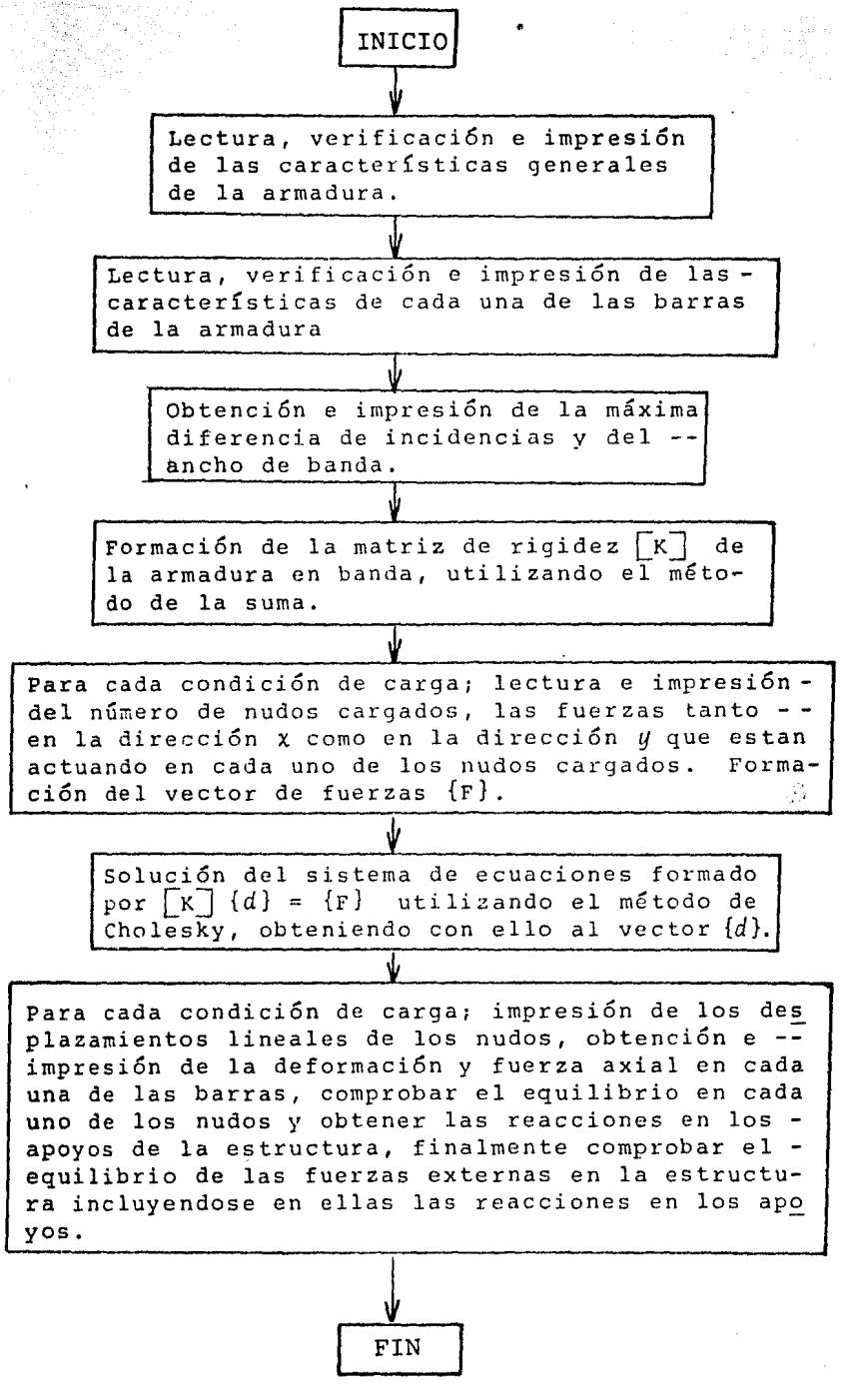
ma que requiera resolver sistemas de ecuaciones lineales -- desarrollado principalmente para microcomputadoras.

4.5 Diagrama de bloque y listado del programa.

Teniendo en cuenta lo expuesto con anterioridad se presenta a continuación el diagrama de bloque que sirvío de ayuda para desarrollar un programa para la solución de armaduras planas por microcomputadora objetivo principal de este trabajo, también se presenta un listado de dicho programa - el cual fué generado por una impresora conectada a una microcomputadora RADIO SHACK modelo II, la cual fué utilizada para probar el programa, esta microcomputadora cuenta actualmente con una capacidad de 64 K bytes de memoria en RAM.

A las variables con subíndice que aparecen en el listado del programa se les dió la mayor dimensión con tal de no exceder la capacidad de memoria en RAM de la microcomputadora, observese en ese listado que como máximo se puede resolver una armadura con nudos, barras, sujeta a condiciones de carga y con un ancho de banda de la matriz de rigidez de la armadura no mayor de , una armadura de tales características relativamente ya puede ser considerada de - buen tamaño.

Sin embargo tales dimensiones pueden ser ajustadas para problemas determinados, aumentando quizá con ello y en algunos casos el tamaño de la armadura que se pueda resolver.


INICIO
Lectura, verificación e impresión de las características generales de la armadura.
Lectura, verificación e impresión de las características de cada una de las barras de la armadura
Obtención e impresión de la máxima diferencia de incidencias y del ancho de banda.
Formación de la matriz de rigidez [K] de la armadura en banda, utilizando el método de la suma.
Para cada condición de carga; lectura e impresión del número de nudos cargados, las fuerzas tanto en la dirección x como en la dirección y que estan actuando en cada uno de los nudos cargados. Formación del vector de fuerzas {F}.
Solución del sistema de ecuaciones formado por [K] {d} = {F} utilizando el método de Cholesky, obteniendo con ello al vector {d}.
Para cada condición de carga; impresión de los desplazamientos lineales de los nudos, obtención e impresión de la deformación y fuerza axial en cada una de las barras, comprobar el equilibrio en cada uno de los nudos y obtener las reacciones en los apoyos de la estructura, finalmente comprobar el equilibrio de las fuerzas externas en la estructura incluyendose en ellas las reacciones en los apoyos.
FIN

```
100 REM  ***** A N A L I S I S   D E   A R M A D U R A S   P L A N A S *****
110 REM
120 REM         TESIS PROFESIONAL.
130 REM
140 REM         AUTOR : FERNANDO MONROY MIRANDA.
150 REM
160 REM         FACULTAD DE INGENIERIA  U N A M.
170 REM
200 DIM K9(160,26),Y9(160,4),F9(160,4),P1(180)
210 DIM I5(180,2),K5(180),T5(180)
220 PRINT "****** ANALISIS DE ARMADURAS PLANAS ******"
222 INPUT "++++++ NOMBRE DE LA ARMADURA : "!T$
224 PRINT "++++++ CARACTERISTICAS GENERALES DE LA ARMADURA ++++++"
226 INPUT "NUMERO DE BARRAS "!N1
228 C=N1: GOSUB 9500: IF C1 = 0 THEN 226
230 INPUT "NUMERO DE NUDOS(INCLUYENDO APOYOS)"!N2
232 C=N2: GOSUB 9500: IF C1 = 0 THEN 230
234 INPUT "NUMERO DE APOYOS "!N3
236 C=N3: GOSUB 9500: IF C1 = 0 THEN 234
238 INPUT "MODULO DE ELASTICIDAD PARA TODAS LAS BARRAS "!E
240 C=E: GOSUB 9500: IF C1 = 0 THEN 238
242 INPUT "NUMERO DE CONDICIONES DE CARGA "!N9
244 C=N9: GOSUB 9500: IF C1 = 0 THEN 242
260 LPRINT "***** ANALISIS DE ARMADURAS PLANAS *****"
264 LPRINT: LPRINT "++++++ ANALISIS DE LA ARMADURA : "!T$
268 LPRINT: LPRINT "++++++ CARACTERISTICAS GENERALES DE LA ARMADURA ++++++"
272 LPRINT: LPRINT "NUMERO DE BARRAS"!N1
276 LPRINT: LPRINT "NUMERO DE NUDOS(INCLUYENDO APOYOS) "!N2
282 LPRINT: LPRINT "NUMERO DE APOYOS"!N3
286 LPRINT: LPRINT "MODULO DE ELASTICIDAD PARA TODAS LAS BARRAS "!E
290 LPRINT: LPRINT "NUMERO DE CONDICIONES DE CARGA "!N9
300 PRINT "++++++ CARACTERISTICAS GEOMETRICAS DE LAS BARRAS ++++++"
310 LPRINT: LPRINT "BARRA","NUDO ORIGEN","NUDO DESTINO","LONGITUD","AREA","ANGULO"
320 FOR J=1 TO N1
330     INPUT "BARRA "!P1(1)
340     INPUT "NUDO ORIGEN "!P1(2)
350     INPUT "NUDO DESTINO "!P1(3)
360     INPUT "LONGITUD"!P1(4)
370     INPUT "AREA "!P1(5)
380     INPUT "ANGULO "!P1(6)
390     PRINT "BARRA         "!P1(1)
400     PRINT "NUDO ORIGEN   "!P1(2)
410     PRINT "NUDO DESTINO  "!P1(3)
420     PRINT "LONGITUD      "!P1(4)
430     PRINT "AREA          "!P1(5)
440     PRINT "ANGULO        "!P1(6)
450     FOR I=1 TO 5
460         C=P1(I): GOSUB 9500: IF C1=0 THEN 500
470     NEXT I
480     GOTO 600
500     PRINT "..... INTRODUZCA LOS DATOS CORRECTOS DE LA BARRA ANTERIOR"
510     GOTO 330
600     INPUT "ESTAN CORRECTOS (SI O NO) "!T$
610     IF T$="NO" THEN 500
620     LPRINT: FOR I=1 TO 6: LPRINT P1(I),: NEXT I: LPRINT
630     I5(P1(1),1)=P1(2): I5(P1(1),2)=P1(3)
640     K5(P1(1))=E*P1(5)/P1(4): T5(P1(1))=P1(6)*0.0174532925
650 NEXT J
2000 REM OBTENCION DEL ANCHO DE BANDA ************************************
```

```
2015 N4=N2-N3:R9=2*N4:S=0
2030 FOR I=1 TO N1
2040      I1=I5(I,1):I2=I5(I,2)
2060      IF I1 > N4 THEN 2150
2070      IF I2 > N4 THEN 2150
2080      S1=ABS(I2-I1)
2120      IF S1 <= S THEN 2170
2130      S=S1
2140      GOTO 2170
2150      LPRINT: LPRINT "LA BARRA ";I;" NO CONTRIBUYE AL ANCHO DE BANDA"
2170 NEXT I
2190 LPRINT :LPRINT "LA MAXIMA DIFERNCIA DE INCIDENCIAS ES ";S
2200 C9=S+1
2220 LPRINT:LPRINT "EL ANCHO DE BANDA ES";C9:LPRINT
2230 C9=2*C9
2240 LPRINT "NUMERO DE COLUMNAS DE LA MATRIZ [K] EN BANDA ";C9: LPRINT
2950 REM        FORMACION DE LA MATRIZ DE RIGIDEZ DE LA ESTRUCTURA ********
3000 FOR I=1 TO N1
3020      I1=I5(I,1): I2=I5(I,2)
3030      C=K5(I)*COS(T5(I))*COS(T5(I))
3040      C1=K5(I)*COS(T5(I))*SIN(T5(I))
3050      S=K5(I)*SIN(T5(I))*SIN(T5(I))
3060      S1=1.0
3070      IF I1 > N4 THEN 3230
3080      IF I2 > N4 THEN 3200
3090      I2=I1
3100      GOSUB 9000                              ' K AA I1,I1, +1.0
3110      I1=I5(I,2) : I2=I5(I,2)
3120      GOSUB 9000
3130      I1=I5(I,1)
3140      S1=-1.0
3150      GOSUB 9000
3160      I1=I2
3170      I2=I5(I,1)
3180      GOSUB 9000
3190      GOTO 3280
3200      I2=I1
3210      GOSUB 9000
3220      GOTO 3280
3230      IF I2 > N4 THEN 3280
3240      I1=I2
3250      GOSUB 9000
3280 NEXT I
4005 REM           LECTURA DE LAS FUERZAS EN LOS NUDOS
4050 FOR J=1 TO N9
4060      INPUT "NOMBRE DE LA CONDICION DE CARGA ";T$
4080      LPRINT:LPRINT "CONDICION DE CARGA # ";J
4090      LPRINT: LPRINT TAB(10);T$
4092      INPUT "NUMERO DE NUDOS CARGADOS, PARA ESTA CONDICION ";L9
4094      LPRINT:LPRINT "NUMERO DE NUDOS CARGADOS ";L9
4100      LPRINT: LPRINT "NUDO","FUERZA EN X","FUERZA EN Y": LPRINT
4120      FOR J1=1 TO L9
4125           INPUT "NUDO CARGADO";I
4130           I2=I*2:I1=I2-1
4150           INPUT "FUERZA EN X";Y9(I1,J): INPUT "FUERZA EN Y ";Y9(I2,J)
4160           LPRINT I,Y9(I1,J),Y9(I2,J)
4170           F9(I1,J)=Y9(I1,J):F9(I2,J)=Y9(I2,J)
4190      NEXT J1
4200      PRINT
4210 NEXT J
```

```
5045 REM        CHOLESKY 1A PARTE
5050 L9=R9-C9+1
5060 FOR I=1 TO R9
5070     J8=C9
5080     IF I <= L9 THEN 5100
5090     J8=R9-I+1
5100     FOR J=1 TO J8
5110         I1=I:J1=1:S=0:J2=J
5150         I1=I1-1:J1=J1+1
5170         IF I1 < 0 THEN 5220
5180         J2=J2+1
5190         IF J2 > C9 THEN 5220
5200         S=S+K9(I1,J1)*K9(I1,J2)
5210         GOTO 5150
5220         IF J > 1 THEN 5250
5222         IF K9(I,J)-S > 0 THEN 5230
5224         LPRINT "***** MATRIZ DE RIGIDEZ NO DEFINIDA POSISTIVAMENTE *****"
5226         LPRINT: LPRINT "                    EL VALOR "K9(I,J)-S : STOP
5230         K9(I,J)=SQR(K9(I,J)-S)
5240         GOTO 5260
5250         K9(I,J)=(K9(I,J)-S)/K9(J,J)
5260     NEXT J
5270 NEXT I
5535 REM         CHOLESKY 2A PARTE
5540 FOR J=1 TO N9
5550     FOR I=1 TO R9
5560         I1=I : J1=1 : S=0.0
5590         I1=I1-1
5600         IF I1<=0 THEN 5650
5610         J1=J1+1
5620         IF J1>C9 THEN 5650
5630         S=S+K9(I1,J1)*Y9(I1,J)
5640         GOTO 5590
5650         Y9(I,J)=(Y9(I,J)-S)/K9(I,1)
5660     NEXT I
5670 NEXT J
5680 REM         CHOLESKY 3A PARTE
5690 FOR J=1 TO N9
5700     FOR I=R9 TO 1 STEP -1
5710         I1=I : J1=1 : S=0.0
5740         I1=I1+1
5750         IF I1>R9 THEN 5800
5760         J1=J1+1
5770         IF J1>C9 THEN 5800
5780         S=S+K9(I,J1)*Y9(I1,J)
5790         GOTO 5740
5800         Y9(I,J)=(Y9(I,J)-S)/K9(I,1)
5810     NEXT I
5820 NEXT J
6000 FOR J=1 TO N9
6020     LPRINT : LPRINT "SOLUCION PARA LA CONDICION DE CARGA # "J :LPRINT
6060     LPRINT "DESPLAZAMIENTOS DE LOS NUDOS"
6080     LPRINT: LPRINT "NUDO","DESPLAZAMIENTO EN X","DESPLAZAMIENTO EN Y"
6085     C=0:C1=0
6090     FOR I=1 TO N4
6110         LPRINT: LPRINT I,Y9(I*2-1,J)," ",Y9(I*2,J)
6112         C=C+F9(I*2-1,J) : C1=C1+F9(I*2,J)
6120     NEXT I
7010     LPRINT : LPRINT "DEFORMACION Y FUERZA AXIAL DE LAS BARRAS"
7030     LPRINT: LPRINT "BARRA","DEFORMACION","FUERZA AXIAL"
```

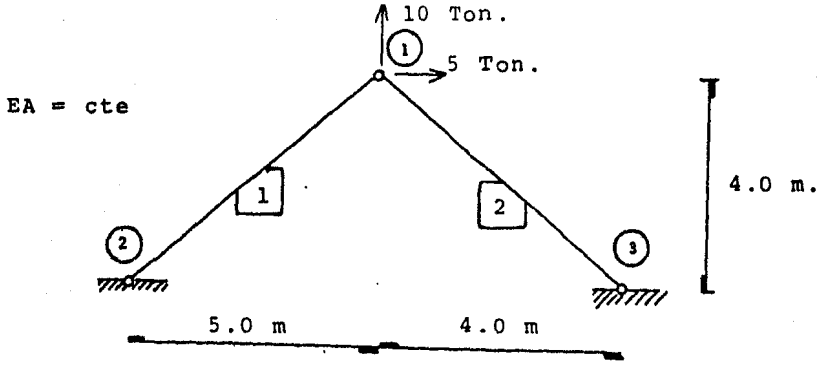
```
7040      FOR I=1 TO N1
7050          I1=I5(I,1) : I2=I5(I,2)
7070          IF I1 > N4 THEN 7110
7080          D1=Y9(I1*2-1,J)
7090          D2=Y9(I1*2,J)
7100          GOTO 7130
7110          D1=0.0 : D2=0.0
7130          IF I2 > N4 THEN 7170
7140          D3=Y9(I2*2-1,J)
7150          D4=Y9(I2*2,J)
7160          GOTO 7200
7170          D3=0.0 : D4=0.0
7200          J8=(D3-D1)*COS(T5(I))+(D4-D2)*SIN(T5(I))
7210          P1(I)=K5(I)*J8
7230          LPRINT : LPRINT I,J8,P1(I)
7240      NEXT I
7500      REM COMPROBACION DEL EQUILIBRIO Y OBTENCION DE LAS REACCIONES
7510      REM                EN LOS APOYOS.
7520      FOR I=1 TO N1
7530          I1=I5(I,1) : I2=I5(I,2)
7560          F1=P1(I)*COS(T5(I))
7570          F2=P1(I)*SIN(T5(I))
7580          F9(I1*2-1,J)=F9(I1*2-1,J) + F1
7590          F9(I1*2,J)=F9(I1*2,J) + F2
7600          F9(I2*2-1,J)=F9(I2*2-1,J) - F1
7610          F9(I2*2,J)=F9(I2*2,J) - F2
7620      NEXT I
7640      LPRINT : LPRINT "COMPROBACION DEL EQUILIBRIO EN LOS NUDOS"
7660      LPRINT: LPRINT "NUDO","SUMA DE FUERZAS EN X","SUMA DE FUERZAS EN Y"
7670      FOR I=1 TO N4
7690          LPRINT :LPRINT I,F9(I*2-1,J)," ",F9(I*2,J)
7700      NEXT I
7720      LPRINT : LPRINT "REACCIONES EN LOS APOYOS"
7740      LPRINT: LPRINT "APOYO","REACCION EN X","REACCION EN Y"
7750      FOR I=N4+1 TO N2
7770          LPRINT : LPRINT I,-F9(I*2-1,J),-F9(I*2,J)
7772          C=C-F9(I*2-1,J) : C1=C1-F9(I*2,J)
7780      NEXT I
7800      LPRINT: LPRINT "COMPROBACION DEL EQUILIBRIO EXTERNO(INCLUYENDO LAS REACCIONES)"
7810      LPRINT: LPRINT "SUMA DE FUERZAS EN X = ";C
7820      LPRINT: LPRINT "SUMA DE FUERZAS EN Y = ";C1
8900  NEXT J
8910  STOP
9000  REM          SUBRUTINA PARA ENSAMBLAR.
9050  I3=2*I1-1 : J3=2*I2-1
9105  IF J3 >= I3 THEN 9125
9115  GOTO 9400
9125  J3=J3-I3+1
9150  K9(I3,J3)=K9(I3,J3)+C*S1
9200  K9(I3,J3+1)=K9(I3,J3+1)+C1*S1
9210  J3=J3-1
9220  IF J3 > 0 THEN 9250
9230  GOTO 9300
9250  K9(I3+1,J3)=K9(I3+1,J3)+C1*S1
9300  K9(I3+1,J3+1)=K9(I3+1,J3+1)+S*S1
9400  RETURN
9500  REM        SUBRUTINA PARA VERIFICAR VALOR DIFERENTE DE CERO
9510  IF C > 0 THEN C1=1 ELSE C1=0: PRINT "***** VALOR(ES) ILEGAL(ES)"
9520  RETURN
9999  END
```

4.6 Ejemplos de Aplicación

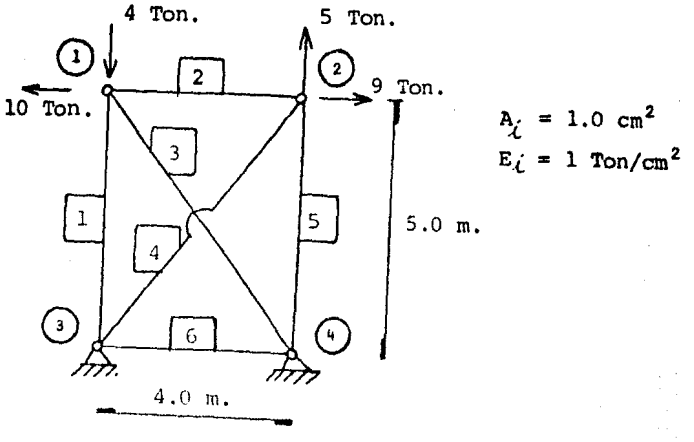
Se presentan a continuación varios ejemplos que fueron resueltos utilizando el programa presentado en el inciso - anterior, no se presenta un instructivo para la utilización del programa debido a que es interactivo, es decir la micro computadora al ejecutar el programa especifica los datos -- que necesita para continuar con la solución del problema. - Solo hay que tener en cuenta que el programa no verifica la congruencia de las unidades utilizadas, por lo que los valo res deben de ser proporcionados en unidades congruentes.

La convención de signos utilizada es la misma que se -- uso en las expresiones desarrolladas en los capítulos III y IV del presente trabajo, los ejemplos son los siguientes:

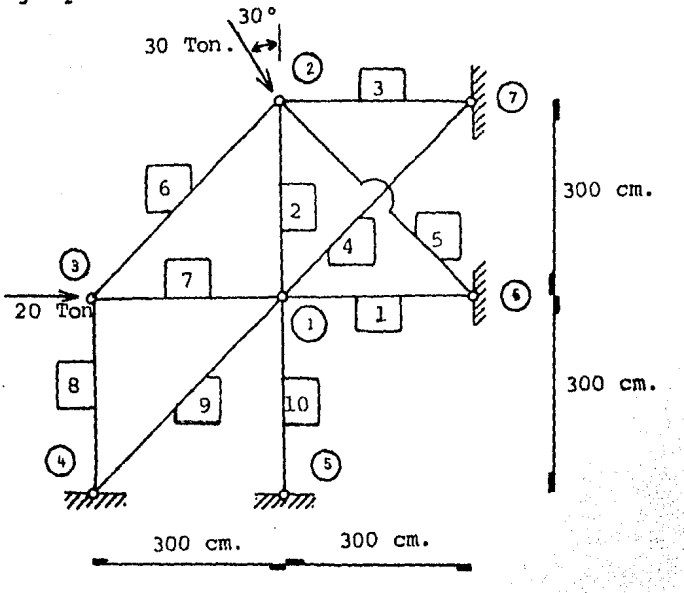
Ejemplo 4.6.1 Resuelva la siguiente armadura



Ejemplo 4.6.2 Resuelva la siguiente armadura



Ejemplo 4.6.3 Resuelva la siguiente armadura



$E = 2.1 \times 10^6$ kg/cm^2

$= 2.1 \times 10^3$ Ton/cm^2

$A_i = 25$ cm^2

$n_c = 1$

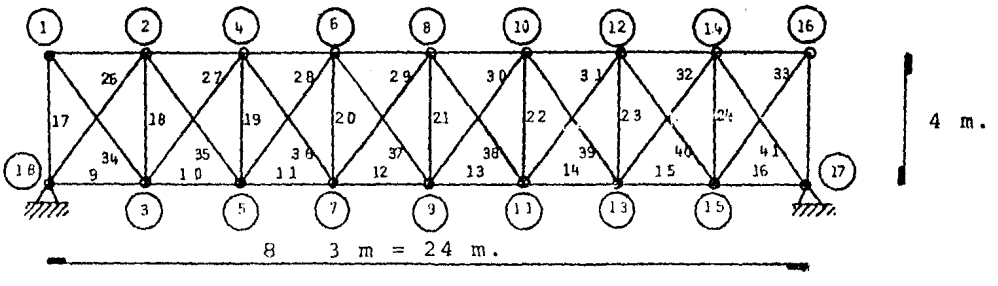
Ejemplo 4.6.4 Resuelva la siguiente armadura

$E = 2.1 \times 10^6$ Kg/cm^2

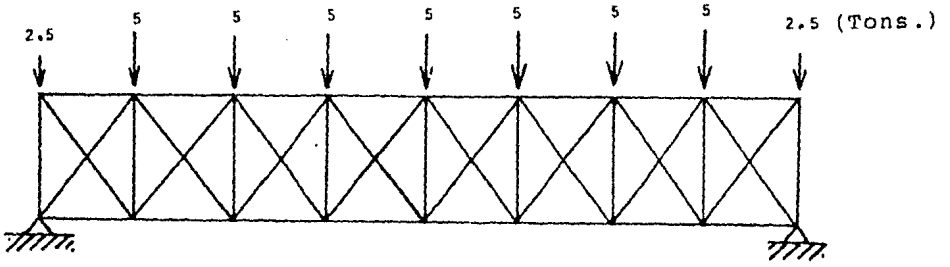
$A_i = 30$ cm^2, $i = 1, 2, \ldots 8$

$A_i = 20$ cm^2, $i = 9, 10, \ldots 16$

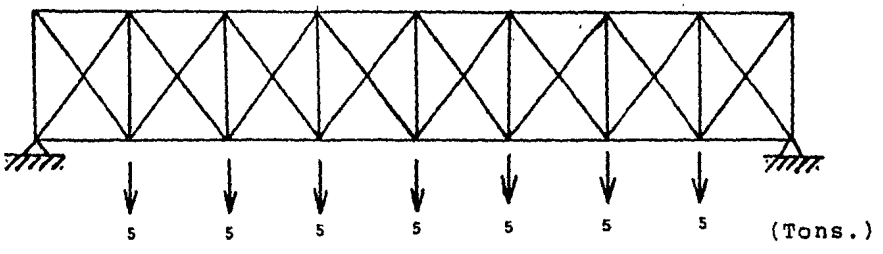
$A_i = 25$ cm^2, $i = 17, 18, \ldots 41$



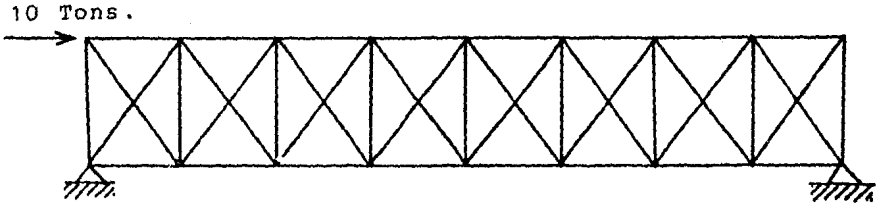
Primera condición de carga



Segunda condición de carga



Tercera condición de carga



***** ANALISIS DE ARMADURAS PLANAS *****

+++++ ANALISIS DE LA ARMADURA : EJEMPLO 4.6.1

+++++ CARACTERISTICAS GENERALES DE LA ARMADURA +++++

NUMERO DE BARRAS 2

NUMERO DE NUDOS(INCLUYENDO APOYOS) 3

NUMERO DE APOYOS 2

MODULO DE ELASTICIDAD PARA TODAS LAS BARRAS 1

NUMERO DE CONDICIONES DE CARGA 1

BARRA	NUDO ORIGEN	NUDO DESTINO	LONGITUD	AREA	ANGULO
1	2	1	6.403	1	38.66
2	1	3	5.657	1	-45

LA BARRA 1 NO CONTRIBUYE AL ANCHO DE BANDA

LA BARRA 2 NO CONTRIBUYE AL ANCHO DE BANDA

LA MAXIMA DIFERNCIA DE INCIDENCIAS ES 0

EL ANCHO DE BANDA ES 1

NUMERO DE COLUMNAS DE LA MATRIZ [K] EN BANDA 2

CONDICION DE CARGA # 1

CARGA VERTICAL

NUMERO DE NUDOS CARGADOS 1

NUDO	FUERZA EN X	FUERZA EN Y
1	5	10

SOLUCION PARA LA CONDICION DE CARGA # 1

DESPLAZAMIENTOS DE LOS NUDOS

NUDO	DESPLAZAMIENTO EN X	DESPLAZAMIENTO EN Y
1	31.8539	69.567

DEFORMACION Y FUERZA AXIAL DE LAS BARRAS

BARRA	DEFORMACION	FUERZA AXIAL
1	60.442	10.6719
2	26.6672	4.71401

COMPROBACION DEL EQUILIBRIO EN LOS NUDOS

NUDO	SUMA DE FUERZAS EN X	SUMA DE FUERZAS EN Y
1	-2.38419E-07	-1.66893E-06

REACCIONES EN LOS APOYOS

APOYO	REACCION EN X	REACCION EN Y
2	-8.33331	-6.66669
3	3.33331	-3.33331

COMPROBACION DEL EQUILIBRIO EXTERNO(INCLUYENDO LAS REACCIONES)

SUMA DE FUERZAS EN X = -2.38419E-07

SUMA DE FUERZAS EN Y = -1.66893E-06

***** ANALISIS DE ARMADURAS PLANAS *****

+++++ ANALISIS DE LA ARMADURA : EJEMPLO 4.6.2

+++++ CARACTERISTICAS GENERALES DE LA ARMADURA +++++

NUMERO DE BARRAS 6

NUMERO DE NUDOS(INCLUYENDO APOYOS) 4

NUMERO DE APOYOS 2

MODULO DE ELASTICIDAD PARA TODAS LAS BARRAS 1

NUMERO DE CONDICIONES DE CARGA 1

BARRA	NUDO ORIGEN	NUDO DESTINO	LONGITUD	AREA	ANGULO
1	3	1	500	1	90
2	1	2	400	1	0
3	1	4	640.31	1	-51.34
4	3	2	640.31	1	51.34
5	4	2	500	1	90
6	3	4	400	1	0

LA BARRA 1 NO CONTRIBUYE AL ANCHO DE BANDA

LA BARRA 3 NO CONTRIBUYE AL ANCHO DE BANDA

LA BARRA 4 NO CONTRIBUYE AL ANCHO DE BANDA

LA BARRA 5 NO CONTRIBUYE AL ANCHO DE BANDA

LA BARRA 6 NO CONTRIBUYE AL ANCHO DE BANDA

LA MAXIMA DIFERNCIA DE INCIDENCIAS ES 1

EL ANCHO DE BANDA ES 2

NUMERO DE COLUMNAS DE LA MATRIZ [K] EN BANDA 4

CONDICION DE CARGA # 1

PRIMERA

NUMERO DE NUDOS CARGADOS 2

NUDO	FUERZA EN X	FUERZA EN Y
1	-10	-4
2	9	5

SOLUCION PARA LA CONDICION DE CARGA # 1

DESPLAZAMIENTOS DE LOS NUDOS

NUDO	DESPLAZAMIENTO EN X	DESPLAZAMIENTO EN Y
1	-5754.72	-2839.07
2	-2292.25	2205.11

DEFORMACION Y FUERZA AXIAL DE LAS BARRAS

BARRA	DEFORMACION	FUERZA AXIAL
1	-2839.87	-5.67975
2	3462.47	8.65618
3	1377.4	2.15114
4	352.41	.550374
5	2285.11	4.57023
6	0	0

COMPROBACION DEL EQUILIBRIO EN LOS NUDOS

NUDO	SUMA DE FUERZAS EN X	SUMA DE FUERZAS EN Y
1	-6.91414E-06	-2.5034E-06
2	2.68221E-07	9.53674E-07

REACCIONES EN LOS APOYOS

APOYO	REACCION EN X	REACCION EN Y
3	-.343817	5.24990
4	1.34381	-6.24998

COMPROBACION DEL EQUILIBRIO EXTERNO(INCLUYENDO LAS REACCIONES)

SUMA DE FUERZAS EN X = -6.67572E-06

SUMA DE FUERZAS EN Y = -1.43051E-06

***** ANALISIS DE ARMADURAS PLANAS *****

+++++ ANALISIS DE LA ARMADURA : EJEMPLO 4.6.3

+++++ CARACTERISTICAS GENERALES DE LA ARMADURA +++++

NUMERO DE BARRAS 10

NUMERO DE NUDOS(INCLUYENDO APOYOS) 7

NUMERO DE APOYOS 4

MODULO DE ELASTICIDAD PARA TODAS LAS BARRAS 2100

NUMERO DE CONDICIONES DE CARGA 1

BARRA	NUDO ORIGEN	NUDO DESTINO	LONGITUD	AREA	ANGULO
1	1	6	300	25	0
2	1	2	300	25	90
3	2	7	300	25	0
4	1	7	424	25	45
5	2	6	424	25	-45
6	3	2	424	25	45
7	3	1	300	25	0
8	4	3	300	25	90
9	4	1	424	25	45
10	5	1	300	25	90

LA BARRA 1 NO CONTRIBUYE AL ANCHO DE BANDA

LA BARRA 3 NO CONTRIBUYE AL ANCHO DE BANDA

LA BARRA 4 NO CONTRIBUYE AL ANCHO DE BANDA

LA BARRA 5 NO CONTRIBUYE AL ANCHO DE BANDA

LA BARRA 8 NO CONTRIBUYE AL ANCHO DE BANDA

LA BARRA 9 NO CONTRIBUYE AL ANCHO DE BANDA

LA BARRA 10 NO CONTRIBUYE AL ANCHO DE BANDA

LA MAXIMA DIFERNCIA DE INCIDENCIAS ES 2

EL ANCHO DE BANDA ES 3

NUMERO DE COLUMNAS DE LA MATRIZ [K] EN BANDA 6

CONDICION DE CARGA # 1

PRIMERA

NUMERO DE NUDOS CARGADOS 2

NUDO	FUERZA EN X	FUERZA EN Y
2	15	-25.98
3	20	0

SOLUCION PARA LA CONDICION DE CARGA # 1

DESPLAZAMIENTOS DE LOS NUDOS

NUDO	DESPLAZAMIENTO EN X	DESPLAZAMIENTO EN Y
1	.0637738	-.0532648
2	.0692388	-.099094
3	.134983	-.0430762

DEFORMACION Y FUERZA AXIAL DE LAS BARRAS

BARRA	DEFORMACION	FUERZA AXIAL
1	-.0637738	-11.1604
2	-.0458292	-8.02011
3	-.0692388	-12.1168
4	-7.43090E-03	-.920109
5	-.119029	-14.7383
6	-.0860989	-10.6608
7	-.0712094	-12.4617
8	-.0430762	-7.53834
9	7.43090E-03	.920109
10	-.0532648	-9.32134

COMPROBACION DEL EQUILIBRIO EN LOS NUDOS

NUDO	SUMA DE FUERZAS EN X	SUMA DE FUERZAS EN Y
1	6.13928E-06	6.67572E-06
2	-2.30419E-06	1.43051E-06
3	1.90735E-06	-9.53674E-07

REACCIONES EN LOS APOYOS

APOYO	REACCION EN X	REACCION EN Y

4	-.650615	6.88773
5	0	9.32135
6	-21.582	10.4216
7	-12.7674	-.650615

COMPROBACION DEL EQUILIBRIO EXTERNO(INCLUYENDO LAS REACCIONES)

SUMA DE FUERZAS EN X = 3.8147E-06

SUMA DE FUERZAS EN Y = 8.04663E-06

***** ANALISIS DE ARMADURAS PLANAS *****

***** ANALISIS DE LA ARMADURA : EJEMPLO 4.6.4

***** CARACTERISTICAS GENERALES DE LA ARMADURA *****

NUMERO DE BARRAS 41

NUMERO DE NUDOS(INCLUYENDO APOYOS) 18

NUMERO DE APOYOS 2

MODULO DE ELASTICIDAD PARA TODAS LAS BARRAS 2.1E+06

NUMERO DE CONDICIONES DE CARGA 3

BARRA	NUDO ORIGEN	NUDO DESTINO	LONGITUD	AREA	ANGULO
1	1	2	300	30	0
2	2	4	300	30	0
3	4	6	300	30	0
4	6	8	300	30	0
5	8	10	300	30	0
6	10	12	300	30	0
7	12	14	300	30	0
8	14	16	300	30	0
9	18	3	300	20	0
10	3	5	300	20	0
11	5	7	300	20	0
12	7	9	300	20	0
13	9	11	300	20	0
14	11	13	300	20	0
15	13	15	300	20	0
16	15	17	300	20	0
17	18	1	400	25	90
18	3	2	400	25	90
19	5	4	400	25	90
20	7	6	400	25	90
21	9	8	400	25	90

22	11	10	400	25	90
23	13	12	400	25	90
24	15	14	400	25	90
25	17	16	400	25	90
26	18	2	500	25	53.13
27	3	4	500	25	53.13
28	5	6	500	25	53.13
29	7	8	500	25	53.13
30	9	10	500	25	53.13
31	11	12	500	25	53.13
32	13	14	500	25	53.13
33	15	16	500	25	53.13
34	1	3	500	25	-53.13
35	2	5	500	25	-53.13
36	4	7	500	25	-53.13
37	6	9	500	25	-53.13
38	8	11	500	25	-53.13
39	10	13	500	25	-53.13
40	12	15	500	25	-53.13
41	14	17	500	25	-53.13

LA BARRA 9 NO CONTRIBUYE AL ANCHO DE BANDA

LA BARRA 16 NO CONTRIBUYE AL ANCHO DE BANDA

LA BARRA 17 NO CONTRIBUYE AL ANCHO DE BANDA

LA BARRA 25 NO CONTRIBUYE AL ANCHO DE BANDA

LA BARRA 26 NO CONTRIBUYE AL ANCHO DE BANDA

LA BARRA 41 NO CONTRIBUYE AL ANCHO DE BANDA

LA MAXIMA DIFERNCIA DE INCIDENCIAS ES 3

EL ANCHO DE BANDA ES 4

NUMERO DE COLUMNAS DE LA MATRIZ [K] EN BANDA 8

CONDICION DE CARGA # 1

PRIMERA

NUMERO DE NUDOS CARGADOS 9

NUDO	FUERZA EN X	FUERZA EN Y
1	0	-2.5
2	0	-5
4	0	-5
6	0	-5
8	0	-5
10	0	-5
12	0	-5
14	0	-5
16	0	-2.5

CONDICIÓN DE CARGA # 2

SEGUNDA

NUMERO DE NUDOS CARGADOS 7

NUDO	FUERZA EN X	FUERZA EN Y
3	0	-5
5	0	-5
7	0	-5
9	0	-5
11	0	-5
13	0	-5
15	0	-5

CONDICION DE CARGA # 3

TERCERA

NUMERO DE NUDOS CARGADOS 1

NUDO	FUERZA EN X	FUERZA EN Y
1	10	0

SOLUCION PARA LA CONDICION DE CARGA # 1

DESPLAZAMIENTOS DE LOS NUDOS

NUDO	DESPLAZAMIENTO EN X	DESPLAZAMIENTO EN Y
1	3.44963E-04	-5.69293E-05
2	3.27206E-04	-4.31832E-04
3	-8.47719E-05	-4.5322E-04
4	2.47846E-04	-8.10446E-04
5	-1.01234E-04	-8.23412E-04
6	1.33107E-04	-1.07912E-03

7	-6.3625E-05	-1.0037E-03
8	-2.02272E-10	-1.17204E-03
9	9.53065E-10	-1.17553E-03
10	-1.33107E-04	-1.07912E-03
11	6.36265E-05	-1.0037E-03
12	-2.47046E-04	-8.10442E-04
13	1.01235E-04	-8.23407E-04
14	-3.27205E-04	-4.3183E-04
15	8.47718E-05	-4.53218E-04
16	-3.44962E-04	-5.69291E-05

DEFORMACION Y FUERZA AXIAL DE LAS BARRAS

BARRA	DEFORMACION	FUERZA AXIAL
1	-1.77573E-05	-3.72904
2	-7.93595E-05	-16.6655
3	-1.14739E-04	-24.0952
4	-1.33107E-04	-27.9526
5	-1.33107E-04	-27.9525
6	-1.14739E-04	-24.0951
7	-7.93508E-05	-16.6653
8	-1.7757E-05	-3.72898
9	-8.47719E-05	-11.8681
10	-1.64622E-05	-2.3047
11	3.7609E-05	5.26527
12	6.3626E-05	8.90764
13	6.36255E-05	8.90757
14	3.76081E-05	5.26514
15	-1.64628E-05	-2.30479
16	-8.47718E-05	-11.8681
17	-5.69293E-05	-7.47197
18	2.13876E-05	2.80712

19	4.9660E-06	.651893
20	4.58688E-06	.602028
21	3.49537E-06	.458767
22	4.58735E-06	.602087
23	4.96657E-06	.651862
24	2.13877E-05	2.80713
25	-5.69291E-05	-7.47194
26	-1.49141E-04	-15.6598
27	-9.26085E-05	-9.72389
28	-6.39592E-05	-6.71571
29	-3.24916E-05	-3.41161
30	-2.73065E-06	-.286718
31	2.53238E-05	2.659
32	5.61983E-05	5.90083
33	5.91898E-05	6.21493
34	5.91903E-05	6.21498
35	5.61988E-05	5.90088
36	2.53239E-05	2.65901
37	-2.73166E-06	-.286824
38	-3.24923E-05	-3.41169
39	-6.39591E-05	-6.71571
40	-9.2608E-05	-9.72384
41	-1.4914E-04	-15.6597

COMPROBACION DEL EQUILIBRIO EN LOS NUDOS

NUDO	SUMA DE FUERZAS EN X	SUMA DE FUERZAS EN Y
1	-3.91007E-05	-7.15256E-06
2	5.00679E-06	4.40227E-05
3	1.35099E-05	-9.53674E-07
4	2.03048E-05	9.29832E-06
5	-3.57628E-06	2.57492E-05

6	7.56655E-06	-7.61449E-06
7	-9.17912E-06	-5.76973E-05
8	3.33786E-06	-1.33030E-04
9	1.2964E-06	-6.58631E-05
10	-4.29153E-06	-1.56079E-04
11	-9.05991E-06	-5.8651E-05
12	5.43594E-05	-2.86102E-06
13	7.62939E-06	-4.24385E-05
14	2.86102E-06	-2.47955E-05
15	1.57356E-05	7.62939E-06
16	1.3113E-05	4.76837E-06

REACCIONES EN LOS APOYOS

APOYO	REACCION EN X	REACCION EN Y
17	-21.2639	19.9997
18	21.264	19.9998

COMPROBACION DEL EQUILIBRIO EXTERNO(INCLUYENDO LAS REACCIONES)

SUMA DE FUERZAS EN X = 0.01006E-05

SUMA DE FUERZAS EN Y = -4.65393E-04

SOLUCION PARA LA CONDICION DE CARGA # 2

DESPLAZAMIENTOS DE LOS NUDOS

NUDO	DESPLAZAMIENTO EN X	DESPLAZAMIENTO EN Y
1	3.6162E-04	-4.56042E-05
2	3.40243E-04	-4.26527E-04
3	-0.39554E-05	-4.68858E-04
4	2.56463E-04	-0.21517E-04
5	-1.00801E-04	-0.45977E-04
6	1.37424E-04	-1.00668E-03
7	-6.33954E-05	-1.11098E-03
8	-2.29671E-10	-1.18115E-03
9	9.45205E-10	-1.20432E-03

10	-1.37425E-04	-1.00668E-03
11	6.33968E-05	-1.11070E-03
12	-2.56463E-04	-8.21514E-04
13	1.00802E-04	-8.45974E-04
14	-3.40242E-04	-4.26525E-04
15	8.39554E-05	-4.68854E-04
16	-3.61619E-04	-4.56039E-05

DEFORMACION Y FUERZA AXIAL DE LAS BARRAS

BARRA	DEFORMACION	FUERZA AXIAL
1	-2.13773E-05	-4.40923
2	-8.37796E-05	-17.5937
3	-1.19039E-04	-24.9981
4	-1.37425E-04	-28.8592
5	-1.37424E-04	-28.8591
6	-1.19038E-04	-24.998
7	-8.37788E-05	-17.5936
8	-2.13769E-05	-4.40915
9	-8.39554E-05	-11.7538
10	-1.6846E-05	-2.35844
11	3.7406E-05	5.23684
12	6.33963E-05	8.87548
13	6.33959E-05	8.87542
14	3.74051E-05	5.23672
15	-1.68466E-05	-2.35852
16	-8.39554E-05	-11.7538
17	-4.56042E-05	-5.98555
18	4.2331E-05	5.55594
19	2.44604E-05	3.21043
20	2.42996E-05	3.18932
21	2.31703E-05	3.0411

22	2.43001E-05	3.18938
23	2.44601E-05	3.21039
24	4.2331E-05	5.55594
25	-4.56039E-05	-5.98551
26	-1.37075E-04	-14.3929
27	-7.78749E-05	-8.17686
28	-4.96273E-05	-5.21087
29	-1.81006E-05	-1.90057
30	1.16602E-05	1.22432
31	3.96555E-05	4.16383
32	7.09319E-05	7.44784
33	7.12561E-05	7.48189
34	7.12567E-05	7.48195
35	7.09324E-05	7.4479
36	3.96556E-05	4.16383
37	1.16593E-05	1.22423
38	-1.81013E-05	-1.90063
39	-4.96274E-05	-5.21087
40	-7.78744E-05	-8.17681
41	-1.37074E-04	-14.3928

COMPROBACION DEL EQUILIBRIO EN LOS NUDOS

NUDO	SUMA DE FUERZAS EN X	SUMA DE FUERZAS EN Y
1	-5.05447E-05	-4.29153E-06
2	1.14441E-05	3.95775E-05
3	1.00136E-05	5.24521E-06
4	2.90871E-05	-1.19209E-05
5	4.76837E-07	5.53131E-05
6	2.92063E-06	-1.74046E-05
7	-7.86781E-06	-6.67572E-05
8	1.3113E-05	-1.4174E-04

9	-5.126E-06	-6.33597E-05
10	-1.90735E-06	-1.48296E-04
11	-1.72853E-05	-5.74589E-05
12	5.57899E-05	-6.19888E-06
13	1.16825E-05	-3.86238E-05
14	1.71661E-05	-2.57492E-05
15	1.14441E-05	1.00136E-05
16	-1.43051E-06	-4.76837E-07

REACCIONES EN LOS APOYOS

APOYO	REACCION EN X	REACCION EN Y
17	-20.3894	17.4997
18	20.3895	17.4998

COMPROBACION DEL EQUILIBRIO EXTERNO(INCLUYENDO LAS REACCIONES)

SUMA DE FUERZAS EN X = 7.82013E-05

SUMA DE FUERZAS EN Y = -4.73022E-04

SOLUCION PARA LA CONDICION DE CARGA # 3

DESPLAZAMIENTOS DE LOS NUDOS

NUDO	DESPLAZAMIENTO EN X	DESPLAZAMIENTO EN Y
1	1.85901E-04	6.8242E-06
2	1.41481E-04	-9.46379E-05
3	3.00107E-05	-9.67657E-05
4	1.03565E-04	-1.42759E-04
5	5.19208E-05	-1.46179E-04
6	7.16559E-05	-1.52032E-04
7	6.49829E-05	-1.55843E-04
8	4.58283E-05	-1.33067E-04
9	6.93103E-05	-1.37399E-04
10	2.60681E-05	-9.70364E-05
11	6.48814E-05	-1.01906E-04
12	1.23973E-05	-5.51839E-05

13	5.17298E-05	-6.03479E-05
14	4.67952E-06	-1.77039E-05
15	2.96508E-05	-2.47988E-05
16	3.81840E-06	-1.83680E-06

DEFORMACION Y FUERZA AXIAL DE LAS BARRAS

BARRA	DEFORMACION	FUERZA AXIAL
1	-4.44203E-05	-9.32826
2	-3.79157E-05	-7.9623
3	-3.19094E-05	-6.70098
4	-2.58276E-05	-5.42379
5	-1.97603E-05	-4.14966
6	-1.36708E-05	-2.87086
7	-7.71778E-06	-1.62073
8	-8.61037E-07	-.180818
9	3.00107E-05	4.20149
10	2.19101E-05	3.06742
11	1.30621E-05	1.82869
12	4.32741E-06	.605837
13	-4.42888E-06	-.620043
14	-1.31516E-05	-1.84123
15	-2.2079E-05	-3.09106
16	-2.96508E-05	-4.15111
17	6.8242E-06	.895677
18	2.12772E-06	.279264
19	3.42031E-06	.448916
20	3.8109E-06	.500181
21	4.33208E-06	.568585
22	4.8699E-06	.639174
23	5.16396E-06	.677769
24	7.09494E-06	.93121

25	-1.83688E-06	-.24109
26	9.17856E-06	.963749
27	7.33829E-06	.77052
28	7.15905E-06	.751701
29	6.72785E-06	.706425
30	6.34462E-06	.666185
31	5.88728E-06	.618165
32	5.80495E-06	.61792
33	2.87012E-06	.301363
34	-1.06620E-05	-1.1196
35	-1.25032E-05	-1.31284
36	-1.26825E-05	-1.33166
37	-1.31136E-05	-1.37693
38	-1.34966E-05	-1.41714
39	-1.39537E-05	-1.46514
40	-1.3956E-05	-1.46538
41	-1.69708E-05	-1.78193

COMPROBACION DEL EQUILIBRIO EN LOS NUDOS

NUDO	SUMA DE FUERZAS EN X	SUMA DE FUERZAS EN Y
1	-1.93119E-05	-1.78814E-07
2	2.32458E-06	9.17912E-06
3	-4.29153E-06	2.68221E-06
4	6.55651E-06	-1.66873E-06
5	4.70877E-06	4.41074E-06
6	8.34465E-06	1.3113E-06
7	-1.78814E-06	-1.00136E-05
8	-8.52347E-06	-1.39475E-05
9	-7.86781E-06	-8.9407E-06
10	2.38419E-06	-1.46627E-05
11	-2.14577E-06	-4.52995E-06

12	-2.74181E-06	-2.38419E-07
13	3.39747E-06	-3.09944E-06
14	-2.38419E-07	-8.34465E-07
15	3.03984E-06	2.38419E-07
16	-2.08616E-07	2.08616E-07

REACCIONES EN LOS APOYOS

APOYO	REACCION EN X	REACCION EN Y
17	-5.22027	1.66664
18	-4.77975	-1.66667

COMPROBACION DEL EQUILIBRIO EXTERNO(INCLUYENDO LAS REACCIONES)

SUMA DE FUERZAS EN X = -1.62125E-05

SUMA DE FUERZAS EN Y = -3.98159E-05

4.7 Conclusiones

Aunque las microcomputadoras poseen ciertas limitaciones principalmente en la de poseer menor cantidad de memoria en comparación con los grandes sistemas, nos pueden ayudar a resolver problemas de razonable magnitud y complejidad como el que se presento en este trabajo. Sin embargo para ello fue necesario tener en cuenta las limitaciones que las microcomputadoras de hoy en día poseen.

Así pues las microcomputadoras aunque de menor capacidad que los grandes sistemas los cuales estabamos acostumbrados a utilizar, siguen siendo útiles como una herramienta para el desarrollo de actividades relacionadas con la Ingeniería Civil.

BIBLIOGRAFIA

CAPITULOS I y II.

1.- Apuntes del curso "Análisis Estructural", CEC Facultad de Ingeniería, UNAM, 1976

2.- Programación BASIC, Byron S. Gottfried Serie Schaums.

3.- Manual del lenguaje BASIC, Burroughs.

4.- Apuntes de Computadoras y Programación, Facultad de Ingeniería, UNAM.

5.- Apuntes del curso "Lenguaje de Programación BASIC con Aplicaciones" (1a. Parte); CEC Facultad de Ingeniería, UNAM, 1983.

6.- Métodos de Computación en Ingeniería Civil, Fenves.

7.- Aplicaciones de la Computación en Ingeniería. M.A. Murray Laso, E. Chicuriel Uziel.

8.- Métodos Numéricos aplicados a la computación digital con FORTRAN. James, Smith.

9.- Revista Información Científica y Tecnológica, Vol. 3 Núm. 56, 1º de Noviembre de 1983, CONACYT.

10.- Revista ICyT, Vol. 4 Núm. 61, 15 de enero de 1983, CONACYT.

11.- Revista ICyT, Vol. 4 Núm. 64, 1º de marzo de 1982, CONACYT.

12.- Revista ICyT, Vol. 3 Núm. 41, 15 de marzo de 1981, CONACYT.

CAPITULOS III y IV

1.- Análisis de Estructuras Reticulares James M. Gere, William Weaver, Jr.

2.- Notas de la clase "Aplicación de las Computadoras al Análisis Estructural" impartidas por el Ing. - Julio E. Damy Rios, Facultad de Ingeniería, UNAM, 1981.

3.- Notas de la clase "Tópicos Estructurales Avanzados y Aplicaciones de las computadoras al Análisis Estructural" J.E. Damy Rios, DEPFI, UNAM, 1982.

4.- Notas de la clase "Teoría General de las Estructuras I" J. E. Damy Ríos, DEPFI, UNAM, 1983.