

DIRECTORIO DE PROFESORES DEL CURSO: LENGUAJE DE PROGRAMACION FORTRAN

Noviembre de 1984.

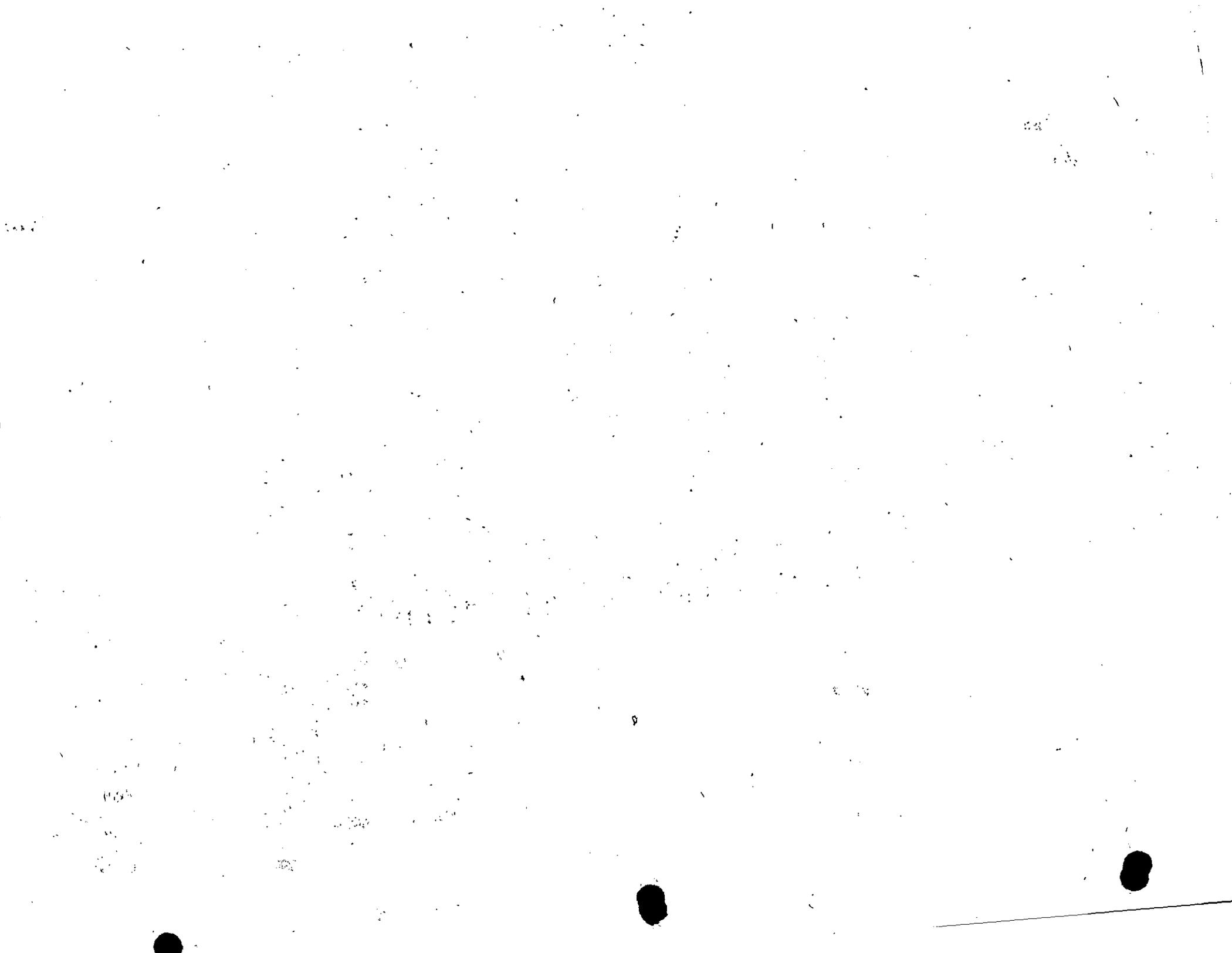
1. ING. HERIBERTO OLGUIN ROMO
DIRECTOR DE CAPTURA Y PROCESO
DIRECCION GENERAL DE POLITICA INFORMATICA
INSTITUTO NACIONAL DE ESTADISTICA
GEOGRAFIA E INFORMATICA
SECRETARIA DE PROGRAMACION Y PRESUPUESTO
ARCOS DE BELEN NO. 2 P.H.
MEXICO, D.F.
761 62 27.

2. ING. JUAN ALEJANDRO JIMENEZ GARCIA
JEFE DEL CENTRO DE CALCULO
UNAM
UNAM
550 57 34

3. M. EN C. JOSE RICARDO CIRIA MERCE (COORDINADOR)
SUBDIRECTOR DE NUEVOS PROYECTOS
COORDINACION DE LA ADMINISTRACION ESCOLAR
U N A M
550 50 46

4. M. EN C. CARLOS AUGUSTO RAMOS LARIOS
JEFE DEL DEPARTAMENTO DE INFORMACION
DIRECCION GENERAL DE PROVEDURIA
U N A M
MEXICO, D.F.
548 97 75

5. ING. ANTONIO PEREZ AYALA
GERENTE ADMINISTRATIVO
PRODUCTOS METALICOS HIERRO MEX, S.A.
CALZ. DE LA VILLA NO. 376
LOCALES A Y B
COL. JAMAICA
15800 MEXICO, D.F.
530 34 08 6 519 30 25



LENGUAJE DE PROGRAMACION FORTRAN 1 9 8 4

Fecha	Tema	Horario	Profesor
Noviembre 9	INTRODUCCION A LA COMPUTADORA DIGITAL BREVE HISTORIA Y UBICACION DEL LENGUAJE ELEMENTOS BASICOS CONSTANTES Y VARIABLES	17 a 21 h	HERIBERTO OLGUIN ROMO RICARDO CIRIA MERCE
Noviembre 10	EXPRESION ARITMETICA ENTRADA SALIDA (PRIMERA PARTE) TRANSFERENCIA DE CONTROL USO DEL EDITOR DEL EQUIPO VAX 11/780	9 a 14 h	ALEJANDRO JIMENEZ O. LUIS MIGUEL MURGUIA MARIN
Noviembre 16	IF ARITMETICO. ARREGLOS DE UNA DIMENSION EJEMPLO PARALELO: PROMEDIO DE CALIFICACIONES PROMEDIO DE CALIFICACIONES USANDO INDICES	17 a 21 h	HERIBERTO OLGUIN ROMO RICARDO CIRIA MERCE
Noviembre 17	ENTRADA/SALIDA (SEGUNDA PARTE) FORMATOS EJEMPLO PARALELO : NOMBRES DE ALUMNOS APLICACIONES NO MATEMATICAS : BUSQUEDA SECUENCIAL BUSQUEDA BINARIA	9 a 14 h	ALEJANDRO JIMENEZ G. LUIS MIGUEL MURGUIA
Noviembre 23	PROPOSICIONES DO Y CONTINUE. ANIDAMIENTO EJEMPLO PARALELO : SE REEMPLAZA INSTRUCCION IF POR INSTRUCCION DO. APLICACIONES MATEMATICAS : AJUSTE A UNA RECTA	17 a 21 h	HERIBERTO OLGUIN ROMO CARLOS RAMOS L.
Noviembre 24	OPERADORES Y EXPRESIONES LOGICAS PROPOSICION IF LOGICO EJEMPLO PARALELO OBTENCION DE PROMEDIOS DETERMINACION DEL NO. DE APROBADOS Y NO APROBADOS APLICACIONES NO MATEMATICAS: ORDENAMIENTO "QUICK-SORT"	9 a 14 h	LUIS MIGUEL MURGUIA MARIN ALEJANDRO JIMENEZ G.

Fecha	Tema	Horario	Profesor
Noviembre 30	ARREGLOS DE N DIMENSIONES. APLICACIONES MATEMATICAS: SOLUCION DE UN SISTEMA DE ECUACIONES	17 a 21 h	HERIBERTO OLGUIN ROMO CARLOS RAMOS L.
Diciembre 1°	FUNCIONES PRE-DEFINIDAS FUNCIONES DEFINIDAS POR EL USUARIO EJEMPLO PARALELO : DESVIACION STANDARD VARIANZA APLICACIONES MATEMATICAS TABLA DE FUNCIONES	9 a 14 h	CARLOS RAMOS LARIOS RICARDO CIRIA MERCE
Diciembre 7	SUBPROGRAMAS FUNCIONES SUBROUTINAS EJEMPLO PARALELO : ORDENAMIENTO ALFABETICO	17 a 21 h	ING. LUIS MIGUEL MURGUIA RICARDO CIRIA MERCE
Diciembre 8	FORTRAN ESTRUCTURADO. APLICACIONES MATEMATICAS : NEWTON RAPSON.	9 a 14 h	ALEJANDRO JIMENEZ G. CARLOS RAMOS L.



**DIVISION DE EDUCACION CONTINUA
FACULTAD DE INGENIERIA U.N.A.M.**

LENGUAJE DE PROGRAMACION FORTRAN

BREVE HISTORIA DEL LENGUAJE

OCTUBRE, 1984

2.- BREVE HISTORIA DEL LENGUAJE

La introducción en el mercado de las computadoras de programas almacenados permitió el nacimiento de una nueva profesión, el programador de computadoras. Desde entonces, se han producido significativos avances en el campo de la programación de computadoras, específicamente en el desarrollo de técnicas para hacer menos dificultoso este trabajo para el programador humano.

Muchos autores reconocen que el honor de haber sido el primer programador (mejor dicho, la primera) corresponde a una simpática dama que murió casi un siglo antes de que apareciera la primera computadora de programas almacenados. Ada Augusta, condesa de Lovelace, quien viviera una vida extraordinaria. Nació en 1815; fué uno de los muchos descendientes del prolífico poeta inglés Lord Byron. Unos pocos meses después de su nacimiento, sus padres se separaron y ella no volvió a ver a su padre nunca más.

Por las fechas de su matrimonio con el conde de Lovelace, en -- 1835, Ada Augusta se relacionó con Charles Babbage, quien estaba en ese momento empezando su proyecto de la máquina analítica. Poseedora de aptitudes para las matemáticas y el pensamiento mecánico, se ofreció a trabajar con Babbage en su proyecto y, en 1842, tradujo del inglés una primera descripción al italiano de la máquina, añadiendo muchas notas de su cosecha. Se refirió a "ciclos de operación" y al repetido uso de las tarjetas como estructuras del tipo de subrutinas y se refirió también a la computación no numérica y a la manipulación simbólica. Observó que la máquina analítica no "originaba nada" y que sólo podía hacer "aquello que uno sabía cómo ordenarle que realizara". Una de sus notas fué una descripción detallada para calcular los números de Bernoulli con la máquina analítica, que para muchos fué el primer "programa". Ada y Charles trabajaron juntos posteriormente en una tréta para aplicar la máquina analítica al problema de los momios en las carreras

de caballos. Durante el resto de su vida Ada se dedicó al juego dilapidando una parte considerable de la fortuna de Lovelace. Murió de cáncer en 1852.

Al igual que los idiomas sirven de vehículo de comunicación entre los seres humanos, existen lenguajes que realizan la comunicación entre los seres humanos y las computadoras. Estos lenguajes permiten expresar los programas o el conjunto de instrucciones que el operador humano desea que la computadora ejecute. Los lenguajes de computadora toman diferentes formas; los de las primeras, como la ENIAC y la EDSAC se componían en el lenguaje real de las máquinas mismas. En lenguaje de máquina, las instrucciones se expresan simplemente como una serie de dígitos binarios, o hits (binary digits). La dificultad de programar las máquinas primitivas de esta manera limitaba drásticamente su utilidad y proporcionaba un fuerte incentivo para que se desarrollaran lenguajes de programación más orientados hacia la expresión de soluciones con la notación de los problemas mismos. Un programa especialmente diseñado podía entonces, ser ejecutado para que realizara la traducción al lenguaje real usado por la máquina.

Los primeros lenguajes de programación se conocieron como lenguajes ensambladores, un ejemplo de los cuales es TRANSCODE, desarrollado para la computadora FERUT de la Universidad de Toronto por Pat Hume y Beatrice Worsley. En los lenguajes ensambladores se define un código especial (llamado mnemónico) para cada una de las operaciones de la máquina y se introduce una notación especial para especificar el dato con el cual debe realizarse tal operación. Un programa especial, denominado ensamblador, traduce las instrucciones simbólicas del lenguaje a las instrucciones de máquina necesarias para que sean ejecutadas. Los lenguajes ensambladores son todavía muy populares en ciertas aplicaciones; a pesar de que se ha avanzado notablemente en los lenguajes de programación de máquina, esto no basta para satisfacer las necesidades de todo lo que el programador desea hacer.

A mediados del decenio 1950-1960 aparecieron los primeros lenguajes de programación de propósito general, uno de los cuales revolucionó muy pronto el campo de la programación. Se llamó FORTRAN (FORmula TRANslating system) y fué publicado en 1954. El líder del proyecto FORTRAN fué John Backus, quien trabajó para la IBM y desarrolló un método formal para definir la sintaxis de los lenguajes de programación, la Forma Backus-Naur o BNF. FORTRAN fué realizado en 1957, con nuevas versiones que fueron apareciendo en 1958, 1960 y 1962, la última de las cuales se conoció con el nombre de FORTRAN IV. Se trata de un lenguaje dirigido a la solución numérica de problemas científicos; es fácil de entender, leer y escribir. Con el FORTRAN, el usuario está capacitado de inmediato para escribir un programa aunque sepa muy poco acerca de las características físicas de la máquina en la cual el programa va a ser ejecutado. Sin duda, el lenguaje es independiente de la máquina y en teoría, a pesar de algunas dificultades prácticas, los programas FORTRAN escritos para una máquina deben ser fáciles de transferir a otra. No ocurre lo mismo con los programas escritos en ensamblador o en código de máquina. El más fuerte impacto que tuvo el lenguaje FORTRAN en la industria de las computadoras se debió a que permitía a los usuarios programar sus propias soluciones, sin necesidad de recurrir a la ayuda de un programador profesional.

En un comienzo, FORTRAN no fué totalmente aceptado a causa, sobre todo, del temor que inspiraban sus altos costos de traducción. Al contrario de los ensambladores, los lenguajes de alto nivel, a causa de su generalidad, requieren traductores más complejos conocidos como compiladores, que por su propia complejidad son también más costosos de ejecutar. A pesar de estos problemas, su empleo aumentó y con el paso de los años los costos de compilación han sido reducidos sustancialmente, por lo que en la actualidad FORTRAN ha llegado a ser el lenguaje de programación más ampliamente utilizado en el mundo, y también un factor muy importante en el cada vez más difundido uso de las computadoras. La aparición de compiladores rápidos orientados al uso de los estudiantes, como PUFFT, desarrollado en la Universidad de Purdue, y WATFOR y WATFIV, desarrollados en la Universidad de Waterloo, han

hecho que la enseñanza de la programación con FORTRAN sea más simple; ello ha permitido que la computación misma llegue a un mayor número de estudiantes.

Otros lenguajes de programación han seguido rápidamente los pasos de FORTRAN; el lenguaje ALGORítmico, ALGOL, fué diseñado por un comité internacional en 1958 y revisado en 1960. Se trata de un lenguaje muy efectivo para resolver una amplia variedad de problemas con aplicaciones en matemáticas numéricas, pero no es adecuado (al igual que FORTRAN) para manejar datos no numéricos. Aún hoy, ALGOL es más popular en Europa que en Norteamérica.

Tanto FORTRAN como ALGOL están dirigidos básicamente a la computación científica; en mayo de 1959, el Departamento de Defensa de Estados Unidos convocó a una reunión para discutir el problema de desarrollar un lenguaje común para aplicaciones en negocios. Respondieron al llamado cerca de 40 representantes de los usuarios, de las dependencias del Gobierno, de los fabricantes de computadoras y de otras partes interesadas. La versión inicial de COBOL (COmmon Business Oriented Language) apareció en diciembre de 1959.

Los objetivos de COBOL consideraban la expresión natural de los programas (es decir, en inglés), lo que permitiría el aprendizaje fácil del lenguaje, la amplia documentación del mismo y la independencia de la máquina, lo cual facilitaría la transferencia de los programas de COBOL de una instalación a otra. A pesar de que las especificaciones de COBOL han sido revisadas varias veces desde su primera versión, el lenguaje mismo ha permanecido esencialmente sin cambios. En la actualidad se utiliza en las aplicaciones de procesamiento de datos para los negocios.

BASIC (Beginner's All-purpose Symbolic Instruction Code), un lenguaje científico de programación que fué diseñado con el objeto de

hacer su aprendizaje y su uso tan fácil como sea posible, lo desarrollaron en Dartmouth College en 1965 John Kemeny y Tom Kurz. El sistema BASIC fué el primero en utilizarse en una red o base distribuída, y -- también el primero en estar disponible en tiempo compartido o modo interactivo. Cada comando proporcionado por el usuario desde una terminal BASIC provoca una respuesta inmediata de la computadora, lo cual le permite al usuario tener un control más estricto sobre el procesamiento de su programa. BASIC continúa siendo muy popular en la actualidad, y el tiempo compartido ha llegado a ser la forma común de operación, con más lenguajes y otras facilidades adicionales para el usuario.

En septiembre de 1963, un comité compuesto por personal de IBM y de clientes se formó con objeto de generar un lenguaje que pudiera atraer a más usuarios, pero que continuara siendo una poderosa herramienta para el ingeniero. En el momento de su inicio, se creyó que el comité únicamente extendería el FORTRAN, pero después de realizar el estudio de FORTRAN, ALGOL y COBOL y de entrevistar a gran variedad de usuarios, el comité decidió desarrollar un nuevo lenguaje. El 10 de marzo de 1964 el comité presentó un informe con el nuevo programa propuesto. (Inicialmente denominado NPL., por New Programming Language, cambió de nombre a petición del National Physical Laboratory) El lenguaje fué revisado en junio y diciembre de ese año y denominado finalmente PL/I. El primer manual oficial se publicó al inicio de 1965 y el primer compilador de PL/I fué terminado en el sistema 360 de IBM en el mes de agosto de 1966.

Debido a que PL/I es un lenguaje muy general, tiene una amplísima variedad de aplicaciones; su uso ya en aumento y muchos piensan que llegará a desplazar a sus progenitores -- FORTRAN, ALGOL y COBOL-- . Recientemente se ha producido algunos compiladores orientados al uso de los estudiantes, como el sistema PL/C de la Universidad de Cornell y el SP/k de la Universidad de Toronto, los cuales se espera que aumenten la aceptación de PL/I como un lenguaje para la enseñanza.

(Tomado de "Ciencia de las Computadoras", J.P Tremblay, P.B. Bunt, Mc. Grace Hill).



**DIVISION DE EDUCACION CONTINUA
FACULTAD DE INGENIERIA U.N.A.M.**

LENGUAJE DE PROGRAMACION FORTRAN

- INTRODUCCION A LA COMPUTADORA DIGITAL
- LENGUAJE FORTRAN
- INTRODUCCION AL SISTEMA VAX - 11/780
- APLICACIONES MATEMATICAS
- APLICACIONES NO MATEMATICAS
- EJEMPLO PARALELO

OCTUBRE, 1984

INTRODUCCION A LA PROGRAMACION Y COMPUTACION ELECTRONICA

CAPITULO I

Introducción a la Computadora Digital

CAPITULO II

Lenguaje FORTRAN

CAPITULO III

Introducción al Sistema VAX -11/780

CAPITULO IV

Aplicaciones Matemáticas

CAPITULO V

Aplicaciones No Matemáticas

CAPITULO VI

Ejemplo Paralelo

CAPITULO I

INTRODUCCION DE LA COMPUTADORA DIGITAL

CONCEPTO DE COMPUTADORA

El objeto de esta breve reseña sobre las computadoras electrónicas y sus múltiples aplicaciones al servicio del hombre, es transmitir al lector una completa visión de conjunto, mediante un lenguaje sencillo que permita comprender conceptualmente los temas tratados, sin necesidad de conocimientos previos en la materia.

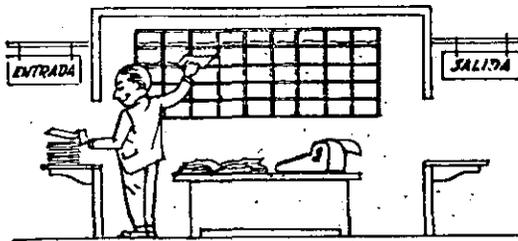
Esperamos que estas páginas, muy simples en apariencia pero con profundo contenido, permitan, a quienes las lean, ingresar al maravilloso mundo de las máquinas automáticas.



Este señor se llama Control. Trabaja en una pequeña habitación. Tiene a su disposición una máquina de calcular que suma, resta, multiplica y divide. Tiene también el señor Control un archivo parecido al casillero que existe en los trenes para clasificación postal.

Hay, además, en la habitación, dos ventanillas identificadas con sendos carteles: "Entrada" y "Salida".

El señor Control tiene un manual que le indica cómo debe desenvolverse con estos elementos, si alguien le pide que haga un trabajo.



3

Una persona quiere saber el resultado de un complicado cálculo. Para ello, escribe ordenada, precisa y detalladamente, cada una de las operaciones que, en conjunto, integran ese cálculo, anota cada instrucción elemental en una hoja de papel y coloca todas las hojas en orden en la ventanilla "Entrada". El señor Control, al ver las hojas, lee en su manual que debe tomar esas hojas con instrucciones, una por una, y colocarlas relativamente en su archivo. Y así lo hace.

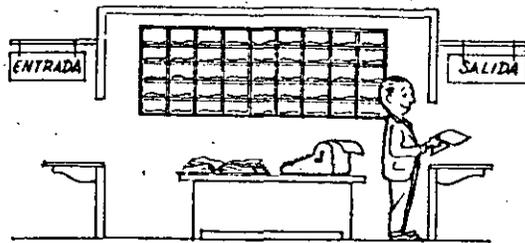


3

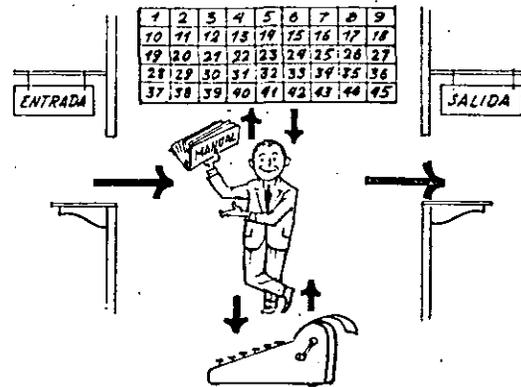
Una vez ubicadas todas las instrucciones en el archivo, el señor Control consulta nuevamente el manual. Allí se le indica que, a continuación, debe tomar la instrucción de la casilla 1 y ejecutarla, luego la de la casilla 2 y ejecutarla, y así sucesivamente hasta ejecutar la última instrucción. Algunas instrucciones indicarán que hay que sumar una cantidad a otra (instrucciones aritméticas); otras, que el señor Control debe ir a la ventanilla "Entrada" para buscar algún dato que intervenga en el cálculo -- (instrucciones de "entrada/salida"), dato que la persona que le formuló el problema habrá colocado ya en dicha ventanilla, en otra hoja de papel.

Finalmente, otras instrucciones indicarán que debe elegirse una de entre dos alternativas (instrucciones lógicas): por ejemplo, supongamos que una parte del cálculo -- desde la instrucción que está en la casilla 5 del archivo hasta la que está en la casilla 9 -- debe ejecutarse 15 veces porque el cálculo así lo exige.

En tal caso, la instrucción que está en la casilla 10 indicará -- que, si los pasos 5 a 9 se han ejecutado menos de 15 veces, se debe volver al paso 5. Cuando se hayan realizado las 15 repeticiones y no antes, el señor Control seguirá con la instrucción de la casilla 11.



Después de ejecutar todas las instrucciones del archivo, haciendo con la máquina de calcular las operaciones en ellas indicadas, el señor Control entrega, a través de la ventanilla "Salida", los resultados obtenidos . . . y se sienta a esperar un nuevo trabajo.



Obsérvese que la actuación del señor Control es puramente mecánica: sólo sigue las indicaciones de su manual y cumple de acuerdo con ellas, las instrucciones que recibe a través de la ventanilla "Entrada". Toma decisiones, pero solamente cuando se le señalan las alternativas que existen y con qué criterio debe elegir una de ellas.

El señor Control puede resolvernos cualquier problema, por complicado que éste sea. Pero para ello debemos indicarle paso a paso, en la forma más elemental y detallada, todo lo que debe hacer para resolverlo, sin olvidarnos absolutamente nada porque, en ese caso, el señor Control no sabría continuar por sí mismo.

Haga el lector la prueba de formular un problema cualquiera de modo tal que una persona que no conozca nada acerca de ese problema, pueda resolverlo sin necesidad de hacer consultas. Verá que es una experiencia interesantísima.

El esquema que acabamos de representar mediante el señor Control y sus elementos de trabajo, corresponde exactamente al esquema de funcionamiento de una computadora electrónica.

A continuación presentaremos una breve descripción de los elementos de la computadora que corresponden a los elementos de trabajo del señor Control.

Las unidades de Entrada (representadas por la ventanilla "Entrada") :-

Son en la computadora, dispositivos capaces de leer información (Instrucciones o Datos) con el objeto de procesarla. Existen una gran variedad de elementos de entrada, entre los cuales tenemos:

Tarjetas de Cartulina y Cintas de Papel: Que son perforadas de manera que cada perforación representa un número, una letra ó un símbolo especial de acuerdo con un código predeterminado.

Cintas magnéticas: Conocidas como "memorias externas" tienen la ventaja de permitir almacenar la información en forma más concentrada (a razón de 80 a 2400 caracteres por pulgada de longitud) y de ser más veloces, ya que pueden enviar o recibir información a la unidad de control a velocidades que van de 10,000 a 680,000 caracteres por segundo. Pueden llegar a tener hasta 730 m. de longitud.

Disco Magnético: También conocidos como "Memoria externa", en general tienen un diámetro aproximado de 30 cm. y pueden grabar hasta 400,000 letras, números, y caracteres especiales, formando palabras, cifras, ó registros completos. Se pueden grabar o leer a razón de 77,000 a 312,000 caracteres por segundo y su tiempo de acceso a un registro alcanza un promedio de 60 mili-segundos.

Una diferencia importante entre las cintas y los discos es la siguiente:

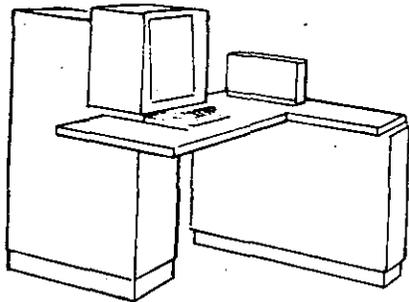
En las cintas los registros se graban o leen secuencialmente..

En los discos se tiene "Libre Acceso" a un registro cualquiera, en forma inmediata, pues cada registro se localiza por su posición física dentro del disco.

Lectora Óptica de Caracteres Impresos: Puede leer un documento impreso por una máquina de escribir, o por una máquina de contabilidad o por la impresora de una computadora, a una velocidad de 30,000 caracteres por minuto.

Unidad de Representación Visual: Esta unidad de entrada/salida sirve para hacer consultas a la computadora, por medio de un teclado de máquina de escribir, y obtener la respuesta reflejada en una pequeña pantalla de televisión.

La imagen está formada por hasta 12 renglones de hasta 80 caracteres (letras, números, ó signos especiales) cada uno.



Vemos aquí otra Unidad de Representación Visual, más evolucionada que la anterior, la comunicación hombre-máquina puede establecerse en ella por medio de gráficas, es decir que la entrada y la salida de datos se hacen por medio de imágenes.

Cuenta esta unidad para ello con un dispositivo con forma de lápiz, que tiene en su punta una célula fotoeléctrica. Un delgado haz de luz parte en determinado momento de un punto de la pantalla y la recorre en forma de zig-zag. Si se apoya el "lápiz" en cualquier posición de la pantalla, su célula fotoeléctrica detectará en algún momento el haz de luz.

Por el tiempo transcurrido desde que el haz de luz comenzó su "barrido" hasta que fue detectado, la computadora determina en qué punto de la pantalla se encuentra apoyado el "lápiz".

Como el barrido dura una fracción de segundo y se realizan muchos barridos por segundo, se puede "escribir" con el "lápiz" sobre la pantalla y el dibujo "ingresa" en la memoria de la computadora como una sucesión de puntos codificados.

La pantalla está imaginariamente dividida en 1.040.576 puntos, de manera que los trazos que se obtienen son prácticamente continuos.

Pueden dibujarse así curvas, estructuras, letras, números y cualquier tipo de gráfico, y esa información ingresa automáticamente a la computadora.

Por otra parte, los resultados obtenidos por la computadora son representados en la pantalla también como curva, letras, etc., bajo control del programa almacenado en la memoria.

Lectora Óptica de Manuscritos: Salvo algunas pequeñas restricciones en cuanto al formato de los caracteres, esta unidad puede "leer" documentos escritos por cualquier persona y con cualquier ejemplo a una velocidad aproximada de 30,000 caracteres por minuto.

El registrador/analizador Fotográfico es una Unidad de Entrada/Salida de datos que realiza las siguientes funciones.

- 1) Registra los resultados de la computadora sobre microfotografías, mediante un tubo de rayos catódicos, que inciden sobre una película fotográfica, y cuyo haz electrónico actúa gobernado por el Programa Almacenado. La película se revela automáticamente dentro de la unidad y 48 segundos después está lista para ser proyectada.
- 2) Proyecta sobre una pantalla translúcida las microfotografías registradas.
- 3) Analiza imágenes reproducidas en negativo sobre película transparente, las digitaliza y las transmite a la Unidad Central de Procesamiento.

La película utilizada tiene 30.5 milímetros de ancho y 120 metros de longitud. La Entrada o Salida de imágenes puede consistir en letras, números, símbolos, dibujos, gráficas, mapas, curvas, etc. En una microfotografía de 30.5 mm X30.5 mm pueden registrarse hasta 30,600 - letras y números, o hasta 16,777,216 puntos correspondientes a imágenes.

La velocidad de Registro/Análisis es de 40,000 letras, números y símbolos por segundo, o su equivalente si se trata de imágenes.

Máquina de Escribir (Teletipo).

Las unidades de almacenamiento o memorias (Representadas por el archivo del señor Control) permiten registrar las instrucciones y los datos para resolver un problema; entre estas se tienen:

Los Anillos Magnetizantes: Estos pueden magnetizarse en un sentido ó en otro "Recordando" así un 1 o un 0 respectivamente. Con 8 de éstos anillos se forma una posición de memoria, en la cual puede registrarse una letra, un dígito ó un carácter especial, según las distintas combinaciones de anillos "En 1" y "En 0", de acuerdo a un código predeterminado.

Las Memorias de Flip -Flops

Las Cintas Magnéticas

Los Discos Magnéticos

El dispositivo aritmético (representado por la máquina de cálculo) que realiza las cuatro operaciones aritméticas.

Las unidades de salida (representadas por la ventanilla "Salida" que pueden ser:

Impresoras

Máquinas de Escribir (Teletipos)

Grabadoras de Cintas Magnéticas

Grabadoras de Discos Magnéticos

Unidad de Representación Visual

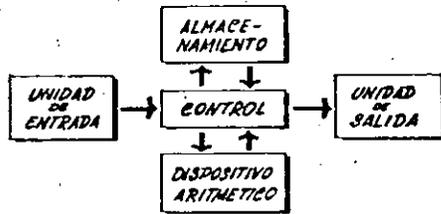
Registrador Analizador Fotográfico

Unidad de Respuesta Oral con la cual la Computadora puede hablar en todo el sentido de la palabra.

Contiene una Cinta magnetofónica en la cual un locutor ha grabado un diccionario de una gran variedad de palabras, en cualquier idioma.

Finalmente, un dispositivo electrónico de control (representado por el señor control) ayudado de un programa especial o sistema operativo (representado por el manual del señor Control), gobierna todas las operaciones de todas las unidades que componen la computadora.

Habiendo descrito las partes que componen la computadora podemos mostrar el siguiente esquema que la representa:



O en forma más resumida :



Siendo :



Hemos hablado hasta este momento de la computadora electrónica desde el punto de vista conceptual. Durante las dos últimas décadas se han producido avances tecnológicos tan extraordinarios en materia de electrónica que la computadora ha sufrido enormes transformaciones. Veremos ahora cómo se ha ido modificando la idea original hasta llegar a los más modernos sistemas de procesamiento de datos.

Las primeras computadoras tenían circuitos con válvulas de vacío. Los tiempos de operación se medían en ellas en milisegundos (milésimas de segundo). Cuando aparecieron los transistores, el diseño de los circuitos se mejoró notablemente y la duración de las operaciones en las computadoras que utilizaban esta "Tecnología de Estado Sólido" se midió en microsegundos (milionésimas de segundo).

El hecho de que las nuevas máquinas fueran miles de veces más rápidas que las anteriores, trajo aparejada la creación de unidades de entrada, salida y memoria externa mucho más veloces.

La invención de un nuevo tipo de transistor ("chip") provocó una verdadera revolución en los circuitos electrónicos y sus procesos de fabricación. El nuevo elemento es tan pequeño que en un dedal de costura caben más de 50,000 chips. Debido a su tamaño, se les denomina circuitos microminiaturizados o microcircuitos. Los tiempos de operación se miden ahora en nanosegundos (milmillonésimas de segundo). Ha nacido en esta forma la tercera generación de computadoras, y las altas velocidades alcanzadas posibilitaron un nuevo enfoque en el diseño de los sistemas de procesamiento de datos.

Enunciaremos brevemente los adelantos que esta tercera generación ha introducido con respecto a la tecnología anterior :

. La computadora se autogobierna y trabaja sin detenerse, pasando de un trabajo a otro sin demora alguna.

. El Operador interviene sólo cuando algún problema excepcional ocurre. La comunicación entre hombre y máquina se realiza sólo sobre la base de "Informes por Excepción" .

. Si ocurre una falla en los circuitos o en la parte electromecánica la máquina realiza un autodiagnóstico e indica cuál es la anomalía.

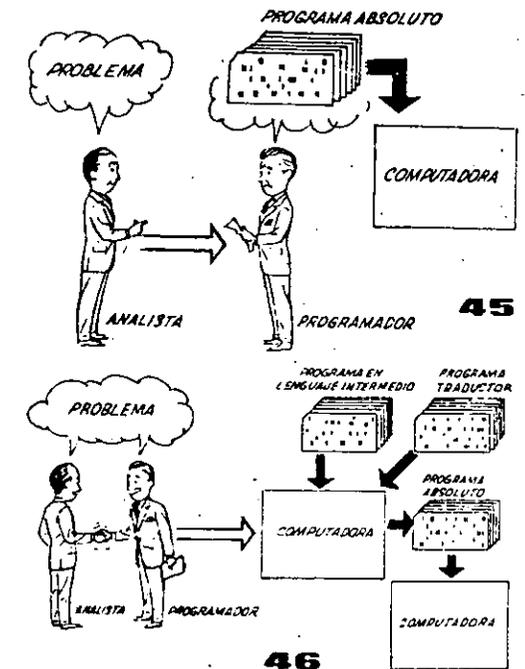
. La velocidad de Entrada-Proceso-Salida se ha incrementado extraordinariamente.

. Todas las operaciones del sistema se realizan en forma simultánea.

. Los lenguajes de programación han evolucionado de manera notable.

. El autocontrol y la autoverificación de operaciones han alcanzado niveles insospechados.

. Pueden realizarse, con máximo rendimiento, varios trabajos distintos simultáneamente. (Multiproceso).



Hasta ahora hemos visto muchas unidades que, en distintas combinaciones, configuran computadoras electrónicas para las más variadas aplicaciones. Ahora nos detendremos para analizar el manejo de dichos sistemas.

El Programa de Instrucciones almacenado en la Unidad Central de Procesamiento, consta de una secuencia de órdenes y comandos, expresados según una codificación especial denominada "Lenguaje Absoluto de Máquina". Las primeras computadoras se "programaban" en este complejo lenguaje. Había entonces una enorme diferencia entre nuestro idioma y aquél según el cual debíamos comunicarnos con la máquina. Esto obligaba a un gran esfuerzo común entre el analista que conocía el problema, y el programador que conocía la computadora, pues ambos hablaban del mismo proceso en distintos lenguajes.

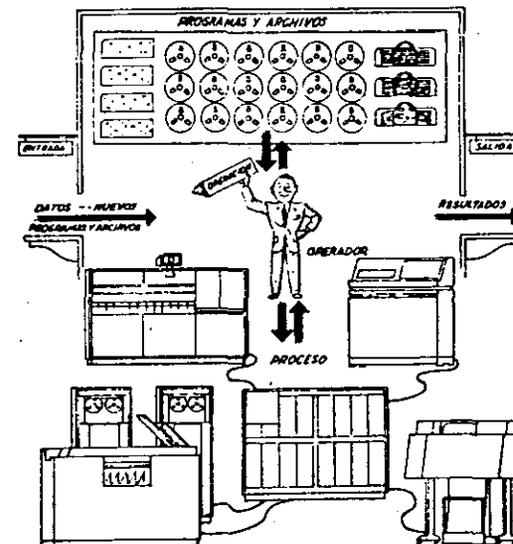
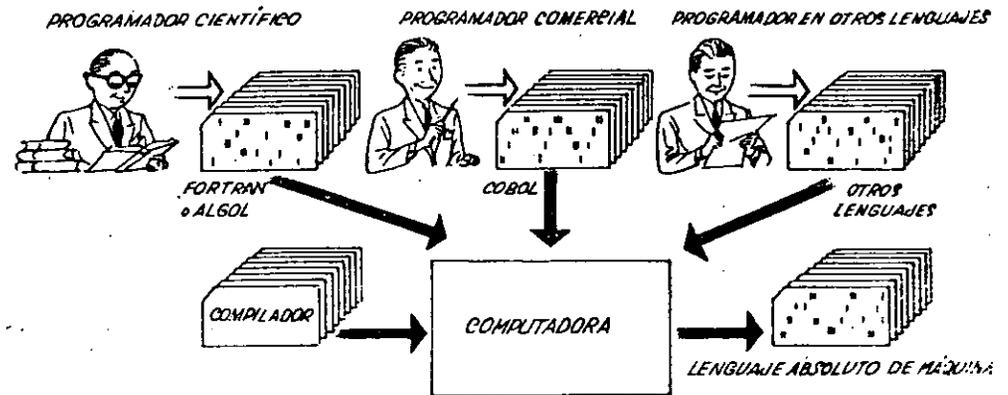
Se crearon, para solucionar el problema, lenguajes intermedios cada vez más parecidos a nuestro idioma. Es decir que cada nuevo lenguaje intermedio se acercaba más al problema y se alejaba más de la máquina. Para cada uno de estos lenguajes se creó un programa traductor llamado "Compaginador" o "Compilador", -- que tenía la misión de traducir el lenguaje intermedio al absoluto de máquina. Ahora, el analista y el programador "hablan un mismo idioma": ambos conocen el problema y la solución.

Pero la computadora seguía desarrollándose, y pronto los lenguajes intermedios fueron insuficientes para formular intrincados problemas científicos o comerciales. Nacieron, entonces, lenguajes especializados: dos de ellos, el FORTRAN y el ALGOL, permiten programar problemas científicos-técnicos utilizando una notación casi idéntica a la notación matemática común. El COBOL es un lenguaje comercial cuyas sentencias configuran oraciones y frases en forma tal que una persona que no sabe qué es una computadora, puede leer un programa y entender perfectamente qué es lo que hará la máquina cuando lo tenga almacenado.

Cada uno de estos lenguajes tiene un programa - Compilador para cada tipo distinto de computadora capaz de procesarlo. Esto significa que un programador que sabe FORTRAN, por ejemplo, puede programar una computadora aún sin conocerla. Es decir que estos tres lenguajes constituyen un "esperanto" de las máquinas.

La tercera generación de computadoras permitió abordar complejos problemas que incluían, entre otros, aspectos comerciales y científicos.

Hemos llegado así a que la computadora nos "entienda", en lugar de que se limite a recibir órdenes en su idioma.



CAPITULO II

(10)

LENGUAJE FORTRAN

ING. HERIBERTO OLGUIN R.

ING. ANTONIO PEREZ A.

M. EN C. RICARDO CIRIA M.

CAPITULO II

(11)

- 1.- INTRODUCCION
- 2.- BREVE HISTORIA DEL LENGUAJE
- 3.- ELEMENTOS BASICOS
 - 3.1.- JUEGO DE CARACTERES
 - 3.2.- NUMEROS
 - 3.3.- ESTRUCTURA DE UN PROGRAMA
 - 3.3.1.- Proposición END
 - 3.3.2.- Proposición STOP
 - 3.4.- HOJA DE CODIFICACION
- 4.- CONSTANTES Y VARIABLES
 - 4.1.- CONSTANTES ENTERAS
 - 4.2.- CONSTANTES REALES
 - 4.3.- VARIABLES ENTERAS
 - 4.4.- VARIABLES REALES
- 5.- EXPRESIONES ARITMETICAS
 - 5.1.- OPERADORES ARITMETICAS
 - 5.2.- EXPRESIONES ARITMETICAS
 - 5.2.1.- REGLAS PARA FORMAR ARIT -EXPR
 - 5.3.- PROPOSICION DE ASIGNACION
- 6.- ENTRADA Y SALIDA (1a. Parte)
 - 6.1.- PROPOSICION PRINT
 - 6.2.- PROPOSICION READ

7.- TRANSFERENCIA DE CONTROL

7.1.- ETIQUETAS

7.2.- PROPOSICION GO TO

7.3.- PROPOSICION IF ARITMETICA

8.- ARREGLOS

8.1.- VARIABLES CON SUBINDICE

8.2.- DECLARACION DIMENSION

8.3.- REGLAS PARA FORMAR SUBINDICES

9.- ENTRADA Y SALIDA (2a. Parte)

9.1.- PROPOSICION READ

9.2.- PROPOSICION WRITE

9.3.- PROPOSICION FORMAT

10.- ITERACIONES

10.1.-PROPOSICION DO

10.2.-ANIDAMIENTO

10.3.-PROPOSICION CONTINUE

11.- PROPOSICION IF LOGICA

11.1.-EXPRESIONES LOGICAS

11.2.-PROPOSICION IF LOGICA

12.- FUNCIONES

12.1.-FUNCIONES PROPORCIONADAS POR EL COMPILADOR

12.2.-FUNCIONES DEFINIDAS POR EL USUARIO

13.- SUBPROGRAMAS

13.1.-SUBP. FUNCTION

13.2.-SUBP. SUBROUTINE

13.3.-PROPOSICION COMMON

1.- Introducción al lenguaje FØRTRAN

El lenguaje FØRTRAN, cuyo nombre corresponde a las primeras letras de las palabras inglesas FØRmula (fØrmula) y TRANslation (traducción), es un lenguaje de programación orientado a problemas matemáticos y se emplea en casi todas las computadoras del mundo. Debido a su parecido con el lenguaje aritmético común, el FØRTRAN simplifica la preparación de problemas que pueden resolverse mediante una computadora. Los datos e instrucciones se pueden organizar mediante una secuencia de enunciados FØRTRAN; éstos constituyen el llamado Programa Fuente.

Todas las computadoras que "entienden" el lenguaje FØRTRAN. tienen lo que se llama un Compilador Fortran, llamado también traductor o intérprete, el cual analiza los enunciados FØRTRAN y los traduce a un Programa Objeto, el cual queda en Lenguaje de Máquina.

Un programa escrito en lenguaje FØRTRAN se puede procesar en cualquier máquina que tenga un Compilador FØRTRAN. Esto nos indica que el lenguaje es independiente para cada máquina, o sea que el compilador se debe preparar en cada caso teniendo en cuenta la máquina que ha de usarse en particular; puesto que las máquinas difieren en su organización interna, se ha desarrollado un número de "dialectos" del Lenguaje FØRTRAN, cada uno de los cuales es apropiado para una clase de máquinas. Las diferencias entre los varios dialectos son mínimas y se ajustan el uno al otro fácilmente.

3.- ELEMENTOS BASICOS

3.1 Juego de Caracteres

El alfabeto FØRTRAN está constituido de caracteres que son símbolos familiares de escritura y de teclados de máquinas de escribir, así como de dispositivos especiales de perforación; dichos caracteres son:

Alfabéticos:	A B C D E F G H
	I J K L M N
	* Ø P Q R S T U V W X Y Z
Númericos:	0 1 2 3 4 5 6 7 8 9
Símbolos:	+ - * / = . , () ' a

De este alfabeto se construyen todos nuestros símbolos, expresiones y enunciados que se utilizan en el lenguaje FØRTRAN.

3.2 Números

Los números pueden representarse en diferentes formas, las cuales se asemejan a los símbolos de la aritmética general; pero debido a la estructura interna de las computadoras se establecen las convenciones de: Punto Fijo y Punto Flotante que proporcionan facilidades para su manejo en FØRTRAN. Los símbolos de punto fijo se usarán solamente con números enteros y los cálculos asociados se denominarán aritmética de los enteros o modo entero; mientras que la aritmética de los números reales se hará en la forma de punto flotante y se llamará aritmética de los reales o modo real. Debido a que también es necesario distinguir las constantes (números que no cambian durante toda la ejecución de un programa) de las variables (Números que pueden cambiar), surgen cuatro clases de símbolos para los números.

3.3.1 El enunciado END

(16)

Este se lee simplemente END e informa al compilador que el programa fuente ha terminado y debe ser el último enunciado de cualquier programa FORTRAN.

3.3.2, El enunciado STØP

Este aparece simplemente como STØP y es el que nos indica que ha terminado la ejecución y en el caso de IBM - 1130 la computadora se detiene y el operador tendrá que hacer que continúe trabajando. Debido a ello se recomienda que se utilice el enunciado CALL EXIT, el cual pasa el control a un programa monitor que hace que la computadora continúe ejecutando los otros programas que siguen a continuación.

Tanto el STØP como el CALL EXIT podrán aparecer después de cualquier enunciado.

4.- Constantes y Variables

(14)

4.1 Constantes enteras.

Dependiendo del tipo de computadora se podrán representar por un cierto número de dígitos, si el entero es negativo, los dígitos deberán ser precedidos del signo menos; si el entero es positivo el signo es opcional.

Ejem. Símbolos para constantes enteras pueden ser entre otras:

1976 +1 0 +1976 -1976

Símbolos que no se aceptan para constantes enteras:
7483282 (dependiendo de la computadora utilizada, puede ser demasiado grande)

1976: (el punto decimal no se permite)

4.2 Constantes reales

Dependiendo del tipo de computadora, las constantes reales se podrán representar por varios dígitos, con punto decimal pudiéndose colocar al principio de los dígitos, al final o entre dos dígitos cualesquiera. Cuando aparece un punto en una constante su tratamiento será de punto flotante. Si la constante real es precedida de un signo menos, se indicará que es negativa, si es positiva el signo es opcional.

Ejem. Símbolos para constantes reales pueden ser entre otras:

1976. -.00001976 +12.345 -12.345
-.007 .007 5.348 0.3

Símbolos que no se aceptan para constantes reales:
123456789.32

5343 (falta el punto decimal)

Para representar las constantes reales existe también la llamada forma exponencial; esta la podemos representar mediante una letra E y una constante entera de uno o dos dígitos, positiva o negativa. Esta constante entera es un exponente del número diez; el signo

menos es para los exponentes negativos y para los positivos, el signo es opcional. En FORTRAN, la presencia del exponente hace que el uso del punto decimal sea opcional.

Ejem.	Forma exponencial	Forma no exponencial
	1.328E2	132.8
	1.328E02	132.8
	1.328E00	132.8
	-4.724E-03	-.004724
	+7.61E3	7610.
	-6432E-3	-6.432

4.3 Variables enteras

Estas se representan por combinaciones de una a seis letras y/o dígitos, no se permiten otros caracteres y el primer caracter deberá ser una de las letras I, J, K, L, M ó N. El primer caracter de una variable es el que indica si es entera o real. Durante la ejecución de un programa, las variables enteras deberán restringirse a valores enteros.

Ejem. Símbolos para variables enteras pueden ser, entre otros:

NUMCT	KILO	N1	N2	M10	KONT
IIALC	JCLAV	MARY	KONTI	L1976	

Símbolos no aceptables para variables enteras:

CUENT (el primer caracter debe ser I, J, K, L, M ó N).

Kontador (demasiados caracteres)

12.34 (sólo se aceptan letras y números)

4.4. Variables reales

Estas se representan por combinaciones de una a seis letras y/o dígitos, no se permiten otros caracteres y el primer caracter tiene que ser necesariamente una letra diferente a I, J, K, L, M ó N. Durante la ejecución de un programa dichas variables se deben restringir a valores reales.

Ejem. Símbolos para variables reales pueden ser, entre otros: ⁽¹⁷⁾

FUERZ	VELOC	ACELI	CUENT	A1	A2
ALFA	VIELA	RA42	XT	PROD	SUMA

Símbolos no aceptables para variables reales:

A3.8 (el punto no es letra o número)
CORRIEN (demasiados caracteres)
3 BASO (el primer caracter debe ser una letra)
MUMCT (el primer caracter no puede ser M)

5.1 Operadores Aritméticos

Las operaciones aritméticas y los símbolos que se utilizan en FØRTRAN son:

	Ejem.	Algebra	FØRTRAN
Adición	+	$a + b$	$A + B$
Sustracción	-	$a - b$	$A - B$
Multiplicación	*	$a b$	$A * B$
División	/	$\frac{a}{b}$	A / B
Exponenciación	**	a^2	$A * B$
		a^2	$A ** 2$

5.2 Expresiones aritméticas

En base a lo expuesto anteriormente podemos ahora formular expresiones aritméticas en lenguaje FØRTRAN y nos daremos cuenta que son muy similares a las expresiones aritméticas del algebra común.

Expresiones FØRTRAN	Expresiones Comunes
$A ** 2 - B ** 2$	$a^2 - b^2$
$B ** 2 - 4 * A * C$	$b^2 - 4ac$
$(A+B)/2.$	$\frac{1}{2} * (a+b)$
$2 * K - J + N$	$2k - j + n$
$C + B - 3 * A$	$c + b - 3a$

5.2.1 Reglas para las expresiones aritméticas

Las reglas a las que debemos sujetar las expresiones aritméticas son necesarias debido a la estructura de las computadoras y al observarlas tendremos un ahorro en el tiempo de ejecución de un programa.

Regla 1 Si nos fijamos en las expresiones FØRTRAN anteriores nos damos cuenta que : Todas las constantes y variables en una expresión deben estar en el mismo modo, esto es, todas deben ser enteras o todas deben ser reales. (existe una excepción que mencionaremos más adelante).

Es necesario consultar los manuales de cada máquina, ⁽²¹⁾ ya que como hemos mencionado anteriormente dependerá esta regla del tipo de computadora. Por lo pronto la consideraremos como se ha indicado.

Si A=5., B=8., C=2. y D=1.6

Entonces (A+B)/C se calcula en el siguiente orden:

$$5.+8.=13. \quad 13./2.=6.5$$

Mientras que A+B/C se calcula en el siguiente orden:

$$8./2.=4. \quad 5.+4.=9.$$

Ahora si deseamos calcular (A+C)**2 Conducirá a:

$$5.+2.=7 \quad 7.**2=49.$$

Mientras que A+C**2 Conducirá a:

$$2.**2=4 \quad 5.+4.=9$$

Ahora si: (A*B)/(C*D)=40./3.2=12.5

Entonces: A*B/C*D=40./C*D=20.*D=32.

Finalmente si tenemos paréntesis dentro de otros paréntesis se tiene:

$$(A*(B+C)).**2=(A*10)**2=50.**2=2500.$$

B+C tiene la más alta prioridad por encontrarse en el paréntesis más interno.

$$(A*B+C)**2=(40.+2)**2=42.**2=1764.$$

$$A*(B+C)**2=A*10.**2=A*100.=500.$$

$$A*B+C**2=A*B+4.=40.+4.=44$$

Debemos tener cuidado en expresar lo que deseamos realizar.

Regla 4 No deberemos colocar un signo de operación antes de un signo más o menos, esto es, no deberemos poner dos signos de operación juntos.

Ejem. A*B I+J M+N A/-B

Estas expresiones deberán sustituirse por:

A*(-B) I+(-J) M-(-N) A/(-B)

5.3 Proposición de Asignación

Se forman con las expresiones presentadas anteriormente y nos indican los cálculos particulares que deben hacerse. Su forma es:

Variable = Expresión aritmética

El significado del signo = es el de asignación, esto es, que deberá calcularse el valor de la expresión a la derecha del signo = y su valor se asignará a la variable que se encuentre a la izquierda del signo, la cual tiene una localidad en la memoria de la computadora.

Ejem. Si $A=5.$, $B=8.$, $C=2.$ y $D=1.6$

$X=(A+B)/C$ se le asignará a la X el valor 6.5

$ALO=(A+B)**2$ se le asignará a ALO el valor 169.

$RAI+SQRT(B*C)$ se le asignará a RAI el valor 4.

Algo diferente al álgebra normal es el enunciado

$A=A+3.$ el cual no debe alarmarnos ya que indica que a la localidad de memoria con el nombre A se le asignará el nuevo valor $A+3.$ esto es:

Si $A=5.$ y $A=A+3.$ entonces :

$A=5.+3.$ $A=8.$ o sea que la variable A se le asigna el valor de 8. y el valor anterior que fué 5. se pierde.

6.- ENTRADA Y SALIDA (1a. PARTE) (23)

Todo programa FORTRAN que realice algún cálculo ó resuelva algún problema, debe informar al usuario el resultado de sus cálculos mediante los dispositivos con que cuenta la computadora para ello, como podría ser una impresora ó una pantalla de televisión. De igual forma, la mayoría de los programas procesan la información que un dispositivo externo les proporciona, como sería una lectora de tarjetas perforadas, una unidad de cinta ó el teclado de una terminal.

Las instrucciones con que cuenta el lenguaje FORTRAN para hacer estas operaciones se llaman instrucciones de entrada y salida debido a que hacen entrar o salir información desde el programa hacia algún dispositivo externo.

Primeramente se muestran las instrucciones de entrada y salida read y PRINT en sus formas más sencillas.

6.1 Proposición PRINT.

La proposición PRINT es la encargada de mostrar en algún dispositivo de salida el valor de las variables que el programador desee; su forma general más sencilla es la siguiente:

PRINT * , out list.

En la forma anterior, el asterisco es un indicador de que la forma de impresión de los valores se hará de acuerdo a un standard definido por el lenguaje, es decir, en vez del asterisco se podrá poner un indicador que especificará la forma en que deseemos que se impriman los valores de las variables (cuantas columnas en blanco, cuantos decimales después del punto, etc).

Este indicador se llama indicador de FORMATO y se estudiará ampliamente más adelante; por el momento, bastará con que utilicemos el asterisco para no preocuparnos por esos detalles.

Lo que está representado por outlist es la lista de las variables que deseamos que se impriman, por ejemplo:

A= 13.5

B= 44.44

I= 123

PRINT *, A, B, I.

Produciría lo siguiente:

1,3500000E+00 4.4440000E+01 123

La impresión en formato exponencial de las variables reales "A" y "B" se debe al standard del lenguaje cuando se utilizó el * en vez del indicador de formato.

6.2 Proposición READ

La proposición READ efectúa la operación contraria de la proposición PRINT, es decir, toma un número que se proporciona por algún dispositivo externo y lo asigna a una variable. La forma general de la proposición READ es completamente similar a la proposición PRINT, es decir:

READ * , in list.

En donde nuevamente el asterisco indica que se tomen los datos de entrada sin importar la forma en la que venga; por ejemplo, dada la instrucción.

READ *, X, Y, Z.

y suponiendo que los datos externos fueron:

22 - 18.37 4.24E+5

Después de ejercitar la instrucción READ la variable X tendría el valor 22, la variable Y valdría -18.37 y la variable Z sería igual a 425000.

7.- Transferencia de Control

25

7.1 Etiquetas

Debido a que los enunciados de un programa FØRTRAN se ejecutan en el orden que aparecen y que en muchas ocasiones queremos transferir la ejecución a otros enunciados si se satisface una cierta condición, FØRTRAN nos permite numerar dichos enunciados. Un número de enunciado debe ser una constante entera de uno a cinco caracteres. sin el signo más ò menos; el número se coloca a la izquierda del enunciado.

Ejem. 3 CONT = CONT+1.
 24 RAIZ = (A**2+B**2)**.5

7.2 Proposición GO TO

Este toma la forma GO TO N en donde N es un número de enunciado.

El GO TO produce un salto incondicional; así GO TO 3 envía la ejecución al enunciado número 3 que puede ser la instrucción de conteo del ejemplo anterior. GO TO 24 pesa el control al enunciado 24 que puede ser el del ejemplo anterior.

Ejem. Supongamos que unos de los enunciados de un programa son:

1 = 1
 ISUM = 0
 1 ISUM = ISUM+1
 I = 1+1
 GO TO 1

Esto nos representa la suma de los números enteros, desde luego es necesario ponerle otros enunciados pero por el momento nos aclara lo indicado.

7.3 IF ARITMETICO

La instrucción IF aritmética es la parte del lenguaje FORTRAN que se encarga de efectuar una transferencia condicional del flujo de control, es decir, de hacer una transferencia de control como la instrucción GOTO pero, a diferencia de éste, que siempre transfiere el control a un solo lugar, el IF aritmético puede transferir el control a uno de tres lugares diferentes, dependiendo del valor que tenga una expresión dada.

Lo que determina el lugar a transferir es el valor de una expresión aritmética, teniéndose tres destinos diferentes que se escogen de acuerdo al valor negativo, igual a cero o positivo que tenga la expresión; de lo anterior se desprende el nombre de la instrucción (IF aritmética), ya que el destino de la transferencia depende de un valor aritmético.

La forma general de la instrucción IF aritmética es la siguiente:

```
IF (arit-expresión) ln1, ln2, ln3
```

En donde arit-expresión es cualquier expresión aritmética y ln1, ln2 y ln3 son etiquetas de líneas de destino; al ejecutarse esta instrucción, se evalúa la expresión aritmética y después se transfiere el control a la línea ln1 si la expresión resultó negativa, a la línea ln2 si la expresión es igual a cero ó a la línea ln3 si la expresión es positiva.

Al escribir programas FORTRAN, es frecuente que se requiera saber si una variable determinada es igual a un cierto valor o no y, dependiendo de la respuesta, efectuar algún cálculo u otro diferente; es por esto que la expresión aritmética más común dentro de un IF aritmético es la resta de una variable menos una constante.

Para averiguar si la variable es igual al valor dado, se resta a la variable la constante dada; de esta manera, si el resultado es igual a cero, sabremos que la variable es igual a la constante; en caso contrario, si el resultado es positivo, la variable es mayor que la constante y si el resultado es negativo, la variable es menor que la constante; veamos un ejemplo:

```
IF ( A - 7.5 ) 12, 25, 88
```

```
C AQUI SE TRANSFIERE EL CONTROL SI "A" ES MENOR QUE 7.5
```

```
12 - - -
```

```
- - -
```

```
C AQUI SE TRANSFIERE CUANDO "A" ES IGUAL A 7.5
```

```
25 - - -
```

```
- - -
```

```
C. Y AQUI CUANDO "A" ES MAYO QUE 7.5
```

```
88 - - -
```

```
- - -
```

Si al programador no le interesan los tres resultados que proporciona el IF aritmético sino sólo dos de ellos, se puede repetir la misma etiqueta en dos de los destinos del IF aritmético. Por ejemplo, si en un programa hay que efectuar un cálculo cuando la variable NUM sea menor o igual a la variable LIMIT, y otro cálculo diferente cuando NUM sea mayor que LIMIT, se utilizaría el siguiente IF aritmético.

```
IF ( NUM - LIMIT ) 20, 20, 40
```

```
C CALCULO CUANDO NUM <= LIMIT
```

```
20. - - -
```

```
- - -
```

```
C. CALCULO CUANDO NUM > LIMIT
```

```
40 - - -
```

```
- - -
```

Finalmente no hay que olvidar que, en forma similar a la instrucción GOTO, la instrucción IF aritmética siempre transferirá el control a partir de ella hacia algún otro punto dentro del programa por lo que la instrucción que siga inmediatamente después de la instrucción IF aritmética sólo se ejecutará si tiene una etiqueta y el control de transferencia a ella desde algún otro lugar del programa, por lo cual, es conveniente que alguno de los destinos del IF aritmético se coloque inmediatamente después de la instrucción IF. (28)

8.- Arreglos

Frecuentemente tratamos con un grupo de variables que forman ó pertenecen a una clase o colección. Cuando las variables forman un conjunto ordenado, pueden relacionarse unas con otras por la notación de subíndices; entonces designamos esa colección como arreglo y las variables que pertenecen a ésta serie son elementos del arreglo. A veces se emplea como sinónimo de arreglo el nombre de matriz y, en consecuencia, hablamos de elementos de la matriz.

8.1 Variables con subíndice

Un conjunto de números que pueda arreglarse en un renglón ó columna se considera como un arreglo lineal ó unidimensional, y ésta serie puede llamarse vector. Identificamos los elementos de un vector renglón ó columna por un sólo subíndice.

Ejem. La columna de números del vector llamado A, consiste de los elementos A, hasta A_n inclusive y se representa como sigue:

Notación acostumbrada	Notación FORTRAN
A ₁	A (1)
A ₂	A (2)
A ₃	A (3)
⋮	⋮
A _i	A(i)
⋮	⋮
A _n	A(N)

Cada una de estas A(I), en donde I varía de 1 a N, son el nombre de una variable, el conjunto de todas ellas es lo que llamamos arreglo.

Si se usan dos subíndices para identificar los elementos de un arreglo se considera éste como un arreglo bidimensional. Los cuadros de un tablero de ajedrez, pueden considerarse como un arreglo bidimensional. Y si llamamos a cualquiera de los cuadros con la variable CTAJ tendremos 64 variables; pero como el tablero tiene 8 renglones y 8 columnas, podemos referirnos al cuadro que se encuentra en el renglón 3 y la columna 5 con la variable CTAJ (3,5).

Dependiendo del tipo de computadora será el número de sub-índices que podremos asignarle a un arreglo; en IBM - 1130 sólo se admiten arreglos con un máximo de tres subíndices. (30)

Las variables que se utilicen para designar arreglos deberán observar las reglas que se dieron anteriormente al hablar de variables enteras y reales considerando que para los cinco caracteres alfanuméricos son independientes de los índices que se encuentran entre paréntesis.

8.2 Declaración DIMENSION

Siempre que en un programa utilicemos variables con sub-índices deberemos poner como primer enunciado el DIMENSION, el cual indica al compilador qué tanto espacio de memoria se debe reservar para las variables con subíndices. Su forma es:

DIMENSION u, v, w, ...

Donde u, v, w, ... son nombres de variables, cada una de las cuales va seguida por el máximo número de elementos en el arreglo correspondiente. Deberán observarse las siguientes reglas:

Regla 1 Cada variable con subíndices se debe mencionar en un enunciado DIMENSION antes de su primer uso en el programa.

Regla 2 Los símbolos representados anteriormente por u, v, w, ... deben tener la forma:

nombre de variable (máximo número de elementos)

el número entre paréntesis debe ser una constante entera sin signo.

Ejem. DIMENSION A(20), B(4,8), CARR(5,3,4)

Esto indica que el compilador reservará 20 localidades para el arreglo A, sus veinte variables serán A(1), A(2), ..., A(20) al mismo tiempo se reservarán 32 (4X8) localidades para las variables B(1,1), B(1,2), B(1,3), ..., B(1,8), B(2,1), B(4,1), ..., B(2,8), B(3,1), B(3,2), ..., B(3,8), B(4,1), B(4,2), ..., B(4,8) y por último se reservarán 60 (5X3X4) localidades para las variables del arreglo CAR, con tres subíndices cada una.

Regla 3 El arreglo que se use en particular, dentro del programa podrá tener menos elementos que los especificados en la magnitud del enunciados DIMENSION, pero no más.

Regla 4

La variable tal como aparece en el enunciado DIMENSION debe tener exactamente el mismo número de subíndices que en cualquier otra parte del programa.

8.3 Reglas para formar subíndices

Regla 1

Un subíndice debe ser un entero, puede ser constante, variable ó una de las expresiones aritméticas siguientes:

$$A * V + b \quad A * V - b$$

en donde v es una variable entera y a y b son constantes enteras sin signo.

Ejem. Algunos subíndices pueden ser:

$$1 \quad 1972 \quad 10 * KONT \quad 2 * I \quad J \\ 1976 * N - 8 \quad 2 * I - 4 \quad 2 * I + 3$$

No se pueden usar como subíndice:

$$1 + I \quad -I \quad 2 - 10 * CONT \quad -1932 \quad -KILO$$

Regla 2

Un subíndice sólo debe tomar valores positivos

Regla 3

Un subíndice en sí no debe ser una variable con subíndices. Así X(I(2)) no es permitido.

Regla 4

Un símbolo que representa un arreglo, una variable con subíndice, no debe usarse sin subíndices para representar otra variable diferente en el mismo programa. Esto es A(I) y A no deben referirse a variables diferentes. Como siempre hay una excepción que por ahora no tocaremos.

Ejem. Los símbolos para variables reales con subíndices podrían incluir:

$$X(I) \quad SUM(K+2) \quad A(I, 2 * J + 1) \quad B(INT) \quad C(I, J)$$

Para variables enteras con subíndices.

podemos tener:

$$INT(M, N) \quad I(J) \quad ICTA(J, 2 * I).$$

9.- Entrada y Salida (2a. Parte) (32)

Estos, como su nombre lo indica, sirven para introducir y sacar información de la computadora.

9.1 Proposición READ

Este enunciado tiene la forma READ (I, N) LISTA I y N son enteros sin signo y LISTA representa una lista de nombres de variables para las cuales se leerán valores. I designa el tipo de periférico de entrada que se utilice (lectora de tarjetas, consola, etc.). N es el número de un enunciado FØRMAT asociado al READ.

Ejem. El enunciado READ (2, 101) J, B, H
Producirá la lectura de tres números: un entero y dos reales y se almacenarán en las localidades de la memoria de la computadora designadas con las variables J, B y H en su orden. Las comas que separa estos nombres de variables en el READ son indispensables, 2 es la unidad de entrada y 101 un FØRMAT.

9.2 Proposición WRITE

Este tiene la forma WRITE (I, N) LISTA I y N son enteros sin signo y LISTA representa una lista de variables para las cuales se imprimen valores. I designa el tipo de periférico de salida que se utilice (impresora, cinta, etc.). N es el número de un enunciado FØRMAT asociado al WRITE.

Ejem. El enunciado WRITE (3,108) L,X,Y
Producirá que se impriman los valores de las variables L, X y Y que se encuentren en las localidades de memoria con esos nombres, en el formato especificado por el enunciado número 108 y por la unidad de salida número 3; las comas que separan estos nombres de variables en el WRITE son indispensables.

9.3 Declaración FORMAT

Este tipo de enunciado no inician por si mismos los cálculos, no producen transferencia de control ni estimulan el flujo de información, pero proveen al compilador FØRTRAN de los detalles esenciales para la traducción del programa fuente en FØRTRAN al programa objeto en lenguaje de máquina ó para la conversión de datos a la entrada o la salida.

Si queremos introducir datos a la computadora lo podemos hacer mediante un enunciado que esté dentro del programa, como $A = 3.1416$, ésto es lo que podríamos llamar inicializar una variable; y el programa se compilaría cada vez que quisieramos darle un valor diferente a A, lo cual resulta muy costoso, ya que las compilaciones son laboriosas. Para evitar esto se usa el enunciado READ y los valores que se le den a A podrán estar en tarjetas de datos, los cuales son independientes del programa fuente.

El enunciado FØRMAT

Este tiene la forma: N FØRMAT (, , , ...) en la cual N es el número del enunciado FØRMAT y corresponde al N de los enunciados READ y WRITE. Los espacios entre las comas están disponibles para las especificaciones del tipo que se describen más adelante, siendo el número de espacios uno o más, de acuerdo a las necesidades del programador.

La especificación I: Iw

Aquí I indica un valor entero y W es un entero que indica el número de columnas o ancho de campo, que ocupa ese valor en la tarjeta de entrada o en el papel de impresión. El número w deberá incluir un lugar para el signo de ese valor, siendo + opcional.

Ejem. Valor de los datos (34)
 de entrada o salida: 1130 +1620 -370 0 +14
 Especificación : 14 15 14 11 13
 La especificación F:Fw.d

Aquí F indica un valor real, w indica el número de columnas que ocupará el valor en la tarjeta de entrada o en el papel de impresión; d indica el número de cifras que se encontrarán después del punto decimal. w deberá incluir un lugar para el signo y otro para el punto decimal.

Ejem. Valor de los datos de
 entrada ó salida: 32.787 -.007 1130. +3.70
 Especificación: F6.3 F5.3 F5.0 F5.2
 La especificación E: Ew.d

Aquí E indica un valor real en forma exponencial y w indica la anchura de campo para ese valor y debe de incluir el signo, si lo hay, el punto decimal, el lugar para la letra E, un lugar para el signo del exponente, si es negativo, y dos lugares para el exponente; d indica el número de dígitos a la derecha del punto decimal.

Ejem. Valor de los datos
 de entrada o salida: .1403E04 -.7E-02 .1442E+04
 Especificación: E8.4 E7.1 E9.4

Es conveniente que cuando seamos sacar información de la computadora, tomemos en cuenta para el ancho del campo lo siguiente:

- 1.- El signo, aún cuando el + generalmente no se imprime.
- 2.- El punto decimal para las especificaciones F y E
- 3.- Por lo menos un dígito a la izquierda del punto decimal, puesto que muchas máquinas imprimirán allí un cero si otro dígito no ocurre.

Suficientes lugares para todos los dígitos significativos deseados, debido a que para los dígitos que no se les deja espacio se truncan o redondean. (35)

- 5.- Cuatro lugares para el exponente de la especificación E.
- 6.- El primer lugar se deja en blanco para el control de carro.

10.- Iteración

10.1 Proposición DØ

Este toma la forma:

$$DØ K I = L, M, N.$$

$$DØ K I = L, M$$

La segunda forma sólo se aplica cuando $N=1$, lo que es bastante frecuente.

K representa un número de enunciado

I representa una variable entera

L, M, N son variables enteras constantes sin signo.

El DØ produce la ejecución repetida de todos los enunciados que le sigue, hasta el enunciado número K.

La primera vez que se ejecutan estos enunciados la variable I es igual a L, en cada paso subsiguiente I se incrementa en la cantidad N, hasta hacerse mayor ó igual a M en el paso final; en este momento se termina el llamado lazo DØ y el control pasa al enunciado que está a continuación del enunciado K. Así, L es el valor inicial de la variable I y M su valor final. I se llama el índice del enunciado DØ y su valor corriente se puede usar en cálculos durante la ejecución del lazo. Todos los enunciados que le siguen al DØ hasta el número K inclusive constituyen el rango del DØ. También es posible que la variable I no se encuentre en ninguno de los enunciados del rango del DØ y esto nos indica que se realice la ejecución de todos los enunciados del rango del DØ M entre N veces (la parte entera de este cociente M/N). Debemos tomar en cuenta que: el índice I se incrementa secuencial y automáticamente durante la ejecución del lazo y que se puede, en estos momentos, tratar como cualquier variable entera; el índice I queda indefinido después de terminado el lazo DØ y puede utilizarse para cualquier uso general. El enunciado K no debe ser un enunciado de especificación ni una transferencia de control esto incluye cosas como GØ TØ, IF y DØ, así como FORMAT, END y algunos otros. Debemos considerar que no se puede desde ningún punto del programa llegar a un enunciado dentro del rango de un DØ.

Ejem. Utilizaremos un DØ para sumar los número enteros del 1 al 100, ejemplo que ya hemos visto anteriormente.

```

ISUM = 0
DØ 1 I = 1,100
1 ISUM = ISUM+1
STOP

```

Nos damos cuenta que el DØ tiene la misma función que un IF, un GØ TØ y un contador; como podrá observarse con el ejemplo anterior.

10.3 PROPOSICIÓN CONTINUE (38)

En ocasiones, un programa FORTRAN requiere que se transfiera el control hacia un punto en el cual no se efectúe ninguna operación; por ejemplo, suponga que se tiene una instrucción DO para iterar a través de los elementos de un arreglo a fin de contar aquellos elementos que sean positivos e ignorar los que sean iguales a cero ó negativos; veamos el ejemplo:

```

NUM = 0
DO 20 I=1,n
IF (VEC (I) ) 20, 20, 10
10 NUM = NUM + 1
20 NUM = NUM

```

La instrucción con la etiqueta número 20 es la salida del IF en la cual el contador no debe incrementarse; al mismo tiempo que es el punto de regreso para la instrucción DO pero, como se ve claramente, esta instrucción no debe realizar ninguna operación, lo cual se obtiene de este ejemplo por medio de la instrucción NUM= NUM lo cual, sin embargo, podría confundirnos.

Afortunadamente, el lenguaje proporciona una instrucción que no hace nada para usarla en estos casos, esta instrucción es CONTINUE y puede usarse en cualquier lugar que el programador desee; regresando al ejemplo anterior;

```

NUM = 0
DO 20 I = 1, n
    IF (VEC (I) ) 20, 20, 10
10 NUM = NUM + 1
20 CONTINUE

```

A pesar de que las construcciones de programa FORTRAN como la anterior, en la que la instrucción CONTINUE se requiere, no son muy comunes, una práctica muy extendida dentro de la programación FORTRAN consiste en cerrar todas las proposiciones DO con proposiciones CONTINUE aunque no se necesiten; de esta ma-

nera el programador puede ver de inmediato en donde termina el rango de las proposiciones DO.

Por ejemplo, si se requiere sumar todos los elementos de una matriz de dos dimensiones con el sig. programa:

```
SUMA = 0.0
```

```
DO 10 I = 1,n
```

```
DO 10 I = 1,M
```

```
10 Suma = SUMA + MAT (I, J)
```

Es mucho más claro hacerlo de la siguiente manera:

```
DO 20 I = 1 , N
```

```
DO 10 J = 1, M
```

```
    SUMA = SUMA + MAT (I, J)
```

```
10    CONTINUE
```

```
20    CONTINUE
```

11.- Proposición IF (Lógico)

(40)

11.1 Expresiones Lógicas

Para formar las expresiones lógicas (L) utilizaremos los operadores de comparación y los de relación.

Operadores de comparación:

<u>Símbolo Matemático</u>	<u>Significado</u>	<u>Símbolo FORTRAN</u>	<u>Significado Inglés.</u>
<	Menor que	.LT.	Less than
>	Mayor que	.GT.	Greater than
≤	Menor o igual a	.LE.	Less or equal
≥	Mayor o igual a	.GE.	Greater or equal

=	igual a	.EQ.	Equal
≠	Diferente a ó No igual a	.NE.	Not equal

Operadores de relación:

∪	Unión	.OR.	ó ("o inclusive)
∩	Intersección	.AND.	y ("al mismo tiempo)
—	Complemento	.NOT.	no

Para valuar una expresión lógica se hará con las siguientes prioridades:

- 1.- Expresiones entre paréntesis
- 2.- Operadores aritméticos
- 3.- Operadores de comparación (.LT., .GT., .LE., .GE., .EQ. H .NE.)
- 4.- .NOT.
- 5.- .AND.
- 6.- .OR.

En caso de igual jerarquía la evaluación será de izquierda a derecha.

(41)

El IF lógico es de la forma:

IF (L) S

L= expresión lógica que puede tener dos valores: Verdadero o Falso.

S= cualquier enunciado FORTRAN diferente de: un DØ, un enunciado de especificación o de otro IF lógico.

Si L es falso (.FALSE.) entonces se ignora S y la computación continúa al siguiente enunciado. Si L es verdadero (.TRUE.) el enunciado S se ejecuta en seguida.

Resulta interesante hacer notar que si L es relativamente complicada, éste IF puede ser el equivalente de varios IF aritméticos.

Ejem. (1)

X=5. y=0.5

IF (X.GT.3..AND..Y .LE.2) Z=X**3+X*Y

Significa que si $X > 3$. y (al mismo tiempo) $y \leq 2$. se asignará a Z el valor que se obtenga al calcular $X^3 + XY$, esto es $Z = 125. + 2.5 = 127.5$

(2)

IF (A.LE.Z.AND.B.GE. Y .OR.C.GT.Z) GØ TØ 12

Significa que si $A \leq Z$ y (al mismo tiempo) $B > Y$ es verdadero ó $C > Z$ es verdadero ó ambos, entonces se transfiere el control al enunciado 12.

(3)

I = 1

ISUM = 0

1 ISUM = ISUM+1

I = I+1

IF (I.LE.100) GØ TØ 1

STOP

Esto nos indica que sólo sumaremos los números

enteros del 1 al 100

12.- Funciones

12.1 Proporcionados por el Compilador

Estas funciones predefinidas que proporciona el lenguaje FØRTRAN son de tipo de biblioteca. Para utilizarlas usaremos el nombre de la función seguido de un argumento que deberá estar entre paréntesis. Dichos argumentos pueden ser variables simples ó con subíndices, constantes, expresiones aritméticas u otras funciones predefinidas en FØRTRAN

Para IBM - 1130 tenemos:

<u>NOMBRE</u>	<u>FUNCION EJECUTADA</u>	<u>NUM. DE ARGUMENTOS</u>	<u>TIPO DE ARGUMENTO(S)</u>	<u>TIPO DE FUNCION</u>
SIN	Seno trigonométrico (argumento en radianes)	1	Real	Real
COS	Coseno trigonométrico (argumento en radianes)	1	Real	Real
ALOG	Logaritmo natural	1	Real	Real
EXP	Argumento de potencia del número a.	1	Real	Real
SQRT	Raíz cuadrada	1	Real	Real
ATAN	Arco Tangente	1	Real	Real
ABS	Valor absoluto	1	Real	Real
IABS	Valor absoluto	1	Entero	Entero
FLOAT	Convertir argumento de entero a real	1	Entero	Real
IFIX	Convertir argumento de real a entero	1	Real	Entero
SIGN	Transferencia de signo (Arg.1 recibe signo de Arg. 2)	2	Real	Real
ISIGN	Transferencia de signo (arg.1 recibe signo de Arg.2)	2	Entero	Entero
TANH	Tangente Hiperbólica	1	Real	Real

Ejem.

SQRT (B**2-4.*4.*A*C) indica que a lo que se encuentra entre paréntesis se le sacará la raíz cuadrada.
SIN (BETA) indica que se obtendrá el seno trigonométrico de el valor de la variable BETA.

13.- Subprogramas

Los subprogramas, también llamados subrutinas, son programas que pueden ser puestos en uso por otros programas cuando sea necesario.

Las función-s de biblioteca ó funciones del sistema constituyen una variedad de subprogramas .

13.1 FUNCIONES

Quando el valor de una variable depende de una ó más variables ó constantes y además de una serie de cálculos, y dicha variable ha de calcularse repetidamente y en diferentes puntos de un programa, es posible definirla como una Función. En otras palabras, Además de las funciones con que cuenta la biblioteca del sistema, el usuario puede escribir sus propias funciones para uso específico de su programa.

Tomemos un ejemplo para visualizar lo anterior:

Supongamos que para un programa en especial, en el cual trabajamos con grados en lugar de radianes, deseamos calcular continuamente $\text{SENØ}(X)$, sin el uso de funciones sería necesario transformar el argumento deseado de grado a radianes y después llamar a la función del sistema $\text{SIN}(X)$. A continuación presentamos una función que calculará $\text{SENO}(X)$, (X en grados):

```
FUNCTION SENØ ( X )
X = X * 3.14 15 92/180
SENØ = SIN (X)
RETURN
END
```

que es llamada desde el programa como:

```
GRAD = SENØ (GRADØS)
```

En base a este ejemplo podemos generalizar el uso de la proposición FUNCTION.

- a) Debe ser codificada en forma independiente del programa que la usará, es decir, no debe aparecer "dentro" del programa.

b) Debe empezar con la palabra FUNCTION

FUNCTION nombre (parámetro)

c) A continuación se escribe el nombre con que será llamada.

d) Después, entre paréntesis y separados por comas, aparecen los argumentos.

EJEMPLOS.-

FUNCTION RAIZ 1 (A,B,C)

RAIZ1= (B+SQRT (B **2 - 4. * A * C)) / (2,* A)

RETURN

END

FUNCTION RAIZ2 (A, B, C)

RAIZ2 = (-B - SQRT (B**2.4 * A * C)) / (2.*A)

RETURN

END

```

C   EC. SEGUNDØ GRADØ
    READ (2,100) A, B, C
100 FORMAT (3F10.5)
    X1 = RAIZ1 (A,B,C)
    X2 = RAIZ2 (A,B,C)
    WRITE (3,200) A,B,C, XI,X2
200 FORMAT (5 , F10.5)
    CALL EXIT
    END
  
```

Este ejemplo es solamente para mostrar el uso de la proposición FUNCTION y no contempla algunas situaciones como raíces complejas, etc.

13.2 SUBROUTINAS

Como es fácil notar, la proposición FUNCTION nos "regresa" un sólo valor y lo hace a través de su nombre. En muchos casos es conveniente ó necesario que se nos regrese más de un valor, para éstos casos usamos la proposición o enunciado:

SUBROUTINE.

Una subrutina es un subprograma que puede "recibir" cualquier número de parámetros (desde cero hasta un número determinado por el tipo de compilador) y puede "regresar" diferentes valores calculados.

Veamos algunos ejemplos:

Supongamos que al imprimir resultados de un cierto programa tenemos que escribir algún título usando los primeros renglones de la hojas. En tal caso podemos hacer uso de una subrutina como sigue:

```

SUBROUTINE ENCA
WRITE (3,200)
200  FORMAT (/ .1X, 'REPORTE SEMANAL' . / )
RETURN
END

```

Como vemos no hemos pasado ningún parámetro ó valor a la subrutina. Para que se ejecute ésta se debe hacer uso de la proposición CALL, de la siguiente forma:

```
CALL ENCA
```

dentro del programa y en el lugar donde deseemos que ocurra la impresión.

Discutamos ahora un ejemplo muy simple para ejemplificar el uso de parámetros. Hagamos una subrutina que "recibe" como entrada dos números, los sume y el resultado lo "regrese" en otra variable. Sean A y B los números a sumar, y C la variable en donde se pondrá el resultado.

```

SUBROUTINE  SUMA  (A,B,C)
  C= A + B
  RETURN
END

```

Es importante detenerse a ver el significado de los parámetros para las subrutinas:

La subrutina anterior SUMA puede ser llamada de diversas formas:

```

CALL  SUMA  (AA,BB,CC)
CALL  SUMA  (4, 7, X )
etc.

```

Como vemos, las variables A,B y C que aparecen en la subrutina son variables mudas o dormidas y solo tienen sentido dentro de la subrutina. Veamos lo anterior:

Supóngase el siguiente programa:

```

X1 = 3.
X2 = 4.
CALL SUMA (X1, X2, X3)
SUM= X3
WRITE (3,200) X1, X2, X3, SUM
200  FØRMAT(4 F10.5)
CALL EXIT
END

```

Se propone como ejercicio al lector que haga las veces de la máquina y escriba lo que ésta imprimiría.

La máquina imprimirá :

3.0 4.0 7.0 7.0

Una de las facilidades más útiles en subrutinas es la de pasar arreglos como parámetros, ej:

```

SUBROUTINE MAXIM     (A, MAX)
DIMENSION           A ( 10)
-----
-----
-----
-----
-----
RETURN

```

END

Supóngase que ésta subrutina encuentra el elemento del arreglo A (10) con mayor valor y lo regresa a través de la variable MAX. Es importante notar que si pasamos como parámetro uno ó más arreglos hay que dimensionarlos otra vez dentro de la subrutina, lo cual se puede hacer de al menos dos formas: 1) poniendo la dimensión que aparece en el programa que lo llama;

2) Poniéndole dimensión 1(unos)

Ejemplo:

```

DIMENSION A (10) , B (20)
-----
-----
-----
-----

```

```

CALL ORDEN     (A)

```

```

CALL MAXIM     (B)

```

```

CALL MAXIM     (A)
-----
-----
-----
-----

```

CALL EXIT

END.

Caso 1:

SUBROUTINE ØRDEN (X)

DIMENSION X (jQ)

RETURN

END.

Caso 2 :

SUBROUTINE MAXIM (Y)

DIMENSION Y (j)

RETURN

END.

13.3 COMMON.-

Como es posible visualizar en los párrafos anteriores, las variables usadas en las subrutinas, o mejor dicho, dentro de las subrutinas, son totalmente independientes a las avariables usadas en el programa principal. - Muchas veces es conveniente que tanto las subrutinas como el programa que las llama tengan variables en COMUN. Para lograr ésto existe la declaración.

COMMON

La forma general de esta proposición es:

COMMON lista de variables

donde "lista de variables" es un conjunto de variables y/o arreglos separados por comas a las cuales queremos adjudicarles la propiedad anterior, es decir, sean comunes a varios subprogramas.

Ej.

```
COMMON A,B, X (10), AB (30)
```

Esta delcaración debe aparecer al principio de cualquier programa o subrutina en que se desee usar. Veamos un ejemplo.

```
C      SUMA DE DOS NUMEROS
      COMMON A, B, C
      A= 3
      B= 7
      CALL SUMA
      Z = C
      WRITE (3,200) A, B, C, Z
200    FORMAT (4 F10.5)
      CALL EXIT
      END.
```

```
SUBROUTINE SUMA
```

51

```
COMMON A, B, C
```

```
C = A + B
```

```
RETURN
```

```
END
```

Este programa debe imprimir:

3.0

7.0

10.0

10.0

Una propiedad importante del COMMON es que si un arreglo es especificado en COMMON que dá automáticamente dimensionado, es decir, no hay que especificar dicho arreglo a través de la declaración DIMENSION.

En las siguientes páginas se muestran veintiún programas, que incluyen sus diagramas de flujo, condificaciones, datos y resultados; el objeto es que al lector pueda complementar la parte teórica con la práctica, amén de que deberá hacer los propios y procesarlos en una computadora a su alcance.

CAPITULO III

INTRODUCCION AL SISTEMA VAX - 11/780

M. EN C. ALEJANDRO JIMENEZ G.

CAPITULO III

Introducción al Sistema

VAX - 11/780

- 1.- Entrada al Sistema
- 2.- Salida del Sistema
- 3.- Consulta de Archivos
- 4.- Remoción de un Archivo
- 5.- Listado de un Programa
- 6.- Creación y Modificación de un Archivo
- 7.- Compilación y Ejecución de un Programa
- 8.- Comandos de Edición
- 9.- Impresión de Resultados

CHAPTER 1

INTRODUCCION AL SISTEMA VAX-11/780

A continuación presentamos un breve resumen de los comandos necesarios para editar, compilar y correr un programa.

1.1 COMO ENTRAR AL SISTEMA.

Oprima la tecla < RETURN >

El sistema responderá:

Que tengas una agradable sesion

Username:

Responda con el nombre de la clave que le fue asignada.

A continuación el sistema pedirá:

Password:

Teclée la clave secreta que le fue asignada a su clave.

N O T A : Al teclear el PASSWORD éste NO aparecerá en la pantalla.

El sistema verificará la validez de su clave y, en caso de aceptarla, responderá:

Bienvenidos al Sistema VAX/VMS V3.1 (C e c a f i)

26-SEP-1983 13:15:08.29

(S)

BUENAS TARDES

1.2 COMO SALIR DEL SISTEMA.

Para poder salir de sesión, bastará con dar el comando

\$ LOG

1.3 COMO VER LOS ARCHIVOS ALMACENADOS EN LA CLAVE.

Teclee el comando.

\$ DIRECTORY

Este comando nos permitirá ver los archivos almacenados en la clave. Estos aparecerán en orden alfabético y de izquierda a derecha.

1.4 COMO BORRAR UN ARCHIVO.

Para poder borrar un archivo que en un momento dado ya no interese, teclee el comando:

\$ DELETE < nombre de archivo >

1.5 COMO SACAR UN LISTADO DE UN PROGRAMA.

Para poder imprimir un programa fuente o un archivo de resultados teclee el comando:

\$ PRINT < nombre de archivo >

1.6 COMO CREAR UN NUEVO ARCHIVO Y COMO HACERLE MODIFICACIONES

A UNO YA EXISTENTE.

Para poder crear o modificar un archivo, deberá dar el comando:

\$ EDIT < nombre de archivo >

Si el archivo no existe, lo creará como nuevo; si el archivo ya existe, "traerá" como archivo de trabajo y lo podremos modificar.

1.7. COMO COMPILAR, LIGAR Y CORRER UN PROGRAMA.

Para poder ejecutar un programa, es necesario primero traducirlo a un lenguaje que entienda la máquina (lenguaje binario), posteriormente "pesarle" algunas rutinas del sistema y luego "cargarlo" a memoria para que se ejecute.

Esto lo lograremos con el comando:

```
$ COLIGO
```

Al dar este comando, el sistema responderá:

PROGRAMA:

A lo cual el usuario indicará el nombre del programa que desea compilar, ligar y correr.

Además el sistema pide el lenguaje en el que está escrito el programa, en nuestro caso FORTRAN.

LENGUAJE: FORTRAN

1.8. COMANDOS DE EDICION.

Para poder modificar un programa es necesario conocer algunos comandos del editor.

Para invocar al editor se debe dar el comando:

```
$ EDIT
```

Con esto el sistema responderá:

*

[EOR]

El asterisco (*) le informa al usuario que está en modo de edición.

[EOB] es una marca que indica el final del archivo. (57)

Para poder insertar un programa es necesario dar el comando:

* INSERT

Y a continuación teclear nuestro programa.

Cuando terminemos, se deberá teclear simultáneamente la tecla CTRL y la Z (CTRL/Z)

Para poder ver lo que hemos insertado se hace con el comando:

* TYPE WHOLE

Para poder cambiar una letra, palabra o palabras de una línea, teclee el comando:

* SUBSTITUTE/TEXTO VIEJO/TEXTO NUEVO/rango de líneas

Para poder borrar una línea, teclee el comando:

* DELETE rango de líneas

Para poder guardar en disco el programa tecleado, teclee el comando:

* EXIT

1.9 COMO MANDAR IMPRIMIR A PAPEL LOS RESULTADOS DE UN PROGRAMA.

Supongamos que el programa se llama PRUEBA.FOR

Realice la siguiente secuencia:

\$ ASIGNA

\$ RUN PRUEBA

\$ DEASIGNA

\$ PRINT PRUEBA.FOR, RES.LIS

Ejemplo de como entrar al sistema, editar un programa, compilarlo,

correrlo y mandarlo a impresion.

(58)

Que tengas una agradable sesion

Username: ALEJANDRO
Password:

Bienvenidos al Sistema VAX/VMS V3.1 (C e c a f i)

22-SEP-1983 14:15:48.46
BUENAS TARDES, YA CASI ES HORA DE COMER. BUEN PROVECHO
SI DESEA VER LAS NOTICIAS DEL DIA
TECLEE: 'NOTICIAS'

SU OPERADOR ES: Modesto Arce Salazar

* dir

Directory DISK\$CECAFI1:ALEJANDRO

APL.DIR;1	BASIC.DIR;1	CECAFI.DIR;1	ED.DIR;1
FAR.DIR;1	FORTRAN.DIR;1	LIBRO.DIR;1	MAIL.MAI;1
PALOMA.DIR;1	PS.DIR;1	SALVADOR.DIR;1	

Total of 11 files.
\$ edit prueba.for
Input file does not exist

[EOB]
*insert

c
\
\

C ESTO ES UN PROGRAMA DE PRUEBA QUE TIENE SOLAMENTE COMENTARIOS
C ESTA ESCRITO EN FORTRAN .

CZ
[EOB]
*TYPE WHOLE

1
c
2

C ESTO ES UN PROGRAMA DE PRUEBA QUE TIENE SOLAMENTE COMENTARIOS

C ESTA ESCRITO EN FORTRAN .
[EOB]

DATE DOCUMENTA A 03/28

DELETE 1
1 line deleted

59

2
C ESTO ES UN PROGRAMA DE PRUEBA QUE TIENE SOLAMENTE COMENTARIOS
*TYPE WHOLE

2
C ESTO ES UN PROGRAMA DE PRUEBA QUE TIENE SOLAMENTE COMENTARIOS

3
C ESTA ESCRITO EN FORTRAN .
LEGRJ
*SUBSTITUTE/ESTO/ESTA/2

2
C ESTA ES UN PROGRAMA DE PRUEBA QUE TIENE SOLAMENTE COMENTARIOS
1 substitution
*TYPE WHOLE

2
C ESTA ES UN PROGRAMA DE PRUEBA QUE TIENE SOLAMENTE COMENTARIOS
3
C ESTA ESCRITO EN FORTRAN .
LEGRJ
*SUBSTITUTE/ESTA/ESTO/2

2
C ESTO ES UN PROGRAMA DE PRUEBA QUE TIENE SOLAMENTE COMENTARIOS
1 substitution
*TYPE WHOLE

2
C ESTO ES UN PROGRAMA DE PRUEBA QUE TIENE SOLAMENTE COMENTARIOS
3
C ESTA ESCRITO EN FORTRAN .
LEGRJ
EXIT

DISK\$CECAFI1:(ALEJANDRO)PRUEBA.FOR;1 2 lines
\$ DIR

Directory DISK\$CECAFI1:(ALEJANDRO)

APL.DIR#1	BASIC.DIR#1	CECAFI.DIR#1	ED.DIR#1
FAR.DIR#1	FORTRAN.DIR#1	LIBRO.DIR#1	MAIL.MAI#1
PALOMA.DIR#1	PRUEBA.FOR#1	PS.DIR#1	SALVADOR.DIR#1

Total of 12 files.

\$
COLIGO PRUEBA
COMPILA * LIGA * COGORREEE !!!

LENGUAJE: FORTRAN

Compilando

Linkando

ZLINK-W-EMPTYFILE, no modules found in file DISK\$CECAFI1:(ALEJANDRO)PRUEBA.FOR

ZLINK-W-USRTFR, image DISK\$CECAFI1:(ALEJANDRO)PRUEBA.EXE;1 has no user transfer

Coocorriendo !!!!!!!!!!!!!

ZUCL-E-NOTFR, no transfer address

\$

Compilando
 ZFORT-E-ZERLENSTR, Zero-length string

61

TYPE *, 'HJ' in module PRUEBA\$MAIN at line 3
 ZFORT-F-MISSDEL, Missing operator or delimiter symbol.

[TYPE *, 'HOLA' in module PRUEBA\$MAIN at line 3
 ZFORT-F-ENDNOBJ, DISK\$CECAFI1:(ALEJANDRO)PRUEBA.FOR:2 completed with 2
 \$

SIGUE CON ERRORES, CORRIJAMOSLOS!!!!!!

EXIT PRUEBA.FOR

1
 C ESTO ES UN PROGRAMA DE PRUEBA QUE TIENE SOLAMENTE COMENTARIOS
 *TYPE WHOLE

1
 C ESTO ES UN PROGRAMA DE PRUEBA QUE TIENE SOLAMENTE COMENTARIOS

2
 C ESTA ESCRITO EN FORTRAN .

3

TYPE *, 'HOLA'

4

END

LEOBJ

*SUBSTITUTE/'//'/3

3

TYPE *, 'HOLA'

substitutions

*EXIT

DISK\$CECAFI1:(ALEJANDRO)PRUEBA.FOR:3 4 lines

\$ COLIGO

COMPILA * LIGA * COORREEE !!!

PROGRAMA: PRUEBA

LENGUAJE: FORTRAN

Compilando

Ligando

Coorriendo !!!!!!!!!!!!!

HOLA

\$

COMO YA FUNCIONO, YA NOS VAMOS, PRIMERO VAMOS A BORRARLO

DIR

Directory DISK\$CECAFI1:(ALEJANDRO)

APL.DIR:1

BASIC.DIR:1

CECAFI.DIR:1

ED.DIR:1

FAR.DIR:1

FORTRAN.DIR:1

LIBRO.DIR:1

MAIL.MAI:1

FALONA.DIR:1

PRUEBA.EXE:2

PRUEBA.EXE:1

PRUEBA.FOR:2

PRUEBA.FOR:2

PRUEBA.FOR:1

PRUEBA.OBJ:2

PRUEBA.OBJ:1

PS:DIR:1

SALVADOR.DIR:1

Total of 18 files.

\$

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

DELETE PRUEBA.FOR;*; PRUEBA.EXE;*; PRUEBA.OBJ;*
\$ DIR

(62)

62

Directorio DISK\$CECAFI\$ALEJANDRO

APL.DIR:1	BASIC.DIR:1	CECAFI.DIR:1	ED.DIR:1
FAR.DIR:1	FORTRAN.DIR:1	LIBRO.DIR:1	MAIL.MAI:1
PALOMA.DIR:1	PS.DIR:1	SALVADOR.DIR:1	

Total of 13 files.

\$ LOG

ALEJANDRO logged out at 22-SEP-1983 14:26:32.72

REC'D TELETYPE UNIT

Que tengas una agradable sesion

(6)

Username: ALEJANDRO
Password:

Bienvenidos al Sistema VAX/VMS V3.1 (C e c a f i)

22-SEP-1983 14:28:38.32
BUENAS TARDES, YA CASI ES HORA DE COMER. BUEN PROVECHO
SI DESEA VER LAS NOTICIAS DEL DIA
TECLEE: *NOTICIAS*

SU OPERADOR ES: Modesto Arce Salazar

* EDIT PRUEBA.FOR
Input file does not exist
[EOB]
*INSERT

C ESTE ES DE PRUEBA

TYPE *, 'HOLA'

END

^Z
[EOB]
*TYPE WHOLE ,

1
C ESTE ES DE PRUEBA

2
TYPE *, 'HOLA'

3
END

[EOB]
*EXIT
DISK\$CECAFI1:[ALEJANDRO]PRUEBA.FOR;1 3 lines
* COLIGO

COMPILA * LIGA * COORREEE !!!

PROGRAMA: PRUEBA
LENGUAJE: FORTRAN
Compilando
Ligando
Coorciendo !!!!!!!!!!!!!

HOLA

ASIGNA
RUN PRUEBA

* REASIGNA
* PRINT PRUEBA.FOR,RES,LIS
Job 163 entered on queue SYS\$PRINT

document-ASCII

DELETE PRUEBA.*!*

64

Print Job 163 PRUEBA completed on 22-SEP-1983 14:30

DELETE PRUEBA.!*RES.LID!*

LOG

ALEJANDRO .logged out at 22-SEP-1983 14:30:40.68

(64)

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

DATA SHEET 1000 4-2023

CAPITULO IV

APLICACIONES MATEMATICAS

M. EN I. CARLOS RAMOS L.

CAPITULO V

APLICACIONES NO MATEMATICAS

M. EN. C. ALEJANDRO JIMENEZ G.

M. EN. C. RICARDO CIRIA M.

1.- BUSQUEDA

- 1.1.- Búsqueda Secuencial
- 1.2.- Búsqueda Binaria
- 1.3.- Búsqueda Arborescente
- 1.4.- Búsqueda Digital

2.- EJEMPLOS BUSQUEDA

- 2.1.- Secuencial (Caso 1)
- 2.2.- Secuencial (Caso 2)
- 2.3.- Secuencial (Caso 3)
- 2.4.- Binaria (Caso 1)
- 2.5.- Binaria (Caso 2)

3.- ORDENAMIENTO

- 3.1.- Conteo
- 3.2.- Intercambio
- 3.3.- Inserción
- 3.4.- Selección
- 3.5.- Distribución
- 3.6.- Mezcla

4.- EJEMPLOS ORDENAMIENTO

- 4.1.- Método de la Burbuja
- 4.2.- Método Quick-Sort

a) MÉTODOS DE BÚSQUEDA POR COMPARACIONES:

En estos métodos de búsqueda se trata de encontrar la llave buscada comparando unas con otras hasta encontrarla, o decir que no se encuentra en la lista.

Existen varios métodos, dependiendo del conocimiento que se tenga de la lista:

- Búsqueda Secuencial.
- Búsqueda Binaria.
- Búsqueda Arborecente.
- Búsqueda Digital.

a i) BUSQUEDA SECUENCIAL:

Se supone que no se tiene absolutamente ninguna información acerca de la lista, y que el elemento que se está buscando puede estar en cualquier lugar de la misma, o no estar.

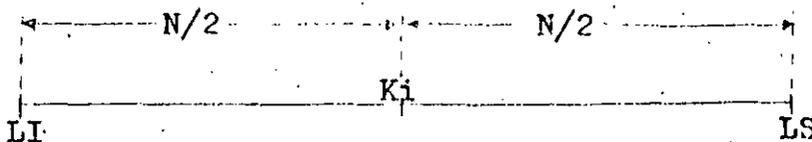
El método consiste en recorrer toda la lista, bajo algún orden (secuencial, por ejemplo), y preguntar por la llave buscada, y por medio de las comparaciones la encontraremos o sabremos a ciencia cierta que no se encuentra en la lista.

a ii) BUSQUEDA BINARIA:

En lo siguiente se supondrá que ya se tiene una tabla ordenada, y que conocemos la longitud de la lista, es decir, se conocen los límites superior e inferior.

Aquí se utilizará la información que se saca al hacer una comparación, ya que: supongamos que preguntamos por la llave K en el elemento de la mitad de la lista (cosa fácil, ya que conocemos su tamaño); Pueden suceder 3 cosas:

- K mayor que K_i ;
- $K = K_i$; ó
- K menor que K_i .



Dependiendo del resultado de esta comparación se continúa el proceso (en caso de que $k \neq k_i$, es decir, no hemos encontrado la llave buscada), pero ahora con una sublista de longitud igual a la mitad de la longitud de la lista original, con lo que el trabajo se reduce notablemente.

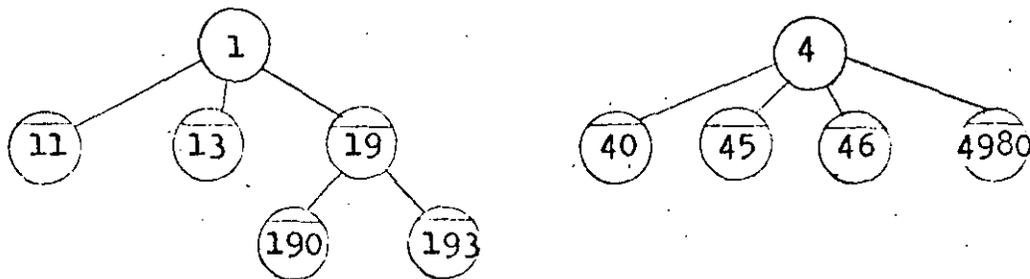
El proceso continúa similarmente hasta encontrar la llave o determinar su ausencia.

a iii) BUSQUEDA ARBORESCENTE:

Suponemos ahora que se tiene un conocimiento grande acerca del contenido de la lista. Supongamos la siguiente lista:

13 45 11 193 190 46 40 4980

Su representación arbórea sería la siguiente:



Si preguntamos por una llave 140, por ejemplo, iremos al árbol de 1's, y preguntamos por un 4 en el segundo nivel, al no existir tal, concluimos que la llave pedida no se encuentra en la lista.

Si preguntamos por un 193, nos dirigiremos al árbol de 1's, y preguntamos por un 9 en el segundo nivel, como sí existe, preguntamos por un 3 en el tercer nivel de la rama, y como sí existe y hemos agotado la llave, hemos encontrado la llave buscada.

Si preguntamos por un 55, preguntamos por un 5 en el primer nivel, y como no existe, concluimos que la llave no se encuentra en la lista.

○ Elementos que sí pertenecen a la lista.

○ Elementos auxiliares en la representación arbórescente, pero que no pertenecen a la lista.

a iv) BUSQUEDA DIGITAL:

(20)

Al igual que en la búsqueda arborescente, suponemos -- que conocemos el contenido de la lista.

La búsqueda digital es prácticamente una representación tabular de una búsqueda arborescente, pues el principio de búsqueda es prácticamente el mismo. Veamos el mismo ejemplo planteado en el inciso anterior:

13 45 11 193 190 46 40 4980

Representándolo ahora tabularmente:

	A	B	C
0		(190)	(40)
1	A	(11)	
2			
3		(13)	(193)
4	C		
5			(45)
6			(46)
7			
8			
9		B	(4980)

Si preguntáramos por la llave 193 nos dirigiremos hacia el renglón de los 1's, y nos encontramos una etiqueta A (el encontrar una llave significa que se necesita más información acerca de la llave. Por lo tanto, nos dirigimos a la intersección de la columna A y el segundo dígito de la llave, es decir, un 9. Ahí encontramos otra etiqueta, una B, por lo tanto, necesitamos más información, y nos dirigimos a la intersección de la columna B con el tercer dígito de la llave, es decir, un 3. Finalmente encontramos que la llave buscada es exactamente la que encontramos en esa localidad de memoria.

Este tipo de representación es muy utilizada para editores, o compiladores, pero se guardan los comandos en lugar de números. En caso de encontrar la llave pedida, se transferirá el

control al dispositivo correspondiente, y en caso de que no se —
encuentre, se ignorará el comando y se seguirá la acción adecua—
da.

72

21-SEP-1983 10:42:22
21-SEP-1983 10:42:18

Y (11) PUEVA
LA 2084 DE 11

```

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 C
0003 C BUSQUEDA SECUENCIAL EN TABLA DESORDENADA
0004 C
0005 C USANDO IF ARITMETICO, GOTO Y ARREGLOS DE UNA DIMENSION
0006 C
0007 C EL VECTOR IR CONTENDRA LOS REGISTROS DONDE SE BUSCARA EL
0008 C ELEMENTO K.
0009 C
0010 C LA BUSQUEDA SE HARA COMPARANDO LA LLAVE DEL REGISTRO CONTRA EL
0011 C ELEMENTO K.
0012 C
0013 C N ES EL NUMERO TOTAL DE REGISTROS.
0014 C
0015 C NOTA: ESTE PROGRAMA NO ESTA ESTRUCTURADO.
0016 C
0017 C
0018 C
0019 C
0020 C
0021 C ALEJANDRO JIMENEZ
0022 C 17 SEPT. 1983
0023 C
0024 C ESTE MODULO LLENA EL VECTOR IR CON N ELEMENTOS, N <= 100
0025 C SE LEEN DE LA TERMINAL TANTO EL VALOR N COMO LOS VALORES DEL VECTOR
0026 C IR.
0027 C DIMENSION IR(100)
0028 C WRITE (6,1000)
0029 C READ(5,1010) N
0030 C WRITE(6,1020)
0031 C I=1
0032 C 10 CONTINUE
0033 C IF (I-N) 20,20,30
0034 C 20 CONTINUE
0035 C WRITE (6,1030) I
0036 C READ (5,1040) IR(I)
0037 C I=I+1
0038 C GOTO 10
0039 C 30 CONTINUE
0040 C
0041 C ESTE MODULO IMPRIME EL VECTOR IR
0042 C
0043 C WRITE (6,1050)
0044 C I=1
0045 C 40 CONTINUE
0046 C IF (I-N) 50,50,60
0047 C 50 CONTINUE
0048 C WRITE (6,1060) I,IR(I)
0049 C I=I+1
0050 C GOTO 40
0051 C 60 CONTINUE
0052 C
0053 C ESTE MODULO REALIZA LA BUSQUEDA SECUENCIAL.
0054 C
0055 C WRITE (6,1100)

```

72

72

```

0056 C
0057 C SE LEE EL ELEMENTO A BUSCAR
0058 C
0059 C READ (5, 1700) K
0060 C
0061 C SI ES NEGATIVO, TERMINA EL PROGRAMA
0062 C
0063 110 CONTINUE
0064 IF (K) 190,120,120
0065 120 CONTINUE
0066 I=1
0067 C
0068 C COMPARA EL ELEMENTO A BUSCAR CON CADA UNO DE LOS ELEMENTOS DEL
0069 C VECTOR IR
0070 C
0071 130 CONTINUE
0072 IF (K-IR(I)) 140,170,140
0073 140 CONTINUE
0074 I=I+1
0075 C
0076 C VERIFICA SI TERMINO CON EL VECTOR COMPLETO
0077 C
0078 IF (I-N) 150,150,160
0079 150 CONTINUE
0080 GO TO 130
0081 160 CONTINUE
0082 C
0083 C FRACASO EN LA BUSQUEDA
0084 C
0085 WRITE (6,1400) K
0086 GO TO 180
0087 170 CONTINUE
0088 C
0089 C EXITO EN LA BUSQUEDA
0090 C
0091 WRITE (6,1300) K,I
0092 180 CONTINUE
0093 WRITE (6,1100)
0094 C
0095 C SE LEE EL ELEMENTO A BUSCAR
0096 C
0097 READ (5, 1200) K
0098 GO TO 110
0099 190 CONTINUE
0100 WRITE (5,1500)
0101 CALL EXIT
0102 C
0103 C ZONA DE FORMATOS
0104 C
0105 1000 FORMAT (' DAME EL NUMERO N DE DATOS.... (FORMATO 12) ')
0106 1010 FORMAT (I2)
0107 1020 FORMAT (' AHORA DAME LOS ELEMENTOS DEL VECTOR IR..(FORMATO 12)')
0108 1030 FORMAT (' DAME EL ELEMENTO ',I2)
0109 1040 FORMAT (I2)
0110 1050 FORMAT (//, ' LOS ELEMENTOS DEL VECTOR IR SON : ',//)

```

```

0111      1060  FORMAT (' IR(',12,') = ',12)
0112      1100  FORMAT (/, ' DAME EL ELEMENTO A BUSCAR ... (FORMATO 12)')
0113      1200  FORMAT (12)
0114      1300  FORMAT (/, ' EL ELEMENTO',13, ' SE ENCONTRO EN LA POSICION',13)
0115      1400  FORMAT (/, ' NO SE ENCONTRO EL ELEMENTO ',12)
0116      1500  FORMAT (//, ' FIN DE PROGRAMA')
0117      END

```

PROGRAM SECTIONS

NAME	BYTES	ATTRIBUTES
0 SCODE	534	PIC CON REL LCL SHR EXE RD COVRT LOAG
1 SPDATA	345	PIC CON REL LCL SHR COEXE RD COVRT LOAG
2 SLOCAL	412	PIC CON REL LCL NUSHR COEXE RD COVRT LOAG
TOTAL SPACE ALLOCATED		1291

ENTRY POINTS

ADDRESS	TYPE	NAME
0-00000000		SECUNUMOSMAIN

VARIABLES

ADDRESS	TYPE	NAME	ADDRESS	TYPE	NAME	ADDRESS	TYPE	NAME
2-00000194	I*4	I	2-00000198	I*4	K	2-00000190	I*4	X

ARRAYS

ADDRESS	TYPE	NAME	BYTES	DIMENSIONS
2-00000000	I*4	IR	400	(100)

LABELS

ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS
0-00000058	10	**	20	0-00000081	30	0-000000CB	40	**
0-00000144	110	**	120	0-00000150	130	**	140	**
0-00000192	170	0-0000019C	180	0-000001F5	190	1-00000000	1000'	1-00000012
1-0000006A	1030'	1-00000081	1040'	1-00000084	1050'	1-0000008E	1060'	1-00000097
1-000000F1	1300'	1-00000122	1400'	1-00000144	1500'			

74

(4)

FUNCTIONS AND SUBROUTINES REFERENCED

TYPE NAME

FORSEXIT

COMMAND QUALIFIERS

FORTRAN /CHECK/LIST SEQUENCING

/CHECK=(ROUNDS, OVERFLOW, UNDERFLOW)

/DEBUG=(NOSYMBOLS, TRACEBACK)

/STANDARD=(NOSYNTAX, NOSOURCE_FORM)

/SHD=(NOPREPROCESSOR, NOINCLUDE, MAP)

/F77 /NOG_FLOATING /I4 /OPTIMIZE /WARNINGS /NO_LINES /NO_CROSS_REFERENCE /NO_MATCH_BLOCKS

COMPILATION STATISTICS

RUN TIME: 1.78 SECONDS
ELAPSED TIME: 4.02 SECONDS
PAGE FAULTS: 292
DYNAMIC MEMORY: 126 PAGES

75

75

```

RUN SECUENUNO
DAME EL NUMERO N. DE DATOS.... (FORMATO I2)
10
AHORA DAME LOS ELEMENTOS DEL VECTOR IR..(FORMATO I2)
DAME EL ELEMENTO 1
20
DAME EL ELEMENTO 2
50
DAME EL ELEMENTO 3
30
DAME EL ELEMENTO 4
90
DAME EL ELEMENTO 5
65
DAME EL ELEMENTO 6
23
DAME EL ELEMENTO 7
38
DAME EL ELEMENTO 8
44
DAME EL ELEMENTO 9
69
DAME EL ELEMENTO 10
12

```

LOS ELEMENTOS DEL VECTOR IR SON :

```

IR( 1) = 20
IR( 2) = 50
IR( 3) = 30
IR( 4) = 90
IR( 5) = 65
IR( 6) = 23
IR( 7) = 38
IR( 8) = 44
IR( 9) = 69
IR(10) = 12

```

```

DAME EL ELEMENTO A BUSCAR ... (FORMATO I2)
45

```

NO SE ENCONTRO EL ELEMENTO 45

```

DAME EL ELEMENTO A BUSCAR ... (FORMATO I2)
12

```

EL ELEMENTO 12 SE ENCONTRO EN LA POSICION 10

```

DAME EL ELEMENTO A BUSCAR ... (FORMATO I2)
20

```

EL ELEMENTO 20 SE ENCONTRO EN LA POSICION 1

```

DAME EL ELEMENTO A BUSCAR ... (FORMATO I2)
65

```

EL ELEMENTO 65 SE ENCONTRO EN LA POSICION 5

```

DAME EL ELEMENTO A BUSCAR ... (FORMATO I2)
23

```

DAME EL ELEMENTO A BUSCAR ... (FORMATO I2)

44

77

EL ELEMENTO 44 SE ENCONTRO EN LA POSICION 8

DAME EL ELEMENTO A BUSCAR ... (FORMATO I2) (72)

70

NO SE ENCONTRO EL ELEMENTO 70

DAME EL ELEMENTO A BUSCAR ... (FORMATO I2)

-1

FIN DE PROGRAMA

\$

```

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 C      BUSQUEDA SECUENCIAL EN TABLA DESORDENADA C
0003 C      USANDO IF ARITMETICO, GOTO Y ARREGLOS DE UNA DIMENSION C
0004 C      EL VECTOR IR CONTENDRA LOS REGISTROS DONDE SE BUSCARA EL C
0005 C      ELEMENTO K. C
0006 C      LA BUSQUEDA SE HARA COMPARANDO LA LLAVE DEL REGISTRO CONTRA EL C
0007 C      ELEMENTO K C
0008 C      N ES EL NUMERO TOTAL DE REGISTROS. C
0009 C
0010 C
0011 C
0012 C
0013 C
0014 C
0015 C
0016 C
0017 C
0018 C
0019 C
0020 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

```

ALEJANDRO JIMENEZ
20 SEPT. 1983

```

0021 C
0022 C      ESTE MODULO LLENA EL VECTOR IR CON N ELEMENTOS, N <= 100 C
0023 C      SE LEEN DE LA TERMINAL TANTO EL VALOR N COMO LOS VALORES DEL VECTOR C
0024 C      IR. C
0025 C      DIMENSION IR(100) C
0026 C      WRITE (6,1000) C
0027 C      READ(5,1010) N C
0028 C      WRITE(6,1020) C
0029 C      I=1 C
0030 C      10 CONTINUE C
0031 C      IF (I-N) 20,20,30 C
0032 C      20 CONTINUE C
0033 C      WRITE (6,1030) I C
0034 C      READ (5,1040) IR(I) C
0035 C      I=I+1 C
0036 C      GOTO 10 C
0037 C      30 CONTINUE C
0038 C      ESTE MODULO IMPRIME EL VECTOR IR C
0039 C      WRITE (6,1050) C
0040 C      I=1 C
0041 C      40 CONTINUE C
0042 C      IF (I-N) 50,50,60 C
0043 C      50 CONTINUE C
0044 C      WRITE (6,1060) I,IR(I) C
0045 C      I=I+1 C
0046 C      GOTO 40 C
0047 C      60 CONTINUE C
0048 C      ESTE MODULO REALIZA LA BUSQUEDA SECUENCIAL. C
0049 C      WRITE (6,1100) C
0050 C
0051 C
0052 C
0053 C
0054 C
0055 C

```

78

78

```

0056 C SE LEE EL ELEMENTO A BUSCAR
0057 C
0058 C READ (5, 1200) K
0059 C
0060 C SI ES NEGATIVO, TERMINA EL PROGRAMA
0061 C
0062 C 100 CONTINUE
0063 C IF (K) 120,110,110
0064 C 110 CONTINUE
0065 C I=I+1
0066 C 120 CONTINUE
0067 C
0068 C VERIFICA SI TERMINO CON EL VECTOR COMPLETO
0069 C
0070 C IF (I-N) 130,130,160
0071 C 130 CONTINUE
0072 C
0073 C COMPARA EL ELEMENTO A BUSCAR CON CADA UNO DE LOS
0074 C ELEMENTOS DEL VECTOR IR
0075 C
0076 C IF (K-IR(I)) 140,150,140
0077 C 140 CONTINUE
0078 C I=I+1
0079 C GO TO 120
0080 C 150 CONTINUE
0081 C
0082 C EXITO EN LA BUSQUEDA
0083 C
0084 C WRITE (6,1300) K,I
0085 C GO TO 170
0086 C 160 CONTINUE
0087 C
0088 C FRACASO EN LA BUSQUEDA
0089 C
0090 C WRITE (6,1400) K
0091 C
0092 C 170 CONTINUE
0093 C WRITE (6,1100)
0094 C
0095 C SE LEE EL ELEMENTO A BUSCAR
0096 C
0097 C READ (5, 1200) K
0098 C GO TO 100
0099 C 180 CONTINUE
0100 C WRITE (6,1500)
0101 C CALL EXIT
0102 C
0103 C ZONA DE FORMATOS
0104 C
0105 C 1000 FORMAT (' DAME EL NUMERO N DE DATOS.... (FORMATO 12) ')
0106 C 1010 FORMAT (I2)
0107 C 1020 FORMAT (' AHORA DAME LOS ELEMENTOS DEL VECTOR IR..(FORMATO 12) ')
0108 C 1030 FORMAT (' DAME EL ELEMENTO ',I2)
0109 C 1040 FORMAT (I2)
0110 C 1050 FORMAT (//, ' LOS ELEMENTOS DEL VECTOR IR SON : ',//)

```

```

0111      1060  .FORMAT (' IR(',12,') = ',12)
0112      1100  .FORMAT ('/, ' DAME EL ELEMENTO A BUSCAR ... (FORMATO 12)')
0113      1200  .FORMAT (12)
0114      1300  .FORMAT ('/, ' EL ELEMENTO',13, ' SE ENCONTRO EN LA POSICION',13)
0115      1400  .FORMAT ('/, ' NO SE ENCONTRO EL ELEMENTO ',12)
0116      1500  .FORMAT ('/, ' FIN DE PROGRAMA')
0117      END
    
```

PROGRAM SECTIONS

NAME	BYTES	ATTRIBUTES
0 SCODE	534	PIC COM REL LCL SHR EXE RD NSBT LONG
1 SFDATA	345	PIC COM REL LCL SHR CODEX RD NSBT LONG
2 SLOCAL	412	PIC COM REL LCL NSHR CODEX RD NSBT LONG
TOTAL SPACE ALLOCATED	1291	

ENTRY POINTS

ADDRESS	TYPE	NAME
0-00000000		SECURDOSSMAIN

VARIABLES

ADDRESS	TYPE	NAME	ADDRESS	TYPE	NAME	ADDRESS	TYPE	NAME
2-00000194	I*4	I	2-00000198	I*4	K	2-00000199	I*4	N

ARRAYS

ADDRESS	TYPE	NAME	BYTES	DIMENSIONS
2-00000000	I*4	IR	400	(100)

LABELS

ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS
0-00000058	10	**	20	0-00000081	30	0-000000C6	40	**
0-00000144	100	**	110	0-00000150	120	**	130	**
0-0000019C	160	0-0000018C	170	0-000001F5	180	1-00000000	1000'	1-00000029
1-0000006A	1030'	1-00000081	1040'	1-00000094	1050'	1-000000AE	1060'	1-00000081
1-000000F1	1300'	1-00000122	1400'	1-00000143	1500'			

RUN SECUENDOS
DAME EL NUMERO N DE DATOS... (FORMATO I2)
5
AHORA DAME LOS ELEMENTOS DEL VECTOR IR..(FORMATO I2)
DAME EL ELEMENTO 1
39
DAME EL ELEMENTO 2 82
35
DAME EL ELEMENTO 3
44
DAME EL ELEMENTO 4
56
DAME EL ELEMENTO 5
67

LOS ELEMENTOS DEL VECTOR IR SON :

IR(1) = 39
IR(2) = 35
IR(3) = 44
IR(4) = 56
IR(5) = 67

DAME EL ELEMENTO A BUSCAR ... (FORMATO I2)
39

EL ELEMENTO 39 SE ENCONTRO EN LA POSICION 1

DAME EL ELEMENTO A BUSCAR ... (FORMATO I2)
35

EL ELEMENTO 35 SE ENCONTRO EN LA POSICION 2

DAME EL ELEMENTO A BUSCAR ... (FORMATO I2)
44

EL ELEMENTO 44 SE ENCONTRO EN LA POSICION 3

DAME EL ELEMENTO A BUSCAR ... (FORMATO I2)
67

EL ELEMENTO 67 SE ENCONTRO EN LA POSICION 5

DAME EL ELEMENTO A BUSCAR ... (FORMATO I2)
66

NO SE ENCONTRO EL ELEMENTO 66

DAME EL ELEMENTO A BUSCAR ... (FORMATO I2)
78

NO SE ENCONTRO EL ELEMENTO 78

DAME EL ELEMENTO A BUSCAR ... (FORMATO I2)
-1

FIN DE PROGRAMA
\$


```

0056. C SE LEE EL ELEMENTO A BUSCAR
0057 C
0058 C
0059 C READ (5, 1200) K
0060 C
0061 C SI ES NEGATIVO, TERMINA EL PROGRAMA
0062 C
0063 100 CONTINUE
0064 110 IF (K) 200,110,110
0065 110 CONTINUE
0066 110 I=1
0067 C
0068 C SE COLOCA EL CENTINELA
0069 C
0070 C
0071 C IR(I+1) = K
0072 C
0073 120 CONTINUE
0074 C
0075 C VERIFICA SI TERMINO CON EL VECTOR COMPLETO
0076 C
0077 130 IF (I-N) 130,130,160
0078 130 CONTINUE
0079 C
0080 C COMPARA EL ELEMENTO A BUSCAR CON CADA UNO DE LOS
0081 C ELEMENTOS DEL VECTOR IR
0082 C
0083 140 IF (K-IR(I)) 140,150,140
0084 140 CONTINUE
0085 140 I=I+1
0086 140 GO TO 120
0087 150 CONTINUE
0088 160 CONTINUE
0089 C
0090 C IF (I-(N+1)) 170,180,170
0091 170 CONTINUE
0092 C
0093 C EXITO EN LA BUSQUEDA
0094 C
0095 C WRITE (6,1300) K,I
0096 180 GO TO 150
0097 180 CONTINUE
0098 C
0099 C FRACASO EN LA BUSQUEDA
0100 C
0101 190 WRITE (6,1400) K
0102 190 CONTINUE
0103 190 WRITE (6,1100)
0104 C
0105 C SE LEE EL ELEMENTO A BUSCAR
0106 C
0107 C READ (5, 1200) K
0108 200 GO TO 100
0109 200 CONTINUE
0110 200 WRITE (6,1500)
0111 200 CALL EXIT

```

```

0111 C
0112 C
0113 C
0114 1000 FORMAT (' DAME EL NUMERO N DE DATOS.... (FORMATO 12) ')
0115 1010 FORMAT (I2)
0116 1020 FORMAT (' AHORA DAME LOS ELEMENTOS DEL VECTOR IR..(FORMATO 12)')
0117 1030 FORMAT (' DAME EL ELEMENTO ',I2)
0118 1040 FORMAT (I2)
0119 1050 FORMAT (///, ' LOS ELEMENTOS DEL VECTOR IR SON : ',///)
0120 1060 FORMAT (' IR(',I2,',) = ',I2)
0121 1100 FORMAT (//, ' DAME EL ELEMENTO A BUSCAR ... (FORMATO 12)')
0122 1200 FORMAT (I2)
0123 1300 FORMAT (//, ' EL ELEMENTO ',I3,', SE ENCONTRO EN LA POSICION ',I3)
0124 1400 FORMAT (//, ' NO SE ENCONTRO EL ELEMENTO ',I2)
0125 1500 FORMAT (///, ' FIN DE PROGRAMA')
0126 END
    
```

PROGRAM SECTIONS

NAME	BYTES	ATTRIBUTES
0 SCODE	562	PIC CON REL LCL SRK EXE ED SCRT LORG
1 SPDATA	345	PIC CON REL LCL SRK WPKXE RE SCRT LORG
2 SLOCAL	412	PIC CON REL LCL SOSR WPKXE ED SCRT LORG
TOTAL SPACE ALLOCATED	1319	

ENTRY POINTS

ADDRESS	TYPE	NAME
0-00000000		CENTINELASMAIN

VARIABLES

ADDRESS	TYPE	NAME	ADDRESS	TYPE	NAME	ADDRESS	TYPE	NAME
2-00000194	I*4	I	2-00000198	I*4	K	2-00000190	I*4	

ARRAYS

ADDRESS	TYPE	NAME	BYTES	DIMENSIONS
2-00000000	I*4	IR	400	(100)

(85)

LABELS

ADDRESS	LABEL								
0-00000058	10	**	20	0-000000F1	30	0-000000CB	40	**	50
0-00000144	100	**	110	0-00000162	120	**	130	**	140
0-00000182	160	**	170	0-00000188	180	0-000001D5	190	0-00000211	200
1-0000002F	1010'	1-00000032	1020'	1-0000005A	1030'	1-00000081	1040'	1-00000084	1050'
1-000000BF	1100'	1-000000EE	1200'	1-000000F1	1300'	1-00000122	1400'	1-00000146	1500'

FUNCTIONS AND SUBROUTINES REFERENCED

TYPE NAME
FORSEXIT

COMMAND QUALIFIERS

FORTRAN /CHECK/LIST CENTINELA
 /CHECK=(ROUND, OVERFLOW, UNDERFLOW)
 /DEBUG=(NOSYMBOLS, TRACEBACK)
 /STANDARD=(NOSYNTAX, NOSOURCE_FORM)
 /SHOW=(NOPREPROCESSOR, NOINCLUDE, MAP)
 /F77 /NOG_FLOATING /I4 /OPTIMIZE /WARNINGS /NOG_LINES /NOCESS_REFERENCE /NO MACHINE_CODE /NO

COMPILATION STATISTICS

RUN TIME: 1.76 SECONDS
 ELAPSED TIME: 4.93 SECONDS
 PAGE FAULTS: 419
 DYNAMIC MEMORY: 126 PAGES

RUN CENTINELA

DAME EL NUMERO N DE DATOS... (FORMATO I2)

8

AHORA DAME LOS ELEMENTOS DEL VECTOR IR..(FORMATO I2)

DAME EL ELEMENTO 1

45

DAME EL ELEMENTO 2

36

DAME EL ELEMENTO 3

78

DAME EL ELEMENTO 4

96

DAME EL ELEMENTO 5

10

DAME EL ELEMENTO 6

23

DAME EL ELEMENTO 7

34

DAME EL ELEMENTO 8

46

87

(87)

LOS ELEMENTOS DEL VECTOR IR SON :

IR(1) = 45

IR(2) = 36

IR(3) = 78

IR(4) = 96

IR(5) = 10

IR(6) = 23

IR(7) = 34

IR(8) = 46

DAME EL ELEMENTO A BUSCAR ... (FORMATO I2)

46

EL ELEMENTO 46 SE ENCONTRO EN LA POSICION 8

DAME EL ELEMENTO A BUSCAR ... (FORMATO I2)

23

EL ELEMENTO 23 SE ENCONTRO EN LA POSICION 6

DAME EL ELEMENTO A BUSCAR ... (FORMATO I2)

45

EL ELEMENTO 45 SE ENCONTRO EN LA POSICION 1

DAME EL ELEMENTO A BUSCAR ... (FORMATO I2)

98

NO SE ENCONTRO EL ELEMENTO 98

DAME EL ELEMENTO A BUSCAR ... (FORMATO I2)

99

NO SE ENCONTRO EL ELEMENTO 99

DAME EL ELEMENTO A BUSCAR ... (FORMATO I2)

96

EL ELEMENTO 96 SE ENCONTRO EN LA POSICION 4

DAME EL ELEMENTO A BUSCAR ... (FORMATO I2)

85

NO SE ENCONTRO EL ELEMENTO 85

88

(88)

DAME EL ELEMENTO A BUSCAR ... (FORMATO I2)

36

EL ELEMENTO 36 SE ENCONTRO EN LA POSICION 2

DAME EL ELEMENTO A BUSCAR ... (FORMATO I2)

34

EL ELEMENTO 34 SE ENCONTRO EN LA POSICION 7

DAME EL ELEMENTO A BUSCAR ... (FORMATO I2)

78

EL ELEMENTO 78 SE ENCONTRO EN LA POSICION 3

DAME EL ELEMENTO A BUSCAR ... (FORMATO I2)

-1

FIN DE PROGRAMA.

\$


```

0055 C
0057 C SE LEE EL ELEMENTO A BUSCAR
0058 C
0059 C READ (5, 1200) K
0060 C
0061 C SI ES NEGATIVO, TERMINA EL PROGRAMA
0062 C
0063 C 100 CONTINUE
0064 C IF (K) 200,110,110
0065 C 110 CONTINUE
0066 C L=1
0067 C IU=0
0068 C
0069 C CALCULA LA MITAD DEL VECTOR
0070 C
0071 C 120 CONTINUE
0072 C IF (IU-L) 180,130,130
0073 C 130 CONTINUE
0074 C I=(L+IU)/2
0075 C
0076 C CHECKA EN QUE MITAD SE PUEDE ENCONTRAR EL ELEMENTO
0077 C BUSCADO
0078 C
0079 C IF (K-1R(I)) 140,170,150
0080 C 140 CONTINUE
0081 C IU=I-1
0082 C GO TO 160
0083 C 150 CONTINUE
0084 C L=I+1
0085 C GO TO 160
0086 C 170 CONTINUE
0087 C WRITE (6, 1300) K,I
0088 C GO TO 190
0089 C
0090 C 160 CONTINUE
0091 C GO TO 120
0092 C 180 CONTINUE
0093 C WRITE (6, 1400) K
0094 C 190 CONTINUE
0095 C WRITE (6,1100)
0096 C
0097 C SE LEE EL ELEMENTO A BUSCAR
0098 C
0099 C READ (5, 1200) K
0100 C GO TO 100
0101 C 200 CONTINUE
0102 C WRITE (6,1500)
0103 C CALL EXIT
0104 C
0105 C ZONA DE FORMATOS
0106 C
0107 C 1000 FORMAT (' DAME EL NUMERO N DE DATOS.... (FORMATO 12) ')
0108 C 1010 FORMAT (I2)
0109 C 1020 FORMAT (' AHORA DAME LOS ELEMENTOS DEL VECTOR TE..(FORMATO 12)')
0110 C 1030 FORMAT (' DAME EL ELEMENTO ',I2)

```

```

0111 1040 FORMAT (I2)
0112 1050 FORMAT (//, ' LOS ELEMENTOS DEL VECTOR IR SON : ',//)
0113 1060 FORMAT (' IR(',I2,') = ',I2)
0114 1100 FORMAT (//, ' DAME EL ELEMENTO A BUSCAR ... (FORMATO I2)')
0115 1200 FORMAT (I2)
0116 1300 FORMAT (//, ' EL ELEMENTO ',I3, ' SE ENCONTRO EN LA POSICION ',I3)
0117 1400 FORMAT (//, ' NO SE ENCONTRO EL ELEMENTO ',I2)
0118 1500 FORMAT (//, ' FIN DE PROGRAMA')
0119 END
    
```

PROGRAM SECTIONS

NAME	BYTES	ATTRIBUTES
0 SCODE	563	PIC CON REL LCL SHR EXE RD REPORT LONG
1 SPDATA	345	PIC CON REL LCL SHR NOEXE RD REPORT LONG
2 SLOCAL	420	PIC CON REL LCL NOSHR NOEXE RD REPORT LONG
TOTAL SPACE ALLOCATED	1328	

ENTRY POINTS

ADDRESS	TYPE	NAME
0-0000000		BINARIA1\$MAIN

VARIABLES

ADDRESS	TYPE	NAME	ADDRESS	TYPE	NAME	ADDRESS	TYPE	NAME
2-00000194	I*4	I	2-000001A0	I*4	IR	2-00000198	I*4	K
2-00000190	I*4	N						

ARRAYS

ADDRESS	TYPE	NAME	BYTES	DIMENSIONS
2-00000000	I*4	IR	400	(100)

LABELS

ADDRESS	LABEL								
0-00000058	10	**	20	0-00000081	30	0-000000C8	40	**	
0-00000144	100	**	110	0-00000154	120	**	130	**	
0-00000187	160	0-00000188	170	0-00000189	180	0-00000109	190	0-00000212	200
1-0000002E	1010	1-00000032	1020	1-0000005A	1030	1-00000081	1040	1-00000084	1050
1-0000008E	1100	1-000000EE	1200	1-000000F1	1300	1-00000122	1400	1-00000144	1500



BINARY MAIN

21-SEP-1983 10:33:44
21-SEP-1983 10:33:38

VAX 11/780
[A] [D] [I] [O] [P] [R] [E] [S] [S] [E] [D]

FUNCTIONS AND SUBROUTINES REFERENCED

TYPE NAME
PURSEEXIT

COMMAND QUALIFIERS

FORTRAN /CHECK/LIST BINARY1

/CHECK=(ROUNDS, OVERFLOW, UNDERFLOW)

/DEBUG=(NOSYMBOLS, TRACEBACK)

/STANDARD=(NOSYNTAX, NOSOURCE_FORM)

/SHOW=(NOPREPROCESSOR, NOINCLUDE, MAP)

/F77 /NOG_FLOATING /I4 /OPTIMIZE /WARNINGS /NO_DUMPS /NO_CROSS_REFERENCE /NO_MACHINE_CODE /O3

COMPILATION STATISTICS

RUN-TIME: 1.87 SECONDS
ELAPSED TIME: 4.16 SECONDS
PAGE FAULTS: 425
DYNAMIC MEMORY: 130 PAGES

(92)
92

RUN BINARIA1
DAME EL NUMERO N DE DATOS.... (FORMATO I2)
5
AHORA DAME LOS ELEMENTOS DEL VECTOR IR..(FORMATO I2)
DAME EL ELEMENTO 1
10
DAME EL ELEMENTO 2 93 (93)
20
DAME EL ELEMENTO 3
30
DAME EL ELEMENTO 4
40
DAME EL ELEMENTO 5
50

LOS ELEMENTOS DEL VECTOR IR SON :

IR(1) = 10
IR(2) = 20
IR(3) = 30
IR(4) = 40
IR(5) = 50

DAME EL ELEMENTO A BUSCAR ... (FORMATO I2)
10

EL ELEMENTO 10 SE ENCONTRO EN LA POSICION 1

DAME EL ELEMENTO A BUSCAR ... (FORMATO I2)
20

EL ELEMENTO 20 SE ENCONTRO EN LA POSICION 2

DAME EL ELEMENTO A BUSCAR ... (FORMATO I2)
30

EL ELEMENTO 30 SE ENCONTRO EN LA POSICION 3

DAME EL ELEMENTO A BUSCAR ... (FORMATO I2)
40

EL ELEMENTO 40 SE ENCONTRO EN LA POSICION 4

DAME EL ELEMENTO A BUSCAR ... (FORMATO I2)
50

EL ELEMENTO 50 SE ENCONTRO EN LA POSICION 5

DAME EL ELEMENTO A BUSCAR ... (FORMATO I2)
55

NO SE ENCONTRO EL ELEMENTO 55

DAME EL ELEMENTO A BUSCAR ... (FORMATO I2)
45

NO SE ENCONTRO EL ELEMENTO 45

DAME EL ELEMENTO A BUSCAR ... (FORMATO I2)
-1


```

0056 C SE LEE EL ELEMENTO A BUSCAR
0057 C
0058 READ (5, 1200) K
0059 C
0060 C SI ES NEGATIVO, TERMINA EL PROGRAMA
0061 C
0062 100 CONTINUE
0063 IF (K) 230,110,110
0064 110 CONTINUE
0065 L=1
0066 IU=N
0067 IENCON=0
0068 C
0069 C CHECA SI LO ENCONTRO O NO
0070 C
0071 120 CONTINUE
0072 IF (1-IENCON) 130,190,130
0073 130 CONTINUE
0074 IF (1-IU) 140,140,190
0075 140 CONTINUE
0076 J=(L+IU)/2
0077 C
0078 C CHECA EN QUE MITAD SE PUEDE ENCONTRAR
0079 C EL ELEMENTO BUSCADO
0080 C
0081 IF (K-IR(1)) 150,160,160
0082 150 CONTINUE
0083 IU=I-1
0084 GO TO 180
0085 160 CONTINUE
0086 IF (K-IR(1)) 170,170,165
0087 165 CONTINUE
0088 L=L+1
0089 GO TO 180
0090 170 CONTINUE
0091 IF (K-IR(1)) 182,185,182
0092 185 CONTINUE
0093 IENCON=1
0094 182 CONTINUE
0095 180 CONTINUE
0096 GO TO 120
0097 190 CONTINUE
0098 IF (IENCON-1) 210,200,210
0099 200 CONTINUE
0100 WRITE (6, 1300) K,I
0101 GO TO 220
0102 210 CONTINUE
0103 WRITE (6, 1400) K
0104 220 CONTINUE
0105 WRITE (6,1100)
0106 C
0107 C SE LEE EL ELEMENTO A BUSCAR
0108 C
0109 READ (5, 1200) K
0110

```

95

95

```

0111          GO TO 100
0112          230 CONTINUE
0113          WRITE (6,1500)
0114          CALL EXIT
0115          C
0116          C
0117          C
0118          1000 FORMAT (' DAME EL NUMERO N DE DATOS.... (FORMATO 12) ')
0119          1010 FORMAT (I2)
0120          1020 FORMAT (' AHORA DAME LOS ELEMENTOS DEL VECTOR IR..(FORMATO 12)')
0121          1030 FORMAT (' DAME EL ELEMENTO ',I2)
0122          1040 FORMAT (I2)
0123          1050 FORMAT (//,' LOS ELEMENTOS DEL VECTOR IR SON : ',//)
0124          1060 FORMAT (' IR(',I2,',') = ',I2)
0125          1100 FORMAT (//,' DAME EL ELEMENTO A BUSCAR ... (FORMATO 12)')
0126          1200 FORMAT (I2)
0127          1300 FORMAT (//,' EL ELEMENTO ',I3,' SE ENCONTRO EN LA POSICION ',I3)
0128          1400 FORMAT (//,' NO SE ENCONTRO EL ELEMENTO ',I2)
0129          1500 FORMAT (//,' FIN DE PROGRAMA')
0130          END
    
```

PROGRAM SECTIONS

NAME	BYTES	ATTRIBUTES
0 SCODE	621	PIC COM REL LCL SHR EXE PD START LENG
1 SPDATA	345	PIC COM REL LCL SHR WDEXE PD START LENG
2 SLOCAL	424	PIC COM REL LCL WOSHR WUWEXE PD START LENG
TOTAL SPACE ALLOCATED	1390	

ENTRY POINTS

ADDRESS	TYPE	NAME
0-00000000		BINARIA2SMAIN

VARIABLES

ADDRESS	TYPE	NAME	ADDRESS	TYPE	NAME	ADDRESS	TYPE	NAME
2-00000194	I*4	I	2-000001A4	I*4	IENCOM	2-000001B0	I*4	IB
2-0000019C	I*4	L	2-00000190	I*4	N			

BINARY 2SMAIN

21-SEP-1983 10:34:24

21-SEP-1983 10:34:19

ARPAYS

ADDRESS	TYPE	NAME	BYTES	DIMENSIONS
2-00000000	I*4	IR	400	(100)

LABELS

ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS
0-00000058	10	**	20	0-000000B1	30	0-000000CB	40	**
0-00000144	100	**	110	0-00000157	120	**	130	**
0-0000018E	160	**	165	0-000001A8	170	0-000001FF	180	0-000001FF
0-000001C1	190	**	200	0-000001F3	210	0-00000213	220	0-00000200
1-0000002F	1010'	1-00000032	1020'	1-0000006A	1030'	1-00000081	1040'	1-00000080
1-0000008F	1100'	1-000000EE	1200'	1-000000F1	1300'	1-00000122	1400'	1-00000145

FUNCTIONS AND SUBROUTINES REFERENCED

TYPE NAME

FORSEXIT

COMMAND QUALIFIERS

FORTRAN /CHECK/LIST BINARIA2

/CHECK=(ROUNDS,OVERFLOW,UNDERFLOW)
 /DEBUG=(NOSYMBOLS,TRACEBACK)
 /STANDARD=(NOSYNTAX,NOSOURCE_FORM)
 /SHOW=(NOCPREPROCESSOR,NOINCLUDE,MAP)
 /F77 /NO_FLOATING /J4 /OPTIMIZE /WARNINGS /NO_D_LINES /NO_CROSS_REFERENCE /NO_MACHINE_CODE /C

COMPILATION STATISTICS

RUN TIME: 2.00 SECONDS
 ELAPSED TIME: 3.12 SECONDS
 PAGE FAULTS: 435
 DYNAMIC MEMORY: 130 PAGES

97

fb

RUN BINARIA2

DAME EL NUMERO N DE DATOS... (FORMATO I2)

6

AHORA DAME LOS ELEMENTOS DEL VECTOR IR..(FORMATO I2)

DAME EL ELEMENTO 1

11

DAME EL ELEMENTO 2 " 98

(98)

22

DAME EL ELEMENTO 3

33

DAME EL ELEMENTO 4

44

DAME EL ELEMENTO 5

55

DAME EL ELEMENTO 6

66

LOS ELEMENTOS DEL VECTOR IR SON :

IR(1) = 11

IR(2) = 22

IR(3) = 33

IR(4) = 44

IR(5) = 55

IR(6) = 66

DAME EL ELEMENTO A BUSCAR ... (FORMATO I2)

23

NO SE ENCONTRO EL ELEMENTO 23

DAME EL ELEMENTO A BUSCAR ... (FORMATO I2)

25

NO SE ENCONTRO EL ELEMENTO 25

DAME EL ELEMENTO A BUSCAR ... (FORMATO I2)

33

EL ELEMENTO 33 SE ENCONTRO EN LA POSICION 3

DAME EL ELEMENTO A BUSCAR ... (FORMATO I2)

45

NO SE ENCONTRO EL ELEMENTO 45

DAME EL ELEMENTO A BUSCAR ... (FORMATO I2)

44

EL ELEMENTO 44 SE ENCONTRO EN LA POSICION 4

DAME EL ELEMENTO A BUSCAR ... (FORMATO I2)

-1

FIN DE PROGRAMA

\$

Algunas veces es necesario ordenar los elementos de una cierta estructura de forma que sus posiciones relativas sigan un orden determinado según un campo al que se llama LLAVE. Estos ordenamientos también se utilizan para organizar los registros de un archivo, lo cual es para hacer más manejable a éste.

Hay una gran diversidad de métodos de ordenamiento, y en general cada método tiene sus ventajas y desventajas. No se puede hablar del mejor método pues la selección de éste depende del tipo de archivo, operaciones a efectuar en los datos, etc.

Los principales factores para la selección de un método de ordenamiento efectivo son:

- El tipo de forma de almacenamiento para los registros.
- La arquitectura de la máquina como aparece para el usuario, del tiempo de acceso, etc.
- La cantidad de datos a ordenar.
- La situación de los datos, es decir, qué tan ordenados están.
- El modo de las llaves (binaria, decimal, alfabética, etc.).
- La distribución de las llaves. El rango de valuación de las llaves. Qué tan cercanos están, duplicación, permutación de las llaves.
- El criterio en el que se basa la eficiencia del algoritmo, que puede ser:
 - a) El número de comparaciones de llaves.
 - b) El número de transferencias de registro.
 - c) El espacio extra de almacenamiento requerido.

Los métodos se pueden clasificar según el tipo de memoria en que se efectúa el ordenamiento:

a) INTERNOS: Cuando todos los datos a ordenar están en memoria principal

b) EXTERNOS: Cuando los datos son tantos que no caben en memoria principal y se guardan en parte de la memoria secundaria durante el proceso de ordenamiento.

Las características de los sorts internos son la complejidad, la combinación de métodos usados y la cantidad de memoria que se requiere.

Por el proceso usado, se clasifican en:

- Métodos de conteo,
- Métodos de intercambio,
- Métodos de inserción,
- Métodos de selección,
- Métodos de distribución, y
- Métodos de mezcla (o intercalación).

a i) CONTEO:

Este algoritmo es muy eficiente cuando existen muchas llaves que son iguales. Sin embargo, es necesario tener un conocimiento previo de la lista a ordenar.

El método consiste en formar un grupo por cada llave, y luego "contar" todos los elementos iguales a la llave, incrementando el número de incidencias de esa llave; Ej:

Ordenar la siguiente lista:

1,7,7,7,1,1,1,2,1,2,2,7,2,2,7,1,7

Se puede observar que se pueden formar 3 grupos: 1, 2 y 7.

grupo 1: 6 incidencias.

grupo 2: 5 incidencias.

grupo 7: 6 incidencias.

Es importante hacer notar que el tamaño de la lista debe ser mucho mayor al número de grupos a formar, para lograr una conveniente eficiencia.

a ii) INTERCAMBIO:

Consiste en comparar dos registros: Si $R_i < R_j$ entonces se intercambian; así el registro con llave mayor se irá al último lugar y, al repetir el proceso, cada llave llega a su lugar, y como los elementos grandes van subiendo como una burbuja, a este método se le llama de la BURBUJA.

Después de cada pasada, ya están en su lugar los más grandes, por lo que no es necesario compararlos con los siguientes. Este algoritmo se detiene al ya no haber intercambios. Este método

es tardado por el gran número de comparaciones que se realizan.

Se observa, sin embargo, que la lista ordenada ocupará exactamente la misma memoria que la obstruida por la lista original.

Ej: Supóngase que se desea ordenar ascendentemente la siguiente lista:

3, 6, 1, 9, 10, 2

Se compara el 3 contra el 6 y se observa que se encuentran correctamente, por lo que no se intercambian. Luego se compara el 6 contra el 1 y se observa que ahora sí habrá que hacer un intercambio de elementos, quedando la lista de la siguiente manera:

3, 1, 6, 9, 10, 2

Se siguen haciendo las comparaciones, ahora el 6 contra el nueve, y se ve que están en orden, por lo que no se hace nada. Luego, se compara el 9 contra el 10, y no se hace nada. Finalmente, comparamos el 10 contra el 2, y se ve que están en desorden, por lo que los intercambiamos:

3, 1, 6, 9, 2, 10

Se ha concluido la "primera pasada", y los elementos de la lista se encuentran aún desordenados (aunque más ordenados que como se dio inicialmente).

Se repite el proceso:

1, 3, 6, 2, 9, 10

Y otra vez:

1, 3, 2, 6, 9, 10

Finalmente:

1, 2, 3, 6, 9, 10

Se observa que ya no hay intercambios, el algoritmo termina, y la lista está ordenada.

Obsérvese que siempre se utilizó la misma regla de decisión: si $R_i < R_j$, donde $i < j$, se intercambian los elementos. Existe otro método, basado en el mismo algoritmo, con la única variación de que cada vuelta o "pasada" que se le da a la lista se cambiará el orden y la pregunta. Es decir, la segunda pasada de la lista anterior se hubiera hecho preguntando si el 10 era mayor que

el 9, y en caso contrario, intercambiarlos, y así sucesivamente. Este método es el llamado de la DOBLE BURBUJA.

El método de SHELL consiste en dividir la lista en una serie de intervalos o sublistas que se van ordenando por separado. Para tomar esas sublistas se da de antemano una sucesión de incrementos adecuada h_t, h_{t-1}, \dots, h_1 de modo que se den pasos mayores que en el algoritmo de inserción simple. Supongamos que se tienen 16 elementos: $e_1, e_2, e_3, \dots, e_{16}$. Primero se divide en cinco grupos de dos, como sigue: $(e_1, e_9), (e_2, e_{10}), (e_3, e_{11}), \dots$. Se comparan esos dos registros y se ordenan.

Luego se dividen los 16 registros en cuatro grupos de cuatro cada uno, y se ordena cada grupo: $(e_1, e_5, e_9, e_{13}), (e_2, e_6, e_{10}, e_{14}), \dots$

Ahora se divide en dos grupos de ocho cada uno, y se ordenan por separado $(e_1, e_3, e_5, e_7, e_9, e_{11}, e_{13}, e_{15})$ y $(e_2, e_4, e_6, e_8, e_{10}, e_{12}, e_{14}, e_{16})$ y se ordenan separadamente.

Por último se considera un sólo grupo de 16 registros que se ordena,

En este ejemplo, la sucesión h_t, \dots, h_1 fue 8, 4, 2, 1, pero se puede tomar cualquiera. Encontrar la mejor sucesión es un problema matemático aún no resuelto completamente, pero debe tratar de reducir los pasos.

Ej:

Ordenar ascendentemente la siguiente lista:

3, 7, 2, 1, 9, 8, 5, 6

Se compara primero el 3 contra el 9, y no se hace ningún intercambio, luego el 7 contra el 8 y tampoco se hace nada; luego el 2 contra el 5 y se dejan igual, finalmente, el 1 contra el 6 y tampoco se hace nada. Se ha terminado la primera pasada.

Ahora se consideran las siguientes comparaciones, haciendo los intercambios correspondientes, en su caso:

El 3 contra el 2, se intercambian.

El 3 contra el 9, se dejan igual.

El 9 contra el 5, se intercambian, y queda la lista como sigue:

2, 7, 3, 1, 5, 8, 9, 6

Ahora:

El 7 con el 1, se intercambian.

El 7 con el 8, se dejan igual.

El 8 con el 6, se intercambian, quedando la lista como se muestra a continuación:

2, 1, 3, 7, 5, 6, 9, 8

Ahora se considera un sólo grupo de ocho elementos y se ordena normalmente:

El 2 contra el 1, se intercambian.

El 2 contra el 3, se dejan igual.

El 3 contra el 7, se dejan igual.

El 7 contra el 5, se intercambian.

El 7 contra el 6, se intercambian.

El 7 contra el 9, se dejan igual.

El 9 contra el 8, se intercambian.

La lista queda así:

1, 2, 3, 5, 6, 7, 8, 9

Como ya no hay intercambios, el algoritmo se detiene, y la lista está completamente ordenada.

El método QUICK-SORT es también un método de intercambio, y consiste en seleccionar un pivote, que puede ser el primer elemento, el último, o la mediana (para que las sublistas sean más o menos iguales), y teniendo dos apuntes i y j , uno a cada extremo de la lista a ordenar, se compara el elemento i contra el elemento j y se hace el intercambio en su caso. En caso de que $e_i < e_j$, es decir, en caso de que no haya intercambio, se decrementa j hasta que esto ocurra. Una vez ocurrido esto, se va incrementando i hasta que haya otro intercambio, y así se sigue procediendo sucesivamente, hasta que $i=j$. Cuando esto sucede, el elemento pivote ha quedado en su posición final y a su izquierda estarán todos los elementos menores que él, y a su derecha los mayores. Así, la lista ha quedado dividida en dos sublistas a cada una de las cuales se vuelve a aplicar el mismo proceso.

Ej: Ordenar ascendentemente la siguiente lista:

1, 7, 8, 2, 5

Tomamos como pivote el 5, y lo comparamos contra el 1.

No hay intercambio. Luego, contra el 7, y ahora sí hay intercambio:

1, 5, 8, 2, 7

Ahora comparamos el 5 contra el 2, y los intercambiamos:

1, 2, 8, 5, 7

Luego, el 5 contra el 8 y los intercambiamos:

1, 2, 5, 8, 7

La lista ha quedado dividida en dos sublistas en las que se puede notar que a la izquierda del elemento pivote (el 5) han quedado todos los elementos menores a él, y a su derecha los mayores. Aplicamos el mismo proceso para cada una de las sublistas:

La sublista izquierda: comparamos el 2 contra el 1 y no hay intercambio; el algoritmo termina.

La sublista derecha: comparamos el 7 contra el 8 y los intercambiamos, y luego el 8 contra el 7, no hay intercambios, y el algoritmo termina.

Como ambas sublistas han quedado ordenadas, podemos asegurar que la lista ha quedado perfectamente ordenada:

1, 2, 5, 7, 8

++ Comparación y recomendaciones para el uso de los algoritmos de la Burbuja, Shell y Quick-Sort. (Según Saúl X. Kletzel):

- El algoritmo de la burbuja es bueno para clasificar archivos de menos de 10 elementos y semiordenados (con menos de $\log_2 N$ fuera de lugar).

- El algoritmo Shell es recomendable para conjuntos de magnitud menor o igual a 50, y la distribución inicial no influye en el comportamiento.

- En general el Quick-Sort es muy bueno para conjuntos de magnitud, no siendo bueno para pequeños. Además es eficiente para conjuntos semiordenados.

a iii) INSERCIÓN:

Los métodos de inserción consisten en insertar un determinado elemento en su lugar relativo correspondiente.

El método más simple de inserción es la directa (o interna), que consiste en tomar el segundo elemento de la lista y o locararlo en su lugar relativo a su izquierda, luego con el tercero, luego con el cuarto, y así hasta ordenar la lista completamente, - lo cual se logra automáticamente al insertar el último elemento.

ej:

- 7, 2, 5, 6, 8, 3
- 2, 7, 5, 6, 8, 3
- 2, 5, 7, 6, 8, 3
- 2, 5, 6, 7, 8, 3
- 2, 5, 6, 7, 8, 3
- 2, 3, 5, 6, 7, 8

Se recomienda que para mayor eficiencia del método se utilice una lista ligada.

También se puede hacer la inserción con banderas, que - consiste en ir distribuyendo los elementos preguntando por el valor de las banderas.

a iv) SELECCIÓN:

Esta familia de ordenamientos está basada en la idea de - seleccionar el elemento mayor o el elemento menor, y ponerlo en su lugar que le corresponde en el área de salida, y sacarlo de la lis ta.

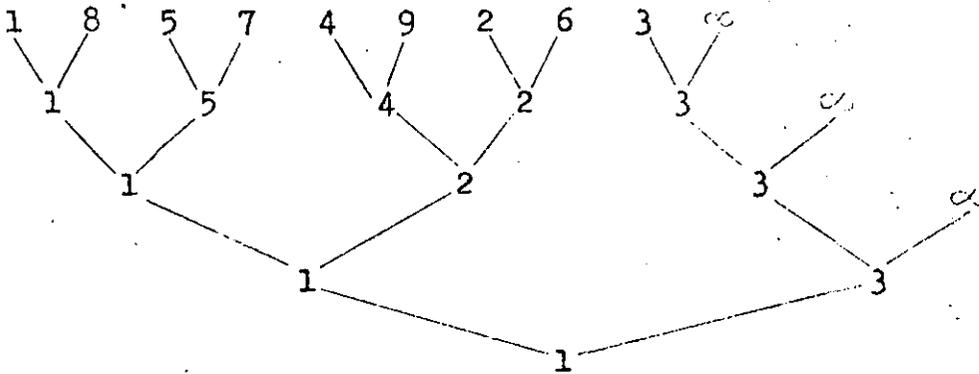
El método más simple es el llamado STRAIGHT-SELECTION, y consiste en ir tomando el elemento de mayor valor, colocarlo en su lugar definitivo de salida y el iminirlo de la lista. El proceso se repite hasta terminar con la lista.

Ej:

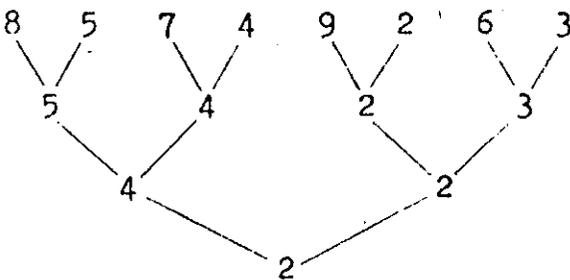
- 3, 4, 9, 7, 2
- 3, 4, 2, 7, 9
- 3, 2, 4, 7, 9
- 2, 3, 4, 7, 9

El método de TORNEO también pertenece a la familia de selección. Este método consiste en comparar de dos en dos elementos, y seleccionar el menor de entre ellos; luego, de entre los ganadores se seleccionan nuevos ganadores, y así sucesivamente hasta lograr obtener un sólo ganador, el cual se saca y se sustituye por un número infinito que queda fuera de competencia para el siguiente torneo; Este proceso se sigue hasta agotar la lista.

Ejemplo:



Se saca el uno de la lista, quedando de la siguiente forma:



Se saca el 2, quedando la lista como sigue:

8 5 7 4 9 6 3

El proceso se sigue análogamente hasta agotar totalmente la lista, con lo cual obtendremos, finalmente, la lista totalmente ordenada.

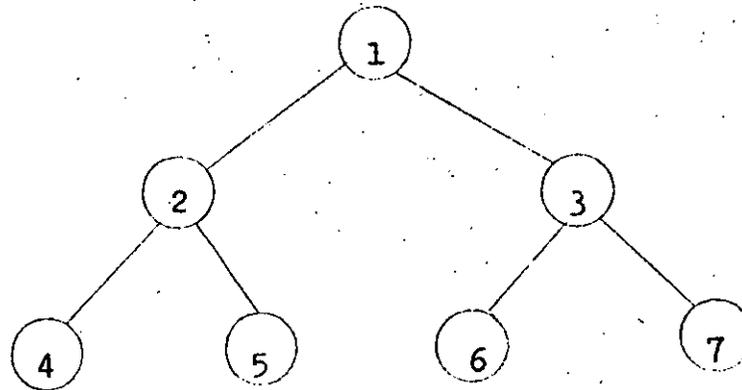
Otro ejemplo de este tipo de algoritmos es el HEAP-SORT.

Se define un heap como un árbol en el que:

$$k(j/2) \geq k_j \quad \text{para } 1 \leq (j/2) < j \leq n,$$

y esto implica que: $k_1 = \max(k_1, \dots, k_n)$.

El árbol se supone que está almacenado como sigue:



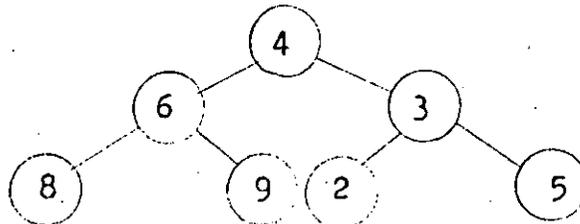
de forma que los hijos del nodo j son $2j$ y $(2j)+1$.

Así un heap es un árbol en que la raíz es mayor o igual a cualquiera de sus hijos.

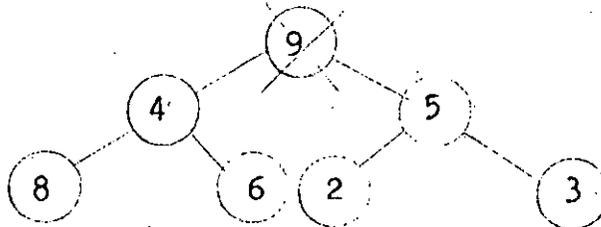
De tal manera, que deberemos formar un heap a partir de nuestra lista original, una vez logrado lo cual, la lista estará perfectamente ordenada; Ejemplo:

4 6 3 8 9 2 5
 (1) (2) (3) (4) (5) (6) (7) ← posición

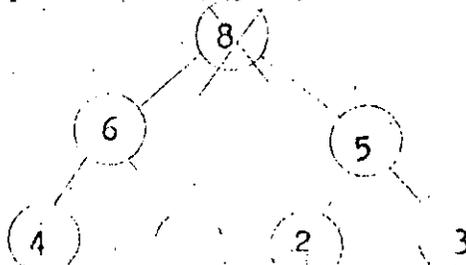
Su representación en un árbol será:



Como no es un heap, lo forzamos a que lo sea:



Aún no lo es, continuamos el proceso; pero ahora sacamos el nodo raíz, ya que es el mayor, dejando que suba el siguiente.



Este método consiste en agrupar los elementos a ordenar según ciertas características comunes, primero generales, y luego yendo a lo particular hasta llegar a lo exclusivo.

Por ejemplo, si tenemos una lista de llaves numéricas y deseamos ordenarlas, las agrupamos primero en grupos dependiendo del dígito menos significativo, luego del segundo dígito, y así sucesivamente, hasta llegar al último dígito. Una vez hecho esto, — basta con ir sacando primero los elementos cuyo dígito más significativo es menor (en caso de que el ordenamiento sea ascendente), — y seleccionar de entre ellos aquél o aquéllos cuyo dígito segundo más significativo sea menor, y de entre ellos, aquél o aquéllos — cuyo siguiente dígito sea menor, y así hasta llegar al menos significativo.

Por ejemplo, supongamos que tenemos la siguiente lista:

23, 96, 25, 66, 9, 5, 6, 89, 90

Clasificando por el último dígito (menos significativo):

						96			
					25	66			89
90			23		5	6			9
0	1	2	3	4	5	6	7	8	9

Ahora por el primer dígito (más significativo):

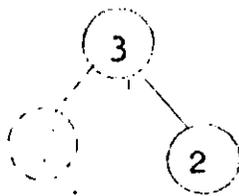
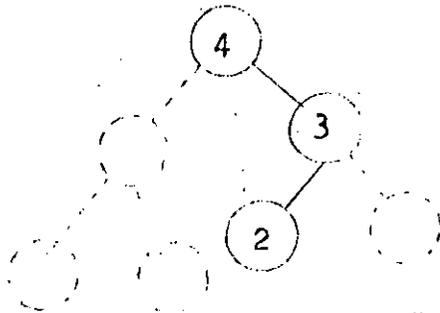
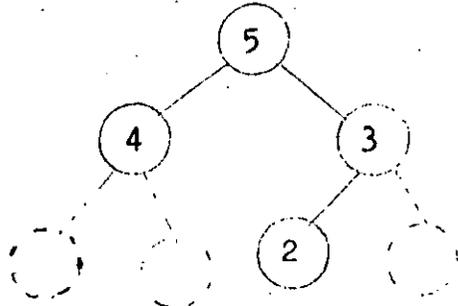
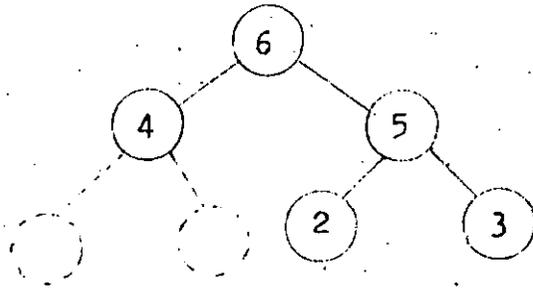
6									
5		25							96
9		23				66		89	90
0	1	2	3	4	5	6	7	8	9

Ahora, sacamos los elementos por grupos del primer dígito (— más significativo), tomando en cuenta los grupos del último dígito, es decir, sacaremos primero los elementos del grupo de los elementos cuyo último dígito sea menor de entre aquéllos cuyo primer dígito sea cero, continuando después con el 1, con el 2, etc.:

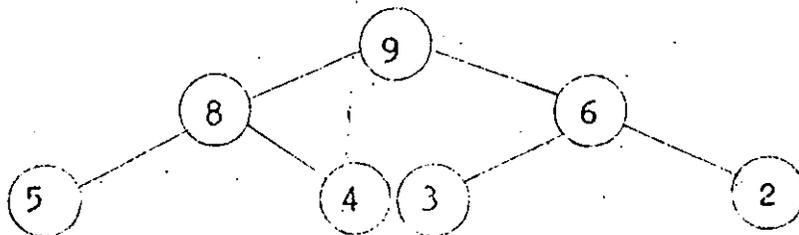
5, 6, 9, 23, 25, 66, 89, 90, 96.

a vi) MEZCLA (o Intercalación):

Esta familia de algoritmos consiste en obtener una lista



Hemos logrado formar el heap, quedando como sigue:



ordenada a partir de dos o más sublistas (llamadas VIAS), previamente ordenadas.

El proceso de mezclado es común para todos estos algoritmos, tanto de ordenamiento interno como externo, y consiste básicamente en tener un apuntador al último elemento de cada lista, y obtener el menor de ellos, sacarlo e incrementar el contador de su lista, subiendo el apuntador a su inmediatamente superior; y luego continuar análogamente hasta agotar todas las listas. Al terminar este proceso tendremos una sólo lista ordenada bajo el mismo criterio.

Pero lo que realmente diferencia un método de otro es la manera de obtener estas sublistas ordenadas, o CORRIDAS.

Entre los ordenamientos internos de mezcla encontramos principalmente dos: MEZCLA NATURAL y MEZCLA FORZADA (o binaria):

Mezcla Natural:

Consiste en tener dos apuntadores en la lista a ordenar: uno al principio y el otro al final de la misma. El primero avanzará siempre y cuando se esté de acuerdo al criterio de ordenamiento, y se detiene al no cumplirse esta condición, completándose así la primera corrida; Similarmente, el apuntador al final de la lista se mueve hacia su izquierda hasta que deje de cumplirse esa condición, formándose así la segunda corrida. El proceso se sigue análogamente hasta haber agotado la lista, es decir, hasta que los dos apuntadores se encuentren. En ese momento habremos obtenido "n" corridas de longitud mayor o igual a uno, las cuales se mezclarán, y obtendremos finalmente la lista perfectamente ordenada. Veamos un ejemplo:

1, 4, 9, 2, 8, 9, 2, 4, 1, 5, 3, 2

Corrida 1: 1, 4, 9.

Corrida 2: 2, 3, 5.

Corrida 3: 2, 8, 9.

Corrida 4: 1, 4.

Corrida 5: 2.

Mezclando 1 y 2: 1, 2, 3, 4, 5, 9.

Mezclando 3 y 4: 1, 2, 4, 8, 9.

Mezclando las dos corridas obtenidas arriba: 1, 1, 2, 2, 3, 4, 4, 5, 8, 9, 9.

Finalmente, mezclamos esta corrida con la corrida 5:

1, 1, 2, 2, 2, 3, 4, 4, 5, 8, 9, 9.

El procedimiento de Mezcla Forzada o Binaria se basa en el hecho de que una corrida de longitud uno (un sólo elemento) está ordenada.

El algoritmo consiste en formar corridas cada vez más grandes, hasta lograr tener una sólo corrida de longitud igual a la de la lista original.

Este hecho requiere que la longitud de la lista a ordenarse, necesariamente, potencia de 2, y en caso contrario, forzarla a que lo sea. Esto se debe a que las corridas que formaremos siguen la función de potencias de 2: 2, 4, 8, 16, 32, 64, etc.

El proceso consiste, pues, en lo siguiente:

Se consideran corridas de longitud 1, ordenadas, y se mezclan de dos en dos entre sí, con lo cual obtendremos nuevas corridas ordenadas, pero de longitud 2; se sigue mezclando, ahora esas nuevas corridas, obteniendo ahora corridas de longitud 4, y luego de longitud 8, y luego de 16, etc, hasta llegar a tener una sólo corrida de longitud igual a la de la lista original.

Veamos un ejemplo: sea la siguiente lista:

```

4, 6, 7, 2, 9, 8, 7, 2
4, 6, 2, 7, 8, 9, 2, 7
2, 4, 6, 7, 2, 7, 8, 9
2, 2, 4, 6, 7, 7, 9, 9

```

COMENTARIOS SOBRE LOS SORTS:

Poder comparar sorts tiene algunos problemas, como son:

- El número de comparaciones y movimientos puede estar al margen, pues aún permutaciones pueden ocasionar cambios.
- El espacio ocupado depende de la implementación del sort, el CPU, el uso de los datos después, etc.
- Las estructuras de datos y estilos de programar pueden afectar los tiempos de corrido.

Se debe tratar de combinar diferentes métodos de ordenamientos, por ejemplo, uno bueno para pocos elementos más o menos ordenados con otro, bueno para muchos y desordenados.

Se pueden combinar las siguientes reglas para seleccionar métodos de ordenamiento:

- No usar inserciones simples para archivos grandes random. Sólo si están casi ordenados.

- Los métodos se deben implementar insensibles a que haya datos duplicados.

- Un mismo algoritmo con distintas implementaciones se comporta diferente.


```

0056      110  CONTINUE
0057      IF (IT .EQ. 0) GO TO 180
0058          IFRONT=IT
0059          IF=0
0060          J=1
0061      120  CONTINUE
0062          IF (J .GT. (IFRONT-1)) GO TO 170
0063              IF (IR(J) .LE. IR(J+1)) GO TO 150
0064                  Iaux=IR(J)
0065                  IR(J)=IR(J+1)
0066                  IR(J+1)=Iaux
0067                  IF=J
0068      150  CONTINUE
0069          J=J+1
0070          GO TO 120
0071      170  CONTINUE
0072          GO TO 110
0073      180  CONTINUE
0074  C
0075  C     ESTE MODULO IMPRIME EL VECTOR IR ORDENADO
0076  C
0077      WRITE (6,1055)
0078      I=1
0079      190  CONTINUE
0080          IF (I-N) 195,195,200
0081      195  CONTINUE
0082          WRITE (6,1060) I,IR(I)
0083          I=I+1
0084          GOTO 190
0085      200  CONTINUE
0086  C
0087      WRITE (6,1100)
0088      CALL EXIT
0089  C
0090  C     ZONA DE FORMATOS
0091  C
0092      1000  FORMAT (' DAME EL NUMERO N DE DATOS.... (FORMATO 12) ')
0093      1010  FORMAT (I2)
0094      1020  FORMAT (' AHORA DAME LOS ELEMENTOS DEL VECTOR IR..(FORMATO 12)')
0095      1030  FORMAT (' DAME EL ELEMENTO ',I2)
0096      1040  FORMAT (I2)
0097      1050  FORMAT (//,' LOS ELEMENTOS DESORDENADOS DEL VECTOR IR SON ',//)
0098      1055  FORMAT (//,' LOS ELEMENTOS ORDENADOS DEL VECTOR IR SON : ',//)
0099      1060  FORMAT (' IR(',I2,',) = ',I2)
0100      1100  FORMAT (//,' FIN DE PROGRAMA')
0101      END

```

BURBUJASMAIN

21-SEP-1983 10:35:26
21-SEP-1983 10:35:22

VA 11 FORT
1A

PROGRAM SECTIONS

NAME	BYTES	ATTRIBUTES
0 SCODE	558	PIC CON REL DCL SHR EXE RD DEVRT LONG
1 SPDATA	275	PIC CON REL DCL SHR LOEXE RD DEVRT LONG
2 SLOCAL	424	PIC CON REL DCL LOSHR LOEXE RD DEVRT LONG
TOTAL SPACE ALLOCATED		1257

ENTRY POINTS

ADDRESS	TYPE	NAME
0-00000000		BURBUJASMAIN

VARIABLES

ADDRESS	TYPE	NAME	ADDRESS	TYPE	NAME	ADDRESS	TYPE	NAME
2-00000194	I*4	I	2-00000144	I*4	IAUX	2-0000019C	I*4	IPRINT
2-000001A0	I*4	J	2-00000190	I*4	K			

ARRAYS

ADDRESS	TYPE	NAME	BYTES	DIMENSIONS
2-00000000	I*4	IR	400	(100)

LABELS

ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS
0-00000058	10	**	20	0-00000081	30	0-000000CE	40	**
0-00000115	110	0-00000131	120	0-00000141	150	0-000001A5	170	0-000001A1
**	195	0-00000200	200	1-00000000	1000'	1-0000002F	1100'	1-00000022
1-00000081	1040'	1-00000084	1050'	1-00000089	1055'	1-000000ED	1060'	1-000000E0

FUNCTIONS AND SUBROUTINES REFERENCED

TYPE	NAME
	FURSEAIT

115

(115)

BURBU (SMAIN

21-SEP-1983 10:35:20
21-SEP-1983 10:35:22

V(-11 FORTIJA
L...BAYB...)

COMMAND QUALIFIERS

FORTRAN /CHECK/LIST BURBUJA

/CHECK=(BOUNDS,OVERFLOW,UNDERFLOW)

/DEBUG=(NOSYMBOLS,TRACEBACK)

/STANDARD=(NOSYNTAX,NCSOURCE_FORM)

/SHOW=(NOPREPROCESSOR,NOTINCLUDE,MAP)

/F77 /NOG_FLOATING /I4 /OPTIMIZE /WARNINGS /NOO_LINES /NOCROSS_REFERENCE /NOBACKUP_CODE /O

COMPILATION STATISTICS

RUN TIME: 1.68 SECONDS
ELAPSED TIME: 2.78 SECONDS
PAGE FAULTS: 365
DYNAMIC MEMORY: 130 PAGES

(116)

116

RUN BURBUJA

DAME EL NUMERO N DE DATOS.... (FORMATO I2)

10

AHORA DAME LOS ELEMENTOS DEL VECTOR IR..(FORMATO I2)

DAME EL ELEMENTO 1

11

DAME EL ELEMENTO 2

117

24

DAME EL ELEMENTO 3

35

DAME EL ELEMENTO 4

42

DAME EL ELEMENTO 5

12

DAME EL ELEMENTO 6

65

DAME EL ELEMENTO 7

07

DAME EL ELEMENTO 8

83

DAME EL ELEMENTO 9

01

DAME EL ELEMENTO 10

15

LOS ELEMENTOS DESORDENADOS DEL VECTOR IR SON

IR(1) = 11

IR(2) = 24

IR(3) = 35

IR(4) = 42

IR(5) = 12

IR(6) = 65

IR(7) = 7

IR(8) = 83

IR(9) = 1

IR(10) = 15

LOS ELEMENTOS ORDENADOS DEL VECTOR IR SON :

IR(1) = 1

IR(2) = 7

IR(3) = 11

IR(4) = 12

IR(5) = 15

IR(6) = 24

IR(7) = 35

IR(8) = 42

IR(9) = 65

IR(10) = 83

FIN DE PROGRAMA

\$

```

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 C ORDENAMIENTO CON EL METODO DE QUICK SORT C
0003 C C C
0004 C USANDO IF LOGICO, GOTO Y ARRREGLOS DE DUA DIMENSION C
0005 C C C
0006 C EL VECTOR IR CONTENDRA LOS REGISTROS QUE SE VAN A ORDENAR C
0007 C AL FINAL DEL PROCESO CONTENDRA AL VECTOR YA ORDENADO C
0008 C C C
0009 C EL ORDENAMIENTO SE HARA DE ACUERDO A LA TEORIA DEL METODO. C
0010 C VER: SEARCHING & SORTING DE DONALD KNUTH VOLUMEN 3 C
0011 C C C
0012 C N ES EL NUMERO TOTAL DE REGISTROS. C
0013 C C C
0014 C C C
0015 C C C
0016 C C C
0017 C C C
0018 C C C
0019 C C C
0020 C C C
0021 C CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0022 C C C
0023 C ESTE MODULO LLENA EL VECTOR IR CON N ELEMENTOS, N <= 100 C
0024 C SE LEEN DE LA TERMINAL TANTO EL VALOR N COMO LOS VALORES DEL VECTOR C
0025 C IR. C
0026 C C C
0027 C DIMENSION IR(100),ISTARL(100),ISTAKR(100) C
0028 C WRITE (6,1000) C
0029 C READ(5,1010) N C
0030 C WRITE(6,1020) C
0031 C I=1 C
0032 C 10 CONTINUE C
0033 C IF (I-N) 20,20,30 C
0034 C 20 CONTINUE C
0035 C WRITE (6,1030) I C
0036 C READ (5,1040) IR(I) C
0037 C I=I+1 C
0038 C GOTO 10 C
0039 C 30 CONTINUE C
0040 C C C
0041 C ESTE MODULO IMPRIME EL VECTOR IR DESORDENADO C
0042 C C C
0043 C WRITE (6,1050) C
0044 C I=1 C
0045 C 40 CONTINUE C
0046 C IF (I-N) 50,50,60 C
0047 C 50 CONTINUE C
0048 C WRITE (6,1060) I,IR(I) C
0049 C I=I+1 C
0050 C GOTO 40 C
0051 C 60 CONTINUE C
0052 C C C
0053 C ESTE MODULO HACE EL ORDENAMIENTO CON EL QUICK C
0054 C C C
0055 C IS=1

```

ALFONSO JIMENEZ
19 SEPT. 1983

118
118


```

0111 C
0112 WRITE (6,1100)
0113 CALL EXIT
0114 C
0115 C
0116 C
0117 1000 FORMAT (' DAME EL NUMERO N DE DATOS.... (FORMATO 12) ')
0118 1010 FORMAT (I2)
0119 1020 FORMAT (' AHORA DAME LOS ELEMENTOS DEL VECTOR IR..(FORMATO 12) ')
0120 1030 FORMAT (' DAME EL ELEMENTO ',I2)
0121 1040 FORMAT (I2)
0122 1050 FORMAT (//,' LOS ELEMENTOS DESORDENADOS DEL VECTOR IR SON ',//)
0123 1055 FORMAT (//,' LOS ELEMENTOS ORDENADOS DEL VECTOR IR SON : ',//)
0124 1060 FORMAT (' IR(',I2,') = ',I2)
0125 1100 FORMAT (//,' FIN DE PROGRAMA')
0125 END
    
```

PROGRAM SECTIONS

NAME	BYTES	ATTRIBUTES
0 SCODE	725	PIC COB REL LOC SHR EXE
1 SBDATA	275	PIC COB REL LOC SHR WORXE
2 SLOCAL	1232	PIC COB REL LOC USER WORXE
TOTAL SPACE ALLOCATED	2232	

ENTRY POINTS

ADDRESS	TYPE	NAME
0-00000000		QUICKSMAN

VARIABLES

ADDRESS	TYPE	NAME	ADDRESS	TYPE	NAME	ADDRESS	TYPE	NAME
2-000004B4	I*4	I	2-000004C0	I*4	LD	2-000004FC	I*4	LD
2-000004CC	I*4	IX	2-000004C8	I*4	IX	2-000004C4	I*4	J

ARRAYS

ADDRESS	TYPE	NAME	BYTES	DIMENSIONS
2-00000000	I*4	IP	400	(100)
2-00000190	I*4	ISTAKL	400	(100)
2-00000320	I*4	ISTAKR	400	(100)

QUICK MAIN

21-SEP-1983 10:31:54
21-SEP-1983 10:31:59

0-11-1000000
0-11-1000000

LABELS

ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS
0-00000058	10	**	20	0-00000081	30	0-000000C0	40	**
0-0000011F	70	0-0000012F	80	0-00000190	90	0-0000018A	100	0-0000012A
0-00000211	130	0-00000218	140	0-0000024C	150	0-00000255	160	0-00000217
**	195	0-000002F4	200	1-00000000	1000	1-0000002F	1010	1-00000000
1-00000081	1040	1-00000084	1050	1-00000089	1055	1-000000FD	1150	1-00000000

FUNCTIONS AND SUBROUTINES REFERENCED

TYPE NAME
FORSEXT

(14)

21-SEP-1983 10:31:54
21-SEP-1983 10:31:55

X 11.00100
37000011

0001

COMMAND QUALIFIERS

FORTRAN /CHECK/LIST QUICK

/CHECK=(NOBADS,OVERFLOW,UNDERFLOW)
/DEBUG=(NOSYMBOLS,TRACEBACK)
/STANDARD=(NOSYNTAX,NOSOURCE_FORM)
/SHOW=(NOPREPROCESSOR,NOINCLUDE,TAP)
/F77 /NOE_FLOATING /I4 /OPTIMIZE /WARNINGS /NOO_LINES /NOO_CROSS_REFERENCE /NOO_MACHINE_CODE /C

COMPILATION STATISTICS

RUN TIME: 7.37 SECONDS
ELAPSED TIME: 5.32 SECONDS
PAGE FAULTS: 366
DYNAMIC MEMORY: 130 PAGES

(22)

122

RUN QUICK

DAME EL NUMERO N DE DATOS.... (FORMATO I2)

10

AHORA DAME LOS ELEMENTOS DEL VECTOR IR..(FORMATO I2)

DAME EL ELEMENTO 1

28

DAME EL ELEMENTO 2

123 (123)

34

DAME EL ELEMENTO 3

05

DAME EL ELEMENTO 4

65

DAME EL ELEMENTO 5

99

DAME EL ELEMENTO 6

76

DAME EL ELEMENTO 7

32

DAME EL ELEMENTO 8

43

DAME EL ELEMENTO 9

56

DAME EL ELEMENTO 10

12

LOS ELEMENTOS DESORDENADOS DEL VECTOR IR SON

IR(1) = 28

IR(2) = 34

IR(3) = 5

IR(4) = 65

IR(5) = 99

IR(6) = 76

IR(7) = 32

IR(8) = 43

IR(9) = 56

IR(10) = 12

LOS ELEMENTOS ORDENADOS DEL VECTOR IR SON :

IR(1) = 5

IR(2) = 12

IR(3) = 28

IR(4) = 32

IR(5) = 34

IR(6) = 43

IR(7) = 56

IR(8) = 65

IR(9) = 76

IR(10) = 99

FIN DE PROGRAMA

\$

```

0001      SUBROUTINE LEE(X,I)
0002      DIMENSION X (100,5)
0003      INTEGER I
0004      INTEGER J,K,NUM
0005      WRITE (6,1010)
0006      READ (5,1020) I
0007      DO J=1,I
0008          WRITE (6,1030) J
0009          READ (5,1050) NUM,(X(J,K),K=2,5)
0010          X(J,1)=FLOAT(NUM)
0011      END DO
0012      RETURN
0013      1010  FORMAT (//, ' DAME EL NUMERO DE ELEMENTOS DE LA LISTA ',/)
0014      1020  FORMAT (I2)
0015      1030  FORMAT (/, ' DAME EL ELEMENTO ',I2)
0016      1050  FORMAT (I2,4A4)
0017      END
    
```

PROGRAM SECTIONS

NAME	BYTES	ATTRIBUTES								
0 SCODE	230	PIC	COM	REL	LCL	SHR	EXE	RD	DDRT	LONG
1 SCDATA	91	PIC	COM	REL	LCL	SHR	NOEXE	RD	DDRT	LONG
2 SLOCAL	32	PIC	COM	REL	LCL	NOshr	NOEXE	RD	DDRT	LONG
TOTAL SPACE ALLOCATED		343								

ENTRY POINTS

ADDRESS	TYPE	NAME
0-00000000		LEE

VARIABLES

ADDRESS	TYPE	NAME	ADDRESS	TYPE	NAME	ADDRESS	TYPE	NAME
AP-000000088	I*4	I	2-00000000	I*4	J	2-00000004	I*4	K

ARRAYS

ADDRESS	TYPE	NAME	BYTES	DIMENSIONS
AP-000000048	R*4	X	2000	(100, 5)

124
124

LEE

27-SEP-1983 08:54:49
27-SEP-1983 08:54:42

VAD .11 FORTIA
DIS:SCFCAP11:10

LABELS

ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS	LABEL
1-00000000	1010'	1-0000002F	1020'	1-00000032	1030'	1-0000004A	1050'

125

27-SEP-1983 08:54:49
 27-SEP-1983 08:54:42

VAX 1 PORTHA
 DISKSCHEM11:1

```

0001      SUBROUTINE IMP(X,I)
0002      DIMENSION X(100,5)
0003      INTEGER I
0004      INTEGER NUM
0005      WRITE (6,1010)
0006      DO J=1,I
0007          NUM=IFIX(X(J,1))
0008          WRITE (6,1020) NUM,(X(J,K),K=2,5)
0009      END DO
0010      RETURN
0011      1010  FORMAT (/, ' LOS ELEMENTOS SON :',//)
0012      1020  FORMAT (10X,13,2X,4A4,/)
0013      END
  
```

PROGRAM SECTIONS

NAME	BYTES	ATTRIBUTES								
0 SCODE	166	PIC	CON	REL	LCL	SHR	EXE	RD	NOART	LONG
1 SPDATA	38	PIC	CON	REL	LCL	SHR	NOEXE	RD	NOART	LONG
2 SLOCAL	32	PIC	CON	REL	LCL	NOshr	NOEXE	RD	ART	LONG
TOTAL SPACE ALLOCATED	236									

ENTRY POINTS

ADDRESS	TYPE	NAME
0-00000000		IMP

VARIABLES

ADDRESS	TYPE	NAME	ADDRESS	TYPE	NAME	ADDRESS	TYPE	NAME
AP-00000008	I*4	I	2-00000004	I*4	J	2-00000008	I*4	K

ARRAYS

ADDRESS	TYPE	NAME	BYTES	DIMENSIONS
AP-00000004	P*4	X	2000	(100, 5)

(126)
 126

IMP. (

27-SEP-1983 08:54:49
27-SEP-1983 08:54:42

VAX 11 PDP11A:
DISK3CPCAM11:1

LABELS

ADDRESS	LABEL	ADDRESS	LABEL
1-00000000	1010'	1-0000001A	1020'

(127)

127

```
0001 SUBROUTINE MERGE(A,H,C,N,M,L)
0002 DIMENSION A(100,5),B(100,5),C(100,5)
0003 INTEGER N,M,L
0004 INTEGER INFIN
0005 INFIN=99999999
0006 A(N+1,1)=INFIN
0007 B(M+1,1)=INFIN
0008 I=1
0009 L=1
0010 J=1
0011 K=1
0012 DO WHILE ( (A(I,1) .NE. INFIN) .OR. (B(J,1) .NE. INFIN) )
0013 DO WHILE (A(I,1) .LT. B(J,1))
0014 CALL COPIA(C,L,A,I)
0015 L=L+1
0016 I=I+1
0017 END DO
0018 DO WHILE ((A(I,1) .EQ. B(J,1)) .AND. (A(I,1) .NE. INFIN))
0019 CALL COPIA(C,L,B,J)
0020 L=L+1
0021 J=J+1
0022 I=I+1
0023 END DO
0024 DO WHILE (A(I,1) .GT. B(J,1))
0025 CALL COPIA(C,L,B,J)
0026 L=L+1
0027 J=J+1
0028 END DO
0029 END DO
0030 RETURN
0031 END
```

(128)

MERGE

27-SEP-1983 08:54:49
27-SEP-1983 08:54:42

VOL 11 P000001
DISKSC00A111:1

PROGRAM SECTIONS

NAME	BYTES	ATTRIBUTES
0 SCODE	398	PIC CON REL LCL SHR EXE RD MWRD LONG
2 SLOCAL	116	PIC CON REL LCL MUSR NORXE RD MWRD LONG
TOTAL SPACE ALLUCATED	514	

ENTRY POINTS

ADDRESS	TYPE	NAME
0-00000000		MERGE

VARIABLES

ADDRESS	TYPE	NAME	ADDRESS	TYPE	NAME	ADDRESS	TYPE	NAME
2-00000004	I*4	I	2-00000000	I*4	INFIN	2-00000008	I*4	J
AP-00000018	I*4	L	AP-00000014	I*4	M	AP-00000010	I*4	B

ARRAYS

ADDRESS	TYPE	NAME	BYTES	DIMENSIONS
AP-00000004	R*4	A	2000	(100, 5)
AP-00000008	R*4	B	2000	(100, 5)
AP-0000000C	R*4	C	2000	(100, 5)

FUNCTIONS AND SUBROUTINES REFERENCED

TYPE	NAME
	COPIA

(129)

```

0001 SUBROUTINE COPIA (X,J,Y,J)
0002 DIMENSION X(100,5),Y(100,5)
0003 INTEGER I,J
0004     INTEGER K
0005     DO K=1,5
0006         X(I,K)=Y(J,K)
0007     END DO
0008     RETURN
0009 END
  
```

PROGRAM SECTIONS

NAME	BYTES	ATTRIBUTES
0 SCODE	110	PIC CON REL LCL SHR EXE PD SORT LONG
2 SLOCAL	44	PIC CON REL LCL BUDEN BODEXE PD YRT LONG
TOTAL SPACE ALLOCATED	154	

ENTRY POINTS

ADDRESS	TYPE	NAME
0-00000000		COPIA

VARIABLES

ADDRESS	TYPE	NAME	ADDRESS	TYPE	NAME	ADDRESS	TYPE	NAME
AP-00000008	I*4	I	AP-00000010	J*4	J	2-00000000	I*4	X

ARRAYS

ADDRESS	TYPE	NAME	BYTES	DIMENSIONS
AP-00000004	R*4	X	2000	(100, 5)
AP-0000000C	R*4	Y	2000	(100, 5)

(130)

27-SEP-1983 08:54:49
 27-SEP-1983 08:54:42

V1 11 FORT...
 DISKSC0411:

```

0001 DIMENSION A (100,5),B (100,5),C (100,5)
0002 INTEGER N,M,L
0003 CALL LEP (A,E)
0004 CALL LEP (A,B)
0005 CALL LEP (B,B)
0006 CALL LEP (B,B)
0007 CALL MERGE (A,B,C,N,M,L)
0008 CALL LEP (C,L-1)
0009 CALL EXIT
0010 END
  
```

PROGRAM SECTIONS

NAME	BYTES	ATTRIBUTES
0 SCODE	80	PIC CON REL LCL SPR EXE RD NOSET LONG
2 SLOCAL	6076	PIC CON REL LCL NOSPR NOEAE RD NOSET LONG
TOTAL SPACE ALLOCATED	6156	

ENTRY POINTS

ADDRESS	TYPE	NAME
0-00000000		MERGESMAIN

VARIABLES

ADDRESS	TYPE	NAME	ADDRESS	TYPE	NAME	ADDRESS	TYPE	NAME
2-00001778	I*4	L	2-00001774	I*4	E	2-00001770	I*4	B

ARRAYS

ADDRESS	TYPE	NAME	BYTES	DIMENSIONS
2-00000000	R*4	A	2000	(100, 5)
2-00000700	R*4	B	2000	(100, 5)
2-00000FA0	R*4	C	2000	(100, 5)

FUNCTIONS AND SUBROUTINES REFERENCED

TYPE	NAME	TYPE	NAME	TYPE	NAME	TYPE	NAME
	FORSEFIT		IMP		LEP		MERGE

131

131

MERGE AIR

27-SEP-1983 08:54:49
27-SEP-1983 08:54:42

DISK SPACE: 11:11

COMMAND QUALIFIERS

FORTRAN /LIST/CHECK MERGE

/CHECK=(ROUNDS,OVERFLOW,UNDERFLOW)

/DEBUG=(NOSYMBOLS,TRACEBACK)

/STANDARD=(FOSYNTAX,NOSOURCE_FORM)

/SHOW=(NOPREPROCESSOR,NOINCLUDE,MAP)

/F77 /NOG_FLOATING /I4 /OPTIMIZE /WARNINGS /NO_LINES /NO_CROSS_REFERENCE /NO_MACHINE_CODE /CO

COMPILATION STATISTICS

PUR TIME: 3.47 SECONDS
ELAPSED TIME: 6.27 SECONDS
PAGE FAULTS: 1582
DYNAMIC MEMORY: 121 PAGES

DAME EL NUMERO DE ELEMENTOS DE LA LISTA

10

133

DAME EL ELEMENTO 1
01 ALEJANDRO JIMENEZ

DAME EL ELEMENTO 2
03 PALOMA ESTRADA

DAME EL ELEMENTO 3
05 ANTONIO PEREZ

DAME EL ELEMENTO 4
07 RICARDO CIRIA

DAME EL ELEMENTO 5
10 CARLOS RAMOS

DAME EL ELEMENTO 6
15 HERIBERTO OLGIN

DAME EL ELEMENTO 7
18 BEATRIZ ENRIQUEZ

DAME EL ELEMENTO 8
26 ROCKY

DAME EL ELEMENTO 9
32 RICHARD NIXON

DAME EL ELEMENTO 10
33 SOCRATES MUNIZ

LOS ELEMENTOS SON :

- 1 ALEJANDRO JIMEN
- 3 PALOMA ESTRADA
- 5 ANTONIO PEREZ
- 7 RICARDO CIRIA
- 10 CARLOS RAMOS
- 15 HERIBERTO OLGIN
- 18 BEATRIZ ENRIQUE
- 26 ROCKY
- 32 RICHARD NIXON
- 33 SOCRATES MUNIZ

DAME EL ELEMENTO 1
01 A. JIMENEZ

134

(134)

DAME EL ELEMENTO 2
04 PABLO NERUDA

DAME EL ELEMENTO 3
15 H. OLGUIN

DAME EL ELEMENTO 4
18 B. ENRIQUEZ

DAME EL ELEMENTO 5
30 SILVIA LARRAZA

DAME EL ELEMENTO 6
34 JUAN PEREZ

DAME EL ELEMENTO 7
45 GISELA GTZ.

LOS ELEMENTOS SON :

1	A. JIMENEZ
4	PABLO NERUDA
15	H. OLGUIN
18	B. ENRIQUEZ
30	SILVIA LARRAZA
34	JUAN PEREZ
45	GISELA GTZ.

LOS ELEMENTOS SON :

1	A. JIMENEZ
3	PALOMA ESTRADA
4	PABLO NERUDA
5	ANTONIO PEREZ
7	RICARDO CIRIA
10	CARLOS RAMOS
15	H. OLGUIN
18	B. ENRIQUEZ
26	ROCKY
30	SILVIA LARRAZA

33 SOCRATES MUNIZ

34 JUAN PEREZ

45 GISELA GTZ.

(135)

135

(136)

CAPITULO VI

EJEMPLO PARALELO

ING. ANTONIO PEREZ A.
M. EN C. RICARDO CIRIA M.

CAPITULO VI

EJEMPLO PARALELO

- 1.- Promedio de Calificación (expresión aritmética)
- 2.- Promedio de Calificación
con subíndices. (Arreglos)
- 3.- Nombres de los Alumnos. (Formatos)
- 4.- Uso de Iteración. (Proposición DO)
- 5.- Calificación Alfabética (Proposición IF lógica)
- 6.- Desviación Standard y
Varianza (Funciones pre-definidas)
- 7.- Ordenamiento Alfabético)Sub-programas)



**DIVISION DE EDUCACION CONTINUA
FACULTAD DE INGENIERIA U.N.A.M.**

LENGUAJE DE PROGRAMACION FORTRAN

FUNCIONES DEFINIDAS POR EL USUARIO

OCTUBRE, 1984

12.2.-

FUNCIONES DEFINIDAS POR EL USUARIO

①

El lenguaje FORTRAN proporciona una manera sencilla por medio de la cual un programador puede definir la función que él desee por medio de lo que se llama proposición de definición de función, siempre y cuando el cálculo que realice la función - pueda escribirse como una expresión aritmética.

Para escribir una proposición de definición de función, deberá escribirse, primero, la forma en que el programador utilizará la función poniendo como argumentos el nombre de cualquier variable, después el signo igual (=), y después la expresión aritmética que el programador desee que ejecute la función cada vez que sea llamada, colocando la variable del argumento en el lugar adecuado dentro de la expresión aritmética.

Por ejemplo, suponga que desea definir una función que obtenga la tangente de un ángulo dividiendo el seno entre el coseno:

$$\text{TAN (X) = SIN (X) / COS (X)}$$

Una vez definida la función, el programador podrá utilizarla como si fuera una función proporcionada por el compilador. P.E.

```
VAR = TAN (ANG)
```

```
XYZ = SQRT (TAN(A1/A2) ** 2 - 1)
```

Para definir correctamente una función, hay que tomar en cuenta lo siguiente:

- La proposición de definición de función deberá colocarse después de las declaraciones de variables, si hay, y antes de la primera proposición ejecutable del programa.
- Todos los argumentos (nombres de variables) que se utilicen a la izquierda del signo igual deberán utilizarse en la expresión aritmética que va a la derecha del signo igual. Sin embargo, sí se permite que en la expresión aritmética se utilice una variable que no sea argumento, en cuyo caso se referirá a la

misma variable que se utilice en el programa. (2)



**DIVISION DE EDUCACION CONTINUA
FACULTAD DE INGENIERIA U.N.A.M.**

LENGUAJE DE PROGRAMACION FORTRAN

PROGRAMA PARA AJUSTE DE CURVAS POR MINIMOS CUADRADOS

ING. CARLOS A. RAMOS LARIOS

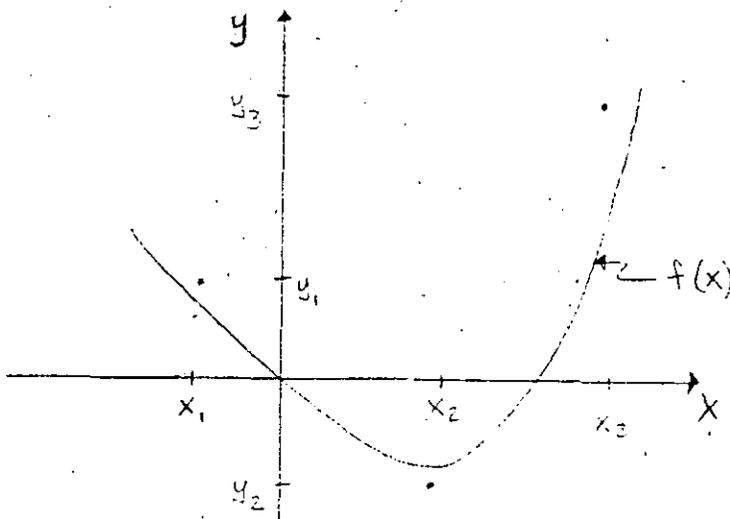
OCTUBRE, 1984.

PROGRAMA PARA AJUSTE DE CURVAS POR MINIMOS CUADRADOS

CARLOS A. RAMOS LARIOS 1983

USO:

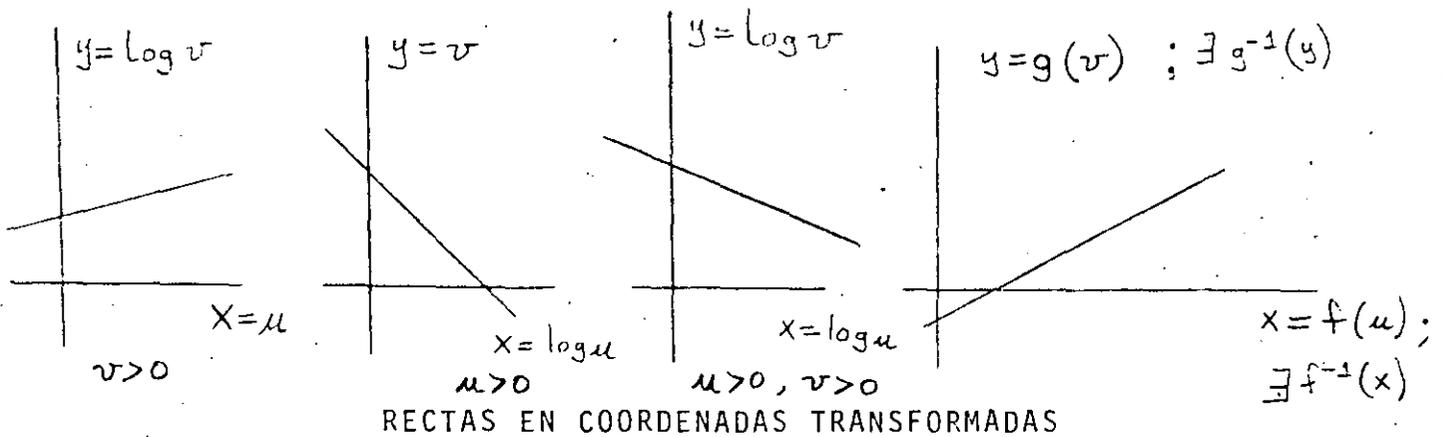
- Obtener expresiones analíticas que representen un fenómeno definido por coordenadas aisladas aplicando el criterio de los mínimos cuadrados.
- Nos limitaremos a fenómenos con una variable independiente y una variable dependiente. (X,Y)
- Las expresiones analíticas por ajustar podran ser: 1) Polinomios enteros de cualquier grado (rectas, parábolas, etc.) en las coordenadas originales, 2) Rectas en las coordenadas (X, LOGY), (LOGX, Y) (LOGX, LOGY), pudiendo utilizarse cualquier base de logaritmos, 3) Rectas con transformaciones biunívocas definidas por el usuario.



$(X_i, Y_i) =$ Coordenadas aisladas

$f(X) =$ Curva ajustada por mínimos cuadrados

$x, y =$ Coordenadas originales



Con las expresiones analíticas obtenidas podrán evaluarse puntos fuera de los observados (Interpolación y Extrapolación)

PLANTEAMIENTO:

Sean: m puntos con abscisa diferente (datos)

n el grado máximo del polinomio que desea ajustarse (dato)

a_n, a_{n-1}, \dots, a_0 los coeficientes del polinomio (incógnitas)

$f(x) = a_n x^n + a_{n-1} x^{n-1} + \dots$ el polinomio de mejor ajuste

$E = \sum_{i=1}^m (f(x_i) - y_i)^2 w_i$ el error cuadrático a minimizar

w_i los factores de ponderación o de repetición de cada observación (dato)

Para resolver el problema, (obtener a_n, a_{n-1}, \dots, a_0), se aplican las condiciones necesarias para la existencia de un mínimo para el error

E:

$$\frac{\partial E}{\partial a_n} = 0 ; \quad \frac{\partial E}{\partial a_{n-1}} = 0 ; \quad \frac{\partial E}{\partial a_0} = 0 \quad (n+1 \text{ condiciones})$$

Para cualquiera de estas condiciones (la jésima):

$$\begin{aligned}
 \frac{\partial E}{\partial a_j} &= \sum_{i=1}^m 2 (f(x_i) - y_i) w_i \left(\frac{\partial f(x_i)}{\partial a_j} \right) \\
 &= 2 \sum_{i=1}^m (f(x_i) - y_i) w_i (x_i^j) \\
 &= 2 \sum (a_n x_i^n + a_{n-1} x_i^{n-1} \dots + a_0 - y_i) w_i (x_i^j) \quad \left(\sum = \sum_{i=1}^m \right) \\
 &= 2 \left(\sum a_n x_i^n w_i x_i^j + \sum a_{n-1} x_i^{n-1} w_i x_i^j + \dots - \sum y_i w_i x_i^j \right) \\
 &= 2 \left(a_n \sum w_i x_i^{n+j} + a_{n-1} \sum w_i x_i^{n-1+j} + \dots + a_0 \sum w_i x_i^j - \sum y_i w_i x_i^j \right)
 \end{aligned}$$

Igualando esta condición a cero, se obtiene una ecuación lineal en las incógnitas a_n, a_{n-1}, \dots, a_0 :

$$a_n \sum w_i x_i^{n+j} + a_{n-1} \sum w_i x_i^{n-1+j} + \dots + a_0 \sum w_i x_i^j = \sum y_i w_i x_i^j$$

La primera condición ($\frac{\partial E}{\partial a_n} = 0$) establece que ($j = n$):

$$a_n \sum w_i x_i^{n+n} + a_{n-1} \sum w_i x_i^{n+n-1} + \dots + a_0 \sum w_i x_i^n = \sum y_i w_i x_i^n$$

La segunda condición ($\frac{\partial E}{\partial a_{n-1}} = 0$) establece que ($j = n-1$):

$$a_n \sum w_i x_i^{n+n-1} + a_{n-1} \sum w_i x_i^{n+n-2} + \dots + a_0 \sum w_i x_i^{n-1} = \sum y_i w_i x_i^{n-1}$$

Al aplicar en la misma forma todas las condiciones necesarias, se puede escribir el siguiente sistema de $n+1$ ecuaciones con $n+1$ incógnitas:

$$\begin{bmatrix}
 \sum w_i x_i^{n+n} & \sum w_i x_i^{n+n-1} & \dots & \sum w_i x_i^{n+1} & \sum w_i x_i^n \\
 \sum w_i x_i^{n+n-1} & \sum w_i x_i^{n+n-2} & & \sum w_i x_i^n & \sum w_i x_i^{n-1} \\
 \cdot & & & \cdot & \cdot \\
 \cdot & & & \cdot & \cdot \\
 \cdot & & & \cdot & \cdot \\
 \sum w_i x_i^{n+1} & \sum w_i x_i^n & & \sum w_i x_i^2 & \sum w_i x_i \\
 \sum w_i x_i^n & \sum w_i x_i^{n-1} & \dots & \sum w_i x_i & \sum w_i
 \end{bmatrix}
 \begin{bmatrix}
 a_n \\
 a_{n-1} \\
 \cdot \\
 \cdot \\
 \cdot \\
 a_3 \\
 a_0
 \end{bmatrix}
 =
 \begin{bmatrix}
 \sum y_i w_i x_i^n \\
 \sum y_i w_i x_i^{n-1} \\
 \cdot \\
 \cdot \\
 \cdot \\
 \sum y_i w_i x_i \\
 \sum y_i w_i
 \end{bmatrix}$$

De la solución del sistema anterior se obtienen los $n+1$ coeficientes a_n, a_{n-1}, \dots, a_0 :

COORDENADAS TRANSFORMADAS:

Para el caso en que se haya hecho un ajuste a una recta, ($n \neq 1$); la expresión tiene la forma:

$$y = f(x) = a_1 x + a_0$$

Y si además los valores de (x_i, y_i) que se utilizaron para plantear el sistema de ecuaciones provienen de una transformación logarítmica sencilla o doble, es posible obtener la forma de la expresión en las coordenadas originales u y v .

CASO I:

$$y = \log_b v \quad ; \quad x = u$$

de donde

$$\log_b v = a_1 u + a_0$$

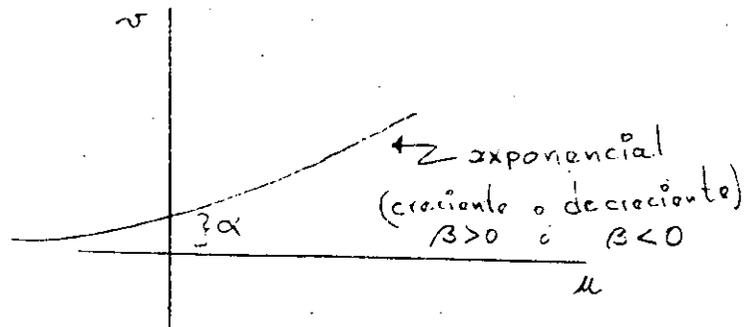
(b =base de logaritmos que se elija, $b > 0$)

$$v = b^{a_1 u + a_0}$$

$$= b^{a_1 u} \cdot b^{a_0}$$

$$= b^{a_0} \cdot b^{a_1 u}$$

$$v = \alpha \cdot b^{\beta u}$$

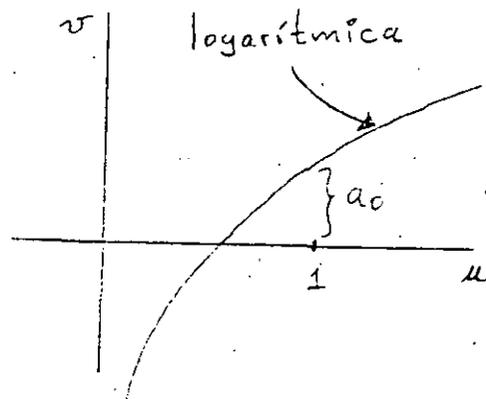


CASO II:

$$y = v ; \quad x = \log_b u$$

de donde

$$v = a_1 \log_b u + a_0$$



CASO III:

$$y = \log_b v ; \quad x = \log_b u$$

de donde

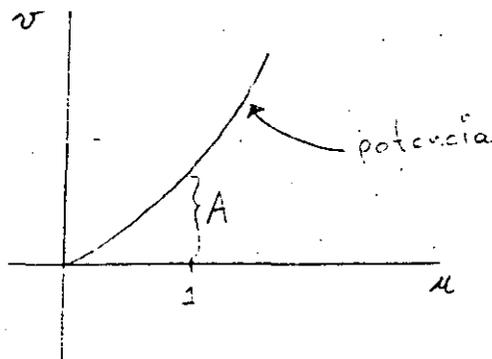
$$\log_b v = a_1 \log_b u + a_0$$

$$v = b^{a_1 \log_b u + a_0}$$

$$= b^{a_0} b^{a_1 \log_b u}$$

$$= b^{a_0} b^{\log_b u^{a_1}} = b^{a_0} u^{a_1}$$

$$v = A u^B$$



Notese que B no tiene que ser necesariamente entero.

Considerando por el momento que el ajuste deseado siempre es una recta (en coordenadas originales ó transformadas), n valdrá 1 y el sistema de ecuaciones a resolver para encontrar a_1 y a_0 siempre será:

$$\begin{bmatrix} \sum w_i x_i^2 & \sum w_i x_i \\ \sum w_i x_i & \sum w_i \end{bmatrix} \begin{bmatrix} a_1 \\ a_0 \end{bmatrix} = \begin{bmatrix} \sum y_i w_i x_i \\ \sum y_i w_i \end{bmatrix}$$

La solución de este sistema por determinarse es:

$$\begin{aligned} a_1 &= \frac{(\sum y_i w_i x_i)(\sum w_i) - (\sum y_i w_i)(\sum w_i x_i)}{(\sum w_i x_i^2)(\sum w_i) - (\sum w_i x_i)^2} \\ &= \frac{(\sum y_i w_i x_i)(\sum w_i) - (\sum y_i w_i)(\sum w_i x_i)}{(\sum w_i x_i^2)(\sum w_i) - [\sum w_i x_i]^2} \\ a_0 &= \frac{(\sum w_i x_i^2)(\sum y_i w_i) - (\sum w_i x_i)(\sum y_i w_i x_i)}{(\sum w_i x_i^2)(\sum w_i) - [\sum w_i x_i]^2} \end{aligned}$$

Una vez definido el problema matemático completamente, se procederá a efectuar un Análisis que conduzca a escribir un programa de computadora que resuelve el problema planteado. A continuación se mencionan algunos pasos que la experiencia demuestra que conviene seguir.

PASO 1.- Definir los límites superiores del tamaño del problema a manejar, normalmente definidos por la capacidad en -

memoria, o en disco, o el tiempo máximo de proceso ocupado, o el volumen de papel a imprimir etc. (en general recursos físicos).

Para este problema, el tamaño queda definido por el número de observaciones o considerar, y por el número de abscisas y ordenadas para evaluar la recta una vez calculada.

Para el ejemplo sea:

$$\max(M) = 20$$

$$\max(NX) = 20$$

$$\max(NY) = 20$$

PASO 2.- Definir claramente (preferentemente por escrito), los resultados (salidas) que son estrictamente necesarios para considerar que el problema está resuelto. En este caso es tos serían:

- A) Un encabezado que haga referencia al problema que se está resolviendo y al programa que se está utilizando incluyendo la versión. Ejemplo. Ajuste a una recta - Por mínimos cuadrados, Programa MINIMOS.FOR versión 1.0
- B) Un eco de los datos que fueron leídos, inmediatamente después de haber sido introducidos al programa.

En este caso por ejemplo:

Número de observaciones a considerar =M=

Tabla de observaciones leídas

PUNTO	X	Y	W
$\frac{1}{2}$			
\vdots			
M			

Número de abscisas para calcular ordenadas =NX=

Tabla de abscisas para calcular ordenadas.

I	XC
1	
2	
3	
\vdots	
NX	

Número de ordenadas para calcular abscisas =NY=

I	YC
1	
NY	

C) Los resultados propiamente dichos del problema en este caso:

Los coeficientes a_0 y a_1 ,

y los valores de Y dada XC

los valores de X dada YC

Nótese que en el paso 2B quedan definidos las entradas ó datos necesarios para el problema.

Se insiste en que en los pasos 2B y 2C se limite a los resultados y datos mínimos, y que una vez que el programa funcione, - se mejore con nuevas versiones ampliándolo por ejemplo con:

Una tabla de observaciones ordenada por abscisa de menor a mayor.

Una tabla de observaciones ordenada por ordenada de menor a mayor.

Leyendo posibilidades de transformaciones en una o dos variables calculando el error E.

Calculando la ordenada al origen y la abscisa al origen etc.

PASO 3.- Definir los posibles errores que pueden ocurrir dentro del programa y la acción que debe tomarse.

A) Errores de incongruencia

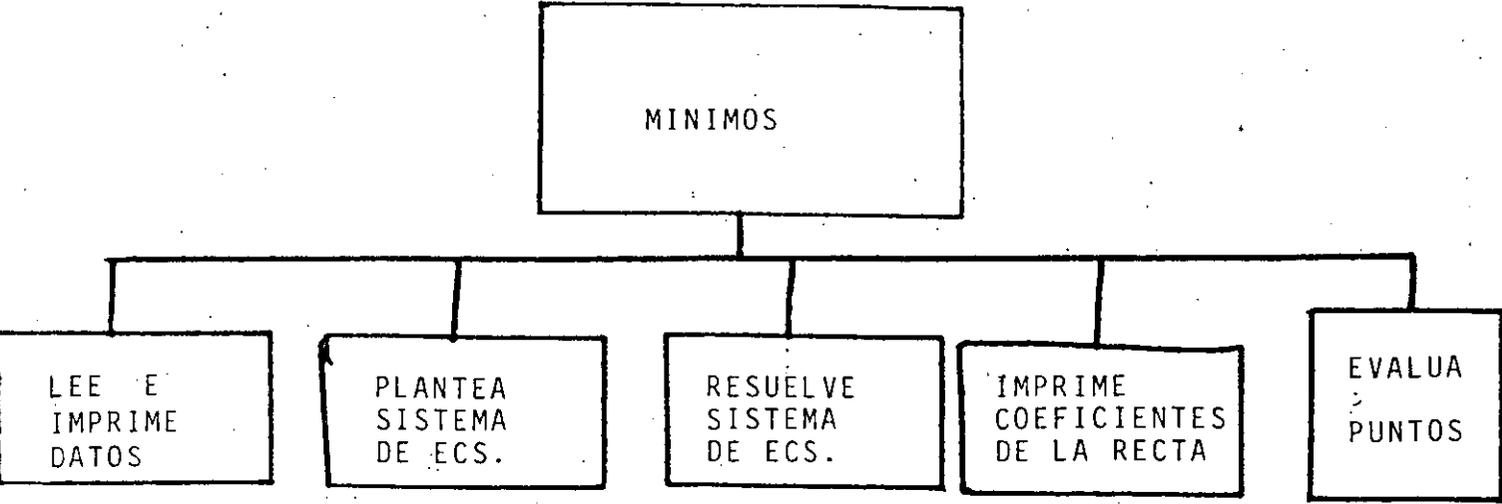
DESCRIPCION	ACCION
Número $M < 2$ o $M > 20$	Enviar mensaje y volverlo a leer
Números NX y/o NY > 5 a 20	Enviar mensaje y volverlo a leer
Determinante del sistema = 0	Enviar mensaje y terminar el programa

Este punto también puede crecer mucho, sin embargo se recomienda iniciar con algo breve y crecer con versiones Subsecuentes. Por ejemplo, puede aumentarse la opción de que al terminar de leer toda la tabla de observaciones, se imprima ésta y se pregunte si se desea hacer algún cambio, o bien puede hacerse que después de imprimir todos los resultados de un problema puedan introducirse más abscisas u ordenadas para evaluar, o que sea posible cambiar solo los valores de los pesos etc.

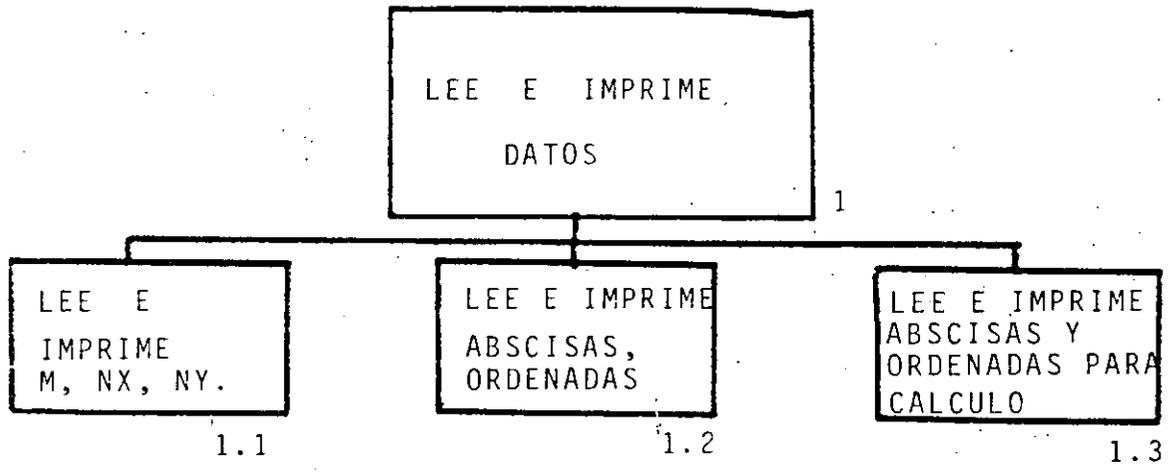
Una vez completados los tres pasos anteriores, o sea lo que llamamos al Análisis para la programación, podemos continuar con lo que se conoce como el Diseño de la Programación. En los pasos de análisis resolvimos el ¿que se va a hacer? En los pasos de diseño se resuelve el ¿como se va a hacer?.

PASO 4.- Separar el problema completo en pocos módulos operativos con máxima "cohesión" y mínimo "acoplamiento". Volver a separar cada uno de los módulos anteriores con el mismo criterio y así sucesivamente hasta obtener módulos "elementales". Este método es la idea central de la técnica conocida con el nombre de "Diseño estructurado de Sistemas "(1).

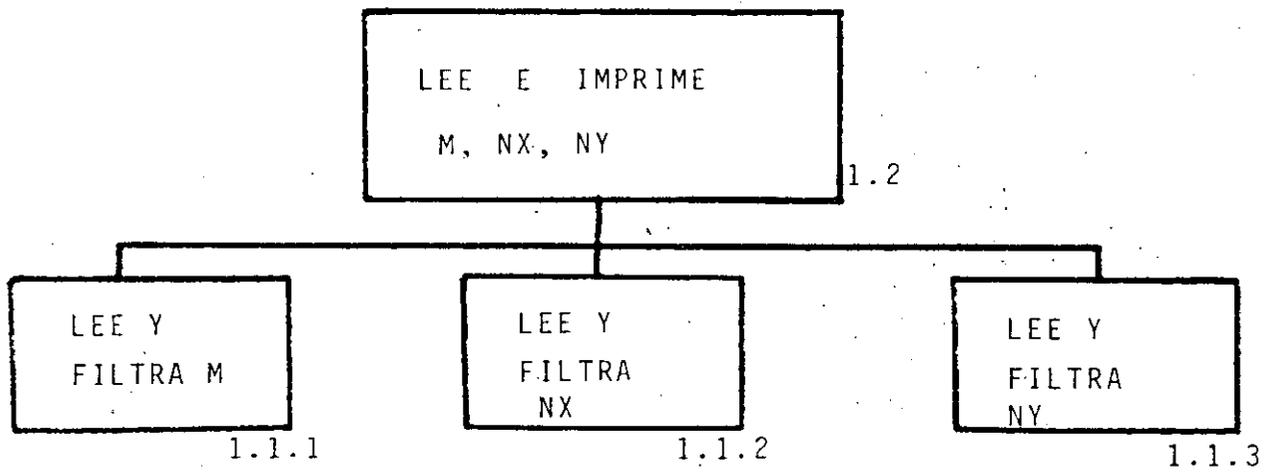
(1) "Structured Design": Fundamentals of a Discipline of Computer Program and Systems Design. Edward Yourdon/Larry L. Constantine, Prentice Hall, 1978.



La segunda separación ó refinamiento podría ser:



Esta misma separación puede aplicarse una vez más a los módulos para obtener niveles inferiores por ejemplo:



Hasta llegar a estos módulos que pueden considerarse elementales

Lo mismo puede hacerse con los módulos restantes hasta completar la llamada "carta de la estructura del programa"

- PASO 5.- Escribir en lenguaje Fortran (codificar en Fortran) los diferentes módulos obtenidos en el paso anterior iniciando con los módulos superiores.
- PASO 6.- Introducir los módulos codificados a la computadora y probarlos.
- PASO 7.- Escribir en Fortran los módulos de menor nivel
- PASO 8.- Repetir pasos 6 y 7 hasta terminar con todos los módulos
- PASO 9.- Agrupar todos los documentos utilizados durante el desarrollo del programa, formando así la documentación técnica.
- PASO 10.- Preparar la documentación de operación o el "Instructivo de usuario", con las notas más relevantes de la documentación técnica y algunos ejemplos y observaciones adicionales.

```

PROGRAMA MINIMOS.FOR
CARLOS A. RAMOS LARIOS
OCTUBRE 1988
DIMENSION X(20),Y(20),W(20),XC(20),YC(20)
PRINT 2010
  LEE E IMPRIME DATOS(1)
  LEE M,NX,NY (1.1)
  LEE Y FILTRA M(1.1.1)
10 PRINT 2020
  READ 1000,M
  IF(2-M)15,15,10
15 IF(20-M)10,10,20
20 PRINT 2030,M
  LEE Y FILTRA NX(1.1.2)
25 PRINT 2040
  READ 1000,NX
  IF(20-NX)25,30,30
30 PRINT 2050,NX
  LEE Y FILTRA NY(1.1.3)
35 PRINT 2060
  READ 1000,NY
  IF(20-NY)35,40,40
40 PRINT 2070,NY
  LEE E IMPRIME ABSCISAS,ORDENADAS Y PESOS (1.2)
  DO 50 I=1,M
  PRINT 2080,I
  READ 1010,X(I),Y(I),W(I)
50 CONTINUE
  PRINT 2090
  DO 60 I=1,M
  PRINT 2100,I,X(I),Y(I),W(I)
60 CONTINUE
  LEE E IMPRIME ABSCISAS Y ORDENADAS PARA CALCULO (1.3)
  IF(NX)100,100,70
70 DO 80 I=1,NX
  PRINT 2110,I
  READ 1020,XC(I)
80 CONTINUE
  PRINT 2120
  DO 90 I=1,NX
  PRINT 2130,I,XC(I)
90 CONTINUE
  IF(NY)140,140,110
110 DO 120 I=1,NY
  PRINT 2140,I
  READ 1020,YC(I)
120 CONTINUE
  PRINT 2150
  DO 130 I=1,NY
  PRINT 2160,I,YC(I)
130 CONTINUE

```

```

C      PLANTEA SISTEMA DE ECUACIONES(2)
140 SW=0
    SWX=0
    SWX2=0
    SWY=0
    SWXY=0
    DO 150 I=1,M
    SW=SW+W(I)
    FAC=W(I)*X(I)
    SWX=SWX+FAC
    SWX2=SWX2+FAC*X(I)
    FAC=W(I)*Y(I)
    SWY=SWY+FAC
    SWXY=SWXY+FAC*X(I)
150 CONTINUE
C      RESUELVE SISTEMA DE ECUACIONES (3)
    DET=SWX2*SW-SWX*SWY
    IF(DET)170,160,170
160 PRINT 2170
    GO TO 250
C      IMPRIME COEFICIENTES DE LA RECTA(4)
170 A1=(SWXY*SW-SWX*SWY)/DET
    A0=(SWX2*SWY-SWX*SWXY)/DET
    PRINT 2180,A1,A0
C      EVALUA PUNTOS(5)
    IF(NX)200,200,180
180 PRINT 2190
    DO 190 I=1,NX
    YY=A1*XC(I)+A0
    PRINT 2200,I,XC(I),YY
190 CONTINUE
200 IF(NY)250,250,210
210 IF(A1)230,220,230
220 PRINT 2210,A0
    GO TO 250
230 PRINT 2220
    DO 240 I=1,NY
    XX=(YC(I)-A0)/A1
    PRINT 2200,I,YC(I),XX
240 CONTINUE
250 CALL EXIT

```

```

FORMATOS
1000 FORMAT(I2)
1010 FORMAT(3F10.0)
1020 FORMAT(F10.0)
2010 FORMAT(1X, 'AJUSTE A UNA RECTA POR MINIMOS CUADRADOS', //,
11X, 'PROGRAMA MINIMOS.FOR VERSION 1.0 03/10/83', //)
2020 FORMAT(1X, 'DAME EL NUMERO DE OBSERVACIONES(MINIMO 2), EN I2')
2030 FORMAT(1X, 'NUMERO DE OBSERVACIONES A CONSIDERAR=M= ', I2)
2040 FORMAT(1X, 'DAME EL NUMERO DE ABSCISAS A LAS QUE SE ', //,
1 ' DESEA CALCULAR ORDENADAS, EN I2')
2050 FORMAT(1X, 'NUMERO DE ABSCISAS PARA CALCULAR ORDENADAS=NX= ', I2)
2060 FORMAT(1X, 'DAME EL NUMERO DE ORDENADAS A LAS QUE SE ', //,
1 ' DESEA CALCULAR ABSCISAS, EN I2')
2070 FORMAT(1X, 'NUMERO DE ORDENADAS PARA CALCULAR ABSCISAS=NY= ', I2)
2080 FORMAT(1X, 'DAME X(I), Y(I), W(I), EN 3F10.0, PARA I= ', I2, //,
11X, '123456789012345678901234567890')
2090 FORMAT(1X, 'TABLA DE OBSERVACIONES LEIDAS: ', //,
1 ' PUNTO ', ' X ', ' Y ', ' W ', //)
2100 FORMAT(1X, I5, 5X, 3F10.3)
2110 FORMAT(1X, 'DAME ABSCISA XC(I), EN F10.0, PARA I= ', I2, //,
11X, '1234567890')
2120 FORMAT(1X, 'TABLA DE ABSCISAS PARA CALCULAR ORDENADAS: ', //,
1 ' PUNTO ', ' XC')
2130 FORMAT(1X, I5, 5X, F10.3)
2140 FORMAT(1X, 'DAME ORDENADA YC(I), EN F10.0, PARA I= ', I2, //,
11X, '1234567890')
2150 FORMAT(1X, 'TABLA DE ORDENADAS PARA CALCULAR ABSCISAS: ', //,
1 ' PUNTO ', ' YC')
2170 FORMAT(1X, 'ERROR: EL DETERMINANTE ES CERO')
2180 FORMAT(//, 1X, 'SOLUCION DEL SISTEMA DE ECUACIONES', //,
11X, 'PENDIENTE= ', F10.3, //,
21X, 'ORDENADA AL ORIGEN= ', F10.3, //)
2190 FORMAT(1X, ' PUNTO ', ' ABSCISA ', ' ORDENADA CALCULADA', //)
2200 FORMAT(1X, I5, 5X, 2F10.3)
2210 FORMAT(1X, 'LA RECTA ES HORIZONTAL. ', //,
11X, 'POR LO QUE NO EXISTEN VALORES DE X SI Y ES DIFERENTE', //,
21X, 'DE: ', F10.3)
2220 FORMAT(1X, ' PUNTO ', ' ORDENADA ', ' ABSCISA CALCULADA', //)
      END

```

AJUSTE A UNA RECTA POR MINIMOS CUADRADOS

PROGRAMA MINIMOS.FOR VERSION 1.0 03/10/83

DAME EL NUMERO DE OBSERVACIONES(MINIMO 2),EN I2
03
#?

NUMERO DE OBSERVACIONES A CONSIDERAR=M= 3
DAME EL NUMERO DE ABCCISAS A LAS QUE SE
DESEA CALCULAR ORDENADAS,EN I2
02
NUMERO DE ABCCISAS PARA CALCULAR ORDENADAS=NX= 2
DAME EL NUMERO DE ORDENADAS A LAS QUE SE
DESEA CALCULAR ABCCISAS, EN I2
04
NUMERO DE ORDENADAS PARA CALCULAR ABCCISAS=NY= 4
DAME X(I),Y(I),W(I), EN 3F10.0,PARA I= 1
123456789012345678901234567890
1.0 3.0 1.0
DAME X(I),Y(I),W(I), EN 3F10.0,PARA I= 2
123456789012345678901234567890
2.0 5.0 1.0
DAME X(I),Y(I),W(I), EN 3F10.0,PARA I= 3
123456789012345678901234567890
3.0 7.0 1.0
TABLA DE OBSERVACIONES LEIDAS:
PUNTO X Y W

1	1.000	3.000	1.000
2	2.000	5.000	1.000
3	3.000	7.000	1.000

DAME ABCCISA XC(I), EN F10.0,PARA I= 1
1234567890
4.0
DAME ABCCISA XC(I), EN F10.0,PARA I= 2
1234567890
5.0

TABLA DE ABSCISAS PARA CALCULAR ORDENADAS:

PUNTO	XC
1	4.000
2	5.000

NOME ORDENADA YC(I), EN F10.0, PARA I= 1
1234567890

13.0

NOME ORDENADA YC(I), EN F10.0, PARA I= 2
1234567890

15.0

NOME ORDENADA YC(I), EN F10.0, PARA I= 3
1234567890

17.0

NOME ORDENADA YC(I), EN F10.0, PARA I= 4
1234567890

19.0

TABLA DE ORDENADAS PARA CALCULAR ABSCISAS:

PUNTO	YC
1	13.000
2	15.000
3	17.000
4	19.000

SOLUCION DEL SISTEMA DE ECUACIONES

PENDIENTE= 2.000

ORDENADA AL ORIGEN= 1.000

PUNTO	ABSCISA	ORDENADA CALCULADA
-------	---------	--------------------

1	4.000	9.000
---	-------	-------

2	5.000	11.000
---	-------	--------

PUNTO	ORDENADA	ABSCISA CALCULADA
-------	----------	-------------------

1	13.000	6.000
---	--------	-------

2	15.000	7.000
---	--------	-------

3	17.000	8.000
---	--------	-------

4	19.000	9.000
---	--------	-------



**DIVISION DE EDUCACION CONTINUA
FACULTAD DE INGENIERIA U.N.A.M.**

LENGUAJE DE PROGRAMACION FORTRAN

PROGRAMA PARA RESOLVER ECUACIONES SIMULTANEAS LINEALES
POR ELIMINACION

M. EN C. CARLOS A. RAMOS LARIOS

OCTUBRE, 1984.

PROGRAMA PARA RESOLVER ECUACIONES SIMULTANEAS LINEALES
POR ELIMINACION

CARLOS A. RAMOS LARIOS 1983.

USO:

Obtener los valores de las incógnitas que satisfacen simultáneamente a un sistema de ecuaciones lineales, aplicando el método de eliminación Gaussiana.

Dada una matriz de coeficientes de un sistema, podrán darse varios vectores de términos independientes para obtener una solución para cada uno de ellos.

PLANTEAMIENTO:

Sea el sistema de n ecuaciones simultaneas:

$$A x = B$$

Donde: A= Matriz de coeficientes del sistema, de n renglones por n columnas.

X= Vector de incógnitas, de n renglones por 1 columna

B= Matriz de términos independientes, de n renglones por m columnas

n= Número de ecuaciones = número de incógnitas

m= Número de columnas de términos independientes = número de veces que desea resolverse el sistema.

O en forma explícita:

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{bmatrix}$$

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

$$B = \begin{bmatrix} b_{11} & b_{12} & b_{13} & \dots & b_{1m} \\ b_{21} & b_{22} & b_{23} & \dots & b_{2m} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ b_{n1} & b_{n2} & b_{n3} & \dots & b_{nm} \end{bmatrix} \quad 2$$

Para resolver el sistema (obtener los valores de X para cada columna de B), se procederá a la eliminación Gaussiana, la cual se basa en 3 reglas básicas:

La solución de un sistema de ecuaciones simultaneas no cambia si:

- 1) Alguna de las ecuaciones del sistema se substituye por otra ecuación que se haya obtenido multiplicando por una constante $\neq 0$ la ecuación original.
- 2) Se intercambia una ecuación por otra.
- 3) Alguna de las ecuaciones del sistema se substituye por otra ecuación que se haya obtenido sumando a la ecuación original, otra ecuación del sistema previamente multiplicada por un constante $\neq 0$.

Las tres reglas básicas anteriores tienen los correspondientes efectos sobre el determinante de la matriz de coeficientes:

- 1) El determinante original queda multiplicado por la constante utilizada.
- 2) El determinante original queda cambiado de signo.
- 3) El determinante original no se afecta.

El método de la eliminación Gaussiana consiste en aplicar reiteradamente las tres reglas básicas, según sea necesario, con el objeto de transformar al sistema original en un sistema equivalente (o sea que tenga la misma solución que el sistema original), pero cuya forma

final permita obtener su solución en forma más sencilla. Una posibilidad es transformar al sistema de tal forma que su matriz de coeficientes se convierta en una matriz identidad. En esta forma final, la solución del sistema puede leerse directamente de la columna de términos independientes.

Ejemplo:

Aplicando las operaciones elementales sobre los renglones del sistema, transformar el siguiente sistema para obtener un sistema equivalente con matriz de coeficientes = I (matriz identidad)

$$\begin{bmatrix} 2 & -1 & 6 \\ 5 & 2 & 8 \\ -1 & -1 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 18 \\ 33 \\ 9 \end{bmatrix} \quad \text{ó bien} \quad \left[\begin{array}{ccc|c} 2 & -1 & 6 & 18 \\ 5 & 2 & 8 & 33 \\ -1 & -1 & 4 & 9 \end{array} \right] = \text{matriz aumentada}$$

Multiplicando el primer renglón por $\frac{1}{2} = 1R(\frac{1}{2})$

$$\left[\begin{array}{ccc|c} 1 & -1/2 & 3 & 9 \\ 5 & 2 & 8 & 33 \\ -1 & -1 & 4 & 9 \end{array} \right] \quad \leftarrow \text{se obtuvo un 1 en } a_{11}$$

Sustituyendo el segundo renglón por la suma de éste y el primero multiplicado por (-5) = $2R + 1R(-5)$

$$\left[\begin{array}{ccc|c} 1 & -1/2 & 3 & 9 \\ 0 & 9/2 & -7 & -12 \\ -1 & -1 & 4 & 9 \end{array} \right] \quad \leftarrow \text{se obtuvo un 0 en } a_{21}$$

Sustituyendo el tercer renglón por la suma de éste y el primero multiplicado por (+1) = $3R + 1R(+1)$

$$\left[\begin{array}{ccc|c} 1 & -1/2 & 3 & 9 \\ 0 & 9/2 & -7 & -12 \\ 0 & -3/2 & 7 & 18 \end{array} \right] \quad \leftarrow \text{se obtuvo un 0 en } a_{31}$$

Hasta aquí se ha completado la eliminación de la primera columna, para la segunda columna puede hacerse lo siguiente:

$$\begin{aligned} & 2R\left(\frac{2}{9}\right) \\ & 1R + 2R\left(\frac{1}{2}\right) \\ & 3R + 2R\left(\frac{3}{2}\right) \end{aligned}$$

De donde se obtiene:

$$\left[\begin{array}{ccc|c} 1 & 0 & 20/9 & 23/3 \\ 0 & 1 & -14/9 & -24/9 = -8/3 \\ 0 & 0 & 28/6 = 14/3 & 14 \end{array} \right]$$

Para eliminar la tercera columna, puede hacerse lo siguiente:

$$\begin{aligned} & 3R\left(\frac{3}{14}\right) \\ & 2R + 3R(14/9) \\ & 1R + 3R(-20/9) \end{aligned}$$

De donde se obtiene:

$$\left[\begin{array}{ccc|c} 1 & 0 & 0 & 9/9 = 1 \\ 0 & 1 & 0 & 6/3 = 2 \\ 0 & 0 & 1 & 3 \end{array} \right]$$

Este sistema puede escribirse como:

$$\begin{aligned} (1)x_1 + (0)x_2 + (0)x_3 &= 1 \\ (0)x_1 + (1)x_2 + (0)x_3 &= 2 \\ (0)x_1 + (0)x_2 + (1)x_3 &= 3 \end{aligned}$$

∴ El determinante del sistema original vale

$$\begin{aligned} \text{Det} &= \frac{1}{1/2} \cdot \frac{1}{2/9} \cdot \frac{1}{3/14} \cdot (1 \times 1 \times 1) \\ &= 42 \end{aligned}$$

De donde puede "leerse" directamente:

$$x_1 = 1; x_2 = 2; x_3 = 3$$

, que es la solución buscada

RESUMEN:

La eliminación utilizada consistió en las siguientes operaciones para cada columna (columna pivote):

- 1) Dividir el renglón que contiene el elemento de la diagonal principal entre dicho elemento, con lo que se logra un 1 en la diagonal principal.
- 2) Substituir los renglones restantes, por la suma de cada uno de ellos y el renglón que contiene la diagonal principal previamente multiplicado por el simétrico del elemento del renglón original que se encuentra en la columna pivote, con lo que se logran ceros en todos los renglones restantes en esa columna.

OBSERVACIONES:

1) De las tres reglas básicas solo se utilizaron dos. El intercambio entre renglones puede requerirse cuando algún elemento de la diagonal principal resulte cero, por lo que antes de dividir el renglón tendrá que intercambiarse éste por algún otro que no tenga un cero en esta columna. (Nótese que solo será posible intercambiarlo por alguno que se encuentre debajo de él, ya que si se intercambia por uno que se encuentre arriba, se afectarán las columnas eliminadas anteriormente que pretenden formar la matriz identidad) Si todos los renglones que se encuentran debajo contienen ceros en la columna pivote, implica que la matriz es singular (Determinante=0), y no es posible obtener una solución única.

La solución en este caso puede no existir o bien pueden existir una familia de soluciones. (Paralelismo y colinealidad respectivamente).

2) Cuando se están substituyendo los renglones restantes de la columna para convertirlos en ceros, la substitución es necesaria sola-

mente cuando en el elemento deseado de la columna no existe un cero. Si lo hay, es posible ahorrar la substitución.

- 3) También es posible ahorrar la substitución de las columnas a la izquierda de la columna pivote, ya que debido a que éstas columnas han sido eliminadas previamente, contienen ceros en el renglón de la diagonal principal de la columna pivote
- 4) Si se eligen como columnas de términos independientes las n columnas de una matriz identidad, las n columnas de soluciones corresponden a la matriz inversa.

Una vez que se han definido el problema matemático completamente, puede pasarse el análisis que conduzca a escribir un programa de computadora para resolver el problema.

PASO 1.- Definir los límites superiores del tamaño del problema a manejar.

$$\max(N) = 20$$

$$\max(M) = 20$$

PASO 2.- Definir los resultados (salidas) estrictamente necesarias

A) Encabezado: Solución de ecuaciones lineales simultáneas por eliminación.

Programa GAUSS.FOR versión 1.0

B) Eco de los datos leídos

Número de ecuaciones = N =

Número de columnas de términos independientes = M =

Coefficientes de la matriz:

1 2 3 ... N

1

2

3

⋮

N

Columnas de términos independientes:

1	2	3	...	M
1				
2				
3				
⋮				
N				

C) Resultados del problema

Columnas de soluciones:

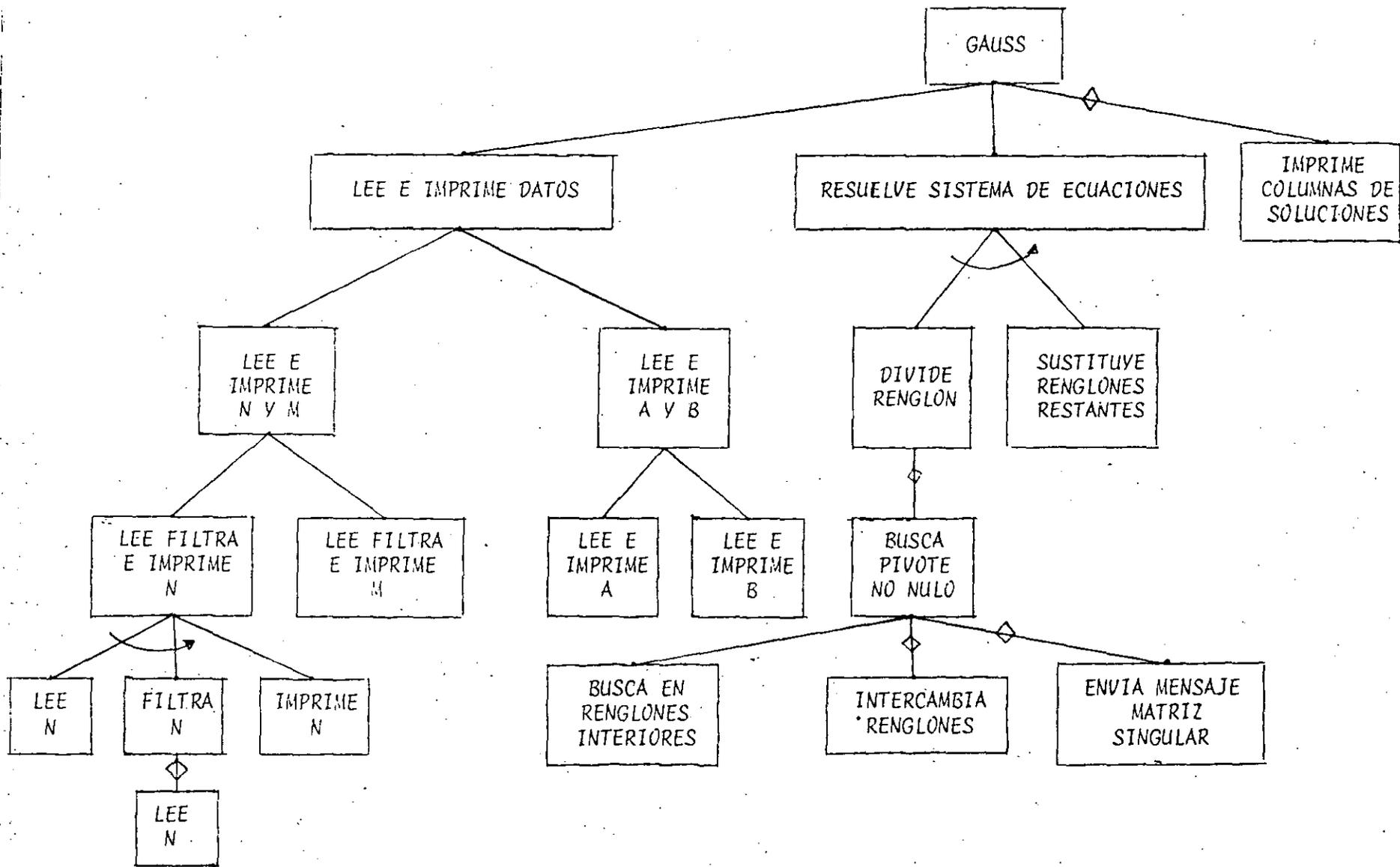
1	2	3	...	M
1				
2				
3				
⋮				
N				

PASO 3.- Definir los posibles errores:

Descripción	Acción
Número $N < 2$ o $N > \text{MAXN}$	Enviar mensaje y volver a leer
Número $M < 1$ o $M > \text{MAXN}$	Enviar mensaje y volver a leer
Determinante del sistema = 0	Enviar mensaje y terminar el programa.

Una vez terminado el análisis, se procederá al Diseño del Programa

PASO 4.- Separar el problema completo en pocos módulos operativos con máxima "cohesión" y mínimo "acoplamiento".



- PASO 5.- Escribir en lenguaje Fortran (codificar en Fortran) los diferentes módulos obtenidos en el paso anterior iniciando con los módulos superiores.
- PASO 6.- Introducir los módulos codificados a la computadora y probarlos.
- PASO 7.- Escribir en Fortran los módulos de menor nivel
- PASO 8.- Repetir pasos 6 y 7 hasta terminar con todos los módulos
- PASO 9.- Agrupar todos los documentos utilizados durante el desarrollo del programa, formando así la documentación técnica.
- PASO 10. Preparar la documentación de operación o el "Instructivo de usuario", con las notas más relevantes de la documentación técnica y algunos ejemplos y observaciones adicionales.

PROGRAMA GAUSS.FOR
CARLOS A. RAMOS LARIOS

OCTUBRE 1983

DIMENSION A(20,20),B(20,20)

10

MAXN=20

MAXM=20

EPS=0.00001

PRINT 2010

LEE E IMPRIME DATOS (1)

LCE E IMPRIME N Y M (1.1)

LEE FILTRA E IMPRIME N (1.1.1)

LCE N (1.1.1.1)

PRINT 2020,MAXN

READ 1000,N

FILTRA N (1.1.1.2)

10 IF(N.GE.2.AND.N.LE.MAXN)GOTO 20

PRINT 2030,N

LCE N(1.1.1.1)

PRINT 2020,MAXN

READ 1000,N

GOTO 10

20 CONTINUE

IMPRIME N (1.1.1.3)

PRINT 2040,N

LEE FILTRA E IMPRIME M (1.1.2)

PRINT 2050,MAXM

READ 1010,M

30 IF(M.GE.1.AND.M.LE.MAXM)GOTO 40

PRINT 2060,M

PRINT 2050,MAXM

READ 1010,M

GOTO 30

40 CONTINUE

PRINT 2070,M

LEE C IMPRIME A Y B (1.2)

LEE E IMPRIME A (1.2.1.)

SE LEE MATRIZ A POR R EN C O L O N E S

DO 50 I=1,N

PRINT 2080,N,I

READ 1030,(A(I,J),J=1,N)

50 CONTINUE

PRINT 2090

SE IMPRIME LA MATRIZ A EN SUBMATRICES DE N X 6

NUMERO DE SUBMATRICES A IMPRIMIR =NSI

NSI=(N-1)/6+1

DO 70 I=1,NSI

CALCULO DE LAS COLUMNAS INICIAL Y FINAL

DE LA SUBMATRIZ I ESIMA

IC1=(I-1)*6+1

IC2=IC1+5

IF(IC2.GT.N)IC2=N

PRINT 2100,(J,J=IC1,IC2)

DO 60 J=1,N

PRINT 2110,J,(A(J,K),K=IC1,IC2)

60 CONTINUE

70 CONTINUE

LEE E IMPRIME B

SE LEE MATRIZ B POR C O L U M N A S

DO 80 I=1,M

PRINT 2120,N,I

READ 1040,(B(J,I),J=1,N)

80 CONTINUE

PRINT 2130

SE IMPRIME LA MATRIZ POR RENGLONES

EN GRUPOS DE 4 COLUMNAS

PRINT 2140,(I,I=1,M)

DO 90 I=1,N

PRINT 2150,I

PRINT 2160,(B(J,I),J=1,M)

90 CONTINUE

RESUELVE SISTEMA DE ECUACIONES(2)

12.

DE 260 I=1,N

I ES EL NUMERO DE LA COLUMNA PIVOTE

DIVIDE RENGLON (2.1)

DIAG=A(I,I)

IF (ABS(DIAG).GT.EPS) GOTO 170

BUSCA PIVOTE NO NULO (2.1.1)

IF (I.LT.N)GOTO 100

ISINGU=1

GOTO 125

CONTINUE

BUSCA EN RENGLONES INFERIORES

ISINGU=1

DO 120 J=I+1,N

IF (ABS(A(J,I)).LE.EPS)GOTO 110

IABAJO=J

ISINGU=0

J=9999

CONTINUE

CONTINUE

CONTINUE

IF (ISINGU.EQ.0) GOTO 130

ENVIA MENSAJE DE MATRIZ SINGULAR (2.1.1.3)

PRINT 2170,I

I=9999

GOTO 160

CONTINUE

INTERCAMBIA RENGLONES (2.1.1.2):

1) DE LA MATRIZ A

DE 140 J=1,N

TEM=A(I,J)

A(I,J)=A(IABAJO,J)

A(IABAJO,J)=TEM

CONTINUE

2) DE LA MATRIZ B

DE 150 J=1,M

TEM=B(I,J)

B(I,J)=B(IABAJO,J)

B(IABAJO,J)=TEM

CONTINUE

CONTINUE

CONTINUE

100

110

120

125

130

140

150

160

170

```
IF (ICINGU.EQ.1) GOTO 250
```

```
DIVIDE EL RENGLON:
```

```
1) DE LA MATRIZ A
```

```
DO 180 J=1,N
  A(I,J)=A(I,J)/DIAG
CONTINUE
```

```
2) DE LA MATRIZ B
```

```
DO 190 J=1,M
  B(I,J)=B(I,J)/DIAG
CONTINUE
```

```
SUSTITUYE RENGLONES RESTANTES (2.2)
```

```
DO 240 J=1,N
```

```
J ES EL NUMERO DEL RENGLON POR SUSTITUIR
SE SUSTITUYEN TODOS EXCEPTO EL DE LA COLUMNA
PIVOTE
```

```
IF(J.EQ.I)GOTO 230
```

```
SE SUSTITUYE SOLO CUANDO NO HAY CERO:
```

```
IF(ABS(A(J,I)).LE.EPS)GOTO 220
SIM=-A(J,I)
```

```
1) DE LA MATRIZ A
```

```
DO 200 K=I,N
  A(J,K)=A(J,K)+A(I,K)*SIM
CONTINUE
```

```
2) DE LA MATRIZ B
```

```
DO 210 K=1,M
  B(J,K)=B(J,K)+B(I,K)*SIM
CONTINUE
```

```
IMPRIME MATRIZ A DESPUES DE SU FORMA FINAL
PARA COMPROBACION
```

```
DO 265 I=1,N
```

```
  PRINT 2175, (A(I,J),J=1,N)
```

```
265 CONTINUE
```

```
2175 FORMAT(6F10.0)
```

IMPRIME COLUMNAS DE SOLUCIONES

14.

PRINT 2100

SE IMPRIME LA MATRIZ B EN SUBMATRICES DE N X 6

NUMERO DE SUBMATRICES A IMPRIMIR =NSI

NSI=(M-1)/6+1

DO 230 I=1,NSI

CALCULO DE LAS COLUMNAS INICIAL Y FINAL
DE LA SUBMATRIZ I ESIMA

IC1=(I-1)*6+1

IC2=IC1+5

IF(IC2.GT.M)IC2=M

PRINT 2170,(J,J=IC1,IC2)

DO 270 J=1,N

PRINT 2200,J,(B(J,K),K=IC1,IC2)

270 CONTINUE

230 CONTINUE

CALL EXIT

1000 FORMAT(I2)

1010 FORMAT(I2)

1030 FORMAT(8F10.0)

1040 FORMAT(8F10.0)

2010 FORMAT(' SOLUCION DE ECUACIONES SIMULTANEAS POR ',

1 'ELIMINACION ',/, ' PROGRAMA GAUSS.FOR ',

2 'VERSION 1.0',//)

2020 FORMAT(' DAME EL NUMERO DE ECUACIONES. EN I2 ',/,

1 ' (MINIMO 2,MAXIMO ',I2,')')

2030 FORMAT(' NUMERO DE ECUACIONES= ',I2, ' INVALIDO')

2040 FORMAT(' NUMERO DE ECUACIONES=N= ',I2)

2050 FORMAT(' DAME EL NUMERO DE COLUMNAS DE TERMINOS',

1 ' INDEPENDIENTES,EN I2 ',/,

2 ' (MINIMO 1, MAXIMO ',I2,')')

2060 FORMAT(' NUMERO DE COLUMNAS DE TERMINOS INDEPENDIENTES=',

1 I2, ' INVALIDO')

2070 FORMAT(' NUMERO DE COLUMNAS DE TERMINOS INDEPENDIENTES=',

1 ' M= ',I2)

2080 FORMAT(' DAME LOS ',I2, ' ELEMENTOS DEL RENGLON ',I2,

1 ', ' DE LA MATRIZ DE COEFICIENTES, EN 8F10.0',/,

2 1X,7('1234567890'))

2090 FORMAT(/, ' COEFICIENTES DE LA MATRIZ:',/)

2100 FORMAT(/,3X,6(7X,I2,3X))

2110 FORMAT(1X,I2,6F12.2)

2120 FORMAT(' DAME LOS ',I2, ' ELEMENTOS DE LA COLUMNA ',I2,

1 ', ' DE TERMINOS INDEPENDIENTES, EN 8F10.0',/,

2 1X,7('1234567890'))

2130 FORMAT(/, ' COLUMNAS DE TERMINOS INDEPENDIENTES:',/)

2140 FORMAT(6(7X,I2,3X))

2150 FORMAT(1X,I2)

2160 FORMAT(6F12.2)

2170 FORMAT(' LA MATRIZ RESULTA SINGULAR AL ELIMINAR LA',

1 ' COLUMNA ',I2,/, ' VALOR DEL CERO RELATIVO=',

2 E15.0)

2180 FORMAT(/, ' COLUMNAS DE SOLUCIONES:',/)

2190 FORMAT(/,3X,6(7X,I2,3X))

2200 FORMAT(1X,I2,6F12.2)

END

RESOLUCION DE ECUACIONES SIMULTANEAS POR ELIMINACION
PROGRAMA GAUSS.FOR VERSION 1.0

15.

DAME EL NUMERO DE ECUACIONES, EN I2
(MINIMO 2,MAXIMO 20)

07

25

NUMERO DE ECUACIONES= 25 INVALIDO

DAME EL NUMERO DE ECUACIONES, EN I2
(MINIMO 2,MAXIMO 20)

01

NUMERO DE ECUACIONES= 1 INVALIDO

DAME EL NUMERO DE ECUACIONES, EN I2
(MINIMO 2,MAXIMO 20)

03

NUMERO DE ECUACIONES=N= 3

DAME EL NUMERO DE COLUMNAS DE TERMINOS INDEPENDIENTES,EN I2
(MINIMO 1, MAXIMO 20)

25

NUMERO DE COLUMNAS DE TERMINOS INDEPENDIENTES=25 INVALIDO

DAME EL NUMERO DE COLUMNAS DE TERMINOS INDEPENDIENTES,EN I2
(MINIMO 1, MAXIMO 20)

NUMERO DE COLUMNAS DE TERMINOS INDEPENDIENTES=M= 2

DAME LOS 3 ELEMENTOS DEL RENGLON 1

DE LA MATRIZ DE COEFICIENTES, EN 0F10.0

12345678901234567890123456789012345678901234567890123456789012345

2. -1. 4.

DAME LOS 3 ELEMENTOS DEL RENGLON 2

DE LA MATRIZ DE COEFICIENTES, EN 8F10.0

12345678901234567890123456789012345678901234567890123456789012345

5. 2. 8.

DAME LOS 3 ELEMENTOS DEL RENGLON 3

DE LA MATRIZ DE COEFICIENTES, EN 0F10.0

12345678901234567890123456789012345678901234567890123456789012345

-1. -1. 4.

COEFICIENTES DE LA MATRIZ:

	1	2	3
1	2.00	-1.00	6.00
2	5.00	2.00	3.00
3	-1.00	-1.00	4.00

DAME LOS 3 ELEMENTOS DE LA COLUMNA 1

DE TERMINOS INDEPENDIENTES, EN 0F10.0

12345678901234567890123456789012345678901234567890123456789012345

18. 33. 9.

DAME LOS 3 ELEMENTOS DE LA COLUMNA 2

DE TERMINOS INDEPENDIENTES, EN 8F10.0

12345678901234567890123456789012345678901234567890123456789012345

1. 2. 3.

16.

COLUMNAS DE TERMINOS INDEPENDIENTES:

	1	2	
1	18.00	1.00	
2	33.00	2.00	
3	9.00	3.00	
	1.	0.	0.
	0.	1.	0.
	0.	0.	1.

COLUMNAS DE SOLUCIONES:

	1	2
1	1.00	-1.14
2	2.00	1.00
3	3.00	0.71

El modelo para la respuesta es:

$$[P] = \frac{1}{12} \begin{bmatrix} -\omega_1 L_1^2 \\ \omega_1 L_1^2 - \omega_2 L_2^2 \\ \omega_2 L_2^2 - \omega_3 L_3^2 \\ \omega_3 L_3^2 - \omega_4 L_4^2 \\ \vdots \\ \omega_{n-1} L_{n-1}^2 \end{bmatrix}$$

$$[K][U] = [P]$$

función de las propiedades de la estructura $(L_i, (EI)_i)$

función de las cargas sobre la estructura y de las

DIMENSION AK(10,10), P(10), AL(10), W(10), ALFA(10)

READ *,N

DO 10 I=1,N-1

READ *,AL(I),W(I),ALFA(I)

10 CONTINUE

DO 20 I=1,N

DIAGONAL PRINCIPAL

IF (I.NE.1) AK(I,I) = AK(I,I) + 4*ALFA(I-1)/AL(I-1)

IF (I.NE.N) AK(I,I) = AK(I,I) + 4*ALFA(I)/AL(I)

DIAGONALES PARALELAS

IF (I.NE.1) AK(I,I-1) = AK(I,I-1) + 2*ALFA(I-1)/AL(I-1)

IF (I.NE.N) AK(I,I+1) = AK(I,I+1) + 2*ALFA(I)/AL(I)

VECTOR DE CARGAS

IF (I.NE.1) P(I) = P(I) + W(I-1)*AL(I-1)*AL(I-1)

IF (I.NE.N) P(I) = P(I) - W(I)*AL(I)*AL(I)

20 CONTINUE

DO 30 I=1,N

PRINT *, (AK(I,J), J=1,N), P(I)

30 CONTINUE

CALL EXIT

END



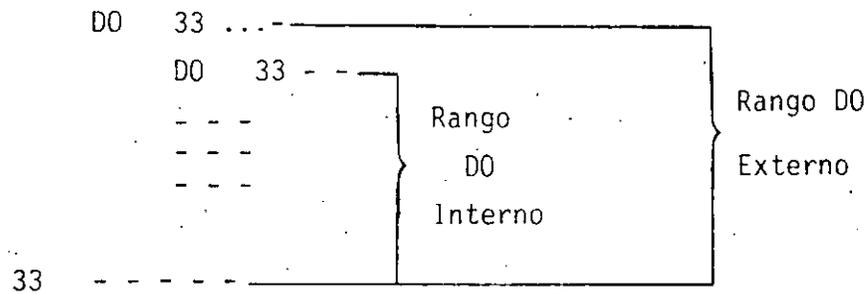
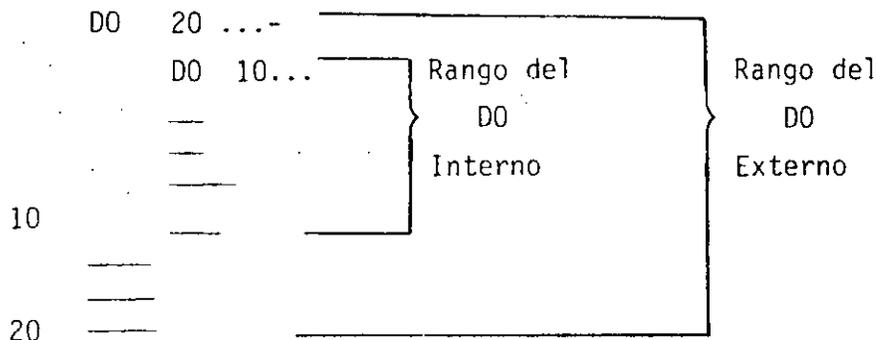
**DIVISION DE EDUCACION CONTINUA
FACULTAD DE INGENIERIA U.N.A.M.**

LENGUAJE DE PROGRAMACION FORTRAN

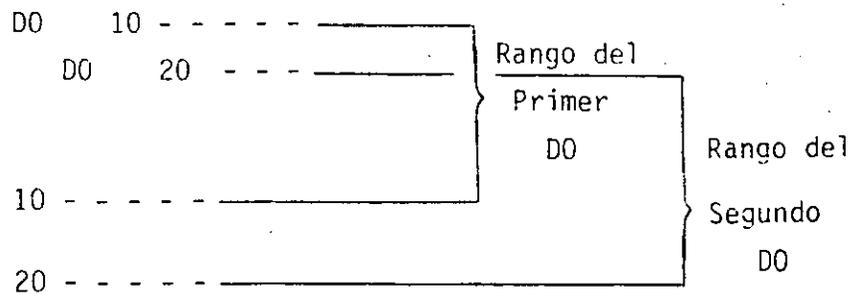
A N I D A M I E N T O

OCTUBRE, 1984.

Hay ocasiones en las que es necesario utilizar una proposición DO dentro del rango de otra proposición DO, a lo cual se le llama - anidamiento . Al utilizar un anidamiento de proposiciones DO hay que tener cuidado con lo siguiente: el final de la proposición DO interna debe estar antes, o cuando mucho, en la misma proposición que en la que termina la proposición DO externa, es decir, el rango de la proposición DO interna (ó anidada) debe estar completamente contenido dentro del rango del DO externo; veamos unos ejemplos:



Lo siguiente es incorrecto:



No hay límites en el nivel de anidamiento, es decir, en el número de (2) DO que se pueden anidar dentro de otro: sin embargo, si hay muchos - DO anidados el programa puede parecer confuso, por lo que se recomienda usar siempre la proposición CONTINUE que se explica en seguida.

C
C EJEMPLO 6.1 EXPRESION ARITMETICA (B)
C

PRINT *, 'TECLEE EL NUMERO DE ALUMNOS DEL GRUPO '
READ *, NUMERO
PROGPO = 0.0

C
C LA VARIABLE "INDICE" ES EL CONTADOR DE ALUMNOS PROCESADOS
C

INDICE = 1
10 PRINT *, 'TECLEE LAS CINCO CALIFICACIONES DEL ALUMNO ', INDICE
READ *, CALIF1, CALIF2, CALIF3, CALIF4, CALIF5
PROALU = (CALIF1+CALIF2+CALIF3+CALIF4+CALIF5)/5.0
PRINT *, 'EL ALUMNO ', INDICE, ' TIENE ', PROALU, ' DE PROMEDIO '
PROGPO = PROGPO + PROALU
INDICE = INDICE + 1

C
C EL SIGUIENTE "IF" PREGUNTA SI LA VARIABLE "INDICE" ANTES
C INCREMENTADA, ES MENOR O IGUAL A LA VARIABLE "NUMERO" PARA
C REGRESAR EL CONTROL A LA PROPOSICION CON LA ETIQUETA 10, 0
C CONTINUAR CON LA SIGUIENTE PROPOSICION (ETIQUETA NUMERO 20)
C

IF (INDICE-NUMERO) 10,10,20
20 PROGPO = PROGPO/NUMERO
PRINT *, 'EL PROMEDIO GENERAL DEL GRUPO ES: ', PROGPO
STOP
END

EJEMPLO 6.2

ARREGLOS

EL VECTOR "CALIF" CONTENDRA LAS CALIFICACIONES DE CADA ALUMNO.
 EL NUMERO DE CALIFICACIONES POR ALUMNO ES CONTROLADO POR
 LA VARIABLE "NUMCAL", PERMITIENDOSE COMO MAXIMO
 EL NUMERO DE ELEMENTOS EN EL VECTOR "CALIF", O SEA, 10

```
DIMENSION CALIF(10)
PRINT *, 'TECLEE EL NUMERO DE ALUMNOS DEL GRUPO '
READ *, NUMALU
PRINT *, 'TECLEE EL NUMERO DE CALIFICACIONES POR ALUMNO '
READ *, NUMCAL
PROGPO = 0.0
```

LA VARIABLE "INDALU" ES EL CONTADOR DE ALUMNOS.

```
INDALU = 1
```

```
10 PRINT *, 'TECLEE LAS ', NUMCAL,
    ' CALIFICACIONES DEL ALUMNO ', INDALU
    PROALU = 0.0
```

LA VARIABLE "INDCAL" ES EL CONTADOR DE
 CALIFICACIONES POR CADA ALUMNO

```
INDCAL = 1
```

```
20 READ *, CALIF(INDCAL)
    PROALU = PROALU + CALIF(INDCAL)
    INDCAL = INDCAL + 1
```

EL SIGUIENTE "IF" CONTROLA LA LECTURA DE
 LAS CALIFICACIONES DE CADA ALUMNO

```
IF ( INDCAL - NUMCAL ) 20, 20, 30
30 PROALU = PROALU / NUMCAL
    PRINT *, 'EL ALUMNO ', INDALU, ' TIENE ', PROALU, ' DE PROMEDIO '
    PROGPO = PROGPO + PROALU
    INDALU = INDALU + 1
```

EL SIGUIENTE "IF" CONTROLA EL PROCESO DE CADA ALUMNO

```
IF ( INDALU - NUMALU ) 10, 10, 40
40 PROGPO = PROGPO / NUMALU
    PRINT *, 'EL PROMEDIO GENERAL DEL GRUPO ES: ', PROGPO
    STOP
    END
```

```

C      EJEMPLO 6.3      FORMATOS- DATOS SEFANUMERICOS
C
C      EL VECTOR "NOMBRE" CONTENDRA EL NOMBRE DE CADA ALUMNO.
C      SE GUARDARA UN CARACTER EN CADA ELEMENTO DEL VECTOR,
C      POR LO QUE SOLO PODRAN GUARDARSE NOMBRES DE 32 O MENOS CARACTERES
C
      DIMENSION CALIF(10),NOMBRE(32)
      WRITE(6,10)
10     FORMAT(' TECLEE EL NUMERO DE ALUMNOS DEL GRUPO (I2)')
      READ(5,20)NUMALU
20     FORMAT(I2)
      WRITE(6,30)
30     FORMAT(' TECLEE EL NUMERO DE CALIFICACIONES POR ALUMNO (I1)')
      READ(5,40)NUMCAL
40     FORMAT(I1)
      PROGFO = 0.0

C
C      LA VARIABLE "INDALU" ES EL CONTADOR DE ALUMNOS
C
      INDALU = 1
50     WRITE(6,60)INDALU
60     FORMAT(' TECLEE EL NOMBRE DEL ALUMNO NUMERO ',I2,' (A32)')
      READ(5,70)NOMBRE
70     FORMAT(32A1)
      WRITE(6,80)NUMCAL,NOMBRE
80     FORMAT(' TECLEE LAS ',I1,' CALIFICACIONES DE: ',32A1,
              '(F3.0)')
      PROALU = 0.0

C
C      LA VARIABLE "INDCAL" ES EL CONTADOR DE
C      CALIFICACIONES POR CADA ALUMNO
C
      INDCAL = 1
90     READ(5,100)CALIF(INDCAL)
100    FORMAT(F3.0)
      PROALU = PROALU+CALIF(INDCAL)
      INDCAL = INDCAL+1

C
C      EL SIGUIENTE "IF" CONTROLA LA LECTURA DE
C      LAS CALIFICACIONES DE CADA ALUMNO
C
      IF ( INDCAL-NUMCAL ) 90,90,110
110    PROALU = PROALU/NUMCAL
      WRITE(6,120)INDALU,NOMBRE,PROALU
120    FORMAT(1X,I2,'.- ',32A1,5X,F5.2)
      PROGFO = PROGFO + PROALU
      INDALU = INDALU+1

C
C      EL SIGUIENTE "IF" CONTROLA EL PROCESO DE CADA ALUMNO
C
      IF ( INDALU-NUMALU ) 50,50,130
130    PROGFO = PROGFO/NUMALU
      WRITE(6,140)PROGFO
140    FORMAT(//,' PROMEDIO GENERAL DEL GRUPO: ',F5.2)
      STOP
      END

```

EJEMPLO 6.4ITERACIONES CON LA PROPOSICION "DO"

(4)

TODAS LAS ITERACIONES SE REALIZAN CON LA PROPOSICION "DO"
INCLUYENDO UN "DO" IMPLICITO PARA LA LECTURA DE LAS CALIFICACIONES

```
DIMENSION CALIF(10),NOMBRE(32)
```

```
WRITE(6,10)
```

```
10 FORMAT(' TECLEE EL NUMERO DE ALUMNOS DEL GRUPO (I2)')
```

```
READ(5,20)NUMALU
```

```
20 FORMAT(I2)
```

```
WRITE(6,30)
```

```
30 FORMAT(' TECLEE EL NUMERO DE CALIFICACIONES POR ALUMNO (I1)')
```

```
READ(5,40)NUMCAL
```

```
40 FORMAT(I1)
```

```
PROGPO = 0.0
```

EL SIGUIENTE "DO" CONTROLA EL PROCESO DE CADA ALUMNO

```
DO 110 INDALU=1,NUMALU
```

```
WRITE(6,50)INDALU
```

```
50 FORMAT(' TECLEE EL NOMBRE DEL ALUMNO NUMERO ',I2,' (A32)')
```

```
READ(5,60)NOMBRE
```

```
60 FORMAT(32A1)
```

```
WRITE(6,70)NUMCAL,NOMBRE,NUMCAL
```

```
70 FORMAT(' TECLEE LAS ',I1,' CALIFICACIONES DE: ',32A1,
```

```
(' ',I1,' F3.0)')
```

EL SIGUIENTE "READ" LEE DE UNA SOLA VEZ TODAS LAS
CALIFICACIONES POR MEDIO DE UN "DO" IMPLICITO DE LECTURA,
POR LO QUE LOS DATOS DEBERAN ESTAR TODOS EN LA MISMA LINEA

```
READ(5,80) ( CALIF(INDCAL),INDCAL=1,NUMCAL )
```

```
80 FORMAT(9F3.0)
```

```
PROALU = 0.0
```

EL SIGUIENTE "DO" ACUMULA TODAS LA CALIFICACIONES
EN LA VARIABLE "PROALU"

```
DO 90 INDICAL=1,NUMCAL
```

```
PROALU = PROALU+CALIF(INDCAL)
```

```
90 CONTINUE
```

```
PROALU = PROALU/NUMCAL
```

```
WRITE(6,100)INDALU,NOMBRE,PROALU
```

```
100 FORMAT(1X,I2,'.- ',32A1,5X,F5.2)
```

```
PROGPO = PROGPO + PROALU
```

```
110 CONTINUE
```

```
PROGPO = PROGPO/NUMALU
```

```
WRITE(6,120)PROGPO
```

```
120 FORMAT(//,' PROMEDIO GENERAL DEL GRUPO: ',F5.2)
```

```
STOP
```

```
END
```



**DIVISION DE EDUCACION CONTINUA
FACULTAD DE INGENIERIA U.N.A.M.**

LENGUAJE DE PROGRAMACION FORTRAN

B I B L I O G R A F I A

OCTUBRE, 1984

BIBLIOGRAFIA

- 1.- TITULO: PROGRAMACION FORTRAN
AUTORES: ROBERT A. STERN
NANCY B. STERN
EDITORIAL: LIMUSA

- 2.- TITULO: PROGRAMACION CON FORTRAN
AUTORES: SEYMOUR LIPSCHUTZ
ARTHUR POE
EDITORIAL: Mc GRAW HILL

- 3.- TITULO: PROGRAMACION FORTRAN 1V
AUTORES: DANIEL D. Mc. CRAKEN
EDITORIAL: LIMUSA WILEY, S.A.

- 4.- TITULO: FORTRAN 1V PROGRAMMING
AUTORES: PAUL W. MURRILL
CECIL L. SMITH
EDITORIAL: INTERNATIONAL TEXTBOOK COMPANY

- 5.- TITULO: FORTRAN PARA INGENIERIA
AUTORES: WILLIAM SCHICK
CHARLES J. MERZ, JR.
EDITORIAL: Mc. GRAW-HILL

- 6.- TITULO: INTRODUCCION AL FORTRAN 1V
AUTORES: ROBERT H. HAMMOND
WILLIAM B. ROGERS
BYARD HOUCK JR.
EDITORIAL: Mc. GRAW-HILL.

7.- TITULO: PROGRAMACION EN FORTRAN 1V
AUTORES: WILLIAM F. SCHALLERT
CARLOL REEDY CLARK
EDITORIAL: FONDO EDUCATIVO INTERAMERICANO, S.A.

8.- TITULO: FORTRAN
AUTORES: DR. ROBERT E. SMITH
DORA E. JOHNSON
EDITORIAL: LIMUSA-WILEY, S.A.



**DIVISION DE EDUCACION CONTINUA
FACULTAD DE INGENIERIA U.N.A.M.**

LENGUAJE DE PROGRAMACION FORTRAN

IF ARITMETICO
COMPLEMENTO

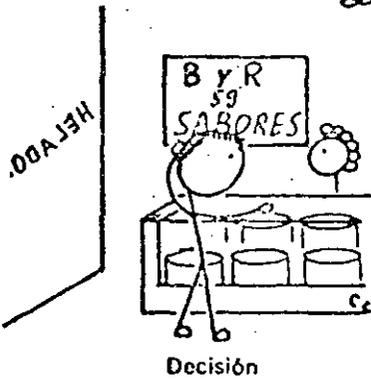
ING. HERIBERTO OLGUIN ROMO

OCTUBRE, 1984.

OBJETIVO: Construir y usar la proposición IF aritmética.

La proposición IF aritmética permite tomar tres caminos alternativos, según una decisión. A continuación se da la forma general de la proposición IF aritmética.

FORMA GENERAL	
1 2 3 4 5 6 7 8 9 ...	nnnnn IF(e) m ₁ , m ₂ , m ₃
COMPONENTES	
nnnnn	es el número de la proposición
IF	es la palabra clave
e	es el argumento de la proposición IF y puede ser cualquier expresión aritmética válida
m ₁	es el número de la proposición a la cual se efectuará la transferencia si e < 0
m ₂	es el número de la proposición a la cual se efectuará la transferencia si e = 0
m ₃	es el número de la proposición a la cual se efectuará la transferencia si e > 0
EJEMPLOS	
1 2 3 4 5 6 7 8 9 ...	IF(A-1.0)5,10,20
	IF(A**2-B**2)5,5,10



3

La e es cualquier expresión aritmética válida. Una expresión aritmética es una secuencia de constantes numéricas y/o de variables conectadas por los operadores aritméticos, como, por ejemplo $X + 2 * Y - 4$. La expresión aritmética tiene un valor numérico; este valor puede ser menor que cero, exactamente igual a cero o mayor que cero. Esta condición determina la proposición que se ha de ejecutar después de la IF. Se hace la transferencia a la proposición m₁ si e < 0, a la proposición m₂ si e = 0, y a la proposición m₃ si e > 0. Esto se muestra en el diagrama de flujo de la fig.

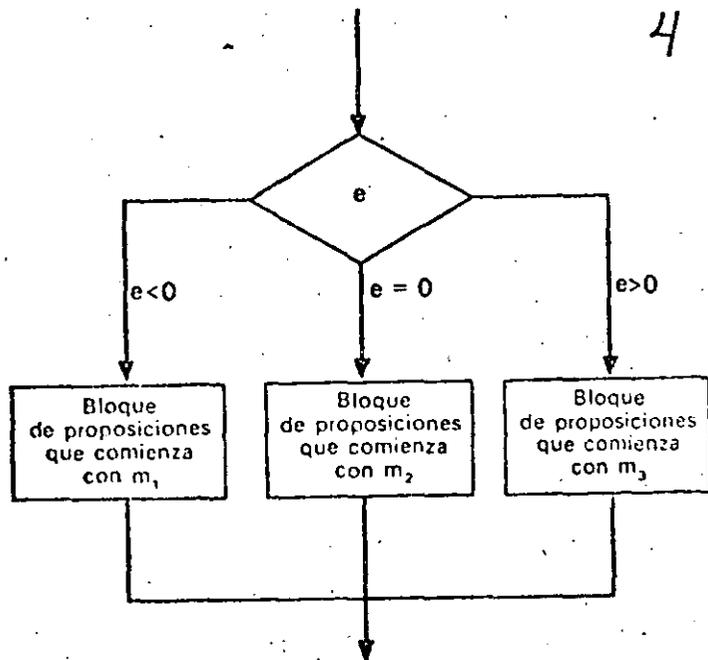


Diagrama de flujo para la proposición IF aritmética.

A continuación se da un ejemplo de la proposición IF aritmética.

EJEMPLO 1

Se desea escribir un programa para extraer las raíces de la ecuación cuadrada $ax^2 + bx + c$ utilizando la fórmula

$$R_1, R_2 = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Las siguientes son las proposiciones necesarias para calcular el radicando ($b^2 - 4ac$) y transferir el control al segmento de programa identificado por la proposición 10 si $(b^2 - 4ac) < 0$, a la proposición 20 si $(b^2 - 4ac) > 0$ y a la proposición 15 si $(b^2 - 4ac) = 0$.

```

1 2 3 4 5 6 7 8 9 ...
      IF(B**2-4.*A*C)10,15,20
10  ...
15  ...
20  ...
    
```

La proposición que sigue a una IF aritmética siempre debe tener un número de proposición.

Para «saltar» los segmentos de programa no deseados se utiliza la proposición GO TO, como se muestra en la próxima solución más completa de la ecuación cuadrática.

EJEMPLO 2

```

1 2 3 4 5 6 7 8 9 ...
      ...
      DISC=B*B-4.*A*C
      IF(DISC)10,15,20
10  WRITE(3,90)
90  FORMAT('NO HAY RAICES REALES')
      GO TO 25
15  X1=-B/(2.*A)
      X2=X1
      GO TO 22
20  X1=(-B+SQRT(DISC))/(2.*A)
      X2=(-B-SQRT(DISC))/(2.*A)
22  WRITE(3,92)X1,X2
92  FORMAT('BLAS RAICES REALES SON', 2F10.1)
25  CONTINUE
    
```

EJEMPLO 3

Escribir proposiciones que sumen los enteros positivos de uno a 100.

```

1 2 3 4 5 6 7 8 9 ...
      I=1
      ISUM=0
8  ISUM=ISUM+I
      I=I+1
      IF(I-100)8,8,6
6  CONTINUE
    
```

Escribir un ciclo propio es especialmente ventajoso cuando no hay ningún «conteo» asociado naturalmente al ciclo y cuando los valores iniciales, terminales y de incremento no son enteros.

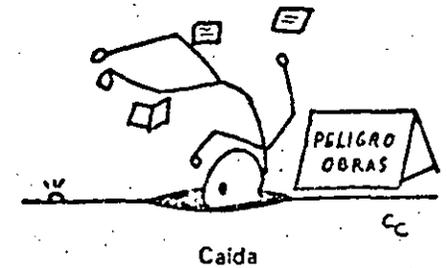
El ejemplo 4 ilustra tal situación.

EJEMPLO 4

Escribir proposiciones que resuelvan $y = \sin x + \cos x$ para valores de x de -10 a 10 radianes en incrementos de 0.1 .

```

1 2 3 4 5 6 7 8 9 ...
      X=-10.1
5  X=X+.1
      IF(X-10.)6,6,10
6  Y=SIN(X) + COS(X)
      WRITE(3,90)X,Y
      GO TO 5
10 CONTINUE
90 FORMAT('b',2E13.5)
      ...
    
```



2

En el ejemplo 5, el valor de la expresión debe ser exactamente cero para una transferencia de la proposición 20.

EJEMPLO 5

Este programa hace que el computador entre en un ciclo sin fin, ya que el

<pre> 123456789... A=0 5 A=A+.1 Y=A**3 WRITE(3,10)A,Y 10 FORMAT('0',2F7.1) IF(A-1.0)5,20,5 20 CONTINUE </pre>	<p>TP*</p> <p>Inspeccionar todos los ciclos iterativos para prevenir los ciclos iterativos sin fin.</p> <ul style="list-style-type: none"> • ¿tiene el ciclo iterativo una condición de salida? • cuidado con los «iguales» como una condición con los reales. Pruebe en condiciones «no iguales» o use los enteros como prueba. <p>* Tácticas preventivas.</p>
---	--

valor de (A-1.0) jamás será exactamente cero. Situaciones como esta pueden evitarse si no se basa nunca la prueba en «iguales», sino en «menor que o iguales» o «mayor que o iguales» al utilizar números reales.

La proposición If en el ejemplo anterior tendrá la forma

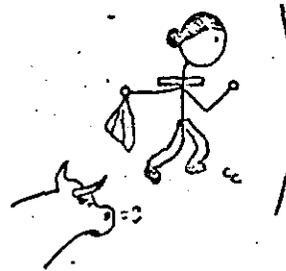
<pre> 123456789... IF(A-1.0)5,20,20 </pre>
--

ya que A siempre excederá a 1.0 en algún punto,

Otro método para evitar el ciclo sin fin es usar aritmética entera o «exacta», como se muestra en el ejemplo 6.

EJEMPLO 6

<pre> 123456789... I=0 5 I=I+1 A=SIN(FLD(I)/3) Y=A**3 WRITE(3,10)A,Y 10 FORMAT('0',2E12.5) IF(I-10)5,20,5 20 CONTINUE </pre>
--



¡No entrar a menos que haya salida!

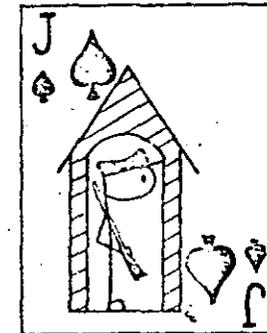
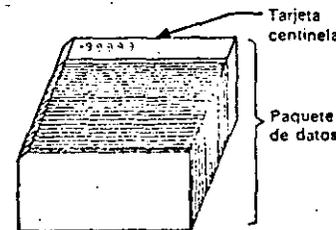
El ejemplo siguiente muestra cómo crear su propio ciclo para la unidad de procesamiento (ver el ejemplo de la sección 10.3) sin usar la opción END=. Un procedimiento para manejar la condición de fin de datos es usar como la última de la serie de tarjetas de datos una *tarjeta centinela*. Esta tarjeta contiene algún valor, tal como 9999, en un campo no usado o un valor fuera de los límites de los valores posibles para uno de los datos leídos. Inmediatamente después de haberse leído el valor del dato, se prueba el campo que contiene el indicador para el fin de datos. La proposición IF aritmética provee un medio para hacer esto.

EJEMPLO 7

Entrar en la memoria todas las tarjetas de identidad de los alumnos e imprimir la información que contienen. El primer campo en cada tarjeta contiene el número de identidad del alumno, un entero entre 100000 y 999999. Puesto que cualquier número negativo es inválido, seleccionamos en forma arbitraria -99999 y lo perforamos en el campo correspondiente al número de identidad del alumno. (Se podría usar cualquier número negativo o simplemente una tarjeta en blanco.) Esta tarjeta centinela se ubica después de la última tarjeta válida de datos.

El programa lee cada tarjeta, comprueba si el campo de números de identidad del alumno contiene un número negativo y transfiere a STOP cuando lo encuentra. Esto se muestra a continuación.

<pre> 123456789... 5 READ(1,90)ID,CLASS,GRPT 90 FORMAT(16,4X,A2,4X,F10.1) IF(ID)99,99,10 10 ...procesar la tarjeta GO TO 5 99 STOP </pre>



Tarjeta centinela

EJERCICIOS

4

(4)

1. Escribir una proposición IF aritmética que transfiera el control a la proposición 10 si A es menor que 25, a la proposición 20 si A es igual a 25 y a la proposición 30 si A es mayor que 25.

123456789...

2. Escribir una proposición IF aritmética que transfiera el control a la proposición 10, 20 ó 30, dependiendo de si $IMO < 0$, $IMO = 0$ o $IMO > 0$, respectivamente.

123456789...



**DIVISION DE EDUCACION CONTINUA
FACULTAD DE INGENIERIA U.N.A.M.**

LENGUAJE DE PROGRAMACION FORTRAN

EXPRESION MATEMATICA
COMPLEMENTO

M. EN C. RICARDO CIRIA MERCE

OCTUBRE, 1984.

```
C
C  EJEMPLO 6.1      EXPRESION ARITMETICA
C
PRINT *, 'TECLEE EL NUMERO DE ALUMNOS DEL GRUPO '
READ *, NUMERO
PROGPO = 0.0

C
C  LA VARIABLE "INDICE" ES EL CONTADOR DE ALUMNOS PROCESADOS
C
INDICE = 1
10 PRINT *, 'TECLEE LAS CINCO CALIFICACIONES DEL ALUMNO ', INDICE
   READ *, CALIF1, CALIF2, CALIF3, CALIF4, CALIF5
   PROALU = (CALIF1+CALIF2+CALIF3+CALIF4+CALIF5)/5.0
   PRINT *, 'EL ALUMNO ', INDICE, ' TIENE ', PROALU, ' DE PROMEDIO'
   PROGPO = PROGPO + PROALU
   INDICE = INDICE + 1

C
C  EL SIGUIENTE "IF" PREGUNTA SI LA VARIABLE "INDICE" ANTES
C  INCREMENTADA, ES MENOR O IGUAL A LA VARIABLE "NUMERO" PARA
C  REGRESAR EL CONTROL A LA PROPOSICION CON LA ETIQUETA 10, O
C  CONTINUAR CON LA SIGUIENTE PROPOSICION (ETIQUETA NUMERO 20)
C
   IF ( INDICE-NUMERO ) 10,10,20
20 PROGPO = PROGPO/NUMERO
PRINT *, 'EL PROMEDIO GENERAL DEL GRUPO ES: ', PROGPO
STOP
END
```

(2)

```

C   EJEMPLO 6.2           ARREGLOS
C
C   EL VECTOR "CALIF" CONTENDRA LAS CALIFICACIONES DE CADA ALUMNO.
C   EL NUMERO DE CALIFICACIONES POR ALUMNO ES CONTROLADO POR
C   LA VARIABLE "NUMCAL", PERMITIENDOSE COMO MAXIMO
C   EL NUMERO DE ELEMENTOS EN EL VECTOR "CALIF", O SEA, 10
C
DIMENSION CALIF(10)
PRINT *, 'TECLEE EL NUMERO DE ALUMNOS DEL GRUPO '
READ *, NUMALU
PRINT *, 'TECLEE EL NUMERO DE CALIFICACIONES POR ALUMNO '
READ *, NUMCAL
PROGPO = 0.0

C
C   LA VARIABLE "INDALU" ES EL CONTADOR DE ALUMNOS
C
INDALU = 1
10 PRINT *, 'TECLEE LAS ', NUMCAL,
      ' CALIFICACIONES DEL ALUMNO ', INDALU
      PROALU = 0.0

C
C   LA VARIABLE "INDCAL" ES EL CONTADOR DE
C   CALIFICACIONES POR CADA ALUMNO
C
INDCAL = 1
20 READ *, CALIF(INDCAL)
      PROALU = PROALU + CALIF(INDCAL)
      INDCAL = INDCAL + 1

C
C   EL SIGUIENTE "IF" CONTROLA LA LECTURA DE
C   LAS CALIFICACIONES DE CADA ALUMNO
C
      IF ( INDCAL - NUMCAL ) 20, 20, 30
30 PROALU = PROALU / NUMCAL
      PRINT *, 'EL ALUMNO ', INDALU, ' TIENE ', PROALU, ' DE PROMEDIO '
      PROGPO = PROGPO + PROALU
      INDALU = INDALU + 1

C
C   EL SIGUIENTE "IF" CONTROLA EL PROCESO DE CADA ALUMNO
C
      IF ( INDALU - NUMALU ) 10, 10, 40
40 PROGPO = PROGPO / NUMALU
      PRINT *, 'EL PROMEDIO GENERAL DEL GRUPO ES: ', PROGPO
      STOP
      END

```

EJEMPLO 6.3

FORMATOS. DATOS ALFANUMERICOS (3)

EL VECTOR "NOMBRE" CONTENDRA EL NOMBRE DE CADA ALUMNO.
SE GUARDARA UN CARACTER EN CADA ELEMENTO DEL VECTOR,
POR LO QUE SOLO PODRAN GUARDARSE NOMBRES DE 32 O MENOS CARACTERES

DIMENSION CALIF(10), NOMBRE(32)

WRITE(6,10)

10 FORMAT(' TECLEE EL NUMERO DE ALUMNOS DEL GRUPO (I2)')

READ(5,20)NUMALU

20 FORMAT(I2)

WRITE(6,30)

30 FORMAT(' TECLEE EL NUMERO DE CALIFICACIONES POR ALUMNO (I1)')

READ(5,40)NUMCAL

40 FORMAT(I1)

PROGPO = 0.0

LA VARIABLE "INDALU" ES EL CONTADOR DE ALUMNOS

INDALU = 1

50 WRITE(6,60)INDALU

60 FORMAT(' TECLEE EL NOMBRE DEL ALUMNO NUMERO ',I2,' (A32)')

READ(5,70)NOMBRE

70 FORMAT(32A1)

WRITE(6,80)NUMCAL,NOMBRE

80 FORMAT(' TECLEE LAS ',I1,' CALIFICACIONES DE: ',32A1,

' (F3.0)')

PROALU = 0.0

LA VARIABLE "INDCAL" ES EL CONTADOR DE
CALIFICACIONES POR CADA ALUMNO

INDCAL = 1

90 READ(5,100)CALIF(INDCAL)

100 FORMAT(F3.0)

PROALU = PROALU + CALIF(INDCAL)

INDCAL = INDCAL + 1

EL SIGUIENTE "IF" CONTROLA LA LECTURA DE
LAS CALIFICACIONES DE CADA ALUMNO.

110 IF (INDCAL - NUMCAL) 90,90,110

PROALU = PROALU / NUMCAL

WRITE(6,120)INDALU,NOMBRE,PROALU

120 FORMAT(1X,I2,'.- ',32A1,5X,F5.2)

PROGPO = PROGPO + PROALU

INDALU = INDALU + 1

EL SIGUIENTE "IF" CONTROLA EL PROCESO DE CADA ALUMNO

IF (INDALU - NUMALU) 50,50,130

130 PROGPO = PROGPO / NUMALU

WRITE(6,140)PROGPO

140 FORMAT(//,' PROMEDIO GENERAL DEL GRUPO: ',F5.2)

STOP

END

EJEMPLO 6.4ITERACIONES CON LA PROPOSICION "DO"

(4)

TODAS LAS ITERACIONES SE REALIZAN CON LA PROPOSICION "DO"
INCLUYENDO UN "DO" IMPLICITO PARA LA LECTURA DE LAS CALIFICACIONES

DIMENSION CALIF(10),NOMBRE(32)

WRITE(6,10)

10 FORMAT(' TECLEE EL NUMERO DE ALUMNOS DEL GRUPO (I2)')

READ(5,20)NUMALU

20 FORMAT(I2)

WRITE(6,30)

30 FORMAT(' TECLEE EL NUMERO DE CALIFICACIONES POR ALUMNO (I1)')

READ(5,40)NUMCAL

40 FORMAT(I1)

PROGPO = 0.0

EL SIGUIENTE "DO" CONTROLA EL PROCESO DE CADA ALUMNO

DO 110 INDALU=1,NUMALU

WRITE(6,50)INDALU

50 FORMAT(' TECLEE EL NOMBRE DEL ALUMNO NUMERO ',I2,' (A32)')

READ(5,60)NOMBRE

60 FORMAT(32A1)

WRITE(6,70)NUMCAL,NOMBRE,NUMCAL

70 FORMAT(' TECLEE LAS ',I1,' CALIFICACIONES DE: ',32A1,

' (',I1,' F3.0)')

EL SIGUIENTE "READ" LEE DE UNA SOLA VEZ TODAS LAS
CALIFICACIONES POR MEDIO DE UN "DO" IMPLICITO DE LECTURA,
POR LO QUE LOS DATOS DEBERAN ESTAR TODOS EN LA MISMA LINEA

READ(5,80) (CALIF(INDCAL),INDCAL=1,NUMCAL)

80 FORMAT(9F3.0)

PROALU = 0.0

EL SIGUIENTE "DO" ACUMULA TODAS LA CALIFICACIONES
EN LA VARIABLE "PROALU"

DO 90 INDICAL=1,NUMCAL

PROALU = PROALU+CALIF(INDCAL)

90 CONTINUE

PROALU = PROALU/NUMCAL

WRITE(6,100)INDALU,NOMBRE,PROALU

100 FORMAT(1X,I2,'.- ',32A1,5X,F5.2)

PROGPO = PROGPO + PROALU

110 CONTINUE

PROGPO = PROGPO/NUMALU

WRITE(6,120)PROGPO

120 FORMAT(//,' PROMEDIO GENERAL DEL GRUPO: ',F5.2)

STOP

END

DIRECTORIO DE ALUMNOS DEL CURSO "LENGUAJE DE PROGRAMACION FORTRAN"
IMPARTIDO EN ESTA DIVISION DEL 9 DE NOVIEMBRE AL 8 DE DICIEMBRE.

- 1.- BOTELLO ESQUIVEL MIGUEL ANGEL
I. M. P.
PASANTE "C"
AV. 100 METROS No. 132
DELEGACION GUSTAVO A. MADERO
567-66-00
LUCERNA No. 72-2
COL. JUAREZ
06600 MEXICO, D.F.
546-93=17
- 2.- CORIA ALVAREZ DIANA
RETORNO 7 DE C. ROBELO
COL. JARDIN BALBUENA
DELEGACION VENUSTIANO CARRANZA
15900 MEXICO, D.F.
768-55-94
- 3.- CORTES RODRIGUEZ JESUS
SEDUE
JEFE DE OFICINA
AV. CONSTITUYENTES No. 947
COL. BELEN DE LAS FLORES
DELEGACION ALVARO OBREGON
271-22-64
MARINA NACIONAL No. 200 E 15-D34
COL. ANAHUAC
DELEGACION MIGUEL HIDALGO
271-31-30
- 4.- ELIAS ZUÑIGA ALEX
AV. DEL TRABAJO No. 100-A
PACHUCA; HGO. 42040
3-04-03
- 5.- GONZALEZ GONZALEZ MARIA DEL CARMEN
CENTROS DE INTEGRACION JUVENIL
JEFE DE OFICINA
JOSE MA. OLLOQUI No. 48
COL. DEL VALLE
DELEGACION BENITO JUAREZ
03100 MEXICO, D.F.
534-34-35
AMORES No. 1636 EDIF. B DEPTO. 302
COL. DEL VALLE
DELEGACION BENITO JUAREZ
03100 MEXICO, D.F.
524-76=93
- 6.- HERNANDEZ JOSE ANTONIO
DIREC. GRAL. ESTUDIOS PROSPECTIVOS
ANALISTA
PLAZA DE LA CONSTITUCION Y PINO SUAREZ
4o. PISO
BAHIA No. 22 MANZ. 28
COL. LAS AGUILAS
DELEGACION ALVARO OBREGON
651-67-61
- 7.- JULIO SANCHEZ JAIME RAMON
S. A. R. H.
COORDINADOR TECNICO
TEPIC No. 40-2o. PISO
COL. ROMA SUR
584-25-67
HIGUERAS No. 14
JARDINES DE SAN MATEO
53240 MEXICO, D.F.
584-25-67

8.- LUNA SAAVEDRA MARIO
TELEFONOS DE MEXICO, S.A.
ANALISTA
RIO SENA No. 49-8o. PISO
COL. CUAUHEMOC
DELEGACION CUAUHEMOC
06500 MEXICO, D.F.
525-15-30

BEETHOVEN No. 46-3
COL. PERALVILLO
DELEGACION CUAUHEMOC
06220 MEXICO, D.F.
597-74-80

9.- MARTINEZ MOLINA NORA ANGELICA
DIRE. GRAL. CONSTRUC. OPERAC. HIDRAULICA
PROGRAMADOR
SAN ANTONIO ABAD No. 231
COL. OBRERA
DELEGACION CUAUHEMOC
06800 MEXICO, D.F.
588-32-27

AV. ACUEDUCTO No. 161-F-14
COL. HUIPULCO, TLALPAN
DELEGACION TLALPAN
655-67-24

10.- MERCADO CERON ROBERTO
ORDENADORES SIST. SOPORTE
JEFE DE INFORMATICA
ALFONSO ESPARZA OTEO No. 144-5o. PISO
COL. GUADALUPE INN
DELEGACION BENITO JUAREZ
01020 MEXICO, D.F.
548-49-65

PRIV. MARTIRES DE TACUBAYA No. 5-12
DELEGACION MIGUEL HIDALGO

11.- MILLONES OLANO JOSE ENRIQUE
PROGRAMA DE LAS NACIONES UNIDAS
PARA EL MEDIO AMBIENTE
CONSULTOR
PRESIDENTE MAZARIK No. 29-5o. PISO
COL. POLANCO
250-15-55 ext. 118

LONDRES No. 62
APARTADO 201
COL. JUAREZ
DELEGACION CUAUHEMOC
250-15-55 ext. 118

12.- MORALES MOLLEMEDO PEDRO
C. F. E.
TECNICO
PRADOS S/N
COL. PRADOS
VILLAHERMOSA, TAB.

HACIENDA DE ATENCO No. 10
COL. FLORESTA-COYOACAN
DELEGACION TLALPAN
14310 MEXICO, D.F.
684-70-94

13.- PEREZ ACUÑA MARIO
PETROLEOS MEXICANOS
INGENIERO DE PROCESO (ANALISTA)
IBSEN No. 43-1er. PISO
COL. CHAPULTEPEC POLANCO
DELEGACION MIGUEL HIDALGO
250-26-11 ext. 2030

MTZ. DE LATORRE No. 28 INT. 4
COL. HEROES DE LA TORRE
DELEGACION GUSTAVO A. MADERO
560-32-00

14.- PEREZ RAMOS EVALIA
S. C.T.
ANALISTA TECNICO
AVS. XOLA Y UNIVERSIDAD
COL. NARVARTE
DELEGACION BENITO JUAREZ
519-51-34

XAVIER SORONDO No. 260
VILLA DE CORTES
DELEGACION BENITO JUAREZ
03530 MEXICO, D.F.
590-54-45

15.- QUINTOS LOPEZ RENE
PRODO COWEN, S.A.
JEFE DEL DEPTO. PROCESO DE DATOS
GRAL. FELIPE ANGELS No. 22
COL. BELLAVISTA
5-15-04-42

CENTENARIO No. 53
COL. PORTALES
DELEGACION BENITO JUAREZ
572-22-47

16.- RAMIREZ GONZALEZ MARGARITA
CENTRO DE INTEGRACION JUVENIL
ANALISTA
JOSE MA. ONOQUI No. 48
COL. DEL VALLE
DELEGACION BENITO JUAREZ
03100 MEXICO, D.F.
534-34-35 al 38

LIMAS No. 12-D
COL. DEL VALLE
DELEGACION BENITO JUAREZ
03100 MEXICO, DF..

17.- RANGEL MARTINEZ ROBERTO
COMISION PLAN NACIONAL HIDRAULICO
PROYECTISTA
TEPIC No. 42
COL. ROMA
574-35-07

XOCHICALCO No. 319 - 12
COL. NARVARTE
DELEGACION BENITO JUAREZ
536-73-15

18.- RUBIO BARRERA RICARDO
HOSPITAL ABC
PROGRAMADOR
SUR 136 ESQ. OBSERVATORIO
COL. AMERICA
277-50-00

PTE 152 ESQ. CEYLAN EDIF. E-1008
COL. UNIDAD MARAVILLAS
368-33-72

19.- RUEDE ALVAREZ ANTONIO
TELEFONOS DE MEXICO, S.A.
INVESTIGADOR ANALISTA
RIO SENA No. 49-80. PISO
COL. CUAUHTEMOC
DELEGACION CUAUHTEMOC
06500 MEXICO, D.F.
525-37-85

MUITLAS No. 8
COL. SAN MATEO TLALTENANGO
CUAJIMALPA
05600 MEXICO, D.F.
812-28-13

20.- SANCHEZ SEGURA EFRAIN LEVIA
CONSTRUCTORA Y URBANIZADORA CUR,S.A.
JEFE DEPTO. ESTIMACIONES Y PRESUPUESTOS
SALTILLO No. 19-50. PISO
COL. EXHIPODROMO CONDESA
06100 MEXICO, D.F.
553-13-22 ext. 532.

LIBERTAD No. 73 INT. 4
COL. PORTALES
DELEGACION BENITO JUAREZ
03300 MEXICO, D.F.
674-20-35

21.- VALDEZ PEREZ GERARDO RAUL

ELEUTERIO MENDEZ No. 25
COL. PORTALES
DELEGACION BENITO JUAREZ
532-76-15